

Fall 12-14-2011

# Multiple Biological Sequence Alignment: Scoring Functions, Algorithms, and Evaluations

Ken D. Nguyen  
*Georgia State University*

Follow this and additional works at: [https://scholarworks.gsu.edu/cs\\_diss](https://scholarworks.gsu.edu/cs_diss)



Part of the [Computer Sciences Commons](#)

---

## Recommended Citation

Nguyen, Ken D., "Multiple Biological Sequence Alignment: Scoring Functions, Algorithms, and Evaluations." Dissertation, Georgia State University, 2011.  
[https://scholarworks.gsu.edu/cs\\_diss/62](https://scholarworks.gsu.edu/cs_diss/62)

This Dissertation is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact [scholarworks@gsu.edu](mailto:scholarworks@gsu.edu).

# MULTIPLE BIOLOGICAL SEQUENCE ALIGNMENT: SCORING FUNCTIONS, ALGORITHMS, AND EVALUATIONS

by

KEN D. NGUYEN

Under the Direction of Dr. Yi Pan

## ABSTRACT

Aligning multiple biological sequences such as protein sequences or DNA/RNA sequences is a fundamental task in bioinformatics and sequence analysis. These alignments may contain invaluable information that scientists need to predict the sequences' structures, determine the evolutionary relationships between them, or discover drug-like compounds that can bind to the sequences. Unfortunately, multiple sequence alignment (MSA) is NP-Complete. In addition, the lack of a reliable scoring method makes it very hard to align the sequences reliably and to evaluate the alignment outcomes.

In this dissertation, we have designed a new scoring method for use in multiple sequence alignment. Our scoring method encapsulates stereo-chemical properties of sequence residues and their substitution probabilities into a tree-structure scoring scheme. This new technique provides a reliable scoring scheme with low computational complexity.

In addition to the new scoring scheme, we have designed an overlapping sequence clustering algorithm to use in our new three multiple sequence alignment algorithms. One of our alignment algorithms uses a dynamic weighted guidance tree to perform multiple sequence alignment in progressive fashion. The use of dynamic weighted tree allows errors in the early alignment stages to be corrected in the subsequence stages. Other two algorithms utilize sequence knowledge-bases and sequence consistency to produce biological meaningful sequence alignments. To improve the speed of the multiple sequence alignment, we have

developed a parallel algorithm that can be deployed on reconfigurable computer models. Analytically, our parallel algorithm is the fastest progressive multiple sequence alignment algorithm.

INDEX WORDS: Abstract, Dissertation, Graduate degree, Georgia State University, Multiple sequence alignments, Parallel multiple sequence alignment Algorithms, Reliability in multiple sequence alignments, Algorithms, Scoring functions

BIOLOGICAL MULTIPLE SEQUENCE ALIGNMENT: SCORING FUNCTIONS, ALGORITHMS,  
AND EVALUATIONS

by

KEN D. NGUYEN

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy  
in the College of Arts and Sciences  
Georgia State University

2011

Copyright by  
Ken Dinh Nguyen  
2011

BIOLOGICAL MULTIPLE SEQUENCE ALIGNMENT: SCORING FUNCTIONS, ALGORITHMS,  
AND EVALUATIONS

by

KEN D. NGUYEN

Committee Chair: Yi Pan

Committee: Guantao Chen

Alex Zelikovsky

Anu G. Bourgeois

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2011

## ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yi Pan, for his long and continuous encouragement and guidance. His everlasting patience and supports give me the opportunity to complete this very long process. He has always been there helping and guiding me in taking initiative steps to success. All the grants, fellowships, and awards given to me are the results of his efforts.

I would like to thank my committee members: Dr. Alex Zelikovsky, Dr. Anu G. Bourgeois and Dr. Guantao Chen for their patience, criticism, and help.

Special thanks to Dr. Raj Sunderraman and Dr. Kimberly N. King whose have been my most favorite professors, especially their passionate teaching styles - One is very detail-oriented and the other is very energetic.

Special thanks to Dr. Martin Fraser, our Computer Science Department first Chair, whose had infused me with the idea that teaching could be a career for me and then offered me a job.

I would like to thank the Molecular Basis of Disease (MBD) program at Georgia State University for their scholarly and financial supports.

I also would like to thank the National Science Foundation for indirect financial supports.

Finally, I would like to thank our computer science department, including all the faculty, staffs, and students who I have been working with. It has been a long and pleasure journey for me. It's been the best place to be in, either for studying or working. I'm sure I'll be missing you all.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b> . . . . .	<b>iv</b>
<b>LIST OF FIGURES</b> . . . . .	<b>x</b>
<b>LIST OF TABLES</b> . . . . .	<b>xviii</b>
<b>LIST OF ABBREVIATIONS</b> . . . . .	<b>xix</b>
<b>Chapter 1 INTRODUCTION</b> . . . . .	<b>1</b>
1.1 Motivation . . . . .	1
1.2 The Organization of This Study . . . . .	2
1.3 Sequence Fundamentals . . . . .	3
1.3.1 Protein . . . . .	3
1.3.2 DNA/RNA . . . . .	4
1.3.3 Sequence Formats . . . . .	5
1.3.4 Motifs . . . . .	7
1.3.5 Sequence Databases . . . . .	8
<b>Chapter 2 PROTEIN/DNA/RNA PAIRWISE SEQUENCE ALIGNMENT</b> . .	<b>10</b>
2.1 Sequence Alignment Fundamentals . . . . .	10
2.2 Dot-Plot Matrix . . . . .	11
2.3 Dynamic Programming . . . . .	13
2.3.1 Needle-Wunch's Algorithm . . . . .	13
2.3.2 Example . . . . .	14
2.3.3 Smith-Waterman's Algorithm . . . . .	15
2.3.4 Affine gap penalty . . . . .	18
2.4 Word Method . . . . .	18
2.5 Searching Sequence Databases . . . . .	20
2.5.1 FASTA . . . . .	20
2.5.2 BLAST . . . . .	22



<b>Chapter 3</b>	<b>QUANTIFYING SEQUENCE ALIGNMENTS</b>	<b>24</b>
3.1	Evolution and Measuring Evolution	24
3.1.1	Jukes and Cantor's Model	25
3.1.2	Measuring Relatedness	27
3.2	Substitution Matrices and Scoring Matrices	27
3.2.1	Identity Scores	27
3.2.2	Substitution/Mutation Scores	28
3.3	Gaps	32
3.3.1	Sequence distances	33
3.3.2	Example	33
3.4	Scoring Multiple Sequence Alignments	34
3.4.1	Sum-of-Pair Score	35
3.5	Circular Sum Score	37
3.6	Conservation Score Schemes	37
3.6.1	Wu and Kabat's Method	38
3.6.2	Jores's Method	38
3.6.3	Lockless and Ranganathan's Method	39
3.7	Diversity Scoring Schemes	39
3.7.1	background	39
3.7.2	Methods	40
3.8	Stereochemical Property Methods	41
3.8.1	Valdar's Method	41
3.9	A New Contribution: Hierarchical Expected matching Probability scoring metric (HEP)	44
3.9.1	Building an AACCH scoring Tree	44
3.9.2	The scoring metric	46
3.9.3	Proof of scoring metric correctness	47
3.9.4	Examples	48
3.9.5	Scoring metric and sequence weighting factor	49
3.9.6	Evaluation Datasets	50
3.9.7	Evaluation Results	53

<b>Chapter 4</b>	<b>SEQUENCE CLUSTERING</b>	<b>60</b>
4.1	Unweighted Pair Group Method with Arithmetic mean - UPGMA	61
4.2	Neighborhood Joining Method - NJ	62
4.3	A new Contribution: Overlapping Sequence Clustering	66
<b>Chapter 5</b>	<b>MULTIPLE SEQUENCES ALIGNMENT ALGORITHMS</b>	<b>70</b>
5.1	Dynamic Programming	71
5.1.1	DCA	71
5.2	Progressive Alignment	73
5.2.1	Clustal Family	73
5.2.2	PIMA: Pattern-induced multi-sequence alignment	75
5.2.3	PRIME: Profile-based Randomized Iteration Method	77
5.2.4	DIALIGN	77
5.3	Consistency and Probabilistic MSA	78
5.3.1	POA: Partial Order graph Alignment	79
5.3.2	PSAlign	79
5.3.3	ProbCons: Probabilistic consistency-based multiple sequence alignment	80
5.3.4	T-COFFEE: Tree-based Consistency Objective Function For alignment Evaluation	82
5.3.5	MAFFT: MSA based on Fast Fourier Transform	83
5.3.6	AVID	84
5.3.7	Eulerian path MSA	85
5.4	Genetic Algorithms	85
5.4.1	SAGA: sequence alignment by genetic algorithm	86
5.4.2	GA and Self-organizing Neural Networks	88
5.4.3	FAlign	89
5.5	New Contributions	89
5.5.1	KB-MSA: Knowledge-based Multiple Sequence Alignment	89
5.5.2	PADT: Progressive multiple sequence Alignment based on Dynamic weighted Tree	99
5.6	Test Data and Alignment Methods	102
5.7	Results	103
5.7.1	Measuring Alignment Quality	103

5.7.2	RT-OSM Results . . . . .	103
<b>Chapter 6</b>	<b>MULTIPLE SEQUENCE ALIGNMENT ON HIGH-PERFORMANCE COMPUTING MODELS . . . . .</b>	<b>107</b>
6.1	Parallel Systems . . . . .	107
6.1.1	Multiprocessor . . . . .	107
6.1.2	Vector . . . . .	107
6.1.3	GPU . . . . .	108
6.1.4	FPGA . . . . .	108
6.1.5	Reconfigurable Mesh . . . . .	108
6.2	Exiting Parallel Multiple Sequence Alignment . . . . .	108
6.3	Reconfigurable-Mesh Computing Models - (R-Mesh) . . . . .	110
6.4	Pair-wise Dynamic Programming Algorithms . . . . .	112
6.4.1	R-Mesh Max Switches . . . . .	112
6.4.2	R-Mesh Adder/Subtractor . . . . .	112
6.4.3	Constant-Time Dynamic Programming on R-Mesh . . . . .	114
6.4.4	Affine Gap Cost . . . . .	118
6.4.5	R-Mesh On/Off Switches . . . . .	119
6.4.6	Dynamic Programming Back-tracking on R-Mesh . . . . .	120
6.5	Progressive Multiple Sequence Alignment on R-Mesh . . . . .	121
6.5.1	Hierarchical Clustering on R-Mesh . . . . .	122
6.5.2	Constant Run-time Sum-of-Pair Scoring Method . . . . .	123
6.5.3	Parallel Progressive MSA Algorithm and Its Complexity Analysis . . . . .	125
<b>Chapter 7</b>	<b>WEB-SERVER FOR SEQUENCE ANALYSIS IMPLEMENTATION</b>	<b>128</b>
7.1	SeqAna: A New Sequence Analysis Web-Server . . . . .	128
7.2	Multiple Sequence Alignment Ranking . . . . .	129
7.3	Multiple Sequence Alignment Analysis . . . . .	130
7.4	Multiple Sequence Format Converter . . . . .	131
7.5	Pairwise Sequence Analysis . . . . .	132
7.6	Sequence Retrieval and other services . . . . .	132

<b>Chapter 8</b>	<b>CONCLUSIONS AND FUTURE RESEARCH DIRECTION . . .</b>	<b>134</b>
<b>REFERENCES . . . . .</b>		<b>136</b>
<b>APPENDICES . . . . .</b>		<b>147</b>
APPENDIX A: PUBLICATIONS RELATED TO THIS RESEARCH . . . . .		147
REFEREED JOURNALS AND CONFERENCE PROCEEDINGS . . . . .		147
POSTERS, ABSTRACTS,PRESENTATIONS AND INVITED TALKS . . . . .		147
PAPERS UNDER REVIEW . . . . .		148

## LIST OF FIGURES

Figure 1.1	The yeast Sec23/24 heterodimer 1M2V: (a) protein structure and (b) primary sequence . . . . .	3
Figure 1.2	This figure shows the four fundamental wobble base pairs . . . . .	5
Figure 1.3	BLOSUM62 Substitution matrix . . . . .	6
Figure 1.4	Transcription and Translation process: DNA $\rightarrow$ RNA $\rightarrow$ Protein (the "central dogma" of biology . . . . .	6
Figure 1.5	NCBI's sequence formats . . . . .	7
Figure 1.6	Illustration of GenBank sequence format . . . . .	8
Figure 1.7	This figure shows (a) Phylis format and (b) Clustal format . . . . .	9
Figure 2.1	Dot plot matrix of two sequence ACACACTA and sequence AGCACACA. The connected diagonal lines indicate matching segments. . . . .	12
Figure 2.2	Structures of the CALM3 protein. Based on PyMOL rendering of PDB 1a29. . .	13
Figure 2.3	Dot plot matrix of human CalM against itself to reveal motifs. . . . .	14
Figure 2.4	Generating a scoring the matrix for sequence "GAATTCAGTTA" and sequence "GGATTCCGA" . . . . .	15
Figure 2.5	Back-tracking,((a), (b), and (c)), to obtain the optimal alignment (d) . . . . .	16
Figure 2.6	Global alignment of the two sequences . . . . .	16
Figure 2.7	Smith-Waterman's local alignment of two sequences: "G A A T T C A G T T A " and "G G A T T C C G A" . . . . .	18
Figure 2.8	This figure shows all available words of size 3 . . . . .	19
Figure 2.9	This figure shows an exact word matched . . . . .	19

Figure 2.10	(a)-filtering dot-plot; (b)- being reevaluated by score matrix; (c) - filtering highest scored diagonals starting from the core-region - identified by an arrow; (d)- assembling the diagonals . . . . .	21
Figure 3.1	A Human Calmodulin wraps around its binding domain in the plasma membrane. $\text{Ca}^{2+}$ atoms are depicted as circles. . . . .	25
Figure 3.2	This figure shows the possible paths a nucleotide can be mutated to with the same probability $\alpha$ in Jukes and Cantor's model . . . . .	26
Figure 3.3	(a) shows the score by edit distance, and (b) shows the score by BLOSUM62 matrix . . . . .	34
Figure 3.4	This figure illustrate the steps in assembling an MSA . . . . .	34
Figure 3.5	Traversal of a tree by SP measurement. The total visits on each edge is shown. Some edges are visited more often than others . . . . .	36
Figure 3.6	Circular traversal of a tree. The dotted arrow indicates the traversal path . . . . .	37
Figure 3.7	Taylor's Venn Diagram of amino acid properties . . . . .	42
Figure 3.8	The True table of Taylor's Venn Diagram . . . . .	42
Figure 3.9	Amino Acid Class Hierarchy (AACH) used in PIMA [106]. Upper case characters are amino acids; lower case characters are amino acid classes. X is the wild-card character of any type, including a gap. . . . .	45
Figure 3.10	HEP scoring tree generated from BLOSUM62 substitution matrix. The matching cardinality of each amino acid class is shown at the bottom of the corresponding class symbol. . . . .	46
Figure 3.11	PIMA Active Scoring Tree for the 3rd column . . . . .	49
Figure 3.12	Matching a Leucine from NNV0 with a similar amino acid in sequence SFV1 [51]. The dotted arrow shows a mismatch, the dashed arrow shows a better match, and the solid arrow shows the best match. . . . .	49

Figure 3.13	The RT OSM sequences. The six motifs of the RT OSM are indicated by roman numeral(I-VI). The bold and capitalized letters represent the core amino acids of each motif. Adapted from [51] . . . . .	52
Figure 3.14	Motif detection accuracy on the RT-OSM data set. Score of 1.0 means a method correctly detects and ranks all six motifs in the data set. . . . .	55
Figure 3.15	Reliability scores on the BAliBASE3.0 benchmark . . . . .	57
Figure 3.16	Reliability scores on 150 references of the PREFAB4.0 benchmark . . . . .	57
Figure 3.17	BAliBASE Total Column (TC) scores . . . . .	59
Figure 3.18	Total Column (TC) scores of 150 random sequence sets from PREFAB4.0 . . . .	59
Figure 4.1	(a) is the input sequences, (b) substitutions on first column, and (c) substitutions on second column . . . . .	61
Figure 4.2	This figure show each step of building the UPGMA phylogeny . . . . .	62
Figure 4.3	Producing NJ tree - not drawn to scale. (a) is initial star tree; (b) shows the combined OTU of A and B; (c) is the NJ tree with the new OTU AB; and (d) is the final NJ evolution tree . . . . .	64
Figure 4.4	This figure illustrates different alignment outcomes from the same input sequences. a) shows the input sequences with the motifs in bold; b) shows the expected alignment; c) and d) show the alignment based on different phylogeny guiding tree. . . . .	66
Figure 4.5	This figure illustrates the clustering of the sequences to generate a sequence pattern. (a) is the all-pairwise alignments, (b) is sequence clusterization, and (c) is sequence pattern identification, where S' is the cluster pattern. The boxes represent the DP local alignment results. . . . .	69
Figure 5.1	Sample alignment of BAliBASE reference 1 [115] ubiquitin set. Red blocks represent alpha helices and green blocks represent beta strands . . . . .	70
Figure 5.2	Illustration of the divide and conquer multiple sequence alignment (DCA) . . . .	71

Figure 5.3	An example of progressive multiple sequence alignment. (a) is the input sequences, (b) is an alignment guiding tree, (c) is external nodes alignment, and (d) is internal nodes alignment . . . . .	74
Figure 5.4	The Amino Acid Class Hierarchy (AACCH) used in PIMA family; X represent a wild-card for matching any symbol, including gap . . . . .	76
Figure 5.5	A pattern generated from a pair-wise alignment . . . . .	76
Figure 5.6	POA algorithm: (a) input sequences, (b)directed graphs for the sequences, (c) graph fusion of sequence S1 and S2, (d) alignment of S1 and S2, (e) graph fusion of S3 to alignment (S1,S2),(f) final partial order graph and final multiple sequence alignment	79
Figure 5.7	PSAlign algorithm: (a) input sequences, (b) minimum spanning tree, (c) undirected graph constructed from pair-wise alignments and spanning three, (d)partial order graph, (e) final alignment result . . . . .	80
Figure 5.8	Basic pair-HMM for aligning two sequences $x$ and $y$ . State $M$ emits two letters that being aligned from each sequence. State $I_x$ and $I_y$ each emits a letter in sequence $x$ and $y$ that are aligned to a gap. . . . .	81
Figure 5.9	Generating extended pair-wise alignment in T-Coffee. (a) sequence SeqA is aligned to sequence SeqC via immediate sequence SeqB. (b) new pair-wise alignment for sequence SeqA and sequence SeqC . . . . .	82
Figure 5.10	T-Coffee multiple sequence alignment schema . . . . .	83
Figure 5.11	Representing 3 sequences in de Bruijn graph . . . . .	85
Figure 5.12	Superpath transformation: (a) detachment and (b) x-cut . . . . .	86
Figure 5.13	SAGA alignment scheme. At any generation $G_i$ the parents $P^i$ are crossed breed or mutated by a random operation $X$ to generate a new set of children $P^{i+1}$ . . . .	87
Figure 5.14	Illustration of SAGA, where the two parents are crossed breed. The dotted boxes represent the consistent alignment between the two parents . . . . .	87
Figure 5.15	Neural networks . . . . .	89



- Figure 5.16 Knowledge-based multiple sequence alignment scheme. Initially, the input sequences are partitioned into semi-related groups. The groups' representative sequences are used to search for any available biological information from sequence knowledge-bases. The sequences are given different weights and re-grouped based on the discovered knowledge. . . . . 91
- Figure 5.17 Clusters alignment by partial order graph, where bold texts represent beta strand blocks, italics represents alpha-helix blocks, and others are non-significant blocks. (a) and (b) show the two clusters to be aligned, each sequence in the cluster is preceding by its name, (c) is the partial order graph represents clusters (a) and (b), (d) is the fusing graph of (a) and (b). These sequences are obtained from BALiBASE . . . 93
- Figure 5.18 Finding the max matching score between two partial order graph nodes by sliding techniques. The overlapping columns are enclosed in the rectangle. Non overlapping columns are considered matching with gaps . . . . . 94
- Figure 5.19 Creation of an annotated sequence for KB-MSA consistency database. (a) shows the input sequences and the final annotated pattern for seq1 having three featured blocks f1, f2, and f3 with weights  $1/3$ ,  $2/3$ , and  $1/3$  respectively; (b) shows the immediate pair-wise local alignments of seq1 and their weight marking vectors . . . . . 94
- Figure 5.20 KB-MSA tested against ClustalW, T-Coffee, PROBCONS, and MAFFT on BALiBASE benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by more than 9%. With 10% sequence knowledge, KB-MSA yields about 4% improvement . . . . . 97
- Figure 5.21 KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT on SABmark benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by at least 8% on the original datasets and more than 6% on false positive datasets . . . . . 97
- Figure 5.22 KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT on HOMSTRAD. With complete sequence knowledge, KB-MSA increases average alignment score about 10% on datasets with less than 20% sequence identity . . . . . 98

Figure 5.23	KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT. With complete sequence knowledge, KB-MSA increases average alignment score at least 7% on datasets with less than 20% sequence identity . . . . .	99
Figure 5.24	This figure illustrates the changeability between branches of an alignment guiding tree. Assuming the distances from A, B, C, and D to G are all less than $\epsilon$ , then A, B, C, and D are interchangeable. . . . .	102
Figure 5.25	Percentage of motifs aligned by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on RT-OSM sequence set PADT-NJ and PADT-UPGMA surpass PROBCONS by 5% and 7% respectively . . . .	104
Figure 5.26	Total column (TC) score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on BALiBASE Ref#1, Ref #2, and Ref#3. PADT-NJ gains at least 9% improvement on Ref#2. . . . .	105
Figure 5.27	Total column (TC) score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on BALiBASE Ref#4 and Ref#5. PADT-NJ gains 6% on Ref#5 over DIALIGN. . . . .	105
Figure 5.28	Quality score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on PREFAB datasets. On the first group, PADT-UPGMA gains 4% improvement over DIALIGN, and more than 10% improvement over others. . . . .	106
Figure 6.1	Allowable configurations on 4 port processing units; (a) shows the ports directions; (b) shows the 15 possible port connections, where the last five port configurations in curly braces are not allowed in Linear R-Mesh (LR-Mesh) models. . . . .	111
Figure 6.2	Two 1-bit max switches. (a) - fusing {NSEW} to find the max of two inputs from North and West ports; (b) - construction of a 1-bit 4-input max switch. . . . .	113
Figure 6.3	An n-bit 3-input max switch, where the rectangle represents a 1-bit 4-input max switch from Figure 6.2. . . . .	113

- Figure 6.4 An  $n$ -bit adder/subtractor that can perform addition or subtraction between two 1UN numbers during a broadcasting time. For additions the inputs are on the North and West borders and the output is on the South border. For subtractions, the inputs are on the West and South borders and the output is on the North border. The number on the West bound is 1-bit left-shifted. The dotted lines represent the omitted processing units that are the same as the ones in the last rows. This figure shows the addition of 3 and 3. Note: the leading 1 bit of input number on the West-bound (left) has been shifted off. The right border is fed with zero (or no signal) during the subtract operation. . . . . 114
- Figure 6.5 A dynamic programming R-Mesh. Each cell  $c_{i,j}$  is a combination of a 3-input max switch and three adder/subtractor units. The "+" and "-" represent the actual functions of the adder/subtractor units in the configuration. . . . . 116
- Figure 6.6 A configuration of a 4-way max switch to solve the longest common subsequence (lcs). The South-East processing unit (in bold) configures NS, E, W if the symbols at row  $i$  and column  $j$  are different; otherwise, it configures N,E,SW. This figure shows a configuration when the two symbols are different. . . . . 118
- Figure 6.7 A configuration to select one of the two inputs in 1UN notation using the right most bit as a selector  $s$ . When  $s = 1$  the switch is turned on to allow the data to flow through and get selected by the max switch. When the selector  $s = 0$ , the on/off switch produces zeros and the other data flow will be selected.  $\epsilon = o - g, o \geq g$ , is the difference between opening gap cost  $o$  and extending gap cost  $g$ . . . . . 119
- Figure 6.8 An  $n \times n + 1$  R-Mesh represents an  $n$ -bit on/off switches. By default, all processing units on the last column (column  $n + 1$ ) configure {NS,E,W}, and fuse {NSEW} when a signal (i.e. 1) travels through them. All cells on the main anti-diagonal cells of the first  $n$  rows and columns fuse {NE,S,W}, cells above the main anti-diagonal fuse {NS,E,W}, and cells below the main anti-diagonal fuse {N,S,E,W}. Figure (a) shows the R-Mesh configuration on a selector bit of 1 ( $s=1$ ) and Figure (b) shows the R-Mesh configuration on a selector bit of 0 ( $s= 0$ ). . . . . 120
- Figure 7.1 Front page of SeqAna Web-Server . . . . . 128

Figure 7.2	Multiple Sequence Alignment Ranking Service . . . . .	130
Figure 7.3	Multiple Sequence Alignment Service . . . . .	131
Figure 7.4	Sequence Format Converting Service . . . . .	131
Figure 7.5	Pair-wise Alignment Service . . . . .	132
Figure 7.6	BLAST Service . . . . .	133

## LIST OF TABLES

Table 1.1	Common Amino Acids . . . . .	4
Table 1.2	Ambiguous Amino Acids . . . . .	4
Table 3.1	DNA/RNA scoring matrix used by NCBI . . . . .	28
Table 3.2	250 PAM substitution matrix . . . . .	30
Table 3.3	BLOSUM 45 substitution matrix . . . . .	31
Table 3.4	GONNET matrix generated from SWISS-PROT database . . . . .	31
Table 3.5	Each label column represents a residue position in a multiple sequence alignment. Amino acids are identified by their one letter code and gaps by a dash (" "). Note: column ( $k$ ) comes from an alignment of 10 sequences where column ( $j$ ) comes from an alignment of only 4 sequences (no gaps in column ( $j$ )). The column score order is $(a) > (b) > (c) > (d) > (e) > (f)$ , then $(g) > (h) > (i)$ , and $(j) > (k)$ (adapted from [121]) . . . . .	52
Table 3.6	Each label column represents a residue position in a multiple sequence alignment. Amino acids are identified by their one letter code and gaps by a dash (" "). The column score correct order is $(a) > (b) > (c) > (d) > (e) > (f)$ , then $(g) > (h) > (i)$ , and $(j) > (k)$ . Note: column ( $j$ ) comes from an alignment of only 4 sequences (no gaps); and the table cannot show all significant digits. . . . .	54
Table 6.1	Summary of Progressive Multiple Sequence Alignment Components along with their time and CPU complexity . . . . .	127

## LIST OF ABBREVIATIONS

- AACCH Amino Acid Covering Class Hierarchy
- ACO Ant Colony Optimization
- ABSMS Average Biological Significant Matching Score
- BLAST Basic Local Alignment Search Tool
- CRCW Concurrent-Read, Concurrent-Write
- CS-MSA Consistency-Based Multiple Sequence Alignment
- DCA Divide and Conquer multiple sequence Alignment
- FAMS Feature average Median Score
- FDCA Fast Divide and Conquer multiple sequence Alignment
- FFT fast Fourier transform
- FPGA Field-Programmable gate Arrays
- GA Genetic Algorithm
- GPU Graphics Processing Unit
- HEP Hierarchical Expected matching Probability
- HMM Hidden Markov Model
- KB-MSA Knowledge-Based Multiple Sequence Alignment
- LCS Longest Common Subsequence
- LR-Mesh Linear Reconfigurable Mesh
- MIMD Multiple-Instruction, Multiple-Data
- OSC Overlapping Sequence Clustering
- PADT Progressive Alignment based on Dynamic guiding Tree

- PAM Point Accepted Mutation
- PARBS Processor Array with Reconfigurable Bus System
- PDP Protein Data Bank
- PE Processing Element
- PIMA Pattern-Induced Multi-sequence Alignment
- PR-Mesh Pipelined Reconfigurable Mesh
- PRAM Parallel random Access Machine
- PRIME Profile-based Randomized Iteration Method
- ProbCons Probabilistic Consistency-based multiple sequence alignment
- PU Processing Unit
- RBT-GA Rubber Band Technique with Genetic Algorithm
- R-Mesh Reconfigurable Mesh
- RT-OSM Reverse-Transcriptase Order-Specific Motifs
- SIMD Single-instruction, Multiple-Data
- T-COFFEE Tree-based Consistency Objective Function For alignmEnt Evaluation
- TC Total Column
- 2D 2 Dimension
- ABSMS Average Biological Significant Matching Score
- BLAST Basic Local Alignment Search Tool
- BLOSUM BLOcks of Amino Acid Substitution Matrix
- CS Circular Sum
- DNA Deoxyribonucleic Acid
- DP Dynamic Programming

- FASTA FAST-All
- GCM Glial Cells Missing
- HMM Hidden Markov Model
- HSP High Score Pair
- mRNA messenger RNA
- MSA Multiple Sequence Alignment
- MSP Maximal Segment Pair
- NCBI National Center for Biotechnology Information
- NJ Neighborhood Joining
- NP Non-deterministic Polynomial time
- OTU Operational Taxonomic Unit
- PDB Protein Data Bank
- RNA Ribonucleic Acid
- rRNA ribosomal RNA
- SP Sum-of-Pair
- tRNA transfer RNA
- TSP Traveling Salesman Problem
- UPGMA Unweighted Pair Group Method with Arithmetic
- WPGMA Weighted Pair Group Method with Arithmetic



## Chapter 1

### INTRODUCTION

Majority of organisms on Earth, though diverse, share a significant biological similarity. There is an abundance of biological sequence data showing that any two mammals can have as many as 99% genes in common. Humans and fruit flies are two very different species that share at least 50% common genes. These striking facts have been discovered largely through biological sequence analysis.

Multiple Sequence alignment is a fundamental task in bioinformatics and sequence analysis. In the early 1970's, Deoxyribonucleic acid (DNA) sequences were obtained using laborious methods based on two-dimensional chromatography. Thus, the number of sequences is limited and often being studied and annotated individually by scientists. By the late 70's, Walter Gilbert [71] and Frederick Sanger [100] proposed DNA sequencing by chemical degradation and enzymatic synthesis, respectively. Their works earned a Nobel Prize in chemistry in 1980. Later, sequences are obtained by many newer methods such as dye-based methods [88], micro arrays, mass-spectrometry, x-ray, ultracentrifugation, etc. Since the development of Sanger's method, the volume of sequences being identified and deposited is enormous. The current commercial sequencing such as "454 sequencing" can read up to 20 million bases per run and produce the sequences in hours. With this vast amount of sequences, manually annotating each sequence is infeasible. However, we need to categorize them by family, analyze them, find features that are common between them, etc. The first step to solve this problem is finding the best way to starts with the sequence fundamentals and then leads readers to the most modern and practical alignment techniques that have been proven to be effective in biological sequence analysis.

#### 1.1 Motivation

There are two popular trends in sequence analysis. One trend focuses primarily on applying rigorous mathematical methods to bring out the optimal alignment of the sequences, thus leading to revelation of possible hidden biological significance between sequences. The other trend stretches on correctly identifying the actual biological significance between the sequences, where some or all biological features may have already been known. These two trends emerge from specific tasks bioinformatics scientists are dealing with. The first trend relates to predicting the sequence structures and homology, species evolution, or determine the relationship between sequences in order to categorizing and organizing sequence databases. The second

trend is to perform a daily task in which scientists want to arrange similar known features of the sequences into the same columns to see how closely they are resemble of each other. The second trend can be seen in evolution analysis, sequence structure and functional analysis or in drug design and discovery. In the later case, for each specific virus sequence, drug designers search for possible drug-like compounds from libraries of simple sequence models annotated with functional sites and specific drug-like compounds that can bind [53, 96] . Hence, aligning a sequence obtained from a new virus against the library of sequences may lead to a manageable set of sequences and compounds to work with.

Consequently, these two distinctive perspectives lead to different approaches to sequence alignment, and the development of sequence alignment algorithms, in turn, allows scientists to automate these tremendous and time consuming tasks.

## 1.2 The Organization of This Study

Multiple sequence alignment study involves many aspects of sequence analysis, and it requires broad and significant background information. Therefore, we present each aspect as a chapter starting with existing methodologies and following by our contributions.

The rest of this chapter provides basic information on biological sequences.

Chapter 2 provides fundamentals in pair-wise sequence alignment.

Chapter 3 describes popular existing quantitative models that have been designed for to quantify multiple sequence alignment. In this chapter we present our new quantitative model along with its analysis and evaluation.

Chapter 4 describes practical clustering techniques that have been used in multiple sequence alignment, following by our new method for categorizing biological sequences into related groups for more reliable and effective alignment.

Chapter 5 describes, characterizes and relates many multiple sequence alignment models including two of our new alignment algorithms.

Chapter 6 describes the latest methods developed to improve the run-time efficiency of multiple sequence alignment. A large section of this chapter is devoted to our parallel alignment model on reconfigurable networks.

Chapter 7 describes our implemented web-server for sequence analysis.

Lastly, Chapter 8 summarizes our contributions in this study and provides insights into our future research directions.

### 1.3 Sequence Fundamentals

DNA is the fundamental that characterizes a living organism and its genome, i.e. its genetic information set. DNA contains thousands of genes which carry the genetic information of a cell. Each gene holds information of how to build a protein molecule, which serves as building blocks for the cell or performs important tasks for the cell functions. The DNA is positioned in the nucleus, which is organized into chromosomes. Since DNA contains the genetic information of the cell, it must be duplicated before the cell divides. This technique is called duplication. When proteins are required, the corresponding genes are transcribed into RNA (transcription), the non-coding parts of the RNA are removed, and the RNA is transported out of the nucleus. Proteins are built outside of the nucleus based on the code in the RNA. Thus, DNA sequence determines protein sequence and its structure; and the protein structure dictates the protein function. In this section, we will present the fundamentals of sequences and their basic components.

#### 1.3.1 Protein

Proteins (also known as polypeptides) are organic compounds consisting of amino acids linked with peptide bonds forming a linear polypeptide chain and folded into a globular form. The linear sequence of amino acids in the protein molecule represents its primary structure. The secondary structure refers to regions of local regularity within a protein fold such as  $\alpha$ -helices,  $\beta$ -turns or  $\beta$ -strands. The size of a protein sequence ranges from a few to several thousand residues. Figure 1.1 shows both the structure and the primary sequence of the yeast 1M2V protein.

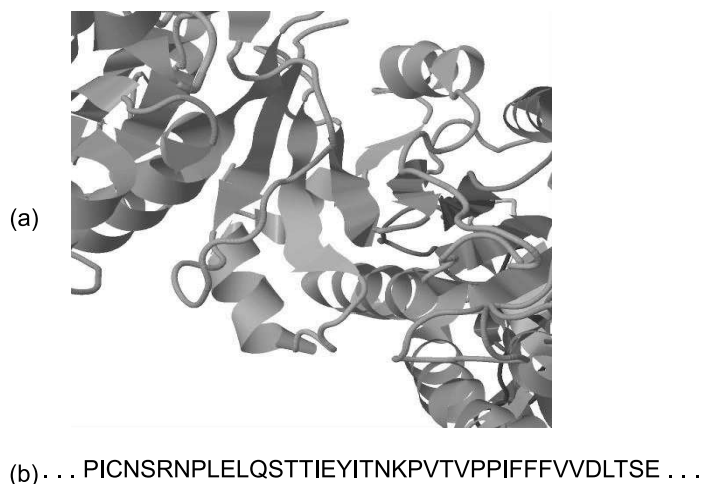


Figure 1.1. The yeast Sec23/24 heterodimer 1M2V: (a) protein structure and (b) primary sequence

Table 1.1. Common Amino Acids

Name	3-Letter	1-Letter
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic Acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V
Unknown or "other"		X

Table 1.2. Ambiguous Amino Acids

Ambiguous Amino Acids	3-Letter	1-Letter
Asparagine or aspartic	Asx	B
Glutamine or glutamic acid	Glx	Z
Leucine or Isoleucine	Xle	J

### 1.3.2 DNA/RNA

Both deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) contain three basic components: (i) a five-carbon sugar, which could be either ribose (as in RNA) or deoxyribose (as in DNA), (ii) a series of chemical groups derived from phosphoric acid molecules, and (iii) four different nitrogen compounds having the chemical properties of bases. The DNA has four bases which are adenine (A), thymine (T), guanine (G) and cytosine (C), and RNA has adenine (A), uracil (U), guanine (G) and cytosine (C) in RNA. Adenine and guanine are double-ring molecules known as purine, while other bases are single-ring molecules known as pyrimidine.

DNA is a linear, double-helix structure, and is composed of two intertwined chains made up of nucleotides. Unlike DNA, RNA is a single-stranded and contains ribose as its sugar. There are three types of RNA: messenger RNA (mRNA), transfer RNA (tRNA) and ribosomal RNA (rRNA). rRNA and tRNA are parts of protein synthesizing engine, and mRNA is a template for protein synthesis. During the process of producing a amino acid chain, the nucleotide sequence of a mRNA is read from one end to the other in a group of three successive bases. These groups are called codons. Each codon is either a coding for a an amino acid or a translation terminate signal. There are 64 (4x4x4) possible codons for the bases.

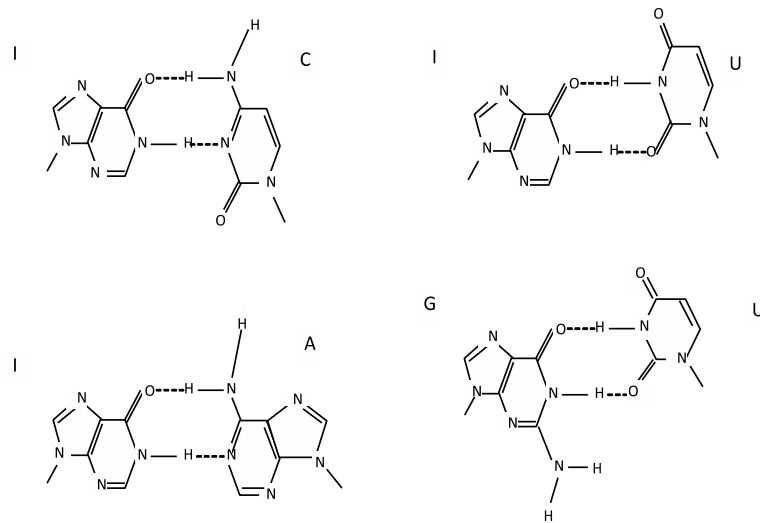


Figure 1.2. This figure shows the four fundamental wobble base pairs

### 1.3.3 Sequence Formats

Protein/DNA/RNA sequences are represented in many different formats such as AB1, ACE, CAF, EMBL, FASTA, FASTAQ, GenBank, PHD, SCF, NEXUS, GFF, Stockholm, SwissProt, etc. In general, the sequence formats can be grouped into four groups: sequencing specific, bared sequence, sequence with features, and alignment sequence. The sequencing specific formats such as AB1 from Applied Biosystems or ACE are used by companies which perform sequencing. The sequence data are obtained in fragments, called reads, and are assembled together. Many sequence formats are specifically designed for different sequencing technologies.

The bared sequence format often represents the sequence residues themselves along with an identification or sequence name. The most classic sequence format is FASTA format (or Pearson format), a text-based format where each sequence is preceded with its name and comments, and the sequence base pairs or amino acids are represented using single-letter codes. Multiple sequences can be included in a file, where each line should be fewer than 120 characters. A comment line starts with character ';' and should only be intended for human. The sequence name, should start with > character, and an asterisk '\*' marks the end of a sequence and can be omitted. Each sequence should be separated by a new line. Following is an example of sequences in FASTA format:

**Example 1** >MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken

ADQLTEEQIAEFKEAFSLFDKDGDTITTKELGTVMRSLGQNPTEIDGDGQVNYEEFVQMMTAK ×



GenBank	<code>gi gi-number gb accession locus</code>
EMBL Data Library	<code>gi gi-number emb accession locus</code>
DDBJ, DNA Database of Japan	<code>gi gi-number dbj accession locus</code>
NBRF PIR	<code>pir  entry</code>
Protein Research Foundation	<code>prf  name</code>
SWISS-PROT	<code>sp accession name</code>
Brookhaven Protein Data Bank (1)	<code>pdb entry chain</code>
Brookhaven Protein Data Bank (2)	<code>entry:chain PDBID CHAIN SEQUENCE</code>
Patents	<code>pat country number</code>
GenInfo Backbone Id	<code>bbs number</code>
General database identifier	<code>gnl database identifier</code>
NCBI Reference Sequence	<code>ref accession locus</code>
Local Sequence identifier	<code>lcl identifier</code>

Figure 1.5. NCBI's sequence formats

Sequences with feature formats allow feature data of the sequence to be included. Figure 1.6 shows a segment of sequence represented in GenBank sequence format. It includes almost everything about the sequence such as the authors of the sequence, the sequence itself, the journal in which it is published, etc.

The alignment sequence formats such as GCG, MSF, Clustal, Phylis, etc. are used to represent sequence alignments. These formats allow the residues from different sequences to be in the same column if they are aligned in the alignment, thus help visualize the alignment of the sequences. For example, Figure 1.7 shows 2 sequences in Phylis and Clustal formats.

### 1.3.4 Motifs

Motifs are short and conserved subsequences of amino acids that characterize a specific biochemical function. Motifs are capable of regulating particular function of a protein, or can determine a sub-structure of a protein. Some sequence motifs are continuous such as the C2H2 zinc finger motifs. However, some motifs are discontinuous and the order in which they occur may be completely different. Thus, identifying these motifs from the sequence is not a simple task. An example of these motifs is the GCM (glial cells missing) motif, found in humans, mice and fruit flies, identified by Akiyama et al. [1]. The GCM motif has the following form: `WDIND*.*P*.*...D.F.*W***.**.IYS**...A.*H*S*WAMRNTNNHN`, where each (.) represents a single amino acid or a gap, and each \* indicates one member of a closely-related family of amino acids.

Since homologous sequences tend to maintain sequence similarity within core-domains [18] and active sites [50], homologous sequences can have low overall sequence similarity [52]. In sequence alignment studies, the terms motif and motifs have broader spectrum, they also refer to conserved segments of

```

LOCUS      SCU49845      5028 bp      DNA      PLN      21-JUN-1999
DEFINITION Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION  U49845
VERSION    U49845.1  GI:1293613
KEYWORDS   .
SOURCE     Saccharomyces cerevisiae (baker's yeast)
ORGANISM   Saccharomyces cerevisiae

REFERENCE  1  (bases 1 to 5028)
AUTHORS    Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
TITLE      Cloning and sequence of REV7, a gene whose function is required for
JOURNAL    Genes Dev. 10 (7), 777-793 (1996)
PUBMED     8846915
FEATURES   Location/Qualifiers
            source          1..5028
                               /organism="Saccharomyces cerevisiae"
                               /db_xref="taxon:4932"
                               /chromosome="IX"
            gene            complement(3300..4037)
                               /gene="REV7"
            CDS              complement(3300..4037)
                               /gene="REV7"
                               /db_xref="GI:1293616"
                               /translation="MNRWVEKWL RVY LKCYINLILFYRNVYPPQSFDTTYQSFNLPQ
                               FVPINRHPALIDYIEELILDVLSKLTHVYRFSICIINKNDLCIEKYVLD FSELQHVD
                               KDDQIITETEVDFEFRSSLNSLIHLEKLPKVNDTITFEAVINAI EELGHLDRNE"
ORIGIN
1  gatcctccat atacaacggt atctccacct cagggttaga tctcaacaac ggaaccattg
61  ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct

```

Figure 1.6. Illustration of GenBank sequence format

sequences that are observed in many sequences without the actual knowledge of the segments' biological functionality.

### 1.3.5 Sequence Databases

Today, there are many sequence databases available. Proteins sequences can be found in SWISS-PROT [11], TrEMBL [11], UniProt [6], PIR [124], and PDP [9] databases. Similarly, DNA and RNA sequences are in Genbank [8], PDB, HOMSTRAD [73], and RefSeq [93] databases. As of 16-Jun-2011 of TrEMBL contains 15,400,876 protein sequence entries comprising 4,982,458,690 amino acids. These entries are automatically annotated and not reviewed. The shortest reported sequence is Q16047-HUMAN from humans containing 4 amino acids (F,P,D, and F), and the longest reported sequence is Q3ASY8-CHLCH containing 36,805 amino acids. TrEMBL's average sequence length is 322 residues. On the other hand, Swiss-Prot is manually annotated and reviewed. It contains 529,056 fully annotated sequence entries, 2.7% of the entries are predicted and about 0.4% of the entries are uncertain. The shortest sequence in Swiss-prot is GWA-SEPOF(P83570) obtained from cuttlefish which contains 2 amino acids (G and W), and the longest sequence is TITIN-MOUSE(A2ASS6) containing 35,213 amino acids. Swiss-prot's average sequence length is 355 residues.



```

MCHU      -----
gi|5524211 LCLYTHIGRN IYGSYLYSE TWNTGIMLLL ITMATAFMGY VLPWQMSFW

-----
GATVITNLFS AIPYIGTNLV EWIWGGFSVD KATLNRFFAF HFILPFTMVA

KDG DGTITTK ELGTVMRSLG QNPTEAELQD MINEVDADGN GTIDFPPEFLT
LAGVHLTFLH ETGSNN-PLG LTSDSDKIPF HPYYTIKDFL GLLILILLLL

MMARKMKD TD SEEEIREAFR VFDK----- DGNGYISAAE LRHVMTNLG-
LLALLSPDML GDPDNHMPAD PLNTPLHIKP EWYFLFAYAI LRSVPNKLGG

-----EKLTD EEVDEMIR-- -----EADIDGD
VLALFLSIVI LGLMPFLHTS KHRSMMLRPL SQALFWTLTM DLLTLTWIGS

GQVNYEEFVQ MMTAK-----
QPVEYPYTII GQMASILYFS IILAFPLIAG XIENY

```

(a)

```

MCHU      -----
gi|5524211|gb|AAD44166.1| LCLYTHIGRNIYGSYLYSETWNTGIMLLLITMATAFMGYVLPWQMSFW 50

MCHU      -----ADQLTEEQIAEFKEAFSLFD 20
gi|5524211|gb|AAD44166.1| GATVITNLFS AIPYIGTNLV EWIWGGFSVD KATLNRFFAF HFILPFTMVA 100
                                     *..  .*  .*:..

MCHU      KDG DGTITTK ELGTVMRSLG QNPTEAELQD MINEVDADGN GTIDFPPEFLT 70
gi|5524211|gb|AAD44166.1| LAGVHLTFLH ETGSNN-PLGLTSDSDKIPF HPYYTIKDFL GLLILILLLL 149
                                     *  :* *;  .** .. . :;  .  *  *  :  : *

MCHU      MMARKMKD TD SEEEIREAFR VFDK----- DGNGYISAAE LRHVMTNLG- 113
gi|5524211|gb|AAD44166.1| LLALLSPDML GDPDNHMPAD PLNTPLHIKP EWYFLFAYAI LRSVPNKLGG 199
::*  *  .:  :  .  :.  :  : *  *  *  .:**

MCHU      -----EKLTD EEVDEMIR-----EADIDGD 133
gi|5524211|gb|AAD44166.1| VLALFLSIVI LGLMPFLHTS KHRSMMLRPLS QALFWTLTMDLLTLTWIGS 249
                                     *..  .* *

MCHU      GQVNYEEFVQ MMTAK----- 148
gi|5524211|gb|AAD44166.1| QPVEYPYTII GQMASILYFS IILAFPLIAG XIENY 284
                                     *:*  :  *.

```

(b)

Figure 1.7. This figure shows (a) Phylis format and (b) Clustal format

## Chapter 2

### PROTEIN/DNA/RNA PAIRWISE SEQUENCE ALIGNMENT

Given a set of biological sequences, it is often a desire to identify the similarities shared between the sequences. This information will give further information about the functionality, originality, or the evolution of the species where these biological sequences are obtained. Theoretically, the best identification technique would be the one that correctly and perfectly aligns all the residues and substructures sharing similar biological and functional features between the sequences and structures. These aligned residue blocks in the sequence alignments suggest possible biological relationships between these sequences and can easily be verified with prior knowledge on these biological data. With some known data, we can logically infer them over to other biological entities involved in the alignment. Without any known information, the alignment of similarities in the sequences obtained from different species could lead to possible discovery of unknown biological meaning. In reality, that technique has yet to exist due to many factors such as the lack of information about the functionality of each region in a sequence, how the regions are correlated, how the two sequences have evolved, or which parts of the sequences are simply random mutations. In practice, many alignment algorithms are design around the two concepts: global alignment and local alignment. In global sequence alignment, all sequences maintain a correspondence over their entire length. On the other hand, in the local alignment only the most similar part of the sequences is aligned. It depends on what is being identified; each concept has its own advantages.

#### 2.1 Sequence Alignment Fundamentals

Sequence alignment problems can be generalized as follows: Let  $W = \{\alpha_1, \dots, \alpha_i, \dots, \alpha_n\}, i > 0$ , be a set of amino acid symbols or nucleotide symbols. A sequence  $s_i = \underbrace{\alpha_i \cdots \alpha_j \cdots \alpha_k}_n$  is a string of  $n$  amino acid symbols. An alignment is a transformation of  $k > 1$  sequences  $\{s_1, s_2, \dots, s_k\}$  into  $\{s'_1, s'_2, \dots, s'_k\}$ , where  $s'_i$  is the  $i^{th}$  sequence  $s_i$  with GAP (-) symbols inserted such that maximizing the similar regions across the sequences. In biological perspective, the gaps represent residue symbols being deleted from the sequences during the course of evolution. In general, when a gap is inserted into a sequence a penalty is applied to the alignment. The weight of the penalty depends on the location of the insertion.

For example, let the sequences to be aligned be  $s_1 : GLISVT$  and  $s_2 : GIVT$ , then a possible alignment is

$$A(s_1, s_2) = \begin{cases} s'_1 : & G & L & I & V & S & T \\ s'_2 : & G & - & I & V & - & T \end{cases}$$

Three common pair-wise alignment techniques are dot-matrix [34], dynamic programming [77, 107], and word method [24]. Each method provides a host of advantages. For example, the dot-matrix provides a good visualization of an alignment; the dynamic program technique guarantees an alignment with optimal score; and the word method (k-tuple) is a time-efficient alignment method that provides reasonable sequence alignments. In this chapter, we will explore the details of these algorithms.

## 2.2 Dot-Plot Matrix

The Dot-Matrix method [34] provides a visualization of potential matching of subsequences between any two sequences. This method is performed as follows:

- (i) For any pair of given sequences of lengths  $m$  and  $n$ , the sequences are mapped into the two perpendicular edges of an  $m \times n$  matrix.
- (ii) A scoring scheme is used to mark the similar between any two residues of the two sequences.

For simplicity, a dot would be placed on the diagonal lines where the two residues are the same, otherwise, that slot is left unmarked. When all  $m \times n$  slots have been considered, dots that resemble contiguous diagonal lines in the matrix represent the similar sections between two sequences. Obviously, two identical sequences will have a diagonal line starting at the beginning joint of the two sequences. Figure 2.2 shows a dot matrix created from aligning sequence ACACACTA against sequence AGCACACA. These two sequences share the same subsequence ACACA. Since this method plots any matching residues between two sequences, it is common that the dot matrix would have many plotted diagonals, and some of these lines are subsequences of longer diagonals. These short diagonals are considered noise. They hinder the capability to recognize the other diagonals and should be removed. The most popular filtering technique is to filter out any diagonal with length less than  $k$ , where  $k$  is arbitrary number or could be derived from the length of the diagonal in the dot matrix. In general,  $k$  is set to the same size of the feature being identified.

Besides providing visualization of possible matches between sequences, another advantage of this method is the capability to show the possible repeated segments between sequences. These segments could be carrying significant biological information sharing between the two sequences such as stem-loops (also

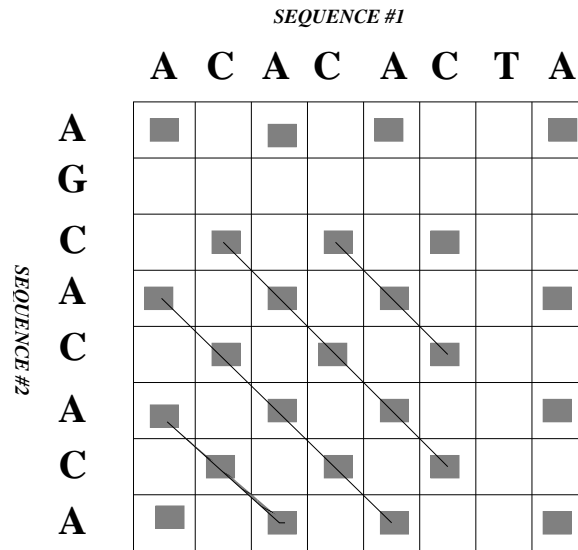


Figure 2.1. Dot plot matrix of two sequence ACACACTA and sequence AGCACACA. The connected diagonal lines indicate matching segments.

known as hairpin loops) or inverted repeats. These are subsequences that are reversed of others subsequences downstream. For example: 5'. . . **GCCGCGGGCCG**AAAAAACCCCCCGGCCCGCGGC . . .3' is an RNA stem-loop. To identify these motifs, a sequence is aligning against itself using dot-matrix. With a filtering window size equal to the length of the motifs being identified, the dot-plot matrix will reveal these repeated segments. Figure 2.2 shows a dot-plot matrix of a human CalM sequence against itself with a window size  $k = 7$  to reveal its fourth EF-hand motifs. [EF-motif contains 2 perpendicular helices E and F wrapping around  $Ca^{2+}$  with a helix loop].

The human CalM sequence is:"MADQLTEEQI AEFKEAFSLF DKDGDGTITT KELGTVMRSL GQNPTEAELQ DMINEVDADG NGTIDFPEFL TMMARKMKDT DSEEEIREAF RVFDKDGNGY ISAAELRHVM TNLGEKLTDE EVDEMIREAD IDGDGQVNYE EFVQMMTAK"

Another advantage, as seen in [87], of dot plot matrix method is to identify the structural information of a sequence. Similar to identifying inverted repeats, a dot-plot matrix is created for a sequence and its reversed sequence. Lines that are perpendicular to the imaginary diagonal represent the stem-loop or hairpin loop. A stem-loop is, common in RNA, a palindromic pattern that exists when two regions of the same molecule base-pair forming a double helix that ends in unpaired loop. In terms of complexity, the dot-plot method takes  $O(nxm)$  for plotting the matrix,  $O(nxm)$  for filtering the noise, and  $O(nxm)$  space for the matrix. Thus, the overall time and space complexity for dot-plot is  $O(nxm)$ . Despite the fact that the dot-plot method is great to identify features being shared between two sequences, it does not provide

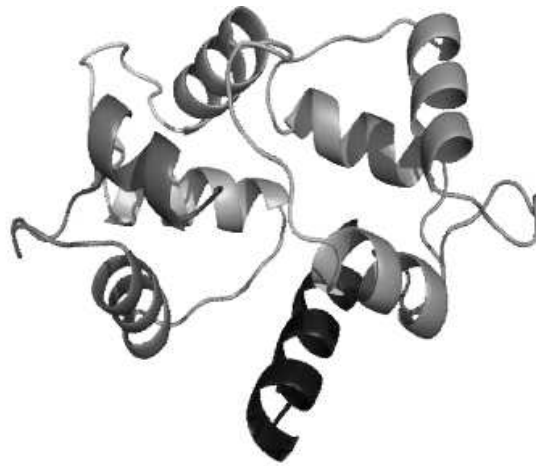


Figure 2.2. Structures of the CALM3 protein. Based on PyMOL rendering of PDB 1a29.

a quantitative similarity measurement between the two sequences. Thus, a simple task such as identifying a pair of the most resembled sequences out of three sequences may not be possible because the sequences do not often share the same segments.

## 2.3 Dynamic Programming

Dynamic programming (DP) is defined by Richard Bellman in 1953 based on an optimal policy such that any subsequence decision must constitute an optimal policy with regard to the result of the first decision, regardless of whatever the first decision is. In sequence alignment term, regardless of the first symbols being chosen to align, all other subsequence aligning symbols guarantee that they yield the optimal alignment score. Dynamic Programming is utilized to produce the global alignment was first introduced in the Needleman-Wunsch's algorithm [77], and a local alignment version is in the Smith-Waterman's algorithm [107]. Similar to the dot-plot technique, two sequences are mapped to the two perpendicular side of an  $m \times n$  matrix. DP then starts from the joined corner of the sequences to calculate all possible matching scores of the residues in the two sequences in the adjacent slots. The process repeated until the scores of all slots are computed. The DP technique takes  $O(nxm)$  for space and time complexity, if the scoring scheme takes constant time to compute.

### 2.3.1 Needle-Wunch's Algorithm

Given two sequences  $a$  and  $b$  of lengths  $m$  and  $n$ , respectively. Their alignment matrix score  $H$  is calculated as follows:

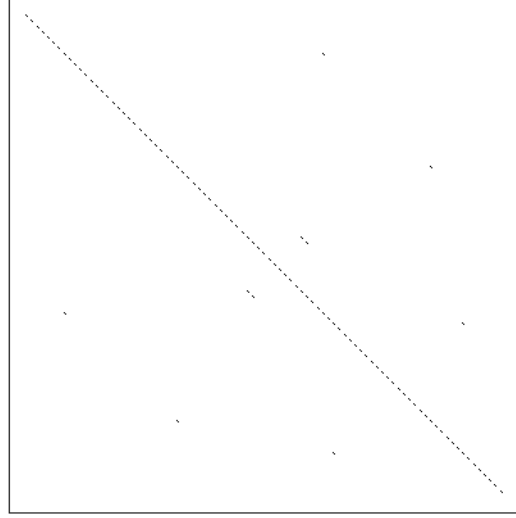


Figure 2.3. Dot plot matrix of human CalM against itself to reveal motifs.

$$H(i, 0) = 0, \leq i \leq m$$

$$H(0, j) = 0, \leq j \leq n$$

$$H(i, j) = \max \left\{ \begin{array}{ll} H(i-1, j-1) + w(a_i, b_j) & \text{Match/Mismatch} \\ H(i-1, j) + w(a_i, -) & \text{Deletion} \\ H(i, j-1) + w(-, b_j) & \text{Insertion} \end{array} \right\}, 1 \leq i \leq m, 1 \leq j \leq n$$

Where:

$a_i$  is the  $i^{th}$  symbol in sequence  $a$

$m = |a|$  is the length of  $a$

$n = |b|$  is the length of  $b$

$H(i, j)$  is the maximum similar score between the subsequence of  $a$  of length  $i$ , and the subsequence of  $b$  of length  $j$ .

$w(c, d), c, d \in \Sigma \cup \{-'\}$ , is the matching score between two residues.

And, the alignment is the trace-back route from the right most end of highest score slot to the starting slot.

### 2.3.2 Example

Given two sequence:

G A A T T C A G T T A (sequence #1)

G G A T T C C G A (sequence #2)

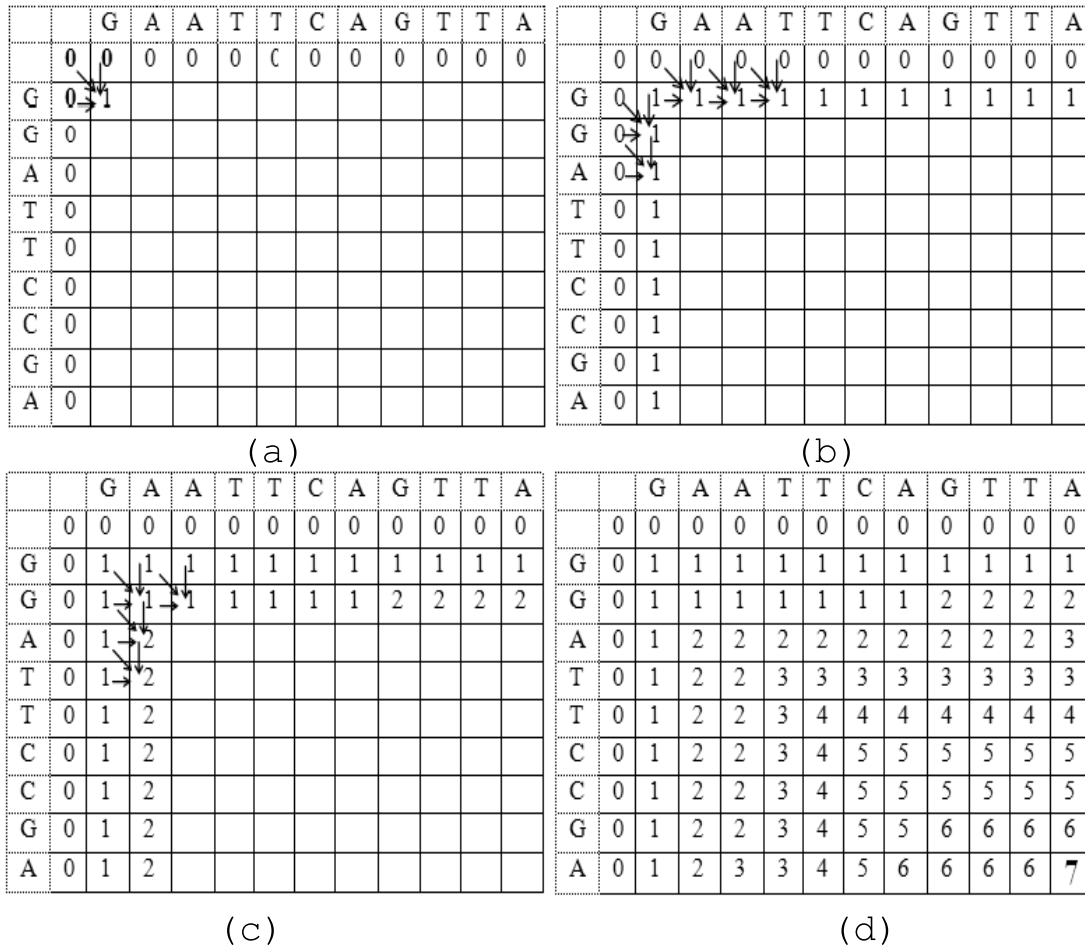


Figure 2.4. Generating a scoring the matrix for sequence "GAATTCAGTTA" and sequence "GGATTCCGA"

and a scoring scheme where a matching pair of symbols get a score of 1, mismatch get a score of 0, and gap penalty is 0. Figures 2.3.2 and 2.3.2 show the steps of Needleman-Wunch's algorithm. And one of the optimal alignment results is shown in Figure 2.3.2

### 2.3.3 Smith-Waterman's Algorithm

From the evolution perspective, two related sequences could evolve independently with many independent mutations lowering the similarity between the sequences. Aligning the sequences with noised information often fails to produce a biologically meaningful alignment. In these cases, the local DP alignment, which proposed by Smith and Waterman, identifies the longest segment pair that yields the best alignment score is more preferable. In the Smith-Waterman's algorithm, the longest segment pair between

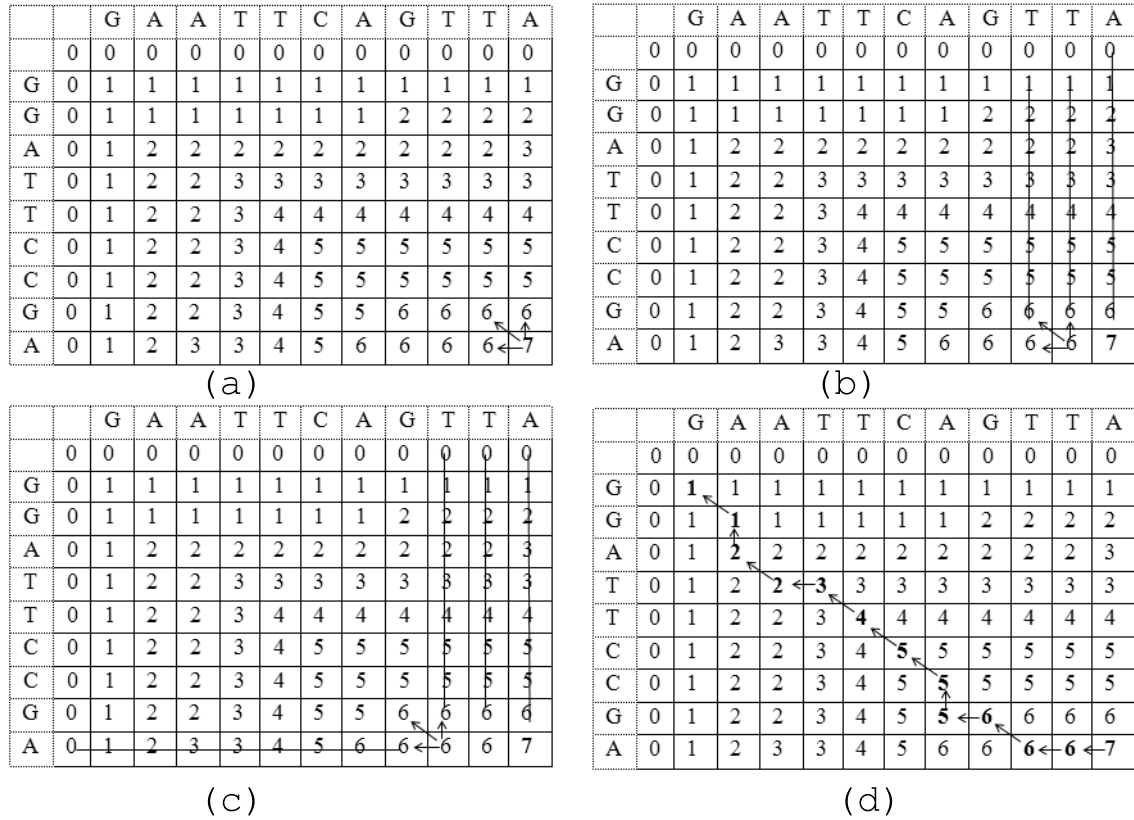


Figure 2.5. Back-tracking,((a), (b), and (c)), to obtain the optimal alignment (d)

```

G - A A T T C - A G T T A (sequence #1 )
|   |   | | |   |   |
G G A - T T C C - G - - A (sequence #2)

```

Figure 2.6. Global alignment of the two sequences



two aligning sequences that yield the optimal alignment is identified by comparing all possible segments of all lengths between the two sequences via DP technique. The main difference between this technique and Needleman-Wunsch's is that negative scores are set to zeroes. This modification produces an alignment score matrix with positive scores. Thus, the backtracking procedure of the algorithm starts at the highest positive score cell and proceeds until it encounters a cell with zero score. The longest segment pair identified in these backtracking steps is the optimal scored local alignment of the two sequences.

Given two sequences  $a$  and  $b$  of lengths  $m$  and  $n$ , respectively. The alignment matrix score  $H$  is built as follows:

$$H(i, 0) = 0, \leq i \leq m$$

$$H(0, j) = 0, \leq j \leq n$$

$$H(i, j) = \max \left\{ \begin{array}{ll} 0 & \\ H(i-1, j-1) + w(a_i, b_j) & \text{Match/Mismatch} \\ H(i-1, j) + w(a_i, -) & \text{Deletion} \\ H(i, j-1) + w(-, b_j) & \text{Insertion} \end{array} \right\}, 1 \leq i \leq m, 1 \leq j \leq n$$

Where:

$a_i$  is the  $i^{th}$  symbol in sequence  $a$

$m = |a|$  is the length of  $a$

$n = |b|$  is the length of  $b$

$H(i, j)$  is the maximum similar score between the subsequence of  $a$  of length  $i$ , and the subsequence of  $b$  of length  $j$ .

$w(c, d), c, d \in \Sigma \cup \{-'\}$ , is the matching score between two residues.

The back-tracking steps in Smith-Waterman is similar to those of Needleman-Wunsch's, however, it starts from the matrix cell with maximum score. Using the same example 2.3.2 with the scoring scheme, +1 for a match, -2 for a gap and -2 for a mismatch, we will get a local alignment of ATTC as in figure 2.3.3. It is important to note that the back-tracking steps will trace the paths that lead to the highest score; therefore, the alignment of "A" to "A" from the fifth row and fourth column (as in Figure 2.3.3) is not on one of these paths.

To align sequences that share multiple conserved regions (motifs), the Smith-Waterman's algorithm can be repeated on the un-aligned segments until all residues in the sequences are aligned. However, this technique is time consuming and does not guarantee an overall optimal alignment.

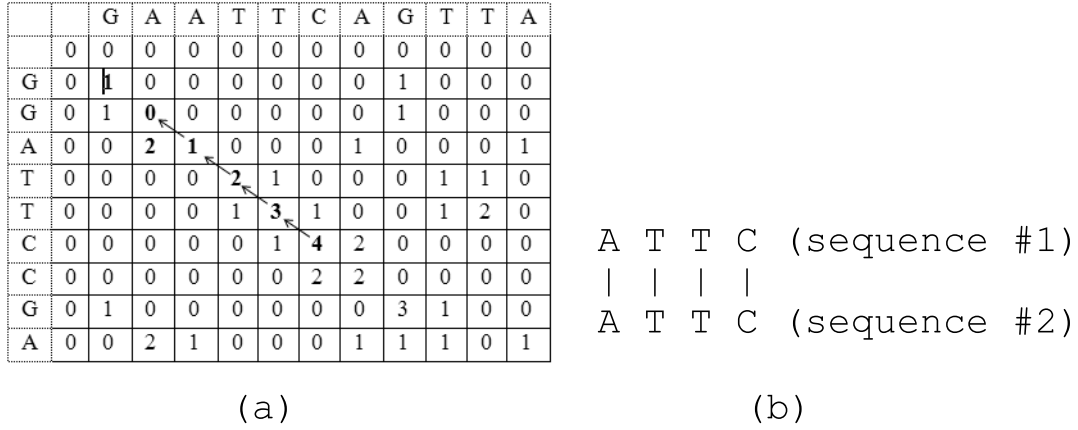


Figure 2.7. Smith-Waterman's local alignment of two sequences: "G A A T T C A G T T A " and "G G A T T C C G A"

### 2.3.4 Affine gap penalty

Statistically, the occurrence of  $d$  consecutive deletions/insertions is more likely than the occurrence of  $d$  isolated mutations. One way to enforce this concept is to penalize the opening gap, the first insertion/delete of a segment, more than consecutive insertions/deletion. This is called the affine gap penalty [41], and it is calculated as:

$$w(g) = o + (l - 1)e \quad (2.1)$$

where  $w(g)$  is the weight, or penalty of gap  $g$ ,  $o$  is opening gap cost,  $l$  is the total length of the gap, and  $e$  is the cost of each insertion/deletion. In general,  $o \leq e \leq 0$ .

## 2.4 Word Method

Word methods, also known as the  $k$  - *tuple* methods [24], are heuristic methods that find a series of short and non-overlapping subsequences ("words"). These methods do not guarantee to find the optimal alignment solution between sequences; however, they are fast and significantly more efficient than dynamic programming and dot-plot methods. Thus, they are more practical to query large databases of sequences. In general, the word methods are as follows:

First, a window of side  $k$  is chosen, normally  $k \geq 3$  for protein sequences or  $k \geq 11$  for DNA and RNA sequences. This  $k$ -window is then applied to one end of a sequence to obtain a  $k$ -length subsequence called

a word. The window is then moved to the other end, one symbol at a time, to obtain all  $k$ -length words in sequential order. The next step is to search these words from other sequences for either exact matches (as in FASTA) or highest scoring words (as in BLAST). Sequences that yield a matching score greater than a predefined threshold  $\lambda$  are considered highly homologous. The following example illustrates how to use the word method in sequence alignments.

**Example:** Given:

Sequence 1 = ACACACTA, Sequence 2 = AGCACACA and  $k = 3$

The 3-word list generated for sequence 1 is as follows: ACA, CAC, ACA, CAC, ACT, and CTA.

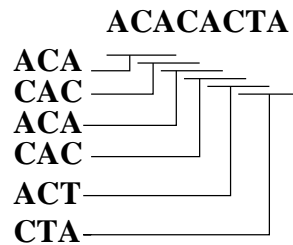


Figure 2.8. This figure shows all available words of size 3

When these words are searched against sequence 2, the first matching word is "CAC".

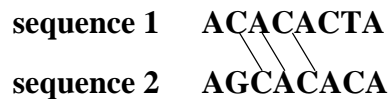


Figure 2.9. This figure shows an exact word matched

Using the exact match scoring scheme as in FASTA, "CAC" and or "ACA" are the matched words of size 3. In practice, a "hit and extend" tactic is used where each matching words can then be extended until there are no more matching symbols. Considering the "hit and extend" scheme, the first longest sharing sub-segment is "CACAC", in which "CAC" is extended with two more matching symbols "AC" (similarly, "ACACA" is extended from "ACA").

The searching time can be significantly reduced when a hashing mechanism is applied where each word and its location in a sequence is stored in a hashing table. With 20 amino acid symbols for protein sequences and 4 for RNA and DNA, each table lookup would yields  $O(1)$  order.

The exact word matching scheme may not be able to detect significant similarity such as wobble base-pairs. Wobble base pairs are fundamental in RNA secondary structure and are critical for proper translation of genetic codes. There are four main wobble base pairs: guanine-uracil, inosine-uracil, inosine-adenine, and inosine-cytosine (or G-U, I-U, I-A, and IC). And due to the codon "wobble", DNA sequences may have the form "XXyXXyXXy" where "X" is conserved and "y" is not. For instance, the following two codes "GGuUCuACgAAg" and "GGcUCcACaAAA" for the same peptide sequence (Gly-Ser-Thr-Lys) will not match any k-tuple of size 3 or longer. In this case, a high scoring matching scheme would be preferable. In this scheme, a word that yields the highest score by summing each pair of matching symbol scores is selected. Effectively, if the following scoring scheme is used:

Score = 2 for X-X, i.e. conserved nucleotide exact match.

Score = 1 for y-x or x-x, i.e. any pair of un-conserved nucleotide.

Score = 0 for X-Y, i.e. any conserved nucleotide mismatched.

Then GGuUCuACgAAg and GGcUCcACaAAA are identical.

## 2.5 Searching Sequence Databases

Most application of pair-wise alignment is not only about finding the similarity between two sequences, but rather taking a sequence and querying it against thousands of other sequences to find any sequence to be homologous. One of the early popular search algorithms is FASTA [65], which is a fast version of dot-plot method. Recently, it has been replaced by BLAST [3], a more efficient algorithm.

### 2.5.1 FASTA

Application of the dot-plot method is seen in FASTA [65] (stand for FAST-All). FASTA and BLAST are heavily used for searching databases for similar sequences. The FASTA procedure is as follows for any given an input sequence  $A, B$  and a word size  $k$ .

Step 1: Dot-plot  $A$  and  $B$  for words with size  $k$

Step 2: Re-score the diagonals using scoring matrix such as PAM matrix [23] for protein and identity matrix for nucleotide sequences. Retain  $d$  diagonals with highest scores (in the original design,  $d = 10$ ). Trimming the ends of those diagonals to include only those contribute to the highest score. These are identified core regions.

Step 3: Join the diagonals by the Needleman-Wunsch's algorithm. For protein sequences  $k = 2$  is frequently used, and for DNA sequences,  $k$  can be in the range from 1 to 6. FASTA uses a lookup table to speedup

processing time. The lookup table is similar to a hash table, where each symbols is mapped into a location in the table such as "A" is 1, "R" is 2, etc. And the values in the table represent the location in the sequence where the symbol occurs.

Figure 2.10 illustrates these steps of FASTA.

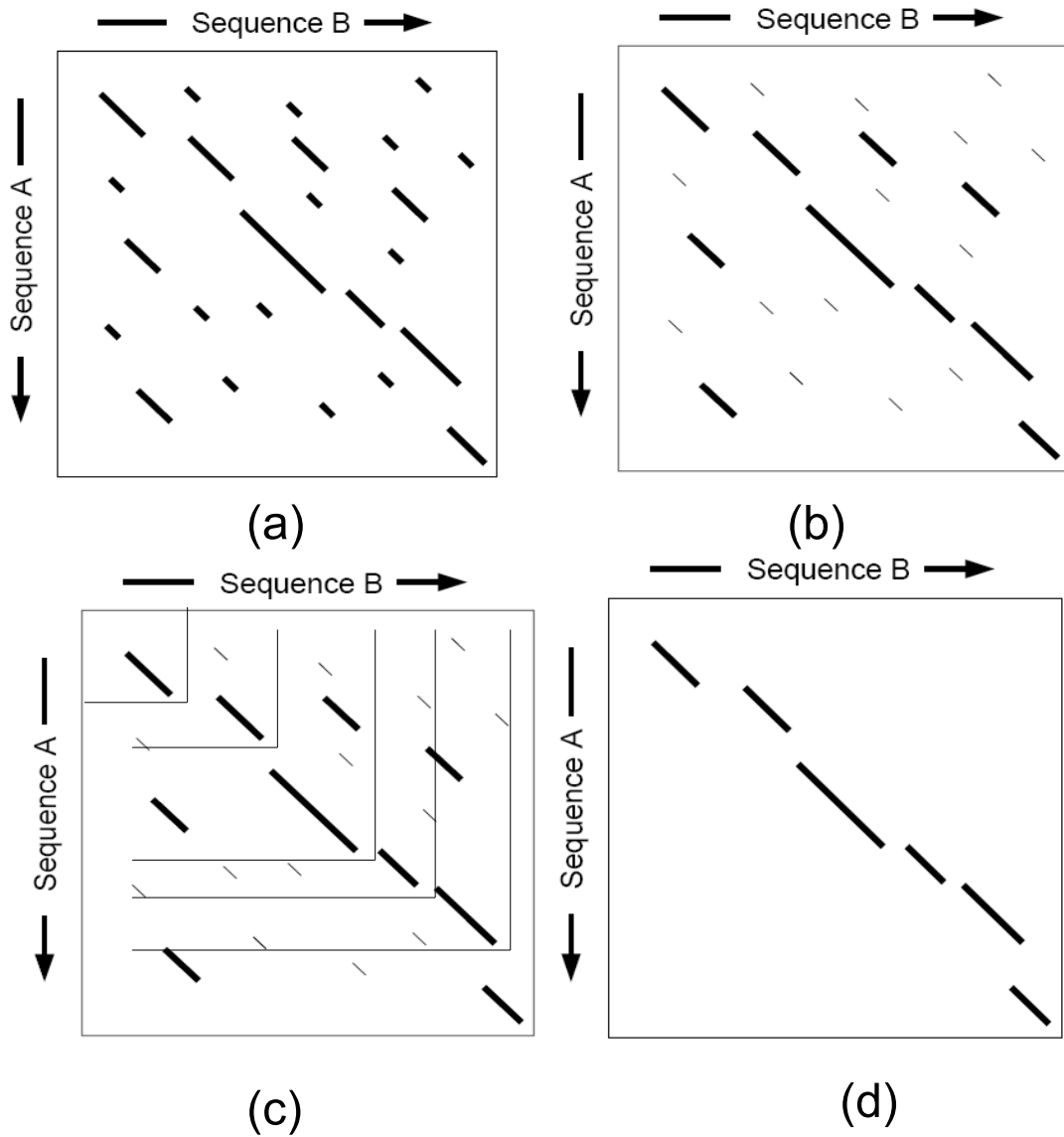


Figure 2.10. (a)-filtering dot-plot; (b)- being reevaluated by score matrix; (c) - filtering highest scored diagonals starting from the core-region - identified by an arrow; (d)- assembling the diagonals

### 2.5.2 BLAST

FASTA has been a good searching tool. As the sizes of the sequence databases increase, the need for better and faster algorithm arises leading to the development of BLAST [3]. BLAST is a heuristic search method which applies the word method to search a database for sequences that are homologous to the query sequence. It finds matching segments between two sequences (or a sequence and the database) with a score greater than or equal to  $S$ , a predetermined cut off score. BLAST handles this task by searching for sequences that have maximal segment pair (MSP) scores greater than a threshold  $T$ . BLAST defines the maximal segment pair to be the highest scoring pair of identical length segments chosen from 2 sequences. The MSP is local maximal such that its score cannot be improved by either extending or shortening both segments. Given a query sequence and a word size  $w$ , BLAST generates a list of overlap words of size  $w$ . Typically,  $w = 3$  for protein sequences and  $4 \leq w \leq 11$  for DNA sequence. A target sequence is scanned for these words with a score greater than or equal to threshold  $T$ . If there is a hit, it is extended to determine whether a segment pair whose score of at least  $S$  can be found. If such segment is found, it is extended until the total score of the segment begins to decrease. This segment pair is called a high scoring segment pair (HSP). Thus, the MSP is the highest score HSP. A low threshold  $T$  results in more hits and slows down the search. When aligning two random sequences of length  $m$  and  $n$ , ( $n$  in this case the total length of all sequences in the database) the probability of finding a segment pair with a score greater than or equal to  $S$  is defined to be:

$$E = 1 - e^{-y} \quad (2.2)$$

where  $y = K m n e^{-\lambda S}$ , and  $K$  and  $\lambda$  are statistical parameters represent the natural scales of for the search space and the scoring system.  $K$  and  $\lambda$  depend upon substitution matrix, gap penalties, the frequencies of the symbols in the sequence. Typically,  $\lambda = 0.3184$  and  $K = 0.13$ .

This threshold is often referred as an E-value, and a typical good E-value should be  $10^{-5}$  or lower. Smaller E-value indicates that the MSP is highly unique and not due to errors.

The probability of finding  $c$  or more distinct segment pairs with scores at least  $S$  is:

$$P = 1 - e^{-y} \sum_{i=0}^{c-1} \frac{y^i}{i!} \quad (2.3)$$

Hence, this probability and the E-value determine the degree of confidence on a BLAST result.

BLAST cannot not guarantee the MSP to be the optimal local alignment; however, the trade off is the greatly improvement of the algorithm speed that needed to search large databases of sequences.

In latter chapters we will discuss the suffix structures and construction techniques that allow a large database of sequences to be stored and retrieved quickly to accommodate sequence searches.

## Chapter 3

### QUANTIFYING SEQUENCE ALIGNMENTS

The concept of measuring the evolution is to calculate the number of evolutions or the number of derivation levels of species from their original ancestors. Hypothetically, to do this, the entire hierarchy of evolution and the species involved must be identified and correctly organized. In the real world, a complete evolution information is not available, and nobody knows for sure when, how, why, and what species really involved in the evolution. Thus, the task of measuring evolution is often derived from the genetic and genomic information that are obtained from certain species, which often comes in the form of sequences. In this chapter, we will study existing methods that have been used to measure species distances and similarities.

#### 3.1 Evolution and Measuring Evolution

In term of sequence evolution, whether the sequence is DNA, RNA, or Protein, the measurement often relies on the sequence primary information. In general, the measurement is focused on four features: sequence similarity, homology, divergence, and convergence. These features involve the sequences, their structures and functionalities.

The simplest of the four is measuring the similarity between sequences. For any given set of sequences, a statistical analysis of sequenced similarity can be derived from the number of common nucleotides in the sequences. However, this measurement has limited use for the randomness involved. For example, finding a highly similarity between a non-functional sequence segment against a functional motif of a sequence may not provide any practical use.

Observing divergence and convergence from groups of nucleotides, segments of sequences, or sequences themselves often lead to discovery of important information such as key elements of a sequence, structures or functions. Therefore, identifying and measuring divergence and convergence between sequences are fundamentals in sequence analysis. Commonly, divergence and convergence often derived on the probability a nucleotide, or amino acid residue, diverges or converges into another.

Lastly, homology measurement is a technique to identify the distance between the sequences and one of their ancestors. And this is the ultimate goal of measuring the evolution. Most often, homology is measured based on available biological information and information derived from the other three measuring



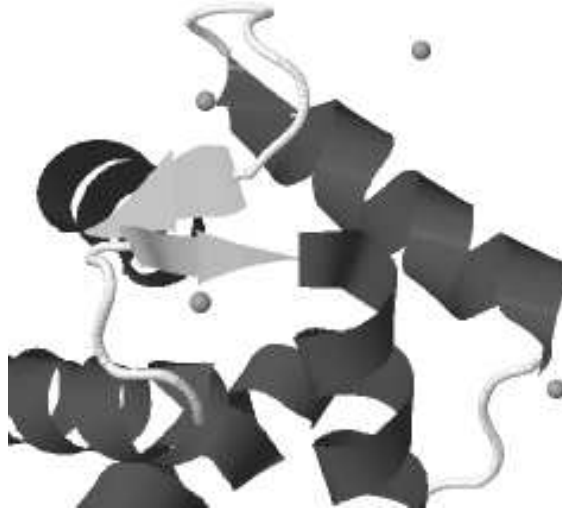


Figure 3.1. A Human Calmodulin wraps around its binding domain in the plasma membrane.  $\text{Ca}^{2+}$  atoms are depicted as circles.

techniques. For example, Calmodulin (CaM: CALcium MODULated proteIN), a calcium-binding protein expressed in all eukaryotic cells to mediate inflammation, metabolism, short-term and long-term memory, nerve growth and immune response, etc, is well known to have approximately 148 amino acids long with four EF-hand motifs, each of which binds a  $\text{Ca}^{2+}$ . When comparing Calmodulin against other sequences that results in identical, or highly similar, segments matching the four EF-hand motifs would give a better confidence that these sequences are homologous. Figure 3.1 depicts of a human CaM 3D structure showing the Calmodulin wrapping its binding domain in the plasma membrane.

Next, we will look into how the evolution can be measured.

### 3.1.1 Jukes and Cantor's Model

In 1966 Jukes and Cantor [55] developed a method modeling the probability of a nucleotide, or a polypeptide, being mutated to, or substituted by, another. This method assumes the rate of substitution or evolution is the same for all four nucleotides (similar for amino acids) as in Figure 3.2, i.e., unrelated DNA/RNA sequences should be 25% identical by chance. The main goal is to find the actual number of mutations that really occur from an ancestor sequence leading to two sequences of fixed length  $n$  with  $k$  different nucleotides, or  $k$  substitutions. In other words, it is to find the number of steps that make these two sequences differed, or the distance between them. For example, observing the following two sequences:  $a$  and  $b$

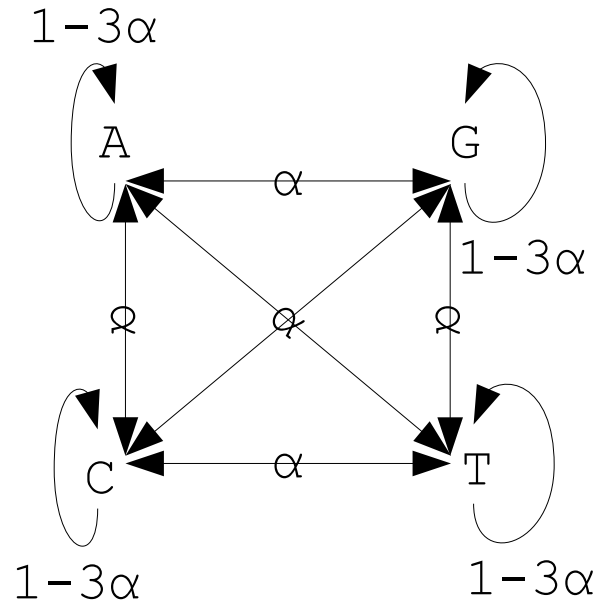


Figure 3.2. This figure shows the possible paths a nucleotide can be mutated to with the same probability  $\alpha$  in Jukes and Cantor's model

$a:$  A A C C A C A C A A C C A C T A A A G A A C C T A C A  
 $b:$  A A G T A C A G T A C T G C T G A T G G A G C A A C T  
 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 C 1 0 0 1

In this representation, 1s represent an incident of mutation. There are 12 substitutions between the two sequences of length 27, which is about 44 percent, (12 is also the distance, or edit distance, between these two sequences). Assuming each nucleotide is equally likely to change to another nucleotide, it is almost certain that the number of mutations is more than 12 since C could change to T then change to G, etc. Jukes and Cantor method estimates the total actual number of mutations is about 18 ( $t = 66$  percent), not 44 percent.

The Jukes and Cantor equation is defined as:

$$d = -\frac{3}{4} \ln\left(1 - \frac{4}{3}p\right), \quad (3.1)$$

where  $t$  is the distance between two sequences, and  $p$  is the fractional dissimilarity defined as the ratio of number of mismatched positions over the length of the sequences.

The terms substitution and mutation referenced in this chapter are used interchangeably, and they both convey the same meaning of a residue symbol being replaced by another symbol.

### 3.1.2 Measuring Relatedness

The most basic task to identify the evolution process is to calculate the evolution distances between the involved species. Initially, species that yield the most confidence in being homologous are grouped together. These groups are then being considered as new taxonomic units, and the grouping process is repeated until all taxonomic units are joined together. The most fundamental task in this process is measuring the evolution distance, or scoring the homogeneousness between species. In terms of sequence analysis, the scoring is often based on the similarity, divergence, and convergence between the sequences. At the sequence residue level, either nucleotides or amino acids, there are available biological and scientific information permitting scientists to rank any pair of matching residues. However, functional units and structures of the sequences are often made of more than one residue and may depend on each other, thus making the ranking very difficult since the units and the dependency between them are being determined. The most popular approach to solve this problem is to accumulate the residue similarity scores, divergent scores and convergent scores, and then normalize these scores based on some weighing schemes. Therefore, the weighing scheme should be refined by experts for each instance of measurement to fit the data and their expected or desired results. Any revealed information obtained from sequence alignment should be incorporated back to further refine the alignment.

## 3.2 Substitution Matrices and Scoring Matrices

Quantifying the distance between sequences often relies on the probability of one residue being mutated to another. Proteins have 20 different amino acids and DNA/RNA has four different nucleotides. For convenience, these mutation/substitution probabilities are often pre-calculated for general uses. The values of these matrices are different and depend on which method is being employed.

### 3.2.1 Identity Scores

This method is probably the simplest scoring scheme. Protein polypeptides and DNA/ RNA nucleotides are classified into two types: identical and non-identical or matched and unmatched. The matched pairs are given a positive score (usually 1) and the unmatched pairs get 0 score.

In case of DNA/RNA, a matched gets a score of 1 and a mismatched gets a score of -1,000 (or any negative penalty score) as in the following Table 3.1

Table 3.1. DNA/RNA scoring matrix used by NCBI

	A	T	G	C
A	1			
T	-1000	1		
G	-1000	-1000	1	
C	-1000	-1000	-1000	1

### 3.2.2 Substitution/Mutation Scores

Substitution scoring scheme is based on the observed residue substitution, or mutation, frequencies seen in sequence alignments. Despite the contradicting fact that alignments must exist before the frequencies can be observed and profiled, the frequencies can be used to align the sequences with relatively reliable results; especially, when the frequencies are obtained from carefully aligning all of the sequences in several families and then construct phylogenetic trees for each family. Each phylogenetic tree is examined to identify the substitution occurred on each branch. A mutation or substitution is more likely to occur between residues with similar biochemical properties. For instance, Isoleucine(I) and Valine(V) get a positive score by this scheme to indicate a likelihood that they would easily mutate to each other. While the hydrophobic Isoleucine(I) will get a negative score with hydrophilic Cystine(C) as their probability of being substituted is very unlikely. A table combining all of these scores is often called a substitution matrix or a scoring matrix. One of the early matrix is Dayhoff's [23, 45] matrix or Point (Percentage) Accepted Mutation (PAM) matrix, developed for protein sequences. The substitution matrix score for all amino acids are generated from observing the likelihood of a particular mutation such as A to D with low successive mutation, i.e., A to x to y to D, where x and y are some other residues over time. The scoring matrix  $M$  contains the probabilities of a residue being mutated to another.  $M_{i,j}$  is defined to be the probability of an amino acid in row  $i$  mutating to an amino acid in column  $j$  after a particular evolution period. For example, matrix 2 PAM represents the two-percentage of acceptable point mutations per  $10^8$  years. The mutation matrix can be generated for any specific evolution period. Thus, PAM is based on mutations that occurred in an overall alignment of sequences, and its sensitivity is based on the distances between sequence pairs. In general, PAM250 - targeting sequences with 250 PAM apart or 250 mutations per 100 amino acids of sequence - is considered a good matrix to use in protein sequence databases searches. The log-odds [20, 31, 75] are applied to the actual values of PAM to normalize it so that PAM1 matrix has one point of mutation per a hundred amino acids. The odds ratio is an approach to determine whether the

probability of a certain event is the same for two groups. The odds ratio of 1 indicates that the event is equally likely in both groups. The log-odds is the natural logarithmic of odds ratio. For independent event  $X$  and  $Y$  as follows:

	Y=1	Y=0
X=1	$n_{11}$	$n_{10}$
X=0	$n_{01}$	$n_{00}$

The log-odds formula for two independent events  $X$  and  $Y$  is:

$$L = \log\left(\frac{p_{11}p_{00}}{p_{10}p_{01}}\right) = \log\left(\frac{n_{11}n_{00}}{n_{10}n_{01}}\right) \quad (3.2)$$

where  $p_{ij} = n_{ij}/n$ , with  $n = n_{11} + n_{10} + n_{01} + n_{00}$

The likelihood of a protein substituting amino acid  $a$  for  $b$  is defined as a ratio of two properties:

$$R(a, b) = \frac{P(a, b|match)}{P(a, b|random)}, \quad (3.3)$$

where  $P(a, b|match)$  is the probability of  $a$  being substituted for  $b$  with the assumption that their positions are biologically equivalent, and  $P(a, b|random)$  is the probability of  $a$  and  $b$  aligned randomly as observed. These ratios for all amino acid symbols are expressed as log values and scaled up to the nearest integers. The formula for generating the value in the substitution matrix is:

$$M(a, b) = \lceil \lambda \log R(a, b) \rceil, \quad (3.4)$$

where  $\lambda$  is a scaling factor. Table 3.2 shows the PAM250 matrix.

Similarly, Blocks Substitution Matrix (BLOSUM) [2, 29] is also designed for scoring alignments between divergent protein sequences used in database search developed along with BLAST. BLOSUM was first mentioned in [23]. Instead of considering each residue independently, BLOSUM considers the significance of local alignment blocks, or highly conserved regions, and their content residues. The exact method is to scan the database for conserved blocks of each protein family and count the relative frequencies of amino acids and their substitutions. The log-odds score is calculated for each possible substitution of the 20 standard protein amino acids. Each BLOSUM matrix is calculated from grouping sequences with  $x$  percentage of similarity from a database. For instance, BLOSUM80 represent the substitution matrix generated from sequences that are more than 80% identical to the query sequence. The following formula is used to generate the BLOSUM score:

Table 3.2. 250 PAM substitution matrix

C	12																		
G	-3	5																	
P	-3	-1	6																
S	0	1	1	1															
A	-2	1	1	1	2														
T	-2	0	0	1	1	3													
D	-5	1	-1	0	0	0	4												
E	-5	0	-1	0	0	0	3	4											
N	-4	0	-1	1	0	0	2	1	2										
Q	-5	-1	0	-1	0	-1	2	2	1	4									
H	-3	-2	0	-1	-1	-1	1	1	2	3	6								
K	-5	-2	-1	0	-1	0	0	0	1	1	0	5							
R	-4	-3	0	0	-2	-1	-1	-1	0	1	2	3	6						
V	-2	-1	-1	-1	0	0	-2	-2	-2	-2	-2	-2	-2	4					
M	-5	-3	-2	-2	-1	-1	-3	-2	0	-1	-2	0	0	2	6				
I	-2	-3	-2	-1	-1	0	-2	-2	-2	-2	-2	-2	-2	4	2	5			
L	-6	-4	-3	-3	-2	-2	-4	-3	-3	-2	-2	-3	-3	2	4	2	6		
F	-4	-5	-5	-3	-4	-3	-6	-5	-4	-5	-2	-5	-4	-1	0	1	2	9	
Y	0	-5	-5	-3	-3	-3	-4	-4	-2	-4	0	-4	-5	-2	-2	-1	-1	7	10
W	-8	-7	-6	-2	-6	-5	-7	-7	-4	-5	-3	-3	2	-6	-4	-5	-2	0	0
	C	G	P	S	A	T	D	E	N	Q	H	K	R	V	M	I	L	F	Y
																			W

$$M_{ij} = \frac{1}{\lambda} \log\left(\frac{p_{ij}}{q_i * q_j}\right), \quad (3.5)$$

where  $p_{ij}$  is the the probability the two amino acids  $i$  and  $j$  replacing each other in a homologous sequence.  $q_i$  and  $q_j$  are the background probabilities of randomly finding the  $i$  and  $j$  in a sequence.  $\lambda$  is a scaling factor that converts the scores to integers for the ease of calculation.

There is an equivalent relationship between PAM and BLOSUM. For example, PAM100 is roughly equivalent to BLOSUM90, so as PAM120 to BLOSUM80, PAM160 to BLOSUM60, PAM200 to BLOSUM52 and PAM250 to BLOSUM45. Table 3.3 represent the BLOSUM45 derived from sequence blocks clustered at the 45% identity level.

GONNET matrix [36] is another matrix developed by Gonnet, Cohen and Benner in 1992 using exhaustive protein pair-wise Needle-Wunsch's alignment algorithm with PAM matrix on entire existing protein sequence database at the time. GONNET matrix is shown in table 3.4.

Table 3.3. BLOSUM 45 substitution matrix

G	7																		
P	-2	9																	
D	-1	-1	7																
E	-2	0	2	6															
N	0	-2	2	0	6														
H	-2	-2	0	0	1	10													
Q	-2	-1	0	2	0	1	6												
K	-2	-1	0	1	0	-1	1	5											
R	-2	-2	-1	0	0	0	1	3	7										
S	0	-1	0	0	1	-1	0	-1	-1	4									
T	-2	-1	-1	-1	0	-2	-1	-1	-1	2	5								
A	0	-1	-2	-1	-1	-2	-1	-1	-2	1	0	5							
M	-2	-2	-3	-2	-2	0	0	-1	-1	-2	-1	-1	6						
V	-3	-3	-3	-3	-3	-3	-3	-2	-2	-1	0	0	1	5					
I	-4	-2	-4	-3	-2	-3	-2	-3	-3	-2	-1	-1	2	3	5				
L	-3	-3	-3	-2	-3	-2	-2	-3	-2	-3	-1	-1	2	1	2	5			
F	-3	-3	-4	-3	-2	-2	-4	-3	-2	-2	-1	-2	0	0	0	1	8		
Y	-3	-3	-2	-2	-2	2	-1	-1	-1	-2	-1	-2	0	-1	0	0	3	8	
W	-2	-3	-4	-3	-4	-3	-2	-2	-2	-4	-3	-2	-2	-3	-2	-2	1	3	15
C	-3	-4	-3	-3	-2	-3	-3	-3	-3	-1	-1	-1	-2	-1	-3	-2	-2	-3	-5
	G	P	D	E	N	H	Q	K	R	S	T	A	M	V	I	L	F	Y	W

Table 3.4. GONNET matrix generated from SWISS-PROT database

C	12																			
S	0	2																		
T	0	2	2																	
P	-3	0	0	8																
A	0	1	1	0	2															
G	-2	0	-1	-2	0	7														
N	-2	1	0	-1	0	0	4													
D	-3	0	0	-1	0	0	2	5												
E	-3	0	0	0	0	-1	1	3	4											
Q	-2	0	0	0	0	-1	1	1	2	3										
H	-1	0	0	-1	-1	-1	1	0	0	1	6									
R	-2	0	0	-1	-1	-1	0	0	0	2	1	5								
K	-3	0	0	-1	0	-1	1	0	1	2	1	3	3	-1						
M	-1	-1	-1	-2	-1	-4	-2	-3	-2	-1	-1	-2	-1	4						
I	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-2	-2	-2	2	4					
L	-2	-2	-1	-2	-1	-4	-3	-4	-3	-2	-2	-2	-2	3	3	4				
V	0	-1	0	-2	0	-3	-2	-3	-2	-2	-2	-2	-2	2	3	2	3			
F	-1	-3	-2	-4	-2	-5	-3	-4	-4	-3	0	-3	-3	2	1	2	0	7		
Y	0	-2	-2	-3	-2	-4	-1	-3	-3	-2	2	-2	-2	0	-1	0	-1	5	8	
W	-1	-3	-4	-5	-4	-4	-4	-5	-4	-3	-1	-2	-4	-1	-2	-1	-3	4	4	14
X	-3	0	0	-1	0	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-4
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

### 3.3 Gaps

Gaps are artificial insertions into sequences to move similar segments of aligning sequences into alignment. It is widely believed that through evolution, segments of a sequence are missing due to several factors, either biologically, physically or chemically, which leads to homologous sequences with different lengths. The missing segments are considered deleted from the sequence, and the sequences retaining these segments are considered inserted with such segments. Thus, to correctly align homologous sequences the missing segments must be accounted for. This is why gaps are inserted into the sequences to get the similar remaining regions aligned. The probability of a segment that is missing from a sequence is rarely studied due to the fact that the exact and true evolution progression including the corresponding protein/DNA/RNA sequences of species involved are unknown. Thus, it is improbable to determine whether segments of a sequence are missing from the original one, are inserted by combining with other sequences of other species or are mutated under some conditions. Sometimes, it is not clear whether the sequences being aligned are homologous or not. The alignment of the sequences may lead to a conclusion.

These problems revolve in a circle, and MSA is an attempt to break that circle. Thus, inserting the right number of gaps into the appropriate locations in the sequences will give the solution we are seeking. Correctly placing the gaps into the sequences is a very difficult problem since there are tremendous possible insertions, or deletions to be considered, and true model often is not available or exist for validating against the resulting alignments. In 1980, Lesk and Chothia [62] showed that deletes/insertions are observed primarily in unstructured coils or surface regions between secondary structure units of homologous protein sequences (i.e.,  $\alpha$ -helices and  $\beta$ -strands) rather than within the units themselves.

We want to at least align similar motifs of the sequences together; thus gap insertions are not avoidable. Nevertheless, unlimited gap insertions are biologically meaningless. As a result, the majority of MSA algorithms penalize gap insertions to avoid unnecessary extension of the alignment. Thus, the overall objective is to maximize the matches in the alignment and minimize inserted gaps.

The gap penalty is experimental value, depending on each specific MSA algorithm, and often left to biologists to decide. Since gaps are artificial, quantifying a match between a residue and a gap is biologically meaningless. Thus, quantifying methods that focus on the available information in the aligning sequences such as known motifs and residue location and mutation constraints are preferable.



### 3.3.1 Sequence distances

All of the matrices mentioned above serve only one purpose, which is how to measure a match between two residue symbols. Almost all sequence analysis depends on some value that represents the distance or the relatedness between two or more sequences, not the residues themselves. In terms of sequence alignment, a pair of sequences will have the same lengths after being aligned. Since a pair of aligned sequence is a pair of two strings of symbols, there are two main approaches to measure the sequence distance. A simple and natural approach is to sum up all scores of each pair. If the score is the identity score, i.e., 1 for matched and 0 for unmatched, then this is the edit distance, or Hamming distance. Edit distance is the number of positions at which the corresponding symbols are different, i.e., the number of substitutions must be done to transform one sequence into the other. The main drawback of this simple method is the biological significance of the sequences is simply ignored which could lead to errors in sequence classification. If a substitution matrix or mutation matrix is being used, the score then represents the compound of biological quantitative measurements of all the residue pairs in the alignment. These sum scores are often called the sum-of-pair score(SP) [15]. Mathematically, sum-of-pair score is define as:

**Definition 1** *The pair-wise alignment score of two induced sequences  $s'_i$  and  $s'_j$ , where  $s'_i, s'_j \in A$  is:*

$$S(A) = S(s'_i, s'_j) = \sum_{k=1}^{|A|} S(s'_i[k], s'_j[k])$$

Where  $A$  is the alignment of sequences  $S_i$  and  $s_j$ ,  $|A|$  is the total length of the alignment,  $S(s'_i[k], s'_j[k])$  is a matching pair-wise score obtained from scoring matrix, and  $s'_i$  and  $s'_j$  are the residue symbols in the aligned sequences  $s_i$  and  $s_j$  (i.e. the sequences with gap (-) inserted), respectively.

### 3.3.2 Example

Given the following sequences to be aligned:

S1:MTEITAAM and

S2: MILIAAM.

After being aligned, a gap is inserted into the second sequence between I and A. The edit distance scoring scheme gives a score of 3, and the scheme uses BLOSUM62 matrix give a score of 8 (assuming the gap-mismatch is -9). Figure 3.3 illustrates these scores.

All of these scoring schemes are only applicable to a pair of residues. In the next Section we will investigate other scoring schemes for groups of more than two residues.

(a)		(b)	
S1:	MTEITAAM	S1:	M T E I T A A M
S2:	MILI-AAM	S2:	M I L I - A A M
	01101000 = 3		5-2-3 4-9 4 4 5 = 8

Figure 3.3. (a) shows the score by edit distance, and (b) shows the score by BLOSUM62 matrix

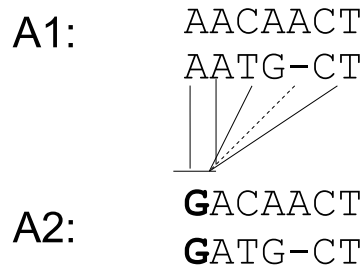


Figure 3.4. This figure illustrate the steps in assembling an MSA

### 3.4 Scoring Multiple Sequence Alignments

Aligning and scoring two sequences are quite primitive with these available substitution matrices and pair-wise alignment algorithms. However, when more than two sequences involved in an alignment, it opens up an entire world of ambiguity. There is no concrete biological information to help us identifying the probability of  $k$  residues mutated to each other at any single sequence site. Without these measurements, it is hard to determine whether an indel (insert/delete) would improve or worsen the alignment of the sequences. And without such measurement, it is hard to determine which alignment of the same set of sequences is the best. Figure 3.4 illustrates this idea, where two preassembled alignments are being aligned together. Each column contains more than one residue symbol, and when matching one column against another it requires a quantitative method to indicate which combination is the best.

One of the simplest scoring methods is the column score, which is similar to the identity score. This method rewards a score of 1 for a column with the same residue symbols and 0 for containing any divergent symbols. This method does not have any practical use except for detecting conserved columns in an alignment. In this Section, we will explore available methods that have been developed to quantify multiple sequence alignment.

### 3.4.1 Sum-of-Pair Score

One of the most popular quantitative measurement for multiple sequence alignment (MSA) is the sum of Sum-of-Pair score, or simply sum-of-pair(SP). It comes with many variations as in [5, 15, 56]. To calculate the SP score of  $n$  sequences, the scores of all  $\binom{n}{2}$  pair-wise alignments within the MSA are added. This sum of pair method is the extension of sum-of-pair method mentioned in scoring pair-wise alignment to cover all pair-wise scores. Formally, the SP score is defined as:

**Definition 2** *The multiple alignment score of  $n$  induced sequences  $s'_1, s'_2, \dots, s'_n$ , where  $s'_i, s'_j \in A$  is:*  

$$S(A) = S(s'_1, s'_2, \dots, s'_n) = \sum_{k=1}^{|A|} \sum_{i,j,j \neq i}^n S(s'_i[k], s'_j[k])$$

The following two examples will illustrate the SP method in details.

**Example 1** Given the following sequences

S1: MTEITA

S2: MILIA

S3: TITIA

and their alignment A:

$$A(S1, S2, S3) = \begin{cases} S1: & M & T & E & I & T & A \\ S2: & M & I & L & I & - & A \\ S3: & T & I & T & I & - & A \end{cases}$$

The SP score of this alignment is the sum of:

$$S(A) = [S(M,M) + S(M,T) + S(M,T)] + [S(T,I) + S(T,I) + S(I,I)] + [S(E,L) + S(E,T) + S(L,T)] + [S(I,I) + S(I,I) + S(I,I)] + [S(T,-) + S(T,-) + S(-,-)] + [S(A,A) + S(A,A) + S(A,A)]$$

The biggest problem with this scoring method is how to score a pair of matching between two gaps. In practice, it is simply ignored.

**Example 2** Aligning a segment of sequence  $s_i = \dots MI \dots$  with pre-aligned sequences  $A = \{s_j = \dots LLM \dots$  and  $s_k = \dots IMI \dots\}$  yields more than one alignments. If BLOSUM62 substitution matrix and a gap penalty of -1 are used to score the alignments, alignment of

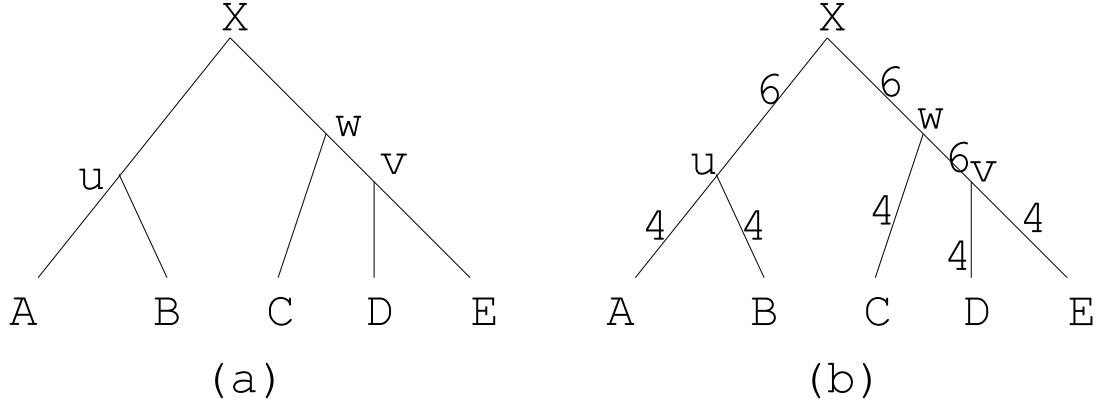


Figure 3.5. Traversal of a tree by SP measurement. The total visits on each edge is shown. Some edges are visited more often than others

$$A(s_i, s_j, s_k) = \begin{cases} s'_j : & L & L & L \\ s'_k : & I & M & I \\ s'_i : & M & - & I \end{cases} \text{ yields a score of -5, which is higher than the alignment of}$$

$$A(s_i, s_j, s_k) = \begin{cases} s'_j : & L & L & L \\ s'_k : & I & M & I \\ s'_i : & M & I & - \end{cases} \text{ yielding a score of -8.}$$

If, instead, the gap penalty score is chosen to be -10, both of these alignments yield the same score.

From the evolution perspective, the SP method is deficient. Consider the following Figure 3.4.1, the left tree represents the evolution tree of the sequences (the leaves), and the right tree represents the SP accounting method for scoring the MSA. The score of a pair-wise alignment  $(A, B)$  evaluates the likelihood evolutionary events on edges  $(A, u)$  and  $(u, B)$  of the tree. Similarly, the score of a pair-wise alignment  $(C, D)$  evaluates the probability of evolution events on edges  $(C, w)$ ,  $(w, v)$  and  $(v, D)$  of the tree. A tick is marked on an edge each time it being accounted while calculating the SP score.

In addition, the SP scoring method calculates all-pair scores of residues in a column at every step of alignment, which increases the runtime of an MSA algorithm by  $O(n^2)$  for aligning  $n$  sequences of length  $n$ . Nevertheless, sum-of-pair is the most popular used scoring function. When the mismatched score and gap penalty are negative values, the SP method seems to work well. The reason is the over-rewarded mismatched scores become larger penalties.

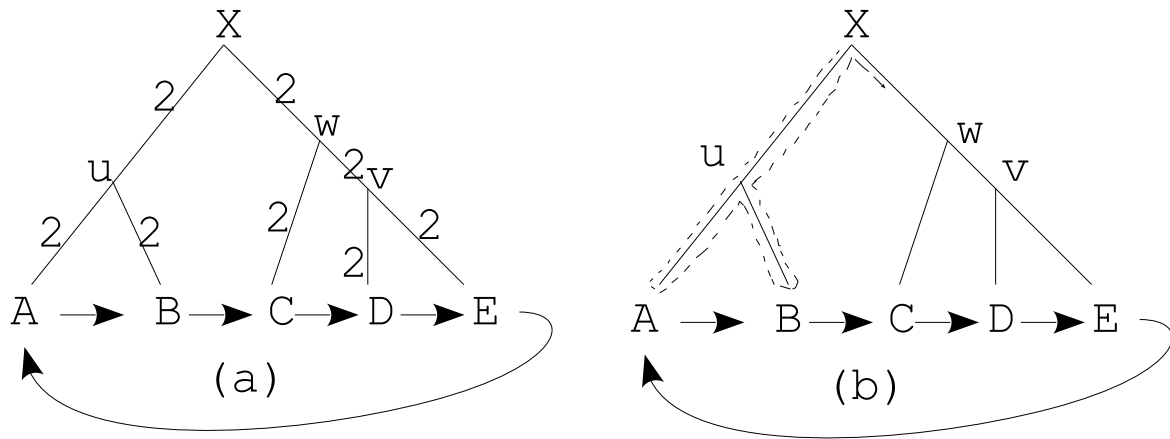


Figure 3.6. Circular traversal of a tree. The dotted arrow indicates the traversal path

### 3.5 Circular Sum Score

Circular Sum (CS) score [39] is an alternative measurement that corrects the redundancy in SP and is based on a solution to the Traveling Salesman Problem (TSP). To calculate the CS score, a circular tour through an evolutionary tree representing the sequences in the MSA must be identified. Given a set  $S$  of  $n$  sequences,  $S = s_1, \dots, s_n$ , the CS score for its alignment is the cost of one half of all the edges in the evolution tree  $T(S)$  being visited during a tour from  $s_1$  to  $s_n$  and back to  $s_1$ . The circular tour visits each leaf once and each edge exactly twice. For example, the circular tour for the tree given in Section 3.4.1, starting from A, is A to B, from B to C, from C to D, from D to E and then back to A.

Similar to SP method, the edges between two residues can be derived from any scoring matrix or substitution matrix.

The CS score gives an upper-bound, the best score that a given set of sequences can be aligned.

One of the drawback of this method is both the evolution tree construction and the TSP are NP-complete. Thus, CS scoring method has limited practical use.

### 3.6 Conservation Score Schemes

When multiple residues are grouped together, scoring them are no longer trivial. Each residue has its own stereo-chemical property and biological meaning. Thus, most scoring schemes rely on the conservation and diversity of the symbols instead.

### 3.6.1 Wu and Kabat's Method

In 1970 Wu and Kabat [126] proposed a method to measure sequence conservation. The method is as follow:

$$S = \frac{k}{nl} \times N \quad (3.6)$$

where  $k$  is the number of amino acid types in a column,  $nl$  is the number of times the most abundant amino acid symbol occurred in the column, and  $N$  is the number of sequences involved during that scoring process. For example, the score for third column of the following alignment:

$$A(s_i, s_j, s_k) = \begin{cases} s'_j: & L & L & L \\ s'_k: & I & M & I \\ s'_i: & M & - & I \end{cases} \quad \text{where } k = 3 \text{ for 3 symbol types, } nl = 2 \text{ since I occurs twice in that column and } N = 3 \text{ as three sequences involved. Thus, the score is: } \frac{3}{2} \times 3 = 4.5$$

This method simply ignores the probability that some other non-conserved symbols could be mutated to the majority symbols. And in the extreme case where there are  $n$  symbols of two types, one type has  $\frac{n}{2} + 1$  residues and the other has  $\frac{n}{2} - 1$  symbols this method will simply ignores the other conserved half.

### 3.6.2 Jores's Method

In 1990, Jores et. al. [54] proposed a different scoring scheme to address the the problem in Wu and Kabat's method. Their scoring method is defined as:

$$S = \frac{k_{pairs}}{n_{pairs}} \times \frac{1}{2}N(N-1) \quad (3.7)$$

where  $\frac{1}{2}N(N-1)$  is the number of all possible pairs of amino acids in a column,  $k_{pairs}$  is the number of distinct pairs and  $n_{pairs}$  is the number of times the most frequent pairs occurs. Using same example in 3.6.1 and scoring the third column, we will have:

$k_{pairs} = 2$  for two distinct pairs (L,I) and (I,I),  $n_{pairs} = 2$  for (L,I) occurs twice. Thus, the score is:

$$S = \frac{2}{2} \times \frac{1}{2}3(3-1) = 3$$

This method carries the same critics as the sum-of-pair score where different symbol pairs get more counts than the conserved ones. Both Wu and Jores method does not take gap penalty into account.

### 3.6.3 Lockless and Ranganathan's Method

Lockless and Ranganathan [69] propose a different scoring method where they measure the conservation of an amino acid in a position extends to the whole alignment. If a symbol  $a$  occurs in the sequence database with a frequency  $q_a$ , then the probability of  $a$  occurring  $n_a$  times in a column of  $N$  residues is  $P(X = n_a)$  where  $X \sim \text{Bin}(N, q_a)$ , where  $\text{Bin}(N, q_a)$  is the binomial probability of  $a$ . For example, if one half of the residues in SWISS-PROT [11] database were all As, then the probability of A occurring eight times in a column with ten residues is the same as the probability of tossing a coin ten times and get eight heads. The function to calculate the difference between the frequency of  $a$  at a specific column and  $a$  across the alignment is defined as:

$$d(n_a, \bar{n}_a) = \ln\left(\frac{P(X = n_a)}{P(X = \bar{n}_a)}\right), \quad (3.8)$$

where  $\bar{n}_a$  is the average frequency of symbol  $a$  in the entire alignment. This gives  $d = 0$  when the frequencies of  $a$  in all columns are the same.

The conservation score is defined to be the root-mean square derivation overall protein amino acids, such as:

$$S = \sqrt{\sum_a d(n_a, \bar{n}_a)^2}. \quad (3.9)$$

Due to its complexity, this method is rarely used in sequence alignment.

## 3.7 Diversity Scoring Schemes

The information theoretic entropy proposed by Shannon in 1948 [102] is frequently employed to measure diversity in alignment columns. In this section we will investigate scoring schemes based on this entropy. First, we will start with the entropy background.

### 3.7.1 background

The Shannon entropy originates from combinatorics and information theoretics. The combinatoric derivation can be seen as follows: Given 10 colored balls of 5 reds, 2 greens and 3 yellows. The total distinct arrangements is  $10!/(5!2!3!) = 2520$ . In general, with  $n$  symbols of  $k$  types, the total distinct

permutations is given as:

$$W = \frac{N!}{\prod_{i=1}^k n_i}, \quad (3.10)$$

where  $n_i$  is the frequency of the  $i$ th type. For large  $n$ ,  $N!$  can be calculated using Sterling's approximation:  $\ln N! \simeq N \ln N - N$ . Therefore:

$$\ln W = -N \sum_i^k p_i \ln p_i, \quad (3.11)$$

where  $p_i = n_i/N$  is the fractional frequency of type  $i$ . And the linearly transformation of this function gives the Shannon's entropy:

$$S = - \sum_i^k p_i \log_2 p_i. \quad (3.12)$$

### 3.7.2 Methods

In 1991, Sander and Schneider [99] defined a conservation score based on a normalized Shannon's entropy as:

$$S = - \sum_i^k p_i \ln p_i \times \frac{1}{\ln k}, \quad (3.13)$$

where  $k$  is the number of symbol types (4 for DNA/RNA sequences and 20 for protein sequences).

Similarly, Shenkin et. al. [103] proposed their score as follow:

$$S = 2^S \times 6, \quad (3.14)$$

where  $S$  is the Shannon's entropy and  $k$  is the the number of symbol types. These two equations are transformations of each other, thus, they are very similar. One problem with these functions is the maximal diversity only occurs when all symbols types are represented evenly. Secondly, these functions will get their max score when there is at least  $k$  sequences involved in a column.

Another scoring scheme based on Shannon's entropy is Gerstein and Altman's [33]. Their function is designed to compare sequence conservation against structural conservation in multiple alignments of



protein sequence structures. The function is define as:

$$S = \sum_i^k \overline{p_i} \log_2 \overline{p_i} - \sum_i^k p_i \log_2 p_i, \quad (3.15)$$

where  $\overline{p_i}$  is the average frequency of residue symbol  $i$  in the alignment.  $k = 20$  for protein sequences.

Like all the previously discussed scoring methods, these entropy functions rely on the frequency of the symbols in sequence databases and ignore the stereochemical and biological properties of the residue symbols, either hydrophobic or hydrophilic.

### 3.8 Stereochemical Property Methods

Scoring methods in this group are designed around the stereo-chemical properties of residue symbols in a column. The stereo-chemical property is first classified by Taylor in 1986 [114]. The classification is based on the residue conservation from the database and the Dayhoff's mutation matrix [23]. The classification is represented in a Venn diagram as depicted in Figure 3.7.

In 1987, Zvelibil et. al. [130] converted Taylor's Venn diagram into a true-table of amino acids against their properties as seen in Figure 3.8. The score is then defined as:

$$S = n_{const} \times \frac{1}{10}, \quad (3.16)$$

where  $n_{const}$  is the constant number of properties with same states that shared between residues in a column. For example, E and D share the same 9 properties,  $n_{const} = 9$ , in the table except D is small and E is not.

#### 3.8.1 Valdar's Method

In 2001, Valdar and Thornton's [121] propose a scoring method to measure the conservation and the frequency of residues in a column as follows:

$$S(x) = \lambda \sum_i^n \sum_{j>i}^n w_i w_j M(s_i(x) s_j(x)), \quad (3.17)$$

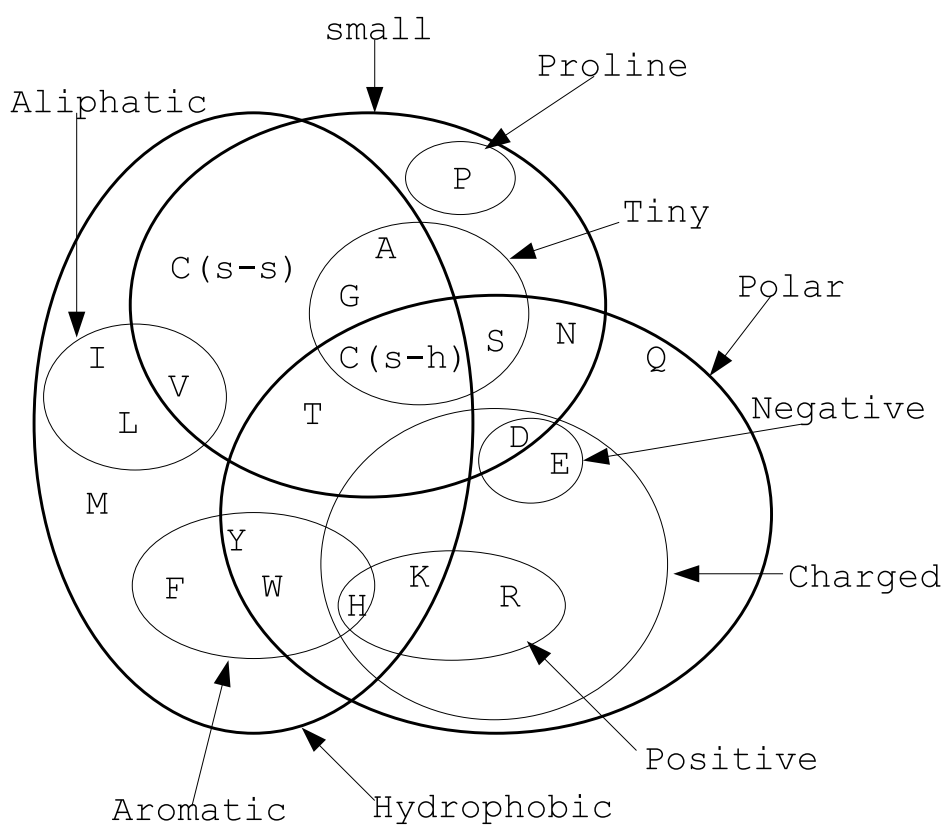


Figure 3.7. Taylor's Venn Diagram of amino acid properties

Properties: Amino Acid	Hydro- phobic	Polar	Small	Charged	Positive	Negative	Tiny	Aliph- atic	Aroma- tic	Proline
I	✓							✓		
L	✓							✓		
V	✓		✓					✓		
C	✓		✓							
A	✓		✓				✓			
G	✓		✓				✓			
M	✓									
F	✓								✓	
Y	✓	✓							✓	
W	✓	✓							✓	
H	✓	✓		✓	✓				✓	
K	✓	✓		✓	✓					
R		✓		✓	✓					
E		✓		✓		✓				
Q		✓								
D		✓	✓	✓		✓				
N		✓	✓							
S		✓	✓				✓			
T	✓	✓	✓							
P			✓							✓
B(Asx)		✓								
Z(Glx)		✓								
X(unknown)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gap	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 3.8. The True table of Taylor's Venn Diagram

where  $n$  is the sequence length and  $\lambda$  scales  $S(A)$  to range  $[0,1]$ , that is,

$$\lambda = \frac{1}{\sum_i^n \sum_{j>i}^n w_i w_j}, \quad (3.18)$$

where  $w_i$  is the weight of sequence  $s_i$  such that

$$w_i = \frac{1}{n} \sum_i^n d(s_i, s_j), \quad (3.19)$$

where  $d(s_i, s_j)$  is the distance between sequence  $s_i$  and  $s_j$  and is calculated as

$$d(s_i, s_j) = 1 - \frac{1}{n \times align_{ij}} \times \sum_{x \in align_{ij}} M(s_i(x), s_j(x)), \quad (3.20)$$

where  $align_{ij}$  is the set of all positions that manifest an amino acid in one or both of  $s_i$  and  $s_j$ , and  $n \times align_{ij}$  is the size of this set.

Matrix  $M$  is a linear transformation of a substitution matrix  $m$  such that all values in  $M$  are in range  $[0-1]$ .  $M$  is defined as:

$$M(a, b) = \begin{cases} \frac{\max(a,b) - \min(m)}{\max(m) - \min(m)}, & \text{if } a \neq \text{gap and } b \neq \text{gap} \\ 0, & \text{otherwise} \end{cases} \quad (3.21)$$

The generalized of this scoring method, called trident, is defined as:

$$S(x) = (1 - t(x))^\alpha (1 - r(x))^\beta (1 - g(x))^\gamma. \quad (3.22)$$

where  $t$  is a function of normalized symbol diversity relating to Shannan's entropy [102],  $r$  is a function of stereo-chemical diversity,  $g$  is a function of gap cost, and  $\alpha$ ,  $\beta$ , and  $\gamma$  are variables. Trident method is far more complex than other methods due to the problem of selecting appropriate values for its variables and the complexity of the three embedded functions.

In the next section, the newest scoring method is presented.

### 3.9 A New Contribution: Hierarchical Expected matching Probability scoring metric (HEP)

To align  $k$  sequences of length  $n$  optimally, we need to explore  $k$  dimensions of solution space to get the optimal solution in  $O(n^k)$  run time. In addition, the probability of aligning a residue  $a$  from sequence  $s_i$  to a residue  $b$  from sequence  $s_j$  is  $p(a|b) \leq 1$ . The probability of a residue from each sequence aligned into one column  $k$  is  $P(k) = \prod_1^n p(a|b)$ ,  $P(k) \rightarrow 0$ . This probability decreases exponentially as the number of aligning sequence increases. Consequently, the scoring function should reward aligned column scores proportionally with the matching probability function.

The Hierarchical Expected matching Probability (HEP) [82] is a scoring method that utilizes the Amino Acid Class Covering Hierarchy (AACCH) used in Smith and Smith's PIMA scoring to calculate the probability of residues being aligned into a column. Thus, provide a scoring mechanism for measuring group-to-group matches. Instead of using the ad hoc cardinalities proposed by Smith and Smith, the cardinalities are generated from any given biological distance matrix, substitution matrix, or any set of quantitative values.

#### 3.9.1 Building an AACCH scoring Tree

Unlike many scoring methods, inspired by sum-of-pair, where all pairwise residue matching scores are calculated at each step, HEP uses a scoring tree. The HEP scoring tree can be built using substitution matrix such as PAM, BLOSSUM, or any scoring matrix. In HEP, the score of matching two residues are assumed to be positive; thus, adjustment of residue matching weighting values is needed. A way to adjust negative weights is scaling up the scores by adding the magnitude of the largest negative weight to all other weights. For example, BLOSUM62 has a weight of -4 for matching C with E, or F with P, etc.; thus, adding 4 to every entry in the substitution matrix making all entries positive. This scaling technique is appropriate for scoring matrices that use log-odds probabilities such as PAM or BLOSUM. It is equivalent to multiply the original probabilities with a constant factor before taking log-odds. The constant factor has to be large enough for the smallest log-odds values to be zero.

Given an  $n \times n$  scoring matrix  $M$  and a set of AACCH classes  $S$ , the scoring tree can be generated as follows:

**Procedure Build AACCH Scoring Tree:**

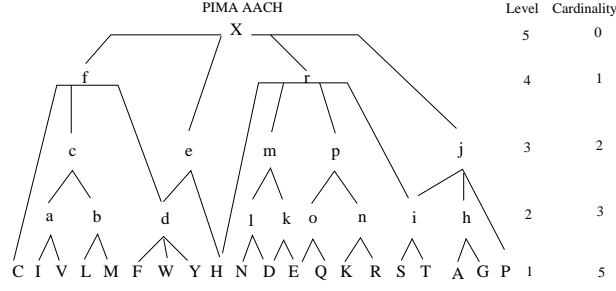


Figure 3.9. Amino Acid Class Hierarchy (AACH) used in PIMA [106]. Upper case characters are amino acids; lower case characters are amino acid classes. X is the wild-card character of any type, including a gap.

INPUT: M - an  $n \times n$  score matrix and an AACCH class S

OUTPUT: T - a tree score

$T \leftarrow S$  if  $\min(M) < 0$  !positively adjust the score

for all  $i$  and  $j$

$$M_{i,j} \leftarrow (M_{i,j} + \text{abs}(\min(M)))$$

endfor

endif

for all  $c \in T$

$$c_{\text{cardinality}} \leftarrow 0$$

endfor

for all  $c \in T$

if  $\text{size}(c) = 1$

$$c_{\text{cardinality}} \leftarrow M(c, c)$$

else

for all  $(i, j) \in c, i \neq j$

$$c_{\text{cardinality}} \leftarrow (c_{\text{cardinality}} + M(i, j))$$

endfor

$$c_{\text{cardinality}} \leftarrow (c_{\text{cardinality}} / \frac{(\text{size}(c) \times (\text{size}(c) - 1))}{2 \times c_{\text{level}}})$$

endif

endfor

if  $X_{\text{cardinality}} > 0$  ! adjusting the root

for all  $c \in Sdo$

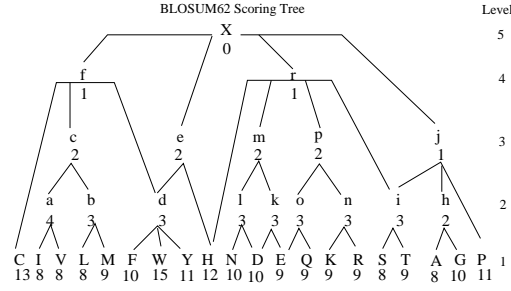


Figure 3.10. HEP scoring tree generated from BLOSUM62 substitution matrix. The matching cardinality of each amino acid class is shown at the bottom of the corresponding class symbol.

$$c_{cardinality} \leftarrow (c_{cardinality} - X_{cardinality})$$

endfor

endif

for all  $c \in S$  and  $c_{level} > 1$  !adjust level cardinality

if  $c_{cardinality} < c(children)_{cardinality}$

$$c_{cardinality} = \min(c(children)_{cardinality})$$

endif

endfor

The leaves of the tree come from the given input values, and the cardinality of each amino acid class is the average pair-wise sums scores of all pairs in the class dividing the level of the class. The tree root is at level 5. An example scoring tree generated from BLOSUM62 is shown in Figure 3.10.

### 3.9.2 The scoring metric

The score of an alignment is the summation of all column scores in the alignment. In order to control the extension of the alignment, the total columns score will be reduced by an extension factor. The extension factor is chosen to be one over the logarithmic overall alignment length. With this factor, the overall alignment score will gradually decreases proportionally to the logarithmic length extension of the alignment. Thus, inserted gaps that cause the alignment to extend will be penalized, which restricts the number of gaps inserting into an alignment to keep the alignment from unnecessarily extending.

$$Score(A_k) = \frac{1}{\log_2 |A_k|} \sum_{i=1}^{|A_k|} cScore(i) \quad (3.23)$$

where:

$k$ : number of sequences,  $k \geq 2$

$A_k$ : an alignment of  $k$  sequences

$|A|$ : length of the alignment

The equation for calculating the column score is as follow:

$$cScore(i) = \begin{cases} 0, \text{ if column } i \text{ matches only at tree root} \\ \frac{1}{(k-1)^{\Gamma_i}} \sum_{l=1}^{\log|T_{col_i}|} \sum_{j=1}^{|T_l|} \frac{(count(node_j)-1)^{c_j}}{l} \end{cases} \quad (3.24)$$

The score (weight) of aligning column  $i$  of alignment  $p$ ,  $p[i]$ , and column  $j$  of alignment  $q$ ,  $q[j]$ , is the column-score of column generated by merging all the residues in  $p[i]$  and  $q[j]$ . The alignment score equation then becomes:

$$Score(A_k) = \frac{1}{\log_2|A_k| \times (k-1)^{\Gamma}} \times \sum_{i=1}^{|A_k|} \sum_{l=1}^{\log|T_i|} \sum_{j=1}^{|T_l^i|} \frac{(count(node_j)-1)^{c_j}}{l}, \quad (3.25)$$

$\forall w_l > 0$

where:

$|T_i|$ : the size of score tree for column  $i$ , i.e. the active scoring tree.

$|T^l|$ : the size of the score tree at level  $l$ .

$c_j$ : the cardinality of node  $j$  in the active scoring tree.

$\Gamma_i$ : the cardinality at the first level of the scoring tree built for column  $i$ .

$count(node_j)$ : the number of residue matched at  $node_j$  in the active scoring tree.

The scoring tree is built before any alignment occurs, which reduces the complexity of alignment algorithm by at least an order factor, comparing to sum-op-pair related scoring methods, for not calculating the match score at every alignment step.

### 3.9.3 Proof of scoring metric correctness

The proof of HEP giving higher weights for columns with higher degree of conservation is as follows:

Given a column of  $k$  aligned sequences,  $k > 1$ , containing  $n$  residues,  $n \leq k$ . Let  $c_l > 0$  be the cardinality of two residues matching at level  $l$ ,  $l \leq \log 5$ ,  $c_1 > 1$ , and  $c_i - c_{i+1} \geq 0$ . The score of the column

is 1 if all the residues are homogeneous; otherwise, the score of the column will be  $\frac{1}{(n-1)^l} \times [(n-1-a_1)^{c_l} + (n-1-a_2)^{c_1} + \dots + (n-1-a_{\frac{n}{2}})^{c_1} + \dots + \frac{(a_i+a_j-1)^{c_t}}{t} \dots + \frac{(n-1)^0}{5}]$ ,  $\forall c_t \geq 1$  and  $a \geq 1, i \neq j, 1 \leq i, j \leq \frac{n}{2}$  and  $1 \leq t \leq \log n$ .

Let  $c = \min(c_l)$  and  $\alpha = c - c_{l+1}$ . If there are only two types of residues (divergent of degree 2), the column score will be  $(k-x-1)^c + (x-1)^c + \frac{n^{c-\alpha}}{2}, 1 \leq x < k$  and  $\alpha \geq 0$ . To prove that the homogenous column has a higher score, we need to prove  $(n-1)^c > (n-x-1)^c + (x-1)^c + \frac{(n-1)^{c-\alpha}}{l}, l \geq 2$ . Since  $(n-1)^c > 2 \times [(n-x-1)^c + (x-1)^c], \forall c > 1$ . To prove  $(n-1)^c > (n-x-1)^c + (x-1)^c + \frac{(n-1)^{c-\alpha}}{l}$  is to prove  $\frac{(l-1)}{l} \times (n-1)^c \geq \frac{(n-1)^c}{2}$ . This is true  $\forall l > 1$ .

Similarly, the proof can be extended to any degree of divergent  $d, d \leq n$ . If there are  $d$  types of amino acids at the leaf level of the scoring tree, the column score will be smaller than a similar tree with divergent degree  $d-1$  because splitting one of the leaf groups into two groups from a  $d-1$  degree divergent tree make it a  $d$  degree divergent tree.

### 3.9.4 Examples

Given the following sequences:  $s_1 = \text{NNN}$ ,  $s_2 = \text{NND}$ , and  $s_3 = \text{NDE}$ . A possible alignment is

$$A_3 = \begin{cases} s1 & \text{NNN} \\ s2 & \text{NND} \\ s3 & \text{NDE} \end{cases}.$$

Using the AACH matching weight, where the matching of N with N = 5, N with D = 3, D with E = 3, N with E = 2, we will have the following score:

$$\text{score}(A_3) = \frac{1}{\log_2(3) \times (3-1)^5} ([ (3-1)^5 ] + [ (2-1)^5 + \frac{(3-1)^3}{2} ] + [ \frac{(2-1)^3}{2} + \frac{(2-1)^3}{2} + \frac{(3-1)^2}{3} ]) = 0.7952$$

(Figure 3.11 shows the active scoring tree using PIMA scoring cardinal scheme.) If instead, we used the scoring tree built from BLOSUM62 substitution matrix, where matching N with N = 10, N with D = 3, D with E = 3, the alignment score would be:

$$\text{score}(A_3) = \frac{1}{\log_2(3) \times (3-1)^{10}} ([ (3-1)^{10} ] + [ (2-1)^{10} + \frac{(3-1)^3}{2} ] + [ \frac{(2-1)^3}{2} + \frac{(2-1)^3}{2} + \frac{(3-1)^2}{3} ]) = 0.6360$$



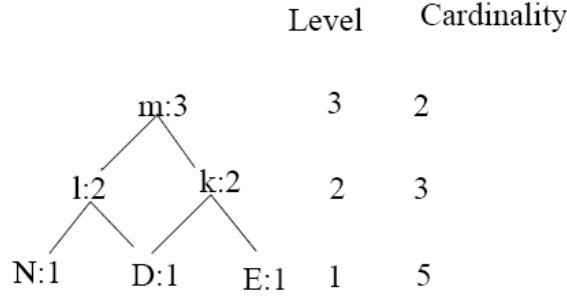


Figure 3.11. PIMA Active Scoring Tree for the 3rd column

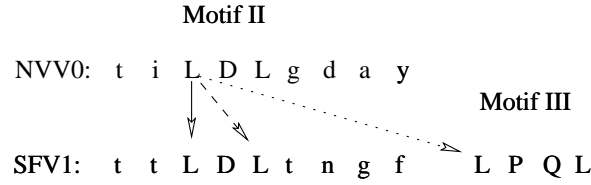


Figure 3.12. Matching a Leucine from NVV0 with a similar amino acid in sequence SFV1 [51]. The dotted arrow shows a mismatch, the dashed arrow shows a better match, and the solid arrow shows the best match.

### 3.9.5 Scoring metric and sequence weighting factor

Sequence weighting is a factor that should be included in a scoring method. Aligning a residue  $a$  from motif  $x$  in sequence  $s_i$  to a similar residue  $b$  in the same core motif in sequence  $s_j$  should have more weight than aligning  $a$  to  $b$  in a different motif or to any other residue from other locations, as in Figure 3.12. This depends on how much biological information about the sequences we have. In some instances, there are groups of similar sequences among aligning sequences. Normalizing this redundancy will help reduce bias in MSA results. However, biologists may include a subset of similar sequences into the aligning sequences to guide the alignment toward known sequences. Normalizing and factorizing out the redundancy in this case is not useful and may lead to unwanted results. Therefore, the weight factor should be from a function that measures the significance of a residue in a motif to a motif rather than the redundancy of similar sequences. If the weight function is known, it can be combined into the scoring metric. The column score will be:

$$cScore(i) = \begin{cases} 0, & \text{if column } i \text{ only matches at tree root} \\ \frac{w(i)}{(k-1)_i^F} \sum_{l=1}^{\log|T_{col_i}|} \sum_{j=1}^{|T_l|} \frac{(count(node_j)-1)^{c_j}}{l} \end{cases} \quad (3.26)$$

where  $w(i)$  is the motif weight of residues in column  $i$  and is calculated as

$$w(i) = \begin{cases} 1, & \text{iff residues are from a similar motif and} \\ & \text{are biological and locational order equivalent} \\ \alpha, & \text{iff residues are from similar motif and} \\ & \text{are biological equivalent} \\ \beta, & \text{otherwise} \end{cases} \quad (3.27)$$

where  $0 < \beta < \alpha \leq 1$ .

### 3.9.6 Evaluation Datasets

The effectiveness and reliability of HEP scoring function are evaluated through three comprehensive tests. The first test is designed to test HEP's capability to detect and rank residue conservation among columns of a multiple sequence alignment. For this test, we utilize the theoretical sequence dataset created by W. S. Valdar [121]. These theoretical residue conservation set is shown in Table 3.5. The second test is to evaluate its reliability on scoring and ranking real biological benchmarks. For this test, we utilize three multiple sequence alignment benchmarks: RT-OSM [51], BALiBASE3.0 (Benchmark Alignment dataBASE) [115], and PREFAB4.0 [30]. Along with these benchmarks, the following multiple sequence alignment tools: ClustalW, PIMA, DCA, DALIGN2, and MAFFT2, T-COFFEE [57, 74, 83, 106, 109, 118] are used to generate alignments for the test. And the third test is to detect whether HEP scoring function can lead to better multiple sequence alignment results. This test is performed by implementing and deploying the HEP scoring function along with SP, Smith's PIMA, Valdar's scoring functions, described in previous Sections, in a progressive multiple sequence alignment program. These three scoring function are chosen for various reasons: SP is the most popular scoring functions and majority of other scoring functions are variations of SP, Smith's PIMA covering hierarchy is used in HEP scoring, and Valdar's scoring function is claimed to be the most reliable one.

**Theoretical Residue Conservation Measurement** In 2001, William Valdar developed a theoretical dataset [121] to evaluate scoring function's capability on measuring residue conservation correctly. This dataset is shown in Figure 3.5

Originally, column (d) in Table 3.5 was considered having higher conservation score than column (c) without supporting justification. From multiple sequence alignment perspective, column (c) indicates that phenylalanine (F) is conserved and shared by nine out of ten sequences. On the other hand, column (d)

fails to indicate which amino acid is conserved among all the sequences. Analytically, the chance aspartic acid (D) will be mutated to glutamic acid (E) is greater than the chance that aspartic acid (D) will change to a phenylalanine (F), the probability for all five residues (either D or E) in column ( $d$ ) mutating to other amino acids is  $(p_{DE})^5 = \prod_1^5 p_{DE}$  [where  $p_{DE}$  is the probability that D is mutated into E], which is much smaller than the probability of residue D in column ( $c$ ) mutating to residue F. This result follows the decomposition of the BLOSUM62 substitution matrix and its log-odd scoring function from [29]

$$S(a, b) = \frac{1}{\lambda} \log \frac{p_{ab}}{f_a f_b} \quad (3.28)$$

where  $S(a, b)$  is the score of aligning residues  $a$  and  $b$ ,  $\lambda = 0.347$  is a constant, and  $f_i$  is the background frequency of residue  $i$ . All amino acid background frequencies are derived from the existing sequences [29]. With the background frequencies  $f_D = 0.0539$ ,  $f_E = 0.0539$  and  $f_F = 0.469$  obtained from [29], we found that  $p_{DE} = 5.8E - 3$  and  $f_{DF} = 9.0E - 4$ . Thus,  $\prod_1^5 p_{DE} = 6.56357E - 12 \ll p_{DF} = 9.0E - 4$ . Intuitively, a column with 100% residue conservation (column j) should have a better conservation score than a column with 40% residue conservation (column k). Thus, we rearrange these columns to the right order.

**RT-OSM Dataset** It is best to measure the reliability of a scoring method based on actual facts. Therefore, we utilize the RT-OSM data-set selected by Hudak and McClure in 1999 [51]. The data-set is shown in Figure 3.13. These are order-specific-motifs (OSM) sequences because they contain a set of motifs occurring in a specific order among the sequences. These selected subsequences contain six different motifs with lengths ranging from 1 to 5 amino acids. This data-set is perfect for evaluating the biological reliability of scoring functions in this study. The function to be evaluated are: SP [15], Valdar’s [121], Trident [121], Smith’s Pima [106], and HEP scoring functions.

**BALiBASE 3.0 and PREFAB 4.0 Datasets** The BALiBASE 3.0 (Protein REference Alignment Benchmark) [115] is a reference benchmark of sequences that are manually aligned with details annotations. The alignment of BALiBASE sequences are based on three-dimensional structure superposition, except the transmembrane sequences. BALiBASE MSA reference benchmark consists of 247 reference alignments, containing over 1,000 sequences. The BALiBASE is categorized into 9 reference sets, and 218 of the sets are in the first 4 reference groups. The last five groups contain sequence sets few short and distanced sequences. Similarly, PREFAB4.0 [30] benchmark contains 1932 alignments averaging 49 sequences of length 240. We randomly select 150 reference alignments to use in our tests. Both BALiBASE3.0 and PREFAB4.0 use a

Table 3.5. Each label column represents a residue position in a multiple sequence alignment. Amino acids are identified by their one letter code and gaps by a dash ("-"). Note: column (k) comes from an alignment of 10 sequences where column (j) comes from an alignment of only 4 sequences (no gaps in column (j)). The column score order is (a) > (b) > (c) > (d) > (e) > (f), then (g) > (h) > (i), and (j) > (k) (adapted from [121])

Columns											
Seq.	(a) >	(b) >	(c) >	(d) >	(e) >	(f)	(g) >	(h) >	(i)	(j) >	(k)
1	D	D	D	D	D	D	I	P	D	L	L
2	D	D	D	D	D	D	I	P	V	L	L
3	D	D	D	D	D	D	I	P	Y	L	L
4	D	D	D	D	D	D	I	P	A	L	L
5	D	D	D	D	D	D	L	W	T		-
6	D	D	D	E	E	E	L	W	K		-
7	D	D	D	E	E	E	L	W	P		-
8	D	D	D	E	E	E	L	W	C		-
9	D	D	D	E	E	F	V	S	R		-
10	D	E	F	E	F	F	V	S	H		-

	I	II	III	IV	V	VI
HT13	pvk <b>K</b> a--	t- <b>IDL</b> kda <b>f</b>	- <b>LPQG</b> -fk	q <b>YMDD</b> Il1	sh <b>GL</b> --	k <b>FLG</b> qii
NVV0	ikk <b>K</b> ---	ti <b>LDI</b> gday	- <b>LPQG</b> -wk	- <b>YMDD</b> Iyi	qy <b>GFM</b> -	k <b>WL</b> Gfel
SFV1	pvp <b>K</b> p--	tt <b>LDL</b> tngf	- <b>LPQG</b> -fl	a <b>YVDD</b> Iyi	na <b>GYV</b> v	e <b>FLG</b> fni
HERVC	pvp <b>K</b> p--	tc <b>LDL</b> kda <b>f</b>	- <b>LPQR</b> -fk	q <b>YVDD</b> L1l	tv <b>GIR</b> c	c <b>YL</b> Gfti
GMG1	mvr <b>K</b> a--	tk <b>VDV</b> raa <b>f</b>	- <b>CPFG</b> -la	a <b>YLDD</b> Ili	-- <b>GLN</b> -	k <b>YL</b> Gfiv
GM17	v-p <b>K</b> kqd	tt <b>IDL</b> akgf	- <b>MPFG</b> -lk	v <b>YLDD</b> Iiv	-- <b>NLK</b> -	t <b>FLG</b> -hv
MDG1	lvp <b>K</b> ksl	sc <b>LDL</b> msgf	- <b>LPFG</b> -lk	l <b>YMDDL</b> vv	-- <b>NLK</b> -	t <b>YL</b> G-hk
MORG	vvr <b>K</b> k--	tt <b>MDL</b> qngf	- <b>APFG</b> -fk	l <b>YMDD</b> Iiv	-- <b>GLK</b> -	h <b>FLG</b> -hi
CAT1	lvd <b>K</b> pkd	eq <b>MDV</b> kta <b>f</b>	k <b>SLYG</b> -lk	l <b>YVDD</b> Mli	-- <b>EMK</b> -	r <b>ILG</b> idi
CMC1	tit <b>K</b> rpe	hq <b>MDV</b> kta <b>f</b>	k <b>AIYG</b> -lk	l <b>YVDD</b> Vvi	--- <b>KR</b> -	h <b>FIG</b> iri
CST4	ftk <b>K</b> rng	t- <b>LDI</b> nhaf	k <b>ALYG</b> -lk	v <b>YVDD</b> Cvi	in <b>KLK</b> -	d <b>ILG</b> mdl
C1095	fnr <b>K</b> rdg	tq <b>LDI</b> ssay	k <b>SLYG</b> -lk	l <b>FVDD</b> Mil	it <b>TLK</b> k	d <b>ILG</b> lei
NDM0	mih <b>K</b> t--	af <b>LDI</b> qqaf	g <b>VPQG</b> svl	t <b>YADD</b> Tav	ts <b>GL</b> --	k <b>YL</b> Gitl
NL13	lip <b>K</b> p--	s- <b>IDA</b> ekaf	g <b>TRQG</b> cpl	l <b>FADD</b> Miv	vs <b>GYK</b> -	k <b>YL</b> Giq1
NLOA	fip <b>K</b> a--	af <b>LDI</b> egaf	g <b>CPQG</b> gvl	g <b>YADD</b> Ivi	ev <b>GLN</b> -	k <b>YL</b> Gvi-
NTC0	vlr <b>K</b> p--	am <b>LDG</b> rnay	g <b>VRQG</b> mvl	a <b>YLDD</b> Vtv	alg <b>IE</b> -	r <b>VLG</b> agv
ICD0	eip <b>K</b> p--	vd <b>IDI</b> k-gf	g <b>TPQG</b> gil	r <b>YADD</b> Fki	rl <b>DL</b> Di	d <b>FLG</b> fkl
IAG0	fkf <b>K</b> t--	ie <b>GDI</b> ks-f	g <b>VPQG</b> gii	r <b>YADD</b> Wlv	el <b>KIT</b> l	- <b>FLG</b> vnl
ICS0	wip <b>K</b> p--	ld <b>ADI</b> sk-c	g <b>TPQG</b> gvi	r <b>YADD</b> Fvi	em <b>GLE</b> l	n <b>FLG</b> fnv
IPL0	yip <b>K</b> s--	le <b>ADI</b> r-gf	g <b>VPQG</b> gpi	r <b>YADD</b> Fvv	sr <b>GLV</b> l	d <b>FVG</b> fnf

Figure 3.13. The RT OSM sequences. The six motifs of the RT OSM are indicated by roman numeral(I-VI). The bold and capitalized letters represent the core amino acids of each motif. Adapted from [51]

TC (Total percentage of matching Columns) score to evaluate the accuracy of an alignment result against its reference alignment.

### 3.9.7 Evaluation Results

Each data set is designed for specific purpose, so each evaluation must be designed for each data-set. For the theoretical sequence set, we rank the sequence columns using HEP and compare its results with the predefined ranks. For the RT-OSM sequence set, we randomly shift the residues and remove the gaps in the sequences to generate six alignments, each contains from one to six of the motifs aligned. It is widely believed that the degree of functional constraint dictates the conservation region of a sequence. This also means that a higher degree of conservation in an aligned region of a multiple sequence alignment result indicates the functional importance of that aligned position. Thus, when the residue in the RT-OSM data set are shifted around leaving only one or few of the motif regions intact, a reliable scoring method should be able to detect the significance of these regions. The score should be higher for formations with more intact motifs.

And for the BALiBASE3.0 and PREFAB4.0 benchmarks, we define a reliability score to evaluate the scoring functions.

**Ranking the Theoretical Set of Sequences** All the columns of the theoretical sets are scored by HEP method using three different scoring matrices and schemes. Table 3.6 summarizes the result of our measurements, where HEP-PIMA is HEP score using Smith’s PIMA cardinality, HEP-P250 is HEP score using generated PAM250 scoring tree, and HEP-BL62 is HEP score using generated BLOSUM62 scoring tree. The correct ranking of the column scores are:  $(a) > (b) > (c) > (d) > (e) > (f)$ , then  $(g) > (h) > (i)$ , and  $(j) > (k)$ . All three tests show that HEP scoring method correctly ranks the columns in these orders. Column  $(b)$  and column  $(c)$  have the same degree of divergence, however, the mutation probability between aspartic acid (D) and glutamic acid (E) are smaller than that between aspartic acid (D) and phenylalanine(F). Thus, column  $(b)$  must have higher score than column  $(c)$ . Although the difference between these two columns is very small, all three variations of HEP method correctly measure it.

**Ranking the RT OSM Sequences** As mentioned earlier, we randomly shift the residues left, right, or delete gaps between the RT-OSM sequences to generate 6 different alignments, each contains from one to six of the motifs aligned. These alignments are scored by the following scoring methods: Sum-of-pair (SP) [15], PIMA [106], Valdar [121], Trident [121], HEP-PIMA, HEP-P250, and HEP-BL62 (as described

Table 3.6. Each label column represents a residue position in a multiple sequence alignment. Amino acids are identified by their one letter code and gaps by a dash ("-"). The column score correct order is  $(a) > (b) > (c) > (d) > (e) > (f) > (g) > (h) > (i) > (j) > (k)$ . Note: column (j) comes from an alignment of only 4 sequences (no gaps); and the table cannot show all significant digits.

Seq.	Columns										
	(a) >	(b) >	(c) >	(d) >	(e) >	(f)	(g) >	(h) >	(i)	(j) >	(k)
1	D	D	D	D	D	D	I	P	D	L	L
2	D	D	D	D	D	D	I	P	V	L	L
3	D	D	D	D	D	D	I	P	Y	L	L
4	D	D	D	D	D	D	I	P	A	L	L
5	D	D	D	D	D	D	L	W	T	-	-
6	D	D	D	E	E	E	L	W	K	-	-
7	D	D	D	E	E	E	L	W	P	-	-
8	D	D	D	E	E	E	L	W	C	-	-
9	D	D	D	E	E	F	V	S	R	-	-
10	D	E	F	E	F	F	V	S	H	-	-
Methods	Column Scores										
HEP-PIMA	1.0	5.6110E-1	5.5493E-1	4.0856E-2	2.5792E-2	2.0805E-2	9.5824E-3	8.2756E-3	5.3628E-5	1.0000E+0	4.1152E-3
HEP-BL62	1.0	3.0795E-1	3.0794E-1	6.0156E-4	3.0645E-4	3.0092E-4	3.1249E-4	7.0552E-8	1.2000E-12	1.0000E+0	1.5242E-4
HEP-P250	1.0	2.4332E-1	2.4330E-1	1.1891E-4	6.1343E-5	5.9447E-5	2.7911E-7	1.2000E-12	0.0000E+0	1.0000E+0	2.0908E-7

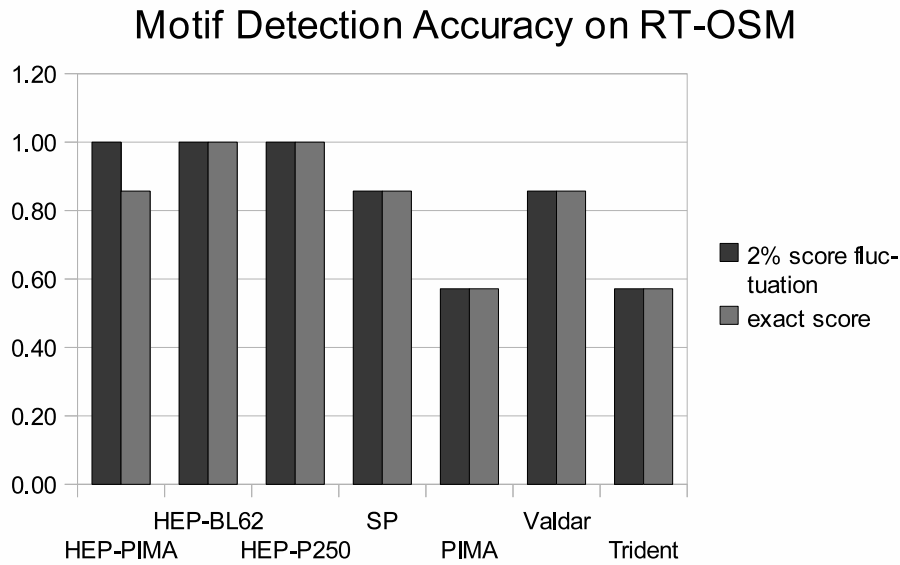


Figure 3.14. Motif detection accuracy on the RT-OSM data set. Score of 1.0 means a method correctly detects and ranks all six motifs in the data set.

in previous Sections). The most reliable scoring method will give the highest score to the alignment with the most intact motifs, and the lowest score to the alignment with the fewest intact motifs. Since the scoring is done on the whole alignment, the columns around the motifs would contribute some weights to the total alignment score. In addition, the fifth motif in the data-set is very diverse, which may contribute a very minimal weight to some scoring methods. Therefore, we allow 2 percents of fluctuation in alignment scores. Thus, if a scoring function gives scores within 2% range to two alignments: one contains four motifs  $I$  to  $IV$  and the other contains five motifs  $I$  to  $V$ , we consider it to be correct. The motif detection accuracy of a scoring function is defined to be the number of correctly identified motifs over all motifs in the data set, i.e:

$$Accuracy = \frac{\# \text{ of motifs correctly identified}}{\# \text{ all real motifs}}$$

Figure 3.14 shows the motif detection accuracy of these scoring methods. Both version of HEP scoring, one uses BLOSUM62 substitution matrix and one uses PAM250 substitution matrix, are able to detect all six motifs without the need of 2% score fluctuation. Allowing the 2% fluctuation improves the HEP method using PIMA score [106] and Valdar's method [121]. Overall, all three versions of HEP scoring method consistently outperform other scoring methods.

**BALiBASE3.0 and PREFAB4.0 MSA Results** The BALiBase3.0 and PREFAB4.0 benchmarks come with a total column score function (TC) that can evaluate an alignment against a reference alignment from the benchmarks. Thus, giving  $n$  sequences to  $k$  different MSA tools, we will get  $k$  different alignments. Scoring these alignments against the benchmark using the total column score (TC) will generate  $k$  scores, one for each alignment, indicating how close each alignment is to the reference alignment, i.e. the expected alignment. Therefore, to evaluate the reliability of a scoring function, we define a reliability score to be the number of the best alignments correctly identified over the total number of all alignments, i.e.:

$$R = \frac{\# \text{ alignments correctly identified}}{\# \text{ total alignments}}$$

For example, if we give a set of  $n$  sequences to ClustalW, MAFFT, and T-COFFEE, we will get three alignments A, B, and C respectively. Let's further assume the reference total columns scores of these alignments are  $TC(A) = 1$ ,  $TC(B) = 0.8$ , and  $TC(C) = 0.3$ . These TC scores indicate A is exactly identical to the reference alignment, B is close, and C is very different from the expected alignment. Thus, any scoring function that scores alignment A with highest score will get one point, and all other function get zero point. For  $n$  different sets of sequences, the reliability score is the number of points each scoring function accumulates over  $n$ .

Since random alignments are often meaningless, we use MAFFT2 [57], ClustalW [117], T-COFFEE [83], PIMA [106], and DALIGN2 [74] MSA tools to generate 5 multiple sequence alignments for each set of input sequences. The alignment results of these tools are scored against the reference alignments in the benchmark. Thus, for 247 reference tests in BALiBASE, we obtained 1235 resulting alignments. And for 150 selected data set from PREFAB4.0, we get 750 resulting alignments. Figures 3.15 and 3.16 show the reliability scores of the tested scoring methods on BALiBASE3.0 and PREFAB4.0, respectively.

Both BALiBASE3.0 and PREFAB4.0 benchmark tests indicate HEP scoring method with BLOSUM62 and PAM250 are more likely to identify and rank the multiple sequence alignments correctly. This feature allows biologists to identify which alignment tools are more accurate and reliable for their multiple sequence alignments. From these results, it seems that HEP scoring with BLOSUM62 is the most reliable method.

### Implementation of HEP Scoring Method in Progressive Multiple Sequence Alignment

The benchmark test results suggest that HEP scoring is more sensitive and reliable than other tested methods. To further confirm that HEP can improve multiple sequence alignment results, we have implemented HEP in a progressive multiple sequence alignment, which is the same alignment technique employed in



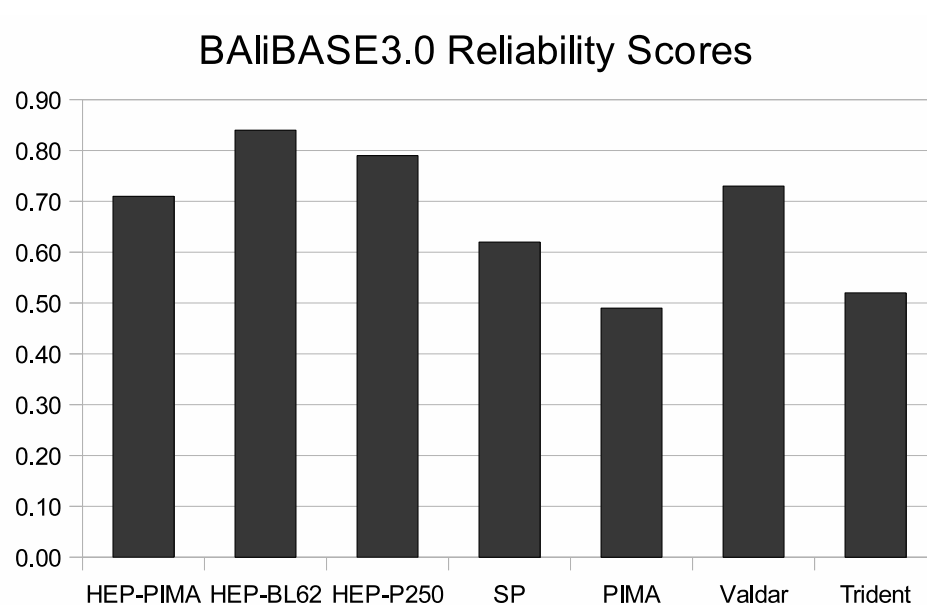


Figure 3.15. Reliability scores on the BALiBASE3.0 benchmark

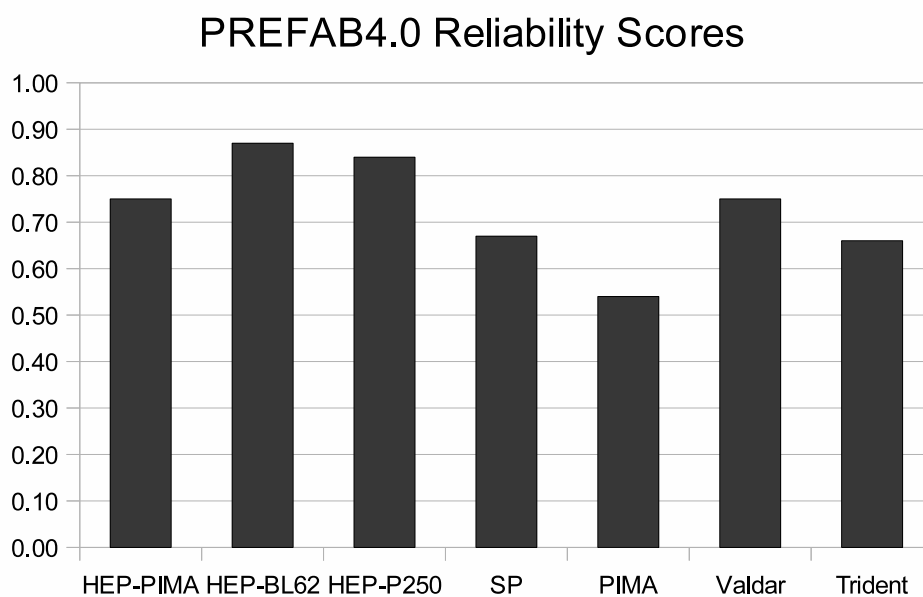


Figure 3.16. Reliability scores on 150 references of the PREFAB4.0 benchmark

ClustalW, PIMA, T-COFFEE, etc. This technique can be summarized as follows:

- (i) Perform all-pairwise sequence alignment to calculate the distance between the input sequences. (ii) Build a dendrogram tree using either UPGMA [108] or NJ [98] hierarchical clustering methods. (iii) Pair-wise align the sequences (or two groups of sequences) following the order of the dendrogram from the leaves to the root.

In our implementation, ClustalW is utilized to perform steps (i) and (ii) since at the pairwise level of sequence alignment HEP score acts exactly the same as sum-of-pair score used in this ClustalW. In step (iii) two versions of Needleman-Wunsch's [77] algorithm are implemented: one uses SP [15] scoring method, one uses Smith's Pima [106] method, one uses Valdar's method [121], and the last one uses HEP method. The Trident method is not implemented since its appropriate parameters are hard to determine; Moreover, Trident method did not perform well in the previous BALiBASE3.0 and PREFAB4.0 tests.

When executing our progressive MSA, a set of input sequences will produce four alignment results. The scoring matrix used in these tests is BLOSUM62, and the gap insertion/delete cost is -10 for the SP and Valdar's methods. We measure the accuracy of a test alignment by using the number of correctly aligned columns of the test alignment divided by the total number of columns in the reference alignment. On all tests, HEP scoring method shows better performance than other scoring methods. On the BALiBASE3.0 benchmark, alignment results that employed HEP scoring yield an average TC score about 7%, 5% and 4% higher than those using Smith's Pima [106], SP [15], and Valdar [121] scoring methods, respectively. Similarly, on the PREFAB4.0 benchmark, HEP yields an average TC score of 4%, 5%, and 3% higher. These results are represented in Figure 3.17 and Figure 3.18. Thus, it is a strong indication that HEP scoring method will improve multiple sequence alignment accuracy.

As a conclusion to this chapter, each scoring scheme is designed for a specific purpose and may work on certain set of sequences. A scoring method that works with all sets of biological sequences may never exist due to the unknown biological nature of the sequences and their evolution.

### BALiBASE3.0: Average Total Column Scores

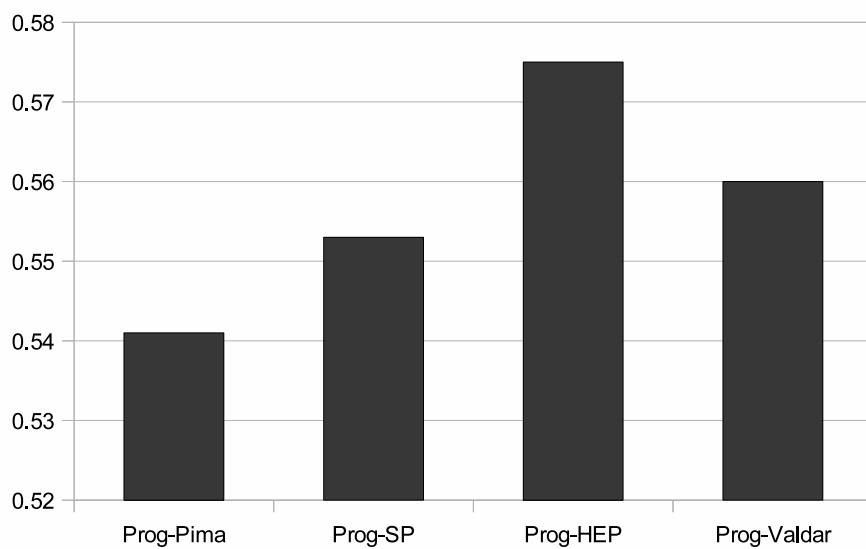


Figure 3.17. BALiBASE Total Column (TC) scores

### PREFAB4.0: Average Total Column Scores

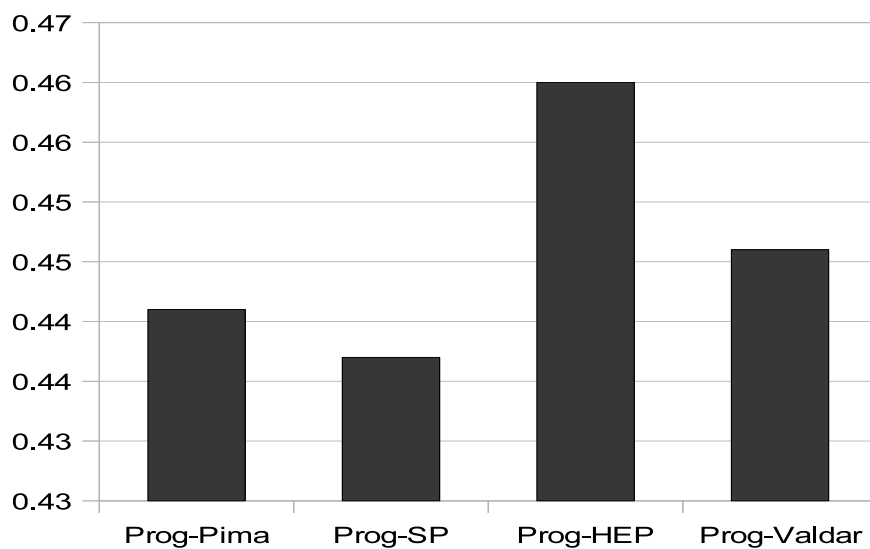


Figure 3.18. Total Column (TC) scores of 150 random sequence sets from PREFAB4.0

## Chapter 4

### SEQUENCE CLUSTERING

It is natural to group closely-related sequences into clusters before performing multiple sequence alignment. Scientists in the field have been debating on how to systematically detect and measure the relatedness between sequences, i.e. the homologous features and functional similarities between sequences. Figure 4.1 shows the mutation steps of residues in different species sequences during evolution. The phylogeny tree indicates at any specific site which symbol is mutated to the other. (b) shows the original symbol was either G or A, after some evolution time, G remains unchanged in sequence A and sequence B; however, this symbol has been changed to C in the family (branch) containing sequences Seq3, Seq4, and Seq5. Finally, C has been changed to A in the subfamily containing sequences Seq4 and Seq5.

There are many algorithms for clustering such as K-means [72], fuzzy C-means [27], hierarchical clustering, and probabilistic clustering [28], etc. However, these methods mainly classify the sequences into groups with smallest distance (edit distance/Hamming distance, Euclidean distance) and often marginalize the biological implication between the sequences. K-Means finds  $k$  clusters by starting with  $k$  random seeds (initial  $k$ -clusters) and grouping the nearest data points (sequences) to the cluster based on the distance between the data points and the centroid of the cluster. Fuzzy-C-Means is almost identical to K-Means method except a data point (sequence) is allowed to participate in more than one clusters. The probabilistic clustering algorithms group the data points into clusters based on their distributions such as Gaussian or Poisson. These clustering algorithms are not very reliable for sequence clustering and very time consuming, for example, K-means is NP-Hard for  $d$  dimensions, and it yields a runtime of  $O(n^{dk+1} \log n)$  for fixed  $d$  and  $k$ .

On the other hand, hierarchical clustering methods are relatively fast and more acceptable in sequence clustering since they treat sequences in pairs to emphasize on the significance between them. The most widely-used methods are Neighborhood Joining [NJ] method [76, 98] and Unweighted Pair Group Method with Arithmetic mean, (also known as average linkage method) [UPGMA]. The weighted version of UPGMA, [WPGMA] [108] also known as maximum linkage, is not popular.

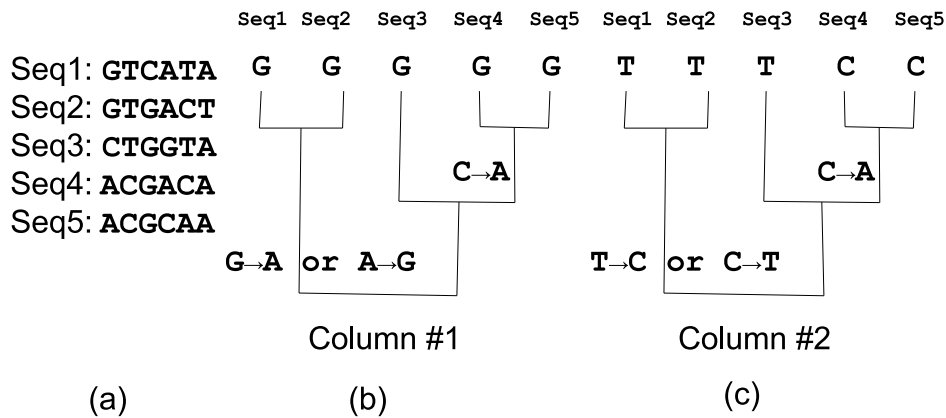


Figure 4.1. (a) is the input sequences, (b) substitutions on first column, and (c) substitutions on second column

#### 4.1 Unweighted Pair Group Method with Arithmetic mean - UPGMA

Like all other methods, Unweighted Pair Group Method with Arithmetic mean (UPGMA) requires the knowledge of the distances between all taxonomies, sequences in this case. These distances naturally come in the form of a matrix. UPGMA uses the distance matrix to construct a phylogenetic tree, or evolutionary tree. In general, each sequence is treated as an Operational Taxonomic Unit (OTU). UPGMA starts with two closest OTUs and combines them to build a new OTU. The distances between all other OTUs to the new OTU are recalculated. The new distances are arithmetic means between the OTU to the all the members of the new OTU. This process is repeated until all OTUs are merged.

Example: Given 4 sequences A, B, C, D and their distance matrix  $d$  as follows:

$$d = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & & & \\ B & 7 & 0 & & \\ C & 11 & 6 & 0 & \\ D & 14 & 9 & 7 & 0 \end{array}$$

First, A and B are grouped together since their distance is minimal ( $d_{AB} = 7$ ) [ note:  $d_{CD} = 7$  could be chosen also]. Next the distances between all other OTUs to AB are calculated. For example, the distance between C to AB and D to AB are:

$$d_{C(AB)} = (d_{AC} + d_{BC})/2 = (11 + 6)/2 = 8.5$$

$$d_{D(AB)} = (d_{AD} + d_{BD})/2 = (14 + 9)/2 = 11.5.$$

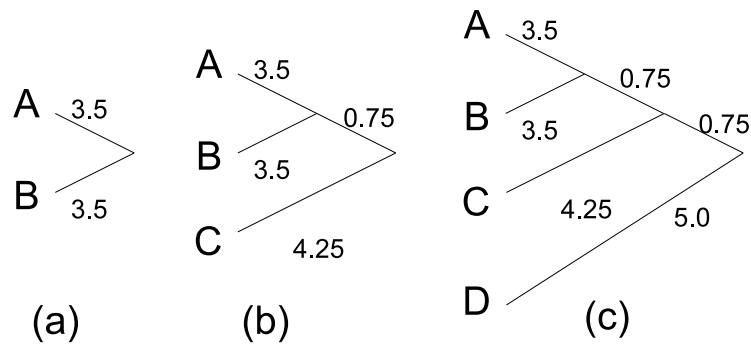


Figure 4.2. This figure show each step of building the UPGMA phylogeny

Thus, the new distance matrix will be:

$$d = \begin{array}{c|ccc} & AB & C & D \\ \hline AB & 0 & & \\ C & \mathbf{8.5} & 0 & \\ D & 11.5 & 9 & 0 \end{array}$$

Next, C is grouped with AB to build OTU ABC and the distance between D to ABC is calculated as:

$$M_{D(ABC)} = (M_{AD} + M_{BD} + M_{CD})/3 = (14 + 9 + 7)/3 = 10.$$

Figure 4.2 illustrates each step to create an UPGMA phylogeny tree for this example. The final tree is shown in (c). One of the weaknesses of UPGMA is its constant rate of evolution. UPGMA has runtime complexity of  $O(n^3)$ .

## 4.2 Neighborhood Joining Method - NJ

Unlike UPGMA, the Neighborhood Joining method (NJ) is a bottom-up sequence clustering approach to construct an unrooted phylogenetic tree with vary evolution rate. Initially, NJ treats all OTUs as a star-tree and computes the average distance between an OTU to all other OTUs. NJ combines a pairs of OTUs that are close to each other and far from all other OTU into a new OTU. The distance between all other OTU to the new OTU is calculated. This process is repeated until only two OTUs remain. This algorithm is detailed as follows: Given a set of  $n$  sequences and a distance matrix  $d$ .

Step 1: Calculate the net divergence for each OTU to all other OTUs as:

$$u_i = \sum_k d_{ik} / (n - 2)$$

Step 2: Calculate the new distance matrix M as:

$$M_{ij} = d_{ij} - u_i - u_j$$

and join two closest neighbors, i.e. the pairs with minimal value from M.

Step 3: Calculate the distance between OTU  $i$  and OTU  $j$  to the new OTU  $ij$  as:

$$d_{i,(ij)} = (d_{ij} + u_i - u_j) / 2$$

$$d_{j,(ij)} = (d_{ij} + u_j - u_i) / 2$$

Step 4: Compute distances between new OTU to all other OTUs as:

$$d_{(ij),k} = (d_{ik} + d_{jk} - d_{ij}) / 2$$

Step 5: Replace  $i$  and  $j$  by  $(i, j)$ .

Step 6: Repeat from step 1 until only 2 OTUs remain.

Example: Given 4 sequences A, B, C, D and their distance matrix d as follows:

d =

	A	B	C	D
A	0			
B	1	0		
C	3	3	0	
D	3	3	4	0

Step 1: The net divergence  $u_i$  are:

$$u_A = (d_{AB} + d_{AC} + d_{AD}) / (4 - 2) = (1 + 3 + 3) / 2 = 3.5$$

$$u_B = (1 + 3 + 3) / 2 = 3.5$$

$$u_C = (3 + 3 + 4) / 2 = 5$$

$$u_D = (3 + 3 + 4) / 2 = 5$$

Step 2: Calculate new distance matrix

$$M_{AB} = d_{AB} - u_A - u_B = 1 - 3.5 - 3.5 = -6$$

$$M_{AC} = d_{AC} - u_A - u_C = 3 - 3.5 - 5 = -5.5$$

$$M_{AD} = d_{AD} - u_A - u_D = 3 - 3.5 - 5 = -5.5$$

$$M_{BC} = -5.5$$

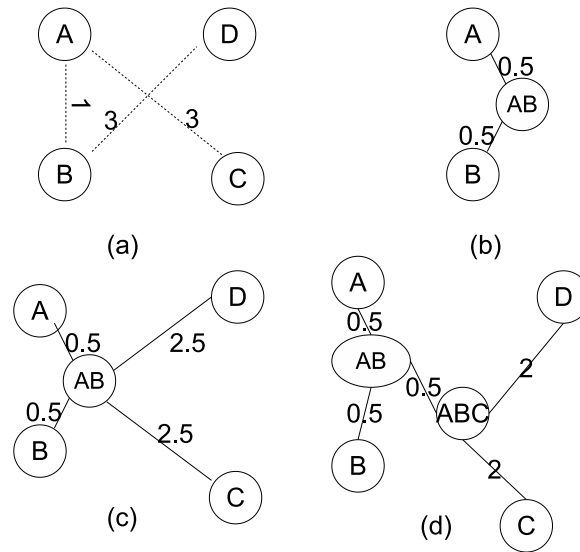


Figure 4.3. Producing NJ tree - not drawn to scale. (a) is initial star tree; (b) shows the combined OTU of A and B; (c) is the NJ tree with the new OTU AB; and (d) is the final NJ evolution tree

$$M_{BD} = -5.5$$

$$M_{CD} = -6$$

Therefore,

$$M = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & & & & \\ B & -6 & & & \\ C & -5.5 & -5.5 & & \\ D & -5.5 & -5.5 & -6 & \end{array}$$

Step 3: Choose the closest neighbors from matrix M. Either AB or CD can be chosen to group into a new OTU since their values are smallest (-6). Let's choose AB as a new OTU. The branch lengths from AB to A and B are:

$$d_{A,(AB)} = (d_{AB} + u_A - u_B)/2 = (1 + 3.5 - 3.5)/2 = 0.5$$

$$d_{B,(AB)} = (d_{AB} + u_B - u_A)/2 = (1 + 3.5 - 3)/2 = 0.5$$

The combination is depicted as (b) in Figure 4.3.

Step 4:

$$d_{(AB),C} = (d_{AC} + d_{BC} - d_{AB})/2 = (3 + 3 - 1)/2 = 2.5$$



$$d_{(AB),D} = (d_{AD} + d_{BD} - d_{AB})/2 = (3 + 3 - 1)/2 = 2.5$$

The new distance matrix is:

d =

	AB	C	D
AB	0		
C	2.5	0	
D	2.5	4	0

The second iteration of the algorithm will produce:

Step 1:

$$u_{AB} = (d_{(AB)C} + d_{(AB)D})/1 = (2.5 + 2.5)/1 = 5$$

$$u_C = (2.5 + 4)/1 = 6.5$$

$$u_D = (2.5 + 4)/1 = 6.5$$

And step 2 gives

$$M_{(AB)C} = d_{(AB)C} - u_{AB} - u_C = 2.5 - 5 - 6.5 = -9$$

Thus, M =

	AB	C	D
AB			
C	-9		
D	-6.5	-6.5	

For step 3, C is chosen to group with AB, and the distances from C and AB to ABC are:

$$d_{C,(ABC)} = (d_{C(AB)} + u_C - u_{AB})/2 = (2.5 + 6.5 - 5)/2 = 2$$

$$d_{(AB),(ABC)} = (d_{C(AB)} + u_{AB} - u_C)/2 = (2.5 + 5 - 6.5)/2 = 0.5$$

Step 4 gives the distance from D to the new cluster ABC as:

$$D_{(ABC),D} = (d_{(AB)D} + d_{DC} - d_{(AB)C})/2 = (2.5 + 4 - 2.5)/2 = 2$$

The algorithm terminates and produces the evolution tree (d) as seen in Figure 4.3.

NJ method has a run time complexity of  $O(n^3)$ .

NJ trees can be converted into rooted trees by estimating the median between the two furthest leaf OTUs.

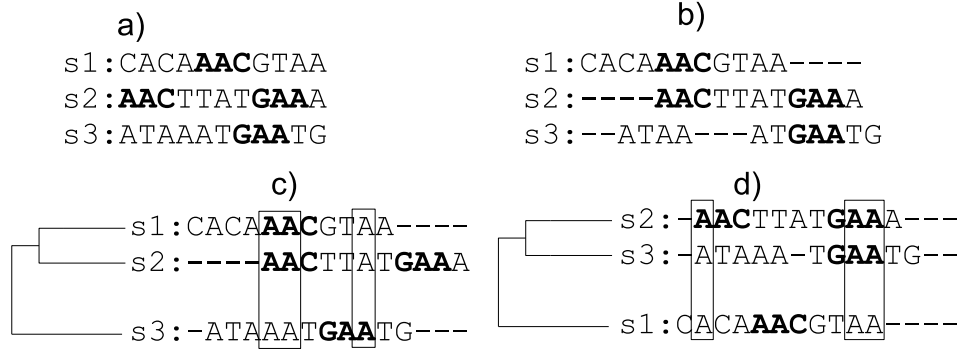


Figure 4.4. This figure illustrates different alignment outcomes from the same input sequences. a) shows the input sequences with the motifs in bold; b) shows the expected alignment; c) and d) show the alignment based on different phylogeny guiding tree.

### 4.3 A new Contribution: Overlapping Sequence Clustering

It is very common to arrange sequences into related groups in sequence analysis. Most of existing clustering algorithms are designed for large scale databases or large data set in which the distance between each data element is known. On contrary, in multiple sequence alignment the distance between sequences is not well defined; and distinctive clusters are not the goal either. Thus, conveniently NJ and UPGMA often are the popular choices for clustering the sequences when needs arise. However, these algorithms are not intended for sequence clustering, but rather predicting the phylogenetic tree of sequences. Alignment techniques following this tree often omit shared features between the sequences since the first pair of sequences dictates the alignment of the next. Figure 4.4 illustrates this concept.

Recently, we have devised a clustering algorithm specifically for multiple sequence alignment called Overlapping Sequence Clustering (OSC) [79]. This technique is based on the facts that an organism can inherit traits from different families. In this case, a sequence from an organism may share similarities with many other sequences. Thus, clustering sequences based on these similarities make more sense than hierarchical sequence clustering. This method uses local DP to find the best pairwise local alignments between sequences. Then from the best scores local alignment, all sequences that yield local alignment scores greater than a preset threshold  $g$  are grouped into a cluster. These clusters can be overlapped. Following are details of this algorithm, which relies on the transitive closure of a set.

The transitive closure is defined as follows:

For any relation  $R$ , the transitive closure of  $R$  is the intersection of all transitive relations containing  $R$  and is defined as:

$$R^+ = \bigcup_{i \in \mathcal{N}} R^i \quad (4.1)$$

where  $R^0 = R$  and, for  $i > 0$

$$R^i = R^{i-1} \cup \{(s_a, s_c) | \exists s_b \wedge (s_a, s_b) \in R^{i-1} \wedge (s_b, s_c) \in R^{i-1}\} \quad (4.2)$$

Similarly, the conditional transitive closure (CTC) is defined with a conditional membership exclusion  $g$ , where  $R^i$  is defined as:

$$R^i = R^{i-1} \cup \{(s_a, s_c) > g | \exists s_b \wedge (s_a, s_b) \in R^{i-1} \wedge (s_b, s_c) \in R^{i-1}\} \cup \{(s_a, s_b) | A(s_a, s_b) \cap A(s_b, s_c) \neq \{\emptyset\} \wedge (s_b, s_c) \in R^{i-1}\} \quad (4.3)$$

Where  $A(s_i, s_j)$  is the local alignment of two sequence  $s_i$  and  $s_j$

The Overlapped Sequence Clustering algorithm is define as follows:

Given a sequence set  $S$ , its local alignment  $A(S)$ , the quantitative relation set  $Q$  between sequences and a membership function (or threshold)  $g$ :

OSC Algorithm( $S, Q, g$ ):

Step 1: Sort the relations in ascending order.

Step 2: Let  $F = \{\emptyset\}$  – cluster set

Step 3:  $C = \{s_i, s_j\}$

where  $s_i, s_j \in S$  and  $(s_i, s_j) \in Q$  and  $(s_i, s_j) > g$  and  $(s_i, s_j) > \forall (s_k, s_l) \in Q$ , and  $i, j \neq k$

Step 4: Find CTC for  $C$

Step 5:

If  $C \neq \{\emptyset\}$  and  $|\{S\}| > 1$

$F = F \cup \{(C)\}$

$S = S - \{C\}$

Repeat step 2

Else:

Reducing cluster overlap by removing cluster membership of sequences that share the same local alignment segment with more than one clusters.

Step 6:

output  $F \cup (S)$  as sequence cluster set

End

The membership function, or threshold  $g$ , can be arbitrary. It can simply be the  $k^{th}$  best score of the input. This algorithm starts with a pair of sequences that yields the highest local alignment score, or shortest distance, as a cluster. It then iteratively finds all other sequences that locally aligned with any sequence in the cluster that: (i) yield scores better than the predefined threshold  $g$  or (ii) have alignment segments overlapped with a local alignment segment of any sequence in the cluster. This process is repeated until no more clusters found. The remaining un-clustered sequences are grouped into a special cluster. The (ii) condition extends the cluster membership to a sequence with low score but share a highly conserved segment to join a cluster. However, this is an exclusive relationship and a sequence can only be a member of one cluster. The cluster overlap reducing step enforces this requirement. This fast technique has a runtime complexity of  $O(kn)$ , where  $n$  is the number of sequences to be clustered and  $k$  is the number of clusters.

Example:

Given sequences set  $S = \{A, B, C, D, E, F, G, H, I\}$ , threshold  $g = 7$ , and the following score matrix:

	A	B	C	D	E	F	G	H	I
A									
B	<b>11</b>								
C	5	4							
D	9	8	7						
E	7	5	5	2					
F	6	7	2	1	9				
G	6	4	4	8	10	8			
H	5	6	3	9	<b>13</b>	9	8		
I	9	8	1	9	7	6	7	6	

The first iteration starts with  $C = \{E, H\}$  for  $(E, H) = 13$  is the highest score. A breadth first search (BSF) from  $C$  adds  $\{E, G\}$ ,  $\{E, F\}$ , and  $\{H, D\}$  by following  $E$ , and  $H$ , respectively. Thus, we will have a cluster of  $\{D, E, F, G, H\}$ ,  $S = \{A, B, C\}$ , and  $F = \{\{D, E, F, G, H\}\}$ .

The second iteration starts with  $C = \{A, B\}$  for  $(A, B) = 11$  is the highest score in this iteration. A BFS from  $C$  adds  $\{A, I\}$  and  $\{D, I\}$ . Thus,  $F = \{\{A, B, D, I\}, \{D, E, F, G, H\}\}$ , and  $S = \{C\}$ .

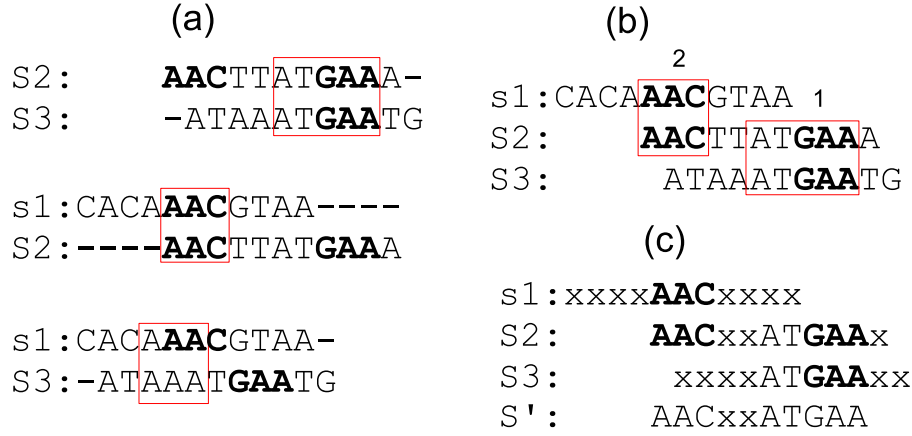


Figure 4.5. This figure illustrates the clustering of the sequences to generate a sequence pattern. (a) is the all-pairwise alignments, (b) is sequence clusterization, and (c) is sequence pattern identification, where  $S'$  is the cluster pattern. The boxes represent the DP local alignment results.

Since (C) is the only sequence left in  $S$ , the algorithm terminates with three clusters  $\{A, B, D, I\}$ ,  $\{D, E, F, G, H\}$  and  $\{C\}$ , where sequence  $D$  is a member of two clusters.

If we enforce the condition that a new member must have all relationships to all other members in the cluster greater than the threshold, then  $D$  would be disjoined from the first cluster. This condition increases the run time complexity of the algorithm, however, it strengthens the relationships in each cluster, i.e., members are closer. In the worst case where all  $n$  sequences overlap in  $n$  clusters, the run-time complexity of this clustering algorithm is  $O(n^3)$ ; Hence, the run-time complexity of the phylogeny estimation is  $O(n^4)$ .

Figure 4.5 shows the application of OSC algorithm on the sequence in 4.4. It also shows a sequence pattern generated from the cluster. This pattern can be used as a key to search databases for annotated sequences with similar pattern. The  $x$  symbol represents a wildcard, which matches any other symbol.

The effectiveness of this clustering algorithm is illustrated in Chapter 5 where it is implemented in our newly developed multiple sequence alignment algorithm.

It is possible to create an optimal phylogenetic tree, or a maximum parsimony tree, from a set of sequences; however, this problem is NP-Complete [42]. Therefore, heuristic clustering and partitioning seem to be the only realistic option. In the next chapter, we will investigate many popular multiple sequence alignment algorithms.

## Chapter 5

### MULTIPLE SEQUENCES ALIGNMENT ALGORITHMS

Multiple sequence alignment (MSA) is the extension of pair-wise sequence alignment as discussed in Chapter 2 in which the number of sequences to be aligned are more than two. [Figure 5.1 shows an alignment of BALiBASE [115] reference 1 ubiquitin set]. With this extension, the MSA problem becomes intractable, in fact, finding the optimal solution for multiple sequence alignment is an NP-complete problem as proved by Wang and Jiang in 1994 [123]. Thus, any attempt to develop a practical algorithmic method to find a mathematical optimal solution for this problem is infeasible. To overcome this problem, various heuristic approaches have been proposed leading to a large number of multiple sequence alignment programs based on different strategies such as progressive, iterative, genetic, probabilistic, or hybrids of these methods. Each strategy focuses on one or few features such as speed, sequence local similarity, sequence overall similarity, sequence structure similarity, etc. In 1999 Thompson et. al. [116] performed a comprehensive comparison of the ten most popular used MSA tools currently available (PIMA [SBPIMA and MLPIMA], MUTAL, MULTIALIGN, PILEUP, CLUSTALX, DIALIGN, SAGA, HMMT and PRRP) at the time. As expected, the tests were not able to distinctively identify the best tool. ClustalX, a progressive MSA were fastest of all with an average reliability. Each method performs well on different reference set of sequences. PRRP tends to perform well on some reference sets containing few sequences in the twilight zone, i.e. sequences that share less than 25% identity.

There are many multiple sequence alignment algorithms have been proposed, many of them are slightly different from each other. In this chapter, only distinctive multiple sequence alignment paradigms are introduced.

```

1ubi      MQIFVKTLTGKTITLEVEPSDTIENVKAKIQDKEGIPPD-----QQR
1guaB     NTIRVFLPNKQRTVVNVRNGMSLHDCLMKALKVRGLQPEC-----CAV
1alo      MIQKVITVNGIEQNLFVDAEALSDVLRQQLGLTGKVGKVGCEQGQCGACSV
1awd      YKVTCLKTPSG-EETIECPEDTYILDAAEEA-GLD-LPYSCRAGACSSCAG

```

Figure 5.1. Sample alignment of BALiBASE reference 1 [115] ubiquitin set. Red blocks represent alpha helices and green blocks represent beta strands

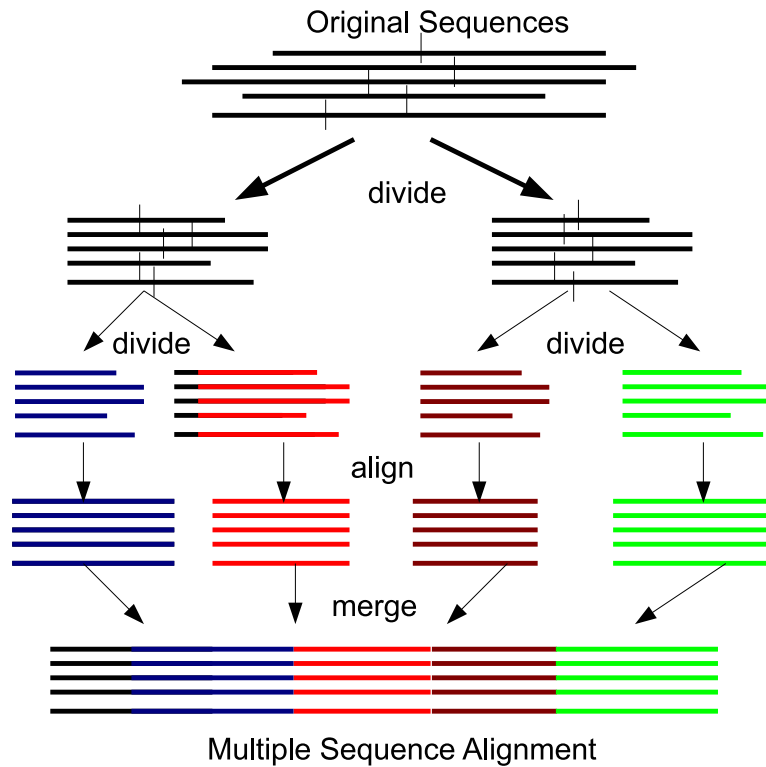


Figure 5.2. Illustration of the divide and conquer multiple sequence alignment (DCA)

## 5.1 Dynamic Programming

Multiple sequence alignment algorithms in this group are extensions of dynamic programming.

### 5.1.1 DCA

Divide and Conquer multiple sequence Alignment (DCA) algorithm is designed by J. Stoye [109] for his dissertation. The basic idea of DCA is rather simple: each sequence is split into two sub-sequences at location near the midpoint. The splits partition the problem into two sub-MSA problems with shorter sequences. This process is repeated until the sequences become sufficiently short, i.e. shorter than a predefined threshold  $L$ . These sub-multiple sequence alignments are then optimally aligned by dynamic programming method. The alignments of these sub-problems are then concatenated yielding a multiple sequence alignment for the original sequences. This algorithm is illustrated in Figure 5.2.

**Determine the cutting position** The most important task of DCA is how to identify the location for splitting each sequence. A perfect set of cuts could lead to an optimal solution. DCA uses a heuristic

based on additional-cost matrices to quantify the compatibility of the cut positions. The cut position is calculated as follows:

For any given sequence  $s = s_1 s_2 \cdots s_n$  of length  $n$  and a cut position  $c (0 \leq c \leq n)$ , the prefix sequence before the cut position  $c$  is denoted as  $s(\leq c)$  and the suffix is  $s(> c)$ . Dynamic Programming is used to compute the additional-cost of  $C_{s_p, s_q}[c_p, c_q]$  for all pair of sequences  $(s_p, s_q)$  and for all cut positions  $c_p$  of  $s_p$  and  $c_q$  of  $s_q$ . The additional-cost is defined as:

$$C_{s_p, s_q}[c_p, c_q] = w_{opt}(s_p(\leq c_p), s_q(\leq c_q)) + w_{opt}(s_p(> c_p), s_q(> c_q)) - w_{opt}(s_p, s_q) \quad (5.1)$$

this is the extra cost imposed by requiring the alignment of  $s_p$  and  $s_q$  to optimally align the two prefixes and suffixes of the sequences rather than the sequences themselves. To extend the cut over all sequences, the multiple-additional-cost is defined as follows:

$$C(c_1, c_2, \dots, c_k) = \sum_{1 \leq p \leq q \leq k} \alpha_{p,q} C_{s_p, s_q}[c_p, c_q] \quad (5.2)$$

where  $\alpha_{p,q}$  is the sequence dependent weight factor. The dependent weight factor is used with the sum-of-pairs score [64] to increase the weight (or score) of a matching pair of residues. It is defined as:

$$\alpha_{p,q} = \begin{cases} 1 & \text{if } \maxScore = 0 \\ 1 - \lambda \frac{\overline{w_{opt}}(s_p, s_q) - \minScore}{\maxScore} & \text{otherwise} \end{cases} \quad (5.3)$$

where  $\minScore$  and  $\maxScore$  are the lowest and highest pair-wise scores of all the columns (positions) in the alignment of sequences  $s_p$  and  $s_q$ , respectively, and  $\overline{w_{opt}}(s_p, s_q)$  is the optimal alignment score of  $s_p$  and  $s_q$ , i.e. score obtained by aligning the sequences via dynamic programming algorithm.

The use of dynamic programming for aligning sequence fragments in DCA increases its processing time exponentially preventing DCA to align more than few sequences ( $< 8$  sequences). A faster version of DCA is FDCA [91] which approximates the cut positions to reduce DCA search space. The approximation is done by preempting any calculation of a tuple of cutting points whenever its partial sum is larger than the minimum found so far. This technique extends the alignment capability of DCA up to 9 sequences.

In general, the suboptimal heuristic cuts often lead to unacceptable alignment results. Nevertheless, DCA is probably the closest algorithm that directly implementing dynamic program technique.



## 5.2 Progressive Alignment

Progressive multiple sequence alignment is introduced by Feng and Doolittle in 1987 [32]. The main idea is that a pair of sequences with minimum edit distance is most likely to originate from a recently diverged species. Thus, optimally aligned these sequences may review the most reliable hidden information. The algorithm is as follows: (i) Calculate pair-wise alignment score and convert them into distances. (ii) Construct a dendrogram, or a guiding tree, from the distances. UPGMA and NJ clustering methods are suitable for this task. (iii) Sequentially align the sequences in their order of addition to the tree. Gap insertions in pair-wise alignments are preserved. A number of alignment programs based on this technique such as: CLUSTALW [117, 118], MULTALIN [21], PILEUP [created by Genetics Computer Groups(GCG), which is commercially known as Accelrys], or PIMA. The different between these programs are minimal. For example, PIMA [106] uses local DP for pair-wise alignments and WPGMA to build the guiding tree, while others use global DP. MULTALIN and PILEUP use UPGMA method to construct their guiding trees, while CLUSTAL uses NJ method (earlier versions of CLUSTALW used UPGMA), and KALIGN [59] uses Wu-Manber [125] and Levenshtein edit distance to calculate distance matrix. Figure 5.3 shows an example of progressive alignment.

The most advantage of progressive alignment algorithms is their capability of aligning a large number of sequences. However, they do not give optimal solution since sequences are iteratively aligned in pair-wise following the order of a guiding tree. Errors made in early stage of alignment will be propagated through the final result.

### 5.2.1 Clustal Family

The most popular program in this group is the CLUSTAL family (including ClustalX and ClustalW) [118]. It is fast and relatively reliable. Like all other progressive alignments, CLUSTAL starts by pair-wise aligning all sequences via global DP with either PAM250 or BLOSUM62 (or any substitution matrix chosen by the user). It guarantees the distances between two sequences are mathematically optimal. CLUSTAL then uses these pair-wise alignment scores to build a neighbor joining tree (NJ) - the early version of CLUSTAL uses UPGMA method. The sequences are then aligned following the tree from the leaves inward. There are some parameters in CLUSTAL that make it more sensitive than other are the gap penalties. For example, CLUSTAL assumes that short stretches of hydrophilic residues (e.g. 5 or more) is an indication loops or random coil regions so it reduces the gap opening penalty for these stretches. Similarly, it increases the opening gap penalty for any gap that are less than 8 residues apart based on the

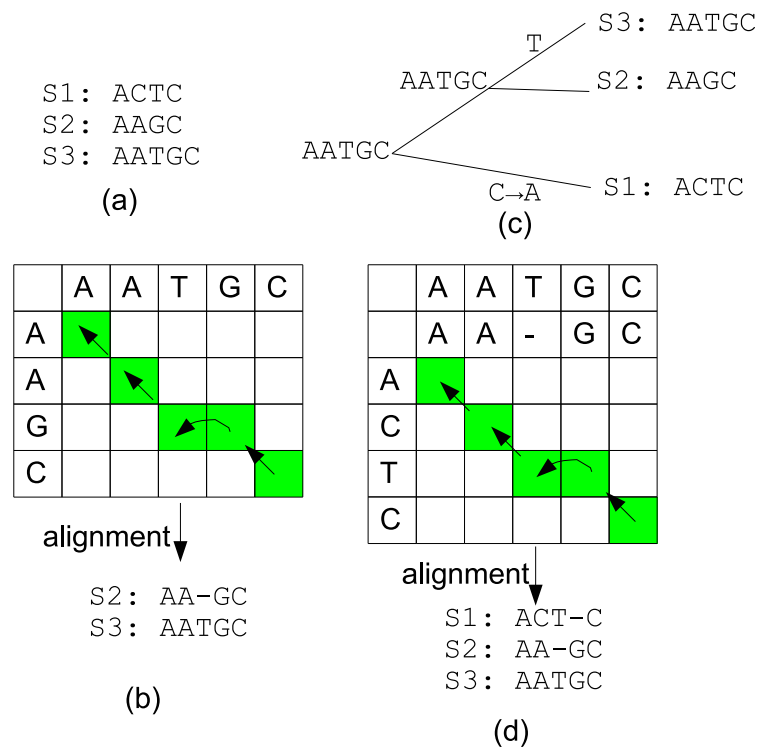


Figure 5.3. An example of progressive multiple sequence alignment. (a) is the input sequences, (b) is an alignment guiding tree, (c) is external nodes alignment, and (d) is internal nodes alignment

observation of alignments between sequences of known structures, which rarely has a gap within 8-residue segments. The initial gap opening penalty is:

$$GOP + \log(\min(n, m)) \times \overline{s(a, b)} \times ID\%, \quad (5.4)$$

where  $GOP$  is the user defined opening gap penalty,  $m$  and  $n$  are sequence lengths,  $\overline{s(a, b)}$ ,  $a \neq b$  is the average mismatched score between any two residues, and  $ID\%$  is the percentage identity scaling factor.  $ID\%$  is the ratio between the numbers of mismatched pairs over the total number of residue pairs.

The extended gap penalty is defined as:

$$GEP \times (1.0 + |\log(\frac{n}{m})|) \quad (5.5)$$

where  $GEP$  is the user defined extended gap cost.

### 5.2.2 PIMA: Pattern-induced multi-sequence alignment

PIMA [105, 106] applies the progressive technique to align protein sequences. The main difference between PIMA and other algorithms is the generation and usage of sequence patterns to align groups of sequences from the internal nodes of the guiding tree. Initially, PIMA performs pair-wise local DP on input sequences to obtain a distance matrix between all the sequences. A dendrogram (guiding tree) is then built from the distance matrix using WPGMA method [108] (Weighted Pair Group Method using Arithmetic averaging). PIMA then progressively aligns pairs of sequences from the leaf level of the tree. A sequence pattern is generated for each pair-wise alignment using the amino acid class covering (AACC) to represent the residues in each column. Thus, aligning any two nodes in PIMA is always a pair-wise alignment of two sequences; either they are the actual input sequences, a sequence and a pattern or a pair of patterns representing two pair-wise alignments. Pattern usage transforms the multiple sequence alignment into a sequence of pair-wise alignments. The amino acid symbol patterns are shown in Figure 5.4, where X represents a wild-card that matches any symbol, and [DE] represents either D or E, etc. Transformation of a column into a pattern symbol is done as: (i) same symbols remain the same; (ii) different symbols are replaced by the minimal covering set symbol; (iii) mixture of gap and other symbol is replaced by  $g$ , a gap symbol. Figure 5.5 shows sequence pattern generated from a pair-wise alignment of two sequences, where a gap is represented as  $g$ .



### 5.2.3 PRIME: Profile-based Randomized Iteration Method

PRIME [127] is an extension of progressive where input sequences are progressively aligned as in CLUSTAL. Next, a distance matrix is generated from the multiple alignment result to construct a phylogenetic tree. A random branch of the tree is cut to partition the sequences in the alignment into two groups. Then Needleman-Wunsch's dynamic programming is used to align these two groups - in this case, two columns being compared rather than two residues. This process is repeated until convergent, i.e. no better alignment score found. In its earlier version, PRRN [40], affine gap penalty and anchoring technique are employed. The anchoring technique keeps conserved columns untouched during progressive alignment, and the group-to-group alignments are done on segments around the anchors.

The group-to-group gap opening penalty is defined as:

$$g(a_i, b_j) = \sum_{1 \leq p \leq m} \sum_{1 \leq q \leq n} w_{p,q} \cdot (-v) \cdot \gamma(a_{p,i}, b_{q,j}) \quad (5.6)$$

where  $a_i$  and  $b_j$  are the column  $i^{th}$  of group  $a$  and column  $j^{th}$  of group  $b$ ;  $w_{p,q}$  is the weighted sum-of-pair score of two sequences represented as rows  $p$  and  $q$ ;  $m$  and  $n$  are the numbers of sequences in the two aligning groups (same as the number of rows in the two columns  $a_i$  and  $b_j$ ),  $p$  and  $q$  are the  $p^{th}$  and  $q^{th}$  rows in the aligning column;  $v$  is a constant opening gap cost (a positive value); and  $\gamma(a, b) = 1$  if either  $a$  or  $b$  is a gap "-", otherwise,  $\gamma(a, b) = 0$ .

### 5.2.4 DIALIGN

Similar to T-COFFEE, DIALign [74] combines both global pair-wise alignments and local pair-wise alignment features. Multiple sequence alignments of DIALign are composed of equal-length segments pairs that exhibit statistical significant similarity. The segments are obtained from local pair-wise alignments. This technique is similar to FASTA alignment. For distantly related sequences, DIALign use global alignment method similar to Needleman's algorithm to align them. For mixture of related and unrelated sequence, DIALign aligns only segments of sequences that are statistical significant. A greedy approach is used to search and align similar segments across the sequences. Gaps are not penalized in DIALign, and segments statistical insignificant are not aligned.

**Tabu Search** Tabu search [35, 38] is an iterative heuristic search scheme that is capable of solving combinatorial optimization problems. Tabu search is deterministic and capable of avoiding local optima

by keeping a list of prohibit solutions, or a tabu list. The Tabu Search for multiple sequence alignment [97] is carried out in two phases.

Phase 1: input sequences are progressively aligned to provide the initial solution for the search. The tabu MSA then moves the gap regions, either locally in a sequence or across multiple sequences, to neighboring columns to generate a new alignment. The alignment is then evaluated with T-COFFEE objective function and added into the tabu list preventing tabu search from repeating the same move. The highest score alignments in tabu list are remove after  $x$  number of iterations so they can be back in the solution space. The moves are repeated until convergent, i.e. no new solution with worse score than the tabu list found, or  $k$  iterations have been performed.

Phase 2: the steps in this phase are similar to those in phase 1 with a modification so that the best moves are kept in the potential solution list, or elite list. The solution is then obtained by applying the moves from the elite list.

$$Score = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^N W_{ij} * Score(A_{ij})}{\sum_{i=1}^{N-1} \sum_{j=i+1}^N W_{ij} * Len} \quad (5.7)$$

where  $N$  is the number of sequences;  $Len$  is the length of the alignment;  $w_{ij}$  is the percent identity between two aligned sequences  $S_i$  and  $S_j$ ;  $A_{ij}$  is the pair-wise projection of sequences  $S_i$  and  $S_j$  obtained from the multiple alignment; and  $Score(A_{ij})$  is the overall level of identity between  $A_{ij}$  and the corresponding pair-wise alignment

### 5.3 Consistency and Probabilistic MSA

Hidden Markov Models (HMMs) are used as a statistical models of the sequence family primary structure consensus as in [7, 58]. The probability that a region being a codon is calculated as:

$$Prob(c_1, \dots, c_k) = \prod_{i=1}^k p(c_i)$$

where  $p(c_i)$  is the probability of codon  $c_i$ . The codon probability is the relative frequencies of the 64 codons from a DNA sequence database. Hidden Markov Model (HMM) is a finite state machine with the triple  $(A, B, \Pi)$ , where  $A$  is the transition probabilities,  $B$  is the output probabilities, and  $\Pi$  is the initial state probabilities. For any given time  $t \geq 1$ ,  $A, B$ , and  $\Pi$  are defined as follows:

$A = \{a_{ij} = P(q_j \text{ at } t+1 | q_i \text{ at } t)\}$ , where  $P(a|b)$  is the conditional probability of a given  $b$ ,  $t \geq 1$  is time, and  $q_i \in Q$

$B = \{b_{ik} = P(o_k | q_i)\}$ , where  $o_k \in \Sigma$ , i.e.  $B$  is the probability that the output is  $o_k$  with given state  $q_i$ .

And  $\pi$  is calculated as:

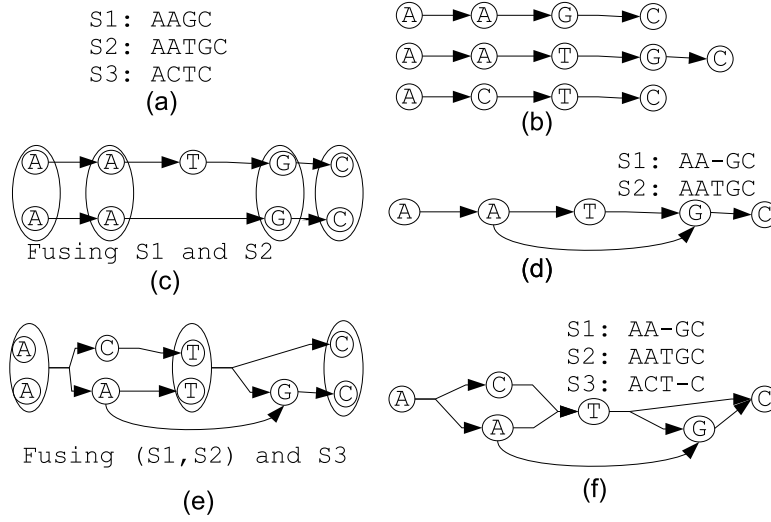


Figure 5.6. POA algorithm: (a) input sequences, (b) directed graphs for the sequences, (c) graph fusion of sequence S1 and S2, (d) alignment of S1 and S2, (e) graph fusion of S3 to alignment (S1, S2), (f) final partial order graph and final multiple sequence alignment

$$\Pi = \{p_i = P(q_i \text{ at } t = 1)\}.$$

### 5.3.1 POA: Partial Order graph Alignment

Generally, HMM-based methods represent an MSA as a directed acyclic graph known as partial-order graph. In this representation, same symbols in each column are coded as a node in the graph. Each node has  $k$  outgoing edges to all distinct symbols in the next column. In terms of Markov model, the observed states are the individual alignment columns. An efficient version of dynamic programming, called Viterbi algorithm, is used to successively align the sequences to grow the MSA. This step is similar to progressive alignment. However, the use of a graph allows the earlier alignment to be updated. POA and PSAlign [43, 110] are developed based on this technique. In POA, each sequence is converted into a directed graph. The graphs are then combined on identical residue symbols. This process is repeated until all graphs are merged together. Figure 5.6 illustrate each step of POA.

### 5.3.2 PSAlign

Similarly, PSAlign [110] is a polynomial time solvable multiple sequence alignment program developed based on a finding the shortest preserving alignment. This method starts by pair-wise alignment the sequences similar to those in T-Coffee [83] and Probcons [25] to incorporate the sequence consistency into the pair-wise scores. Instead of using NJ or UPGMA algorithms to build the guiding tree, it simply builds

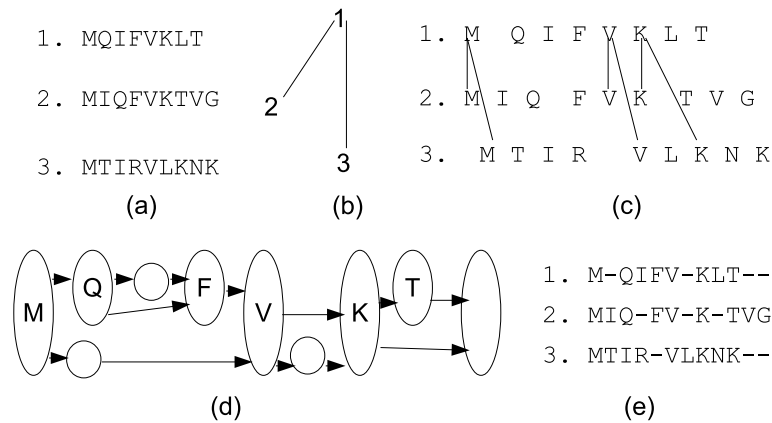


Figure 5.7. PSAlign algorithm: (a) input sequences, (b) minimum spanning tree, (c) undirected graph constructed from pair-wise alignments and spanning three, (d) partial order graph, (e) final alignment result

the minimum spanning tree. An undirected graph is built based on the tree where the leaf nodes are the residues in the sequences. An edge is added between two residues of two sequences if they are aligned in the previous pair-wise alignment step. The next step is to find the optimal alignment of the topological partial order graphs. Figure 5.7 illustrate this algorithm.

The main different between PSAlign and POA is the use of pair-wise alignments while generating the partial order graphs. The advantage of employing partial order graph technique is the speed. It has been shown that POA can align 5000 sequences in 4 hours on a Pentium II computer [43].

### 5.3.3 ProbCons: Probabilistic consistency-based multiple sequence alignment

ProbCons [25] algorithm is based on the probabilistic consistency of the sequences. It is a pair-hidden Markov model-based progressive alignment algorithm that uses Needleman-Wunsch's algorithm with no gap penalty (maximum expected accuracy) rather than traditional Viterbi algorithm. The emission probability of a residue is based on BLOSUM62 substitution matrix and the transitional probabilities, which is corresponding to the gap penalty. The transitional probabilities obtained by unsupervised expectation maximization (EM) [28] method.

Given the input sequence set  $S$ , the algorithm is as follows

Step 1: For every pair of sequences  $x$  and  $y \in S$ , a posterior probability matrix represents the probabilities of two residues, one from sequence  $x$  and one from sequence  $y$ , are paired in an alignment, and it is



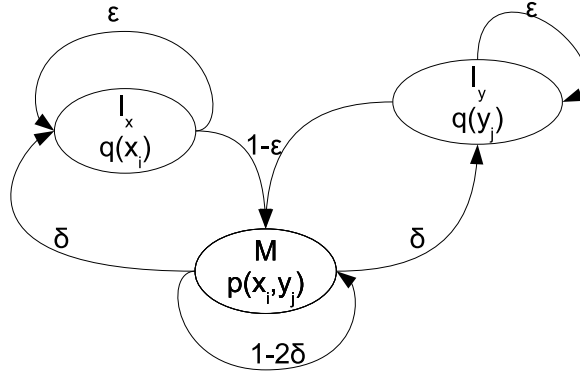


Figure 5.8. Basic pair-HMM for aligning two sequences  $x$  and  $y$ . State  $M$  emits two letters that being aligned from each sequence. State  $I_x$  and  $I_y$  each emits a letter in sequence  $x$  and  $y$  that are aligned to a gap.

calculated as follows:

$$P_{x,y} = \mathbf{P}(x_i \sim y_j \in a * | x, y) \quad (5.8)$$

where  $x_i$  and  $y_j$  are residues at location  $i$  and  $j$  in the sequences, and  $a*$  is a pair-wise alignment of  $x$  and  $y$  generated by the model.

Step 2: The expected accuracy of an alignment  $a$  is defined to be the expected number of correctly aligned pairs of residues, divided by the length of the shorter sequence:

$$E_A(\text{accuracy}(a, a*) | x, y) = \frac{1}{\min|x|, |y|} \sum_{x_i \sim y_j \in a} \mathbf{P}(x_i \sim y_j \in a * | x, y) \quad (5.9)$$

It then computes the alignment  $a$  that maximizes the expected accuracy by dynamic programming, and set  $E(x, y) = \mathbf{E}_{a*}(\text{accuracy}(a, a*) | x, y)$ .

Step 3: The probabilistic consistency transformation is applied to all pair-wise comparisons to estimate the match quality scores  $\mathbf{P}(x_i \sim y_j \in a * | x, y)$ , which is calculated as:

$$P'_{xy} \leftarrow \frac{1}{|S|} \sum_{z \in S} P_{xz} P_{zy} \quad (5.10)$$

step 4: Construct a guide tree for the sequences through hierarchical clustering such as NJ or UP-GMA. Alignments are scored using the sum-of-pair method and  $P'$  matrix.

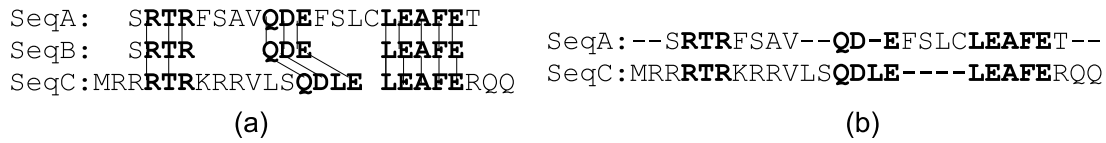


Figure 5.9. Generating extended pair-wise alignment in T-Coffee. (a) sequence SeqA is aligned to sequence SeqC via immediate sequence SeqB. (b) new pair-wise alignment for sequence SeqA and sequence SeqC

Step 5: progressively align the sequences specified by the tree order. Gap penalty is set to zero.

Step 6: Randomly partition the alignment into two groups of sequences and realign until convergent.

### 5.3.4 T-COFFEE: Tree-based Consistency Objective Function For alignment Evaluation

T-COFFEE [83] is a progressive multiple sequence alignment. It starts by using ClustalW and LALIGN [48] (LALIGN is a fast version of Smith and Waterman's algorithm - LALIGN has two versions NAlign and PAlign, one for protein sequences and one for nucleotide sequences) to generate a primary pair-wise sequence alignment library that combines both global pair-wise alignment (ClustalW) and local pair-wise alignment (LALIGN). Duplicate pairs are removed and remaining pairs of the duplicates get double weights. An extended library is built based on triplets of the primary library where a new pair-wise alignment is created if two sequences aligned via the third sequence, as seen in Figure 5.9. The extended library is a list of weighted pair of residues. From the extended library, progressive alignment is performed to generate the final multiple sequence alignment. T-Coffee algorithm is illustrated in Figure 5.10. 3D-COFFEE [89] is an extended version of T-COFFEE in which the three dimension structure alignment of every sequence pair is obtained from server [<http://www.cryst.bioc.cam.ac.uk/fugue/>], and the superposition, [locations where two structures overlapped], of the two sequences are used to increase the weights of residues pair in the extended library.

**Example** Using the same sequences in Figure 5.9:

sequence a : SRTRFSAVQDEFSLCLEAFET

sequence b: SRTRQDELEAFE

sequence c: MRRRTRKRRVLSQDLELEAFERQQ

Assuming the alignment score of {a,b} is 88, {a,c} is 100 and {b,c} is 77. The triplet weight is calculated

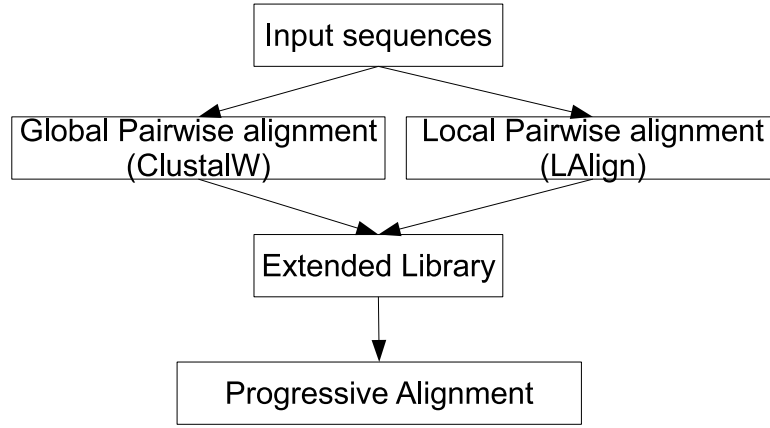


Figure 5.10. T-Coffee multiple sequence alignment schema

as: aligning of  $a_2 = R$  to  $c_4 = R$  gets a weight of 100, and aligning of  $a_2 = R$  to  $c_4 = R$  through  $b_2 = R$  gets a weight of  $\min(\{a, b\}, \{b, c\}) = \min(88, 77) = 77$ . These mentioned residues are displayed in bold.

### 5.3.5 MAFFT: MSA based on Fast Fourier Transform

MAFFT [57] is a time efficient progressive multiple sequences based on fast Fourier transform (FFT) method. The general idea of fast Fourier transform is to find peaks in data and then use matrix transformation to combine all the peaks into a matrix form. The transformation is similar to matrix multiplication where only the peak regions remain. Applying to the multiple sequence alignment problem, fast Fourier transform greatly reduces the solution space and speedups the process. The MAFFT technique is as follows: Each nucleotide  $a$ , or amino acid, is represented as a normalized vector whose components are the volume value  $v(a)$  and polarity value  $p(a)$  as:  $\hat{v}(a) = [v(a) - \bar{v}]/\delta_v$  and  $\hat{p}(a) = [p(a) - \bar{p}]/\delta_p$ , where overbar represent the average over the number of different symbols (20 for protein and 4 for DNA/RNA),  $\delta_v$  and  $\delta_p$  are the standard derivation of polarity and volume.

The correlation between two sequences is defined as :

$$c(k) = c_v(k) + c_p(k) \quad (5.11)$$

for any pair of sequences,  $c_v(k)$  and  $c_p(k)$  are:

$$c_v(k) = \sum_{1 \leq n \leq N, 1 \leq n+k \leq M} \hat{v}_1(n) \hat{v}_2(n+k) \quad (5.12)$$

$$c_p(k) = \sum_{1 \leq n \leq N, 1 \leq n+k \leq M} \hat{p}_1(n) \hat{p}_2(n+k) \quad (5.13)$$

where  $N$  and  $M$  are the lengths of the sequences. In FFT form,  $c_v(k)$  is represented as  $c_v(k) \Leftrightarrow V_1 * (m) \cdot V_2(m)$  where  $V_i$  is the FFT transformation of  $\hat{v}_i(m)$ ,  $\Leftrightarrow$  denotes transformed pairs, and the asterisk represents complex conjugation.

MAFFT uses a windows size of 30 residues to scan every pairs of sequences for homologous segment pairs, i.e. the peaks in  $c(k)$  is greater than 0.7. Dynamic programming is then used to optimally align these segments. This technique is extended two groups of sequences to align multiple sequences. The vector for the group is defined as:

$$\hat{v}_{group1}(n) = \sum_{i \in group1} w_i \cdot \hat{v}_i(n) \quad (5.14)$$

and

$$\hat{p}_{group1}(n) = \sum_{i \in group1} w_i \cdot \hat{p}_i(n) \quad (5.15)$$

where  $w_i$  is the weighting factor for sequence  $i$  as in CLUSTALW.

The substitution matrix used in MAFFT is a normalized of PAM200, which is calculated as:

$$\hat{M}_{ab} = [(M_{ab} - average2)/(average1 - average2)] + S^a \quad (5.16)$$

where  $M_{ab}$  is PAM200,  $average1 = \sum_a f_a M_{aa}$ ,  $average2 = \sum_{a,b} f_a \cdot f_b \cdot M_{ab}$ ,  $f_a$  is the frequency of occurrence of symbol  $a$ , and  $S^a$  is gap extension penalty.  $S^a$  is 0.06,  $f_a$  is 0.25 and  $S^{op}$  is 2.4 for gap opening penalty.

### 5.3.6 AVID

AVID [13] is an iterative alignment approach using anchoring technique as seen earlier. It used Smith and Waterman's algorithm to find maximal local alignment segments. These segments are used to build a suffix tree, and the maximal repeated segments are found by solving the suffix tree problem [44]. These segments are then used as anchors. This technique is repeated for segments around the anchors.

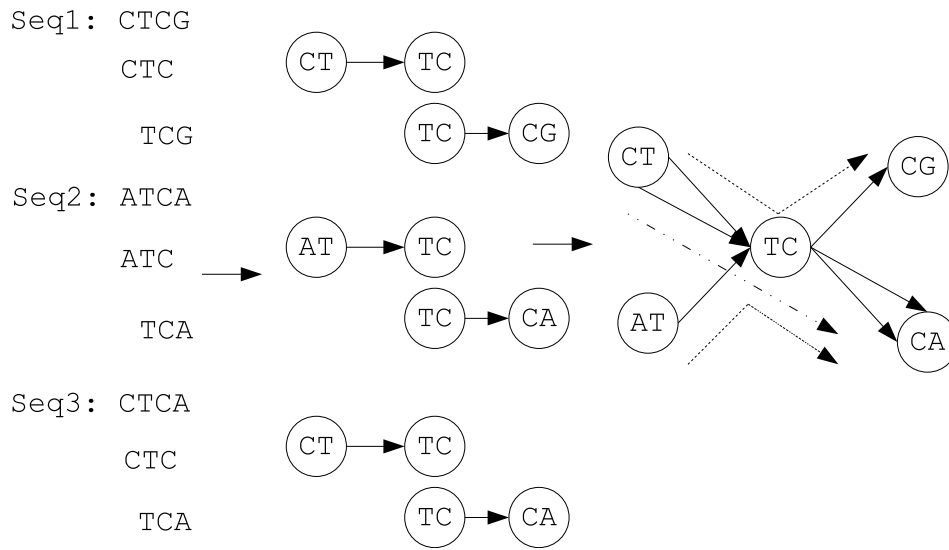


Figure 5.11. Representing 3 sequences in de Bruijn graph

### 5.3.7 Eulerian path MSA

The Eulerian path method [129] maps segments of sequences into nodes of a connected graph and find the Eulerian path through the graph nodes. In this method, a window size  $k$  is used to generated a list of words from each sequence similar to those in FASTA. The words are then used to build a de Bruijn graph, (as in Figure 5.11), which is a directed graph representing sequences of symbols. This graph has  $m^k$  vertices, where  $m$  is the number of symbols in the sequences and  $k$  is the word size. Each vertex has exactly  $m$  incoming and  $m$  outgoing edges. The graph is then transformed into a new graph with superpaths [92]. A superpath is a path equivalent to all sub-paths going from vertex  $x$  to vertex  $y$ .

Figure 5.12 illustrates the transformation of sub-paths into superpaths, where  $P_{x,y}$  in (a) represents superpath of  $P_x$  and  $P_y$  after detachment of path  $xy$ ; and (b) shows x-cut. The superpath transformation eventually shortens all paths from  $x$  to  $y$  to a single path. The alignment is obtained by finding the Eulerian path for the transformed graph.

## 5.4 Genetic Algorithms

Alignment methods in this group revolve around the genetic algorithm [37, 46] to solve the multiple sequence alignment problem. The GA algorithm mimics the natural selection of the nature where the most fitness species have better chance of survive, thus generating more offspring. There are many implementations such as GA [17], GA-DP [128], SAGA [84], RBT-GA [111], GA-ACO [61], and in [14, 67, 78],etc.

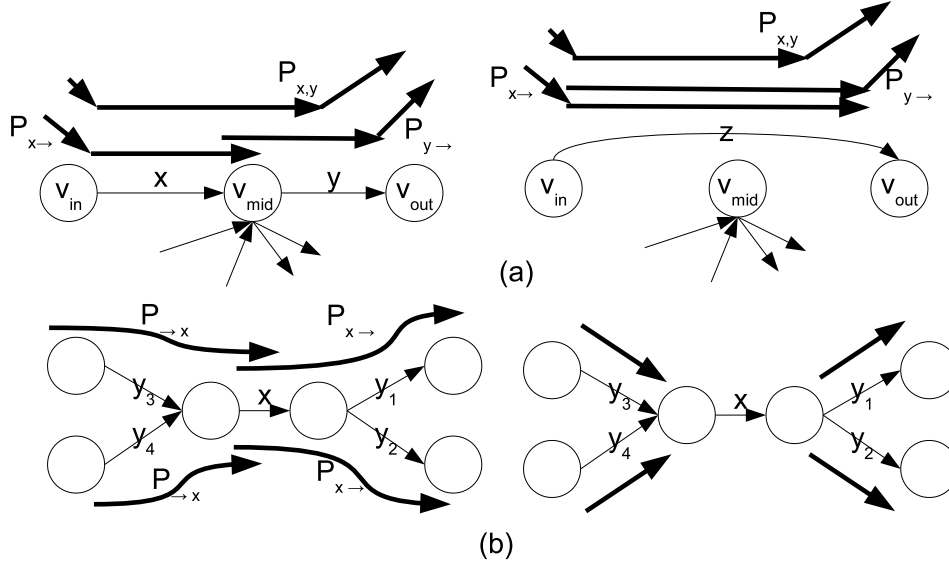


Figure 5.12. Superpath transformation: (a) detachment and (b) x-cut

#### 5.4.1 SAGA: sequence alignment by genetic algorithm

SAGA [84] optimizes its multiple sequence alignment by applying genetic algorithm (GA) [37, 46]. The SAGA algorithm starts with an initial population containing some random alignments of the sequences. These initial alignments represent the first generation,  $g_0$ , of the alignments. In generation  $n, n > 0$ , SAGA ranks each member in the population using the sum-of-pair method as its objective function and applies the GA algorithm to select pairs of parents from generation  $G_{n-1}$  to create new offsprings for the  $n^{th}$  generation. A pair of parents with a good fitness score, based on the objective function, is allowed to have more offsprings. The offsprings are created from any combination of crossover, gap insertion and block shuffling techniques. The crossover allows blocks of the parents to be swapped as in Figure 5.14; the gap insertion method creates a child by inserting gaps into a parent alignment; and the block shuffling allows a block of gaps to be shuffled to its left or right locations. The population is kept constant by replacing the population members with new offsprings. No duplicated members are allowed in the population. The algorithm terminates when no new offspring with better fitness score can be found in the  $n+k$ th generation.  $n$  and  $k$  are predefined parameter by users before executing the algorithm. This algorithm is depicted in Figure 5.13

Similarly, GA and RBT-GA (Rubber Band Techniques with GA) utilize genetic algorithm as their engine. The term 'rubber band', used in RBT-GA method, represents a path from location  $(0,0)$  to location  $(m,n)$  in a back-tracking matrix similar to the one created in dynamic programming. The optimal rubber

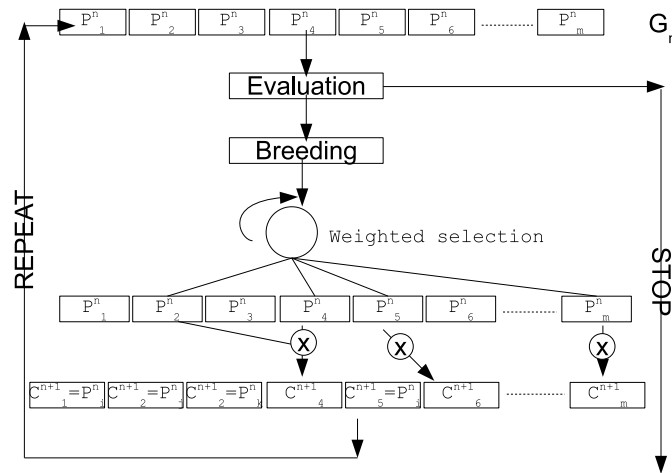


Figure 5.13. SAGA alignment scheme. At any generation  $G_i$  the parents  $P^i$  are crossed breed or mutated by a random operation  $X$  to generate a new set of children  $P^{i+1}$

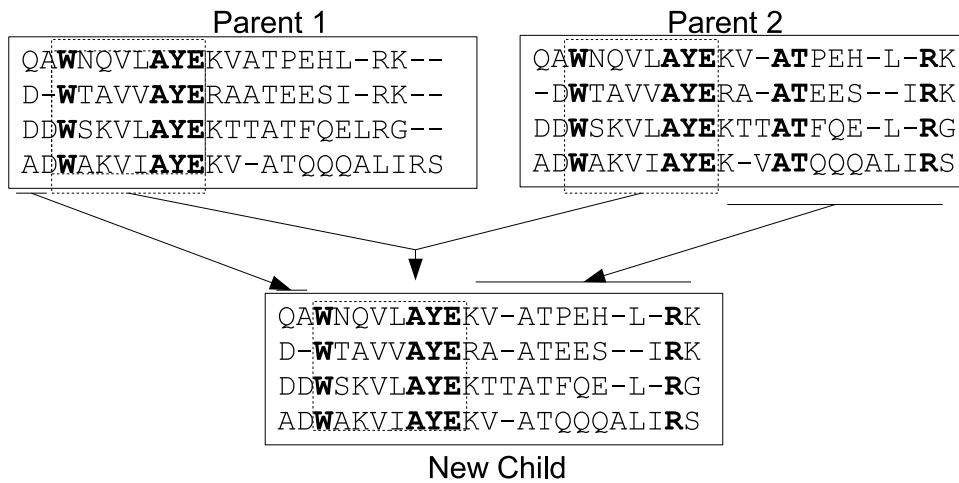


Figure 5.14. Illustration of SAGA, where the two parents are crossed breed. The dotted boxes represent the consistent alignment between the two parents

band is exactly the same as the DP back-tracking path. Instead of finding the optimal solution via DP, the rubber band technique iteratively selects pairs of residues from the aligning pairs of sequences as anchor points (poles) and tries to find the best scored path between the anchors.

Genetic algorithm with ant colony optimization (GA-ACO) combines the GA technique with ant colony optimization (ACO) [26] to prevent local optima. The ACO is an iterative heuristic algorithm that simulates ants' behavior. When an ant moves, it secretes pheromone on its path for other ants to follow. The amount of pheromone secreted is proportional to the goodness or amount of the food being found. The pheromone decays over time. Ants follow paths with the highest intensity of pheromone. Overtime, only frequent paths remain. When applying ACO to GA,  $k$  ants are assigned to random columns of the parent alignments in GA for traversing across the sequences. After  $x$  iterations, the remaining paths are aligned, preserved, and passed to future offspring generations.

#### 5.4.2 GA and Self-organizing Neural Networks

The GA-SNN [66] utilizes self-organized neural networks and genetic algorithm to align a set of sequences. A self-organizing neural network is composed of two layers, an input layer and an output layer. Each node/neuron in the input layer is connected to every node in the output layer with a certain weight. The nodes/neurons in the output layer are interconnected to their neighbors. Figure 5.15 depicts a neural network used in this method. The neural networks are used to identify conserved regions in the sequences allowing genetic algorithm to select offsprings that have these motifs aligned. The neural networks identify the sequence motifs as follows: (i) generates a list of words using window sliding method with length 3 for each input sequence; (ii) feeds the words into the input nodes of the neural networks. (iii) classifies the words that emerge from the third sub-network as motifs and gives them more weights.

The decision of which word is allowed to emerge from the third sub-network is based on the weight of the pattern. When the words are fed into the neural networks, the input nodes/neurons calculate the distances between the words and classified them into groups at the top level. Each word in these groups are passed down to the next level for further classification. The distances between the words are defined to be the average distance between their overlapped words of length 2 (each word is split into two overlapping words for comparison). The words are then classified and grouped similarly to the technique done at the top-level sub-network. A pattern arrives at the bottom of the neural networks will be pair-wise aligned to the pattern stored at that node. The pattern with the highest alignment pair-wise sum-of-pair score at the node is kept as a winner, or a motif.



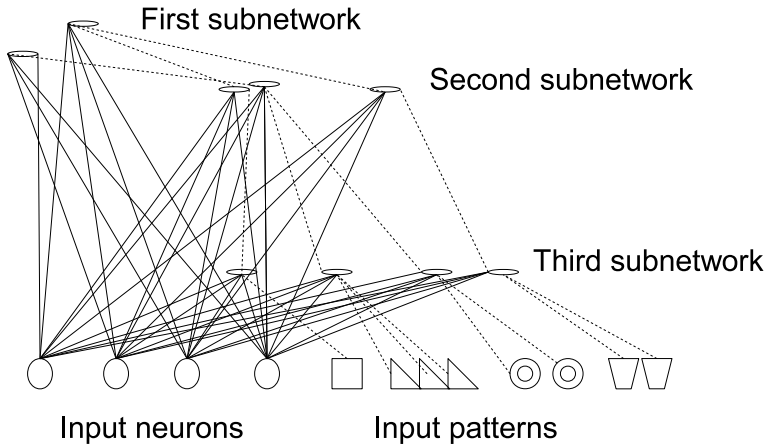


Figure 5.15. Neural networks

### 5.4.3 FAlign

FALIGN [16] combines both progressive and iterative refinement algorithms into MSA. This method requires users to identify and define the motif regions in the sequences. The sequences are split at the motifs' boundaries across all sequences into segments, and each segment of the sequences is aligned progressively. The segments are assembled back after being aligned to generate an alignment. FALIGN uses BLOSUM62 score matrix and sum-of-pair score. The last step of the algorithm is randomly and iteratively shifting the gaps and residues in non-motif regions to improve the alignment score.

## 5.5 New Contributions

This Section is devoted to our multiple sequence alignment algorithms designed and implemented for this project. Each algorithm will be described in detailed and followed by its evaluation comparing to popular existing algorithms.

### 5.5.1 KB-MSA: Knowledge-based Multiple Sequence Alignment

KB-MSA method [79] utilizes the existing biological sequence knowledge databases such as Swiss-Prot, UniProt, or Homstrad, etc to guide sequence alignment. The algorithm schema is depicted in Figure 5.16, which follows these steps:

Step 1: globally pair-wise align the sequences by Needleman and Wunsch's algorithm [77] and categorize them into semi-related groups using the overlapping clustering algorithm as described in Section 4.3.

Step 2: choose a representative sequence, or pivot, from each cluster to query knowledge database.

Step 3: query the knowledge database for annotated sequences with best hit scores, or scores that are higher than some predefined threshold  $\alpha$ . The biological annotated blocks from these sequences are extracted to generate a set of meta-sequences.

Step 4: locally pair-wise align the semi-related sequences in each group to their meta-sequences by Smith and Waterman's algorithm [107]. Sequences that yield low alignment scores will be aligned with other meta-sequences and regrouped to their highest alignment scored group where the meta-sequences come from. Subsequences that are identical, or similar, to the annotated blocks of the meta-sequences are marked as significant blocks and given extra weights for further alignments. Gap insertions in significant blocks are kept intact.

Step 5: remove meta-sequences, choose new pivots, and repeat step 2 to step 5 until a stopping criteria is met. The stopping criteria is either: (i) no new annotated blocks found from knowledge database, (ii) no more sequence regrouping occurred in the last iteration, or (iii) steps 2-5 have been performed  $x$  iterations, where  $x$  is predefined by the user.

Step 7: align the sequences in their own groups by arranging the equivalent significant blocks together. Significant blocks from the sequences are considered equivalent when they are identified from the same annotated block of a meta-sequence.

Step 8: represent each cluster as a partial order graph, where each significant block is a node in the graph.

Step 9: built a guiding tree from pair-wise cluster alignment scores via Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [108].

Step 10: pair-wise align the clusters following the guiding tree.

Step 11: use MAFFT [57] to align non-significant blocks to improve alignment score.

**Assembling the Sequence Alignment** Step 8, 9 and 10 of the KB-MSA algorithm deserve more details, while other steps are straight forward. When all the sequence clusters have been refined and their sequence members have been marked, a partial order graph is built for each cluster (step 8). Each node of the graph represents segments from different sequences in the cluster that resemble a specific biological annotated block from knowledgebase queried sequence. In other terms, each graph node is the best match between the sequences and the actual biological data, and they should not be altered. This partial order graph representation is different from [43] where each node represents a single residue.

For example, assuming the input sequences are: (obtained from BALiBASE [115])

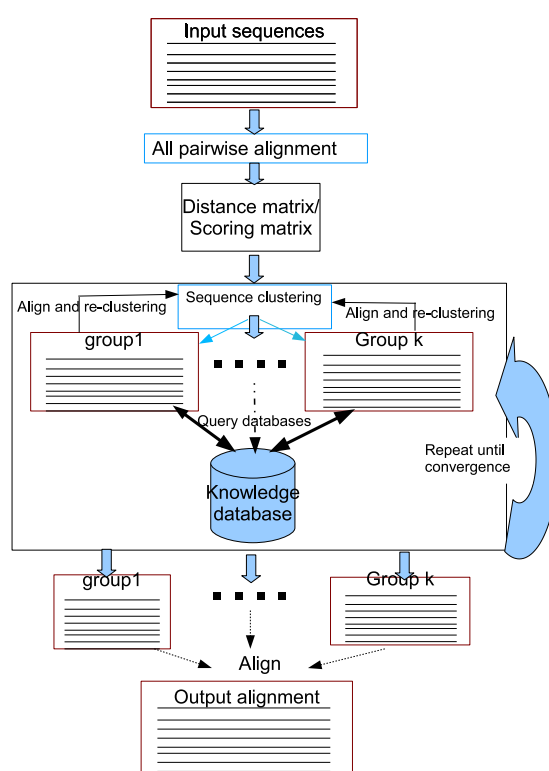


Figure 5.16. Knowledge-based multiple sequence alignment scheme. Initially, the input sequences are partitioned into semi-related groups. The groups' representative sequences are used to search for any available biological information from sequence knowledge-bases. The sequences are given different weights and re-grouped based on the discovered knowledge.

1ABOA WCEAQTKNQGQGWVPSNYITPVN  
 1YCSB WWWARLNDKEGYVPRNLLGLYP  
 1PHT WLNGYNETTGERGDFPGTYVEYIGRKKISP  
 1IHVA AVVIQDNSDIKVVPRRKAKIIRD  
 1VIE YAVESEAHPGSVQIYPVAALERIN

where each sequence is prefixed with its name. After applying steps 1-8 of KB-MSA algorithm, two clusters (1ABOA, 1YCSB, 1PHT) and (1IHVA, 1VIE) are created and annotated with features as in Figure 5.17. In this example, both clusters have the same partial order graph representation.

After generating a partial order graph for each cluster of sequences, the next step (step 9) is to generate a guiding tree indicating which pair of clusters is closest. To do this, all pair-wise distances (or  $\frac{1}{\text{alignment score}}$ ) between the clusters must be computed. The Needleman and Wunsch's algorithm [77] is modified to use a node sliding technique to find the best matching score between any two nodes. In this technique, a node is slid against another, one residue column at a time to find the best matching location, i.e. the location at which the sum of all column scores is maximum. Figure 5.18 shows the sliding steps between two beta strand nodes of previous example (Figure 5.17). The overlapping columns are enclosed in the rectangle. To avoid the complication of gap penalty incurring in non-overlapping columns, the Hierarchical Expected matching Probability (HEP) [82] is used to calculate the matching score between columns of two nodes. The sliding technique allows the arrangement of residues in each graph node remain intact throughout the alignment process.

**Scoring the matches** To determine whether a sequence yields a good enough alignment score with a biological feature obtained from the knowledgebase, its significant threshold must be measured. The significant threshold is the feature average median score (FAMS) and is defined as  $\frac{1}{2}$  of the average alignment score of the feature to itself. Any sequence aligning with the biological feature and yielding an average significant matching score (ABSMS) less than this threshold is considered insignificant.

For any given  $i$ th significant feature  $sig_i$  queried from the knowledgebase and its matching pair-wise optimal local alignment  $A(x, y)$ , the average biological significant matching score (ABSMS) is defined as:

$$\overline{S(sig_i, A(x, y))} = \frac{S(A(x, y)) + S(A(y, y))}{S(A(sig_i, sig_i)) + S(A(y, y))} \quad (5.17)$$

where  $x$  is a segment from feature  $sig_i$  and  $y$  is a segment from the aligning sequence,  $A(x, y)$  is an optimal pair-wise alignment between two segments  $x$  and  $y$ , and  $S(A(x, y))$  is the alignment score of

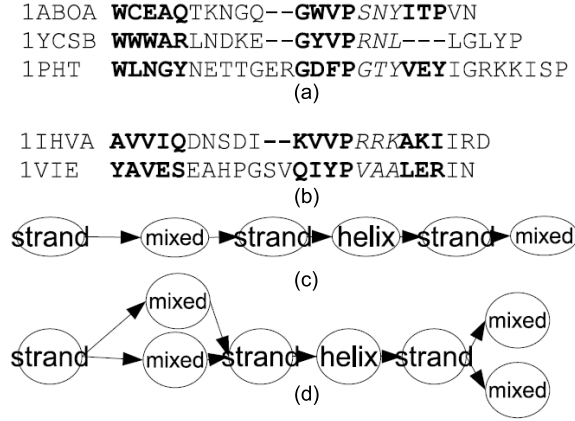


Figure 5.17. Clusters alignment by partial order graph, where bold texts represent beta strand blocks, italics represents alpha-helix blocks, and others are non-significant blocks. (a) and (b) show the two clusters to be aligned, each sequence in the cluster is preceding by its name, (c) is the partial order graph represents clusters (a) and (b), (d) is the fusing graph of (a) and (b). These sequences are obtained from BALiBASE

$A(x, y)$ . Note, all matches between two gap symbols are not scored.

For example, assuming the following optimal pair-wise local alignment:

$$A = \begin{cases} L & F & V & A \\ L & F & - & A \end{cases} \text{ is derived from the beta strand feature containing residues: "NLFVAL" (top) and}$$

another sequence (bottom). For simplicity, let assume the matching score of the same residues is 1, different residues is 0, and a gap is -1. Thus, the feature average median score (FAMS) is  $\frac{1}{2} = 0.5$ , and the alignment

score ( $S(A)$ ) is 2 for:

$$A = \begin{cases} L & F & V & A \\ L & F & - & A \\ 1 & +1 & -1 & +1 = 2 \text{ ( = alignment score)} \end{cases}$$

Similarly, the score of aligning the beta strand to itself is 6 (there are six residues in the strand), and the segment from the sequence to itself is 3 (there are 3 residues from the sequence appearing in the alignment).

The average biological significant matching score (ABSMS) is:  $\bar{S} = \frac{2+3}{6+3} = 0.55$ . Since  $\bar{S} = 0.55$  is greater the feature average median score (0.5), the segment of the sequence involved in this alignment is marked as significant.

**A Consistency Variation of KB-MSA** When sequence knowledge databases are not available, KB-MSA can be modified slightly to utilize the consistency information from the input sequence. To do

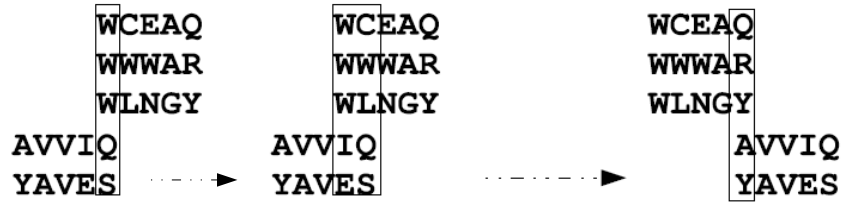


Figure 5.18. Finding the max matching score between two partial order graph nodes by sliding techniques. The overlapping columns are enclosed in the rectangle. Non overlapping columns are considered matching with gaps

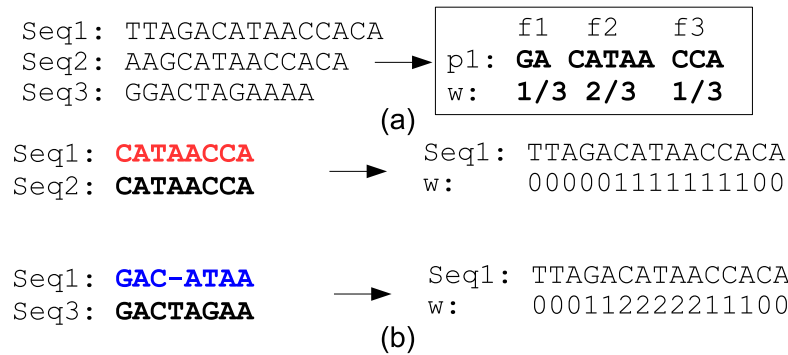


Figure 5.19. Creation of an annotated sequence for KB-MSA consistency database. (a) shows the input sequences and the final annotated pattern for seq1 having three featured blocks f1, f2, and f3 with weights 1/3, 2/3, and 1/3 respectively; (b) shows the immediate pair-wise local alignments of seq1 and their weight marking vectors

this, a temporary database containing simulated sequences representing the consistency information across the input sequences must be built. The consistency database can be built through the following steps:

- Step 1: perform all-pair-wise local sequence alignment by dynamic programming.
- Step 2: for each sequence, annotate aligned segments as significant blocks (or consistency blocks).
- Step 3: calculate matching weight vector for each sequence
- Step 4: generate a pattern for each sequence containing annotated significant blocks and their weights.

Figure 5.19 illustrates the process of creating a consistency database for KB-MSA. Every time a residue in a sequence is identified as a part of a local pair-wise alignment, 1 is added to its weight vector  $w$ . The differences in the weight vector indicate where the cuts should be. Each significant block is represented as a triplet containing the block residues, the block featured identification and the block weight factor. The weight factor is the total number of blocks being identified in the pair-wise local alignment divides the number of all input sequences. The weight factor is set to 1 for all featured blocks of sequences from knowledge database. This is the main difference between the sequences from the two types of databases.

The technique of using sequence consistency information in multiple sequence alignment is fundamentals in T-COFFEE, PROBCONS, and MAFFT. However, the consistency blocks in this algorithm represent the common contiguous residues across many sequences, unlike T-COFFEE or PROBCONS and MAFFT, where the sequence consistency is a library of local/global pair-wise alignment scores or matrices of transitional probabilities.

**Alignment Benchmarks** To validate the accuracy and reliability of the alignment generated by knowledge-based multiple sequence alignment algorithm (KB-MSA), the BALiBASE [115], PREFAB [30], HOMSTRAD [73], and SABmark [122] multiple sequence alignment benchmarks are utilized.

BALiBASE alignment benchmark contains 214 reference alignments that are corrected and verified alignments based on 605 structures from the Protein Data bank (PDB). These references are partitioned into nine reference sets. Each set is designed for a different type of alignment problem. Ref1 alignments contain similar length sequence subsets with no large insertions or extensions. Each subset contains fewer than 6 sequences with similar percent of identity. Ref2 alignments contain highly divergent sequences. Ref3 references contain pairs of subfamilies with less than 25% identity between the two subfamilies in each pair. Ref4 references contain long terminal extensions, and Ref5 references contain large internal insertions and deletions. References 6-8 are designed for transmembrane regions, inverted domains, and repeat sequences. The first 5 references are fully inspected and verified thus making them an appropriate benchmark for evaluating the new alignment algorithms. The accuracy of an alignment is accessed based on the number of columns across the sequences correctly aligned.

The second utilized benchmark is the PREFAB [30] which contains 1682 alignments. These alignments are generated by pair-wise align two sequences with known 3D structures and include up to 24 high scoring homologous sequence to these pairs. The accuracy of an alignment is accessed based on the pair of sequences with known 3D structures.

The third benchmark used in SABmark [122], which contains two subsets: the superfamily and the twilight. Each superfamily group represents SCOP superfamily with 25-50 percent identity. Each group in the twilight subset contains sequences with 0-25 percent identity. These two subsets are extended to include non-homologous sequences as false positives. SABmark accesses a multiple sequence alignment accuracy by taking the average pair-wise alignment scores against the reference alignment pair-wise reference set.

The fourth alignment benchmark used is HOMSTRAD(HOMologous STRucture Alignment Database) [73] which contains 130 protein families and 590 aligned structures that are selected from x-ray structure anal-

ysis. Despite that HOMSTRAD is not designed as a benchmark, its sequences are clustered and aligned by families makes it a good candidate for multiple sequence alignment.

**Results and Discussions** In these evaluation tests, default parameters are used on CLUSTALW [118], MAFFT [57], PROBCONS [25], and T-COFFEE [83]. BLOSUM62 substitution matrix is used to quantify residue matches with a gap penalty of -4 for pair-wise alignments. In addition, the biological annotated features being queried from the knowledge database are sequence secondary structures (beta-strand and alpha-helices) and motifs. Since majority of the benchmark sequences are from protein data bank (PDB), the SWISSPROT [11], (built on PDB), sequence knowledge database is used in these benchmark tests. It is possible that the knowledge database may contain incomplete sequence information, i.e., biological knowledge of some sequences are not available. To simulate this situation, benchmark tests is performed with 10% (KB-MSA 10%) sequence knowledge where maximum of 10 percent of the sequences being used to query the sequence knowledge database are used. The 10% is obtained from our experiments (ranging from 50%, 30%, 20%, 10%, 5%) on HOMSTRAD random datasets at which majority of alignment accuracy drops significantly.

The first five fully verified references (Ref1, Ref2, Ref3, Ref4 and Ref5) from the BALiBASE [115] are used. Figure 5.20 shows the average BALiBASE benchmark alignment scores. The KB-MSA tested with full support of SWISSPROT database yields the highest score among all the tested algorithms. It surpasses CLUSTALW and T-COFFEE by an average score of 10% and PROBCONS and MAFFT by 8% and 7%, respectively. The KB-MSA 10% represents KB-MSA tested with partial knowledge database in which only 10% of all aligning sequences are allowed to query the database. These sequences are chosen at random to simulate a partial knowledge database where at most 10 percent of the queries get any valid result from the database. CB-MSA is a derivation of KB-MSA, in which the sequence knowledge database is replaced with the consistency database. Without the actual biological sequence knowledge database, CB-MSA algorithm is still comparable to CLUSTALW and T-COFFEE on the BALiBASE benchmark tests.

Similarly, the same tests are performed on the sequence subsets of SABmark. The result is shown in Figure 5.21. With the full support of the SWISSPROT knowledgebase, the new algorithm (KB-MSA) outperforms other algorithm by a margin between 8-15%. With the simulation of 10 percent query hits, the KB-MSA (KB-MSA 10%) loses some accuracy; however, it still yields comparable accuracy as PROBCONS - The most accurate existing alignment algorithm. Without the support of sequence knowledge database, the consistency-based (CB-MSA) algorithm is comparable to T-COFFEE, CLUSTALW and MAFFT.



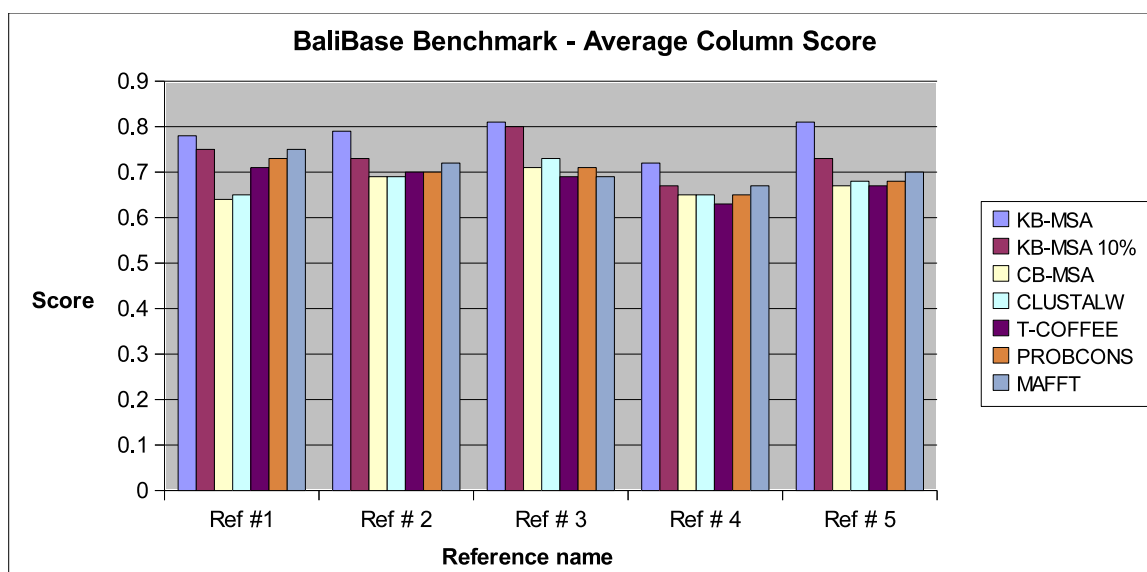


Figure 5.20. KB-MSA tested against ClustalW, T-Coffee, PROBCONS, and MAFFT on BALiBASE benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by more than 9%. With 10% sequence knowledge, KB-MSA yields about 4% improvement

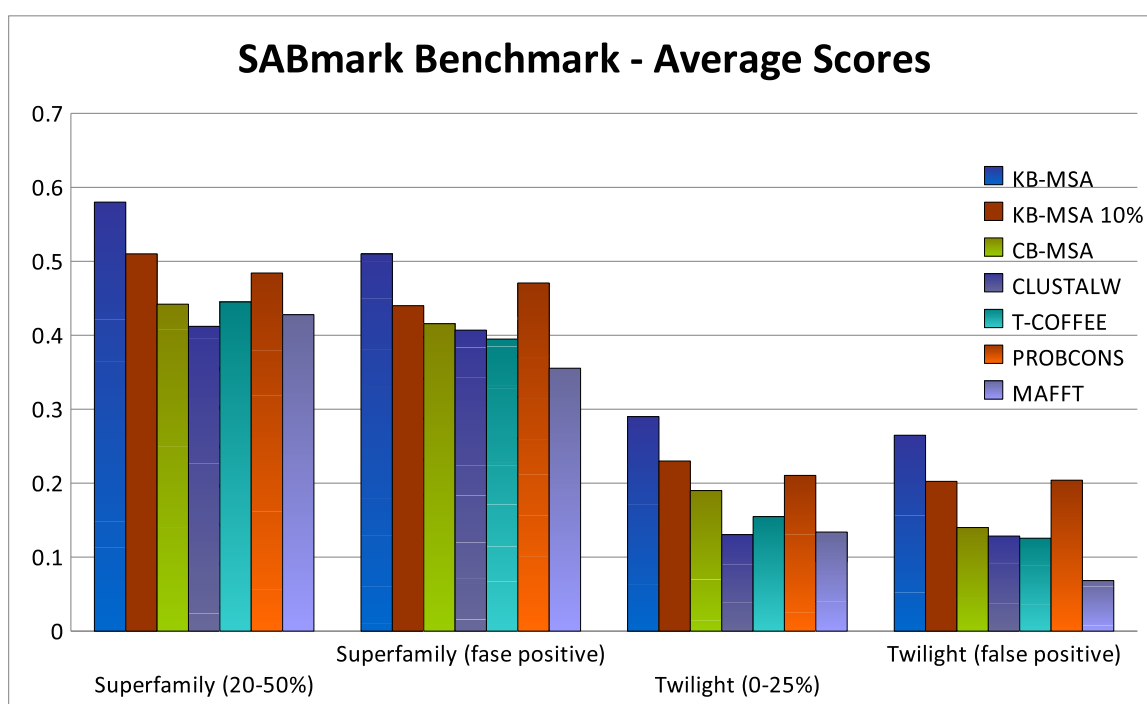


Figure 5.21. KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT on SABmark benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by at least 8% on the original datasets and more than 6% on false positive datasets

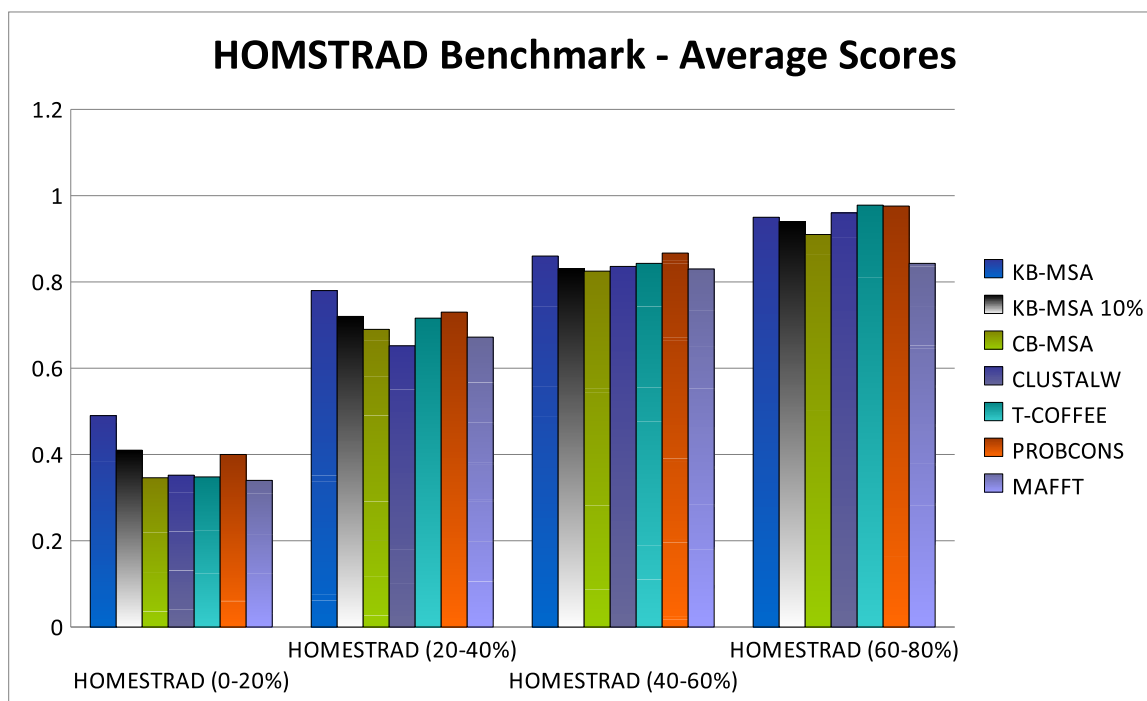


Figure 5.22. KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT on HOMSTRAD. With complete sequence knowledge, KB-MSA increases average alignment score about 10% on datasets with less than 20% sequence identity

Sequences in the HOMSTRAD and PREFAB benchmarks are categorized into test groups based on their percent identity, or sequence similarity percentage. Each group contains sequences that are in 20 percent identity range. Figure 5.22 and Figure 5.23 show the results of these two benchmark tests. From these results, the new alignment algorithm with full knowledgebase support (KB-MSA) outperforms all other algorithms on datasets containing sequences with low percent identity. These improvements are derived directly from the sequence knowledgebase where other algorithms do not have access to. This advantage is fading out as the sequence percent identity increases. For datasets with sequence percent identity greater than 40%, the new algorithm accuracy is comparable to other testing algorithms. The KB-MSA with 10% knowledgebase support yields similar average alignment score to PROBCONS in most cases, and the KB-MSA using consistency database (CB-MSA) performs relatively comparable to CLUSTALW and T-COFFEE in many cases.

Observing the experimental data from the benchmark tests, the knowledge-based multiple sequence alignment algorithm (KB-MSA) performs consistently well on almost all datasets, especially, when the aligning sequences share at most twenty percent identity. In the PREFAB benchmark tests, it is reasonable to expect that the new algorithm will perform much better on datasets in the twilight zone since scoring is

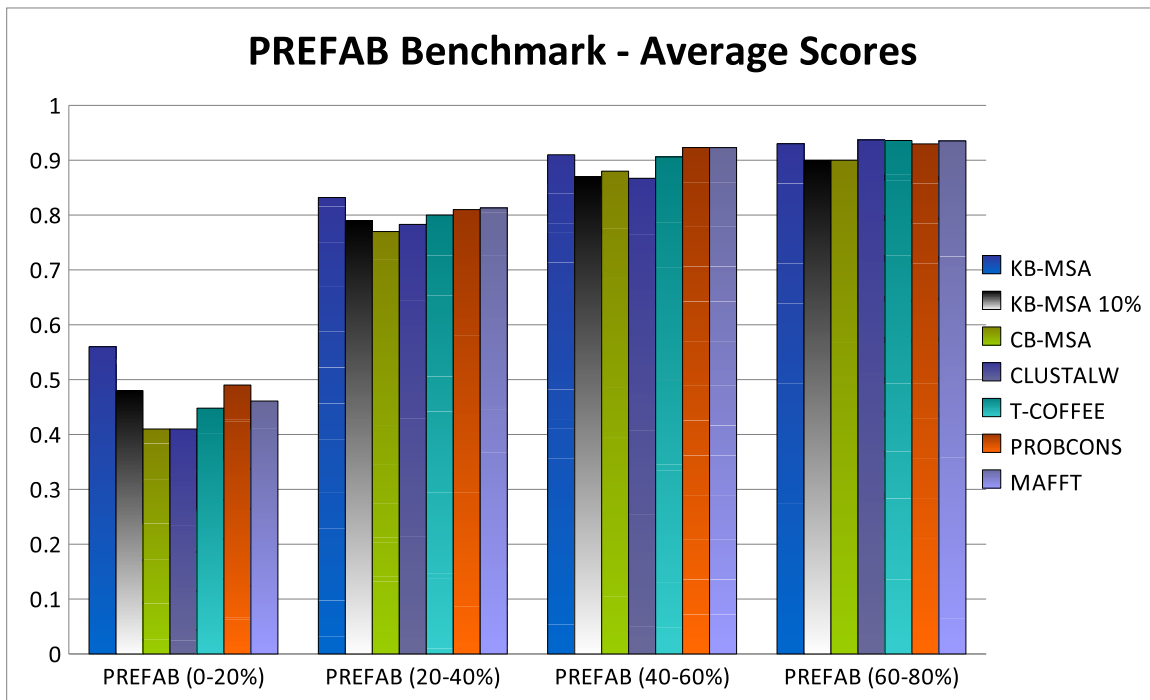


Figure 5.23. KB-MSA testing against ClustalW, T-Coffee, PROBCONS, and MAFFT. With complete sequence knowledge, KB-MSA increases average alignment score at least 7% on datasets with less than 20% sequence identity

done only on the core-pair of each dataset. And the KB-MSA should get a near perfect score if it queries the database with the cored-pair. However, KB-MSA is not able to detect the cored-pairs from these groups. On datasets with high percentage of sequence identity, most tested algorithms perform very well, and the little extra sequence knowledge many not contribute much to the alignment outcomes.

Comparing the three consistency-based algorithms (CB-MSA, PROBCONS, and T-COFFEE), PROBCONS comes out to be the best in the group, while the different between CB-MSA and T-COFFEE are not very clear.

### 5.5.2 PADT: Progressive multiple sequence Alignment based on Dynamic weighted Tree

[P]rogressive multiple sequence [A]lignment based on [D]ynamic weighted [T]ree (PADT) [80] is a special variation of progressive multiple sequence alignment that has been employed in T-COFFEE [83] and CLUSTALW [118]. What makes this algorithm distinctively different from other is the capability to change its sequence alignment order to reflect the best pair-wise matching whenever new information are available; thus correcting alignment errors as soon as they are found.

Initially, the biological information and local pair-wise alignment scores are obtained for all input sequences. The biological information can be obtained by querying sequence knowledge databases such as SWISS-PROT or TrEMBL [11], etc. To combine this information, each residue in the sequences will be represented as a triplet,  $(r, \alpha, \beta)$ , containing the residue  $r$ , the local alignment score  $\alpha$ , and the biological score  $\beta$ . This information are utilized to enhance the global pair-wise alignments and estimating the sequence alignment guiding tree. Moreover, the guiding tree in our algorithm can dynamically reorder its branches depending on the result of alignments on lower branches.

**The PADT's Algorithm** The algorithm to align  $n$  input sequences is as follows:

Step 1: each sequence is represented as an array of triplets. The  $i$ th triplet of sequence  $j$  represents the residue  $i$ th of sequence and two values  $\alpha$  and  $\beta$ . These values represent the local alignment score and biological score of this residue, respectively. Initially, these scores are zero.

Step 2: for each input sequence, extract its sequence biological information. Residues being identified as parts of a conserved or functional region are given  $\beta = 1$  score for its biological score.

Step 3: perform all pair-wise sequence alignments via Smith-Waterman's method [107] to identify the longest common subsequence between any pair of sequences. Residues included in the common subsequence are given scores of  $\alpha$ . These  $\alpha$  scores are accumulative. And a residue with local alignment score of  $n \times \alpha$  indicates that this residue is common to all pair-wise local alignments.

Step 4: perform global all pair-wise sequence alignments via the weighted Needleman-Wunsch's algorithm as described earlier. The inverse of these alignment scores represents the distance matrix between the sequences.

Step 5: estimate the phylogenetic tree by using the distance matrix generated in step 4. The phylogenetic tree can be estimated by either the Neighbor Joining (NJ) or Weighted/Unweighted Pair Group Method with Arithmetic mean (UPGMA/WPGMA) algorithms as discussed in Chapter 4. Branches of the guiding tree that are less than  $\epsilon$  distance apart are considered interchangeable.

Step 6: align the sequences following the order specified by the estimated phylogenetic tree from the tree leaves to the tree root via the Needleman-Wunsch's algorithm with weighted scoring scheme described earlier. On any section of a tree where there are more than two interchangeable branches, all pair-wise alignments between these branches are performed. The highest score pairs is selected and gap-refined; the process is repeated until all branches are aligned.

**Scoring method** Traditionally, Needleman-Wunsch's algorithm (Needleman & Wunsch, 1970) maps two sequences  $x$  and  $y$  of lengths  $n$  to the two connecting side of a scoring matrix  $H$  with size  $(n + 1) \times (n + 1)$ . The values in first row and column of the matrix are set to the corresponding gap cost, i.e.  $H[0, i] = H[i, 0] = i \times \text{gap}$ , where  $\text{gap}$  is the gap-aligned penalty (a negative value). A pair of residues from row  $i$ th and column  $j$ th is the maximum of three values:

$$H[i, j] = \begin{cases} H[i - 1, j - 1] + s(x_i, y_j) \\ H[i - 1, j] + \text{gap} \\ H[i, j - 1] + \text{gap} \end{cases}$$

where  $s(x_i, y_j)$  is the substitution score (or pair-wise residue score) between residues at row  $i$  and residue at column  $j$ .

To incorporate the local alignment scores and the biological scores into this dynamic pair-wise alignment, the pair-wise score between two triplets  $(x_i, \alpha_i, \beta_i)$  and  $(y_j, \alpha_j, \beta_j)$ , representing residues  $y_j$  and  $x_i$ , is defined as:

$$s((x_i, \alpha_i, \beta_i), (y_j, \alpha_j, \beta_j)) = s(x_i, y_j) + \left[ \frac{\alpha_i + \alpha_j}{2} + B(\beta_i, \beta_j) \right] \times s(x_i, y_j)$$

where  $B(\beta_i, \beta_j)$  is the biological score of the two residues  $y_j$  and  $x_i$ , which is defined as:

$$B(\beta_i, \beta_j) = \begin{cases} 1 & \text{if } \beta_i \neq 0 \text{ and } \beta_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

To align two pre-aligned groups of sequences, the sum-of-pair scoring method is used in place of substitution matching score and the average local alignment score is extended to the average of all local alignment scores in the two aligning columns. The sequence group column biological score is set to 1 if any of the residues in a column has a biological score of 1; otherwise, it is set to zero. This group biological score helps dynamic programming to favor matching between two columns that each has at least a residue with biological significance. These biological scores ( $\beta$ ) and local alignment scores ( $\alpha$ ) are used as weight control parameters. They can be increased or decreased to adjust the weights of these two features.

**Dynamic Alignment Guiding Tree** Similar to traditional method, an alignment guiding tree is built from either UPGMA [108] or neighbor-joining (NJ) [98] method from all pair-wise sequence distances. However, in this method, the branches of the alignment guiding tree that are less than  $\epsilon$  distance apart are considered interchangeable. These branches indicate that the pair-wise distances between these sequences are small. Thus, when the sequences are aligned, these distance will change depending on gap insertions and the arrangement of the columns these alignments. In this design, the branches are allowed to be reordered dynamically based on the best pair-wise alignment of any two branches. This step is done by performing all pair-wise alignments at the tree level where there are more than two interchangeable branches. The best

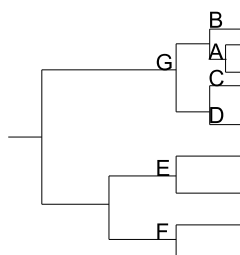


Figure 5.24. This figure illustrates the changeability between branches of an alignment guiding tree. Assuming the distances from A, B, C, and D to G are all less than  $\epsilon$ , then A, B, C, and D are interchangeable.

aligned pair is selected for gap refinement. The process is repeated until all the branches at this level are aligned. Therefore, this guiding tree guarantees that alignments at any tree level yield highest pair-wise alignment scores.  $\epsilon$  is defined to be one half of the average distance between all internal nodes of the alignment guiding tree. Nevertheless, this value is a parameter and can be changed to extend or restrict the changeability range on the tree levels. Figuratively, the interchangeability is shown in Figure 5.24, where the distances from branches A, B, C, and D to G are all less than  $\epsilon$ . Thus, the pair-wise alignment preceding branch A may make it closer to either C or D. Thus, A, B, C, and D are interchangeable.

The gap refinement step is as follows: for any alignment with more than two sequences, gaps inserted prior to the last pair-wise alignment are rearranged to improve the alignment score. These gaps, while needed in alignment steps prior to the last alignment, may not be so significant after the last alignment; thus, rearrange them may improve the alignment score. Gaps are randomly selected and moved to the nearest border of a contiguous conserved region. The border is indicated by the difference in residue biological scores  $\beta$ , or their local alignment scores  $\alpha$ , between two adjacent residues. In case both biological score and local alignment score exist, both borders indicated by these scores are tested. A gap will be repositioned to the location that improves the alignment score.

## 5.6 Test Data and Alignment Methods

Five different existing progressive alignment tools, CLUSTALW [118], DIALIGN [74], T-COFFEE [83], MUSCLE [30] and PROBCONS [25], are used to test the new algorithm performance. Among these, CLUSTALW and MUSCLE utilize global alignment method; DIALIGN employs local alignment technique; T-COFFEE combines both global and local strategies; and PROBCONS employs global technique with hidden Markov model. These alignment algorithms are performed on the same datasets from three different reference benchmarks: the Reverse-Transcriptase Order-Specific-Motifs (RT-OSM) sequences [51](as

seen in Figure 3.13), BALiBASE3.0 [115], and PREFAB (Protein REference Alignment Benchmark version 4.0) [30]. Details of these benchmarks are described earlier in Chapter 3.

In the next Section, we will describe the experimental results of the new multiple sequence alignment algorithm.

## 5.7 Results

All alignment tools are performed with their default parameters. The parameters used in the new alignment method (PADT) are: the local alignment score  $\alpha$  is set to  $\frac{1}{n}$ , where  $n$  is the number of input sequences,  $\beta$  is set to 1 for residues with biological significance, gap penalty is -10, and the substitution scores are from and BLOSUM62(Henikoff & Henikoff,1992).

The new algorithm PADT (Progressive multiple sequence Alignment based on Dynamic weighted Tree) is implemented to generate two alignments from the same input sequences: one is constructed from following the phylogeny tree created by UPGMA method [108], and the other is constructed from following the tree created by neighbor-joining (NJ) method [98].

### 5.7.1 Measuring Alignment Quality

There are two popular quantitative methods for measuring such conformity are Total Columns (TC) [115] and percentage of conserved regions, or motifs, correctly aligned. TC score is most the popular in many alignment benchmark tests. TC score is the percentage of reference aligned columns that are correctly aligned by other methods.

### 5.7.2 RT-OSM Results

The RT-OSM (Reverse-Transcriptase Order-Specific-Motifs) [51] test is designed to find alignment algorithms that are sensitive to sequence motifs. These motif regions can be very short, thus making it very hard to detect. This is biological information should be integrated into multiple sequence alignment technique, if it is available. In RT-OSM benchmark, all motifs have been annotated. As expected, both versions of the algorithm (PADT-NJ is the new multiple sequence alignment utilizing neighbor-joining phylogeny estimation method and PADT-UPGMA is the new multiple sequence alignment algorithm utilizing the UPGMA phylogeny estimation method) yield highest alignment scores. The result is shown in Figure 5.25.

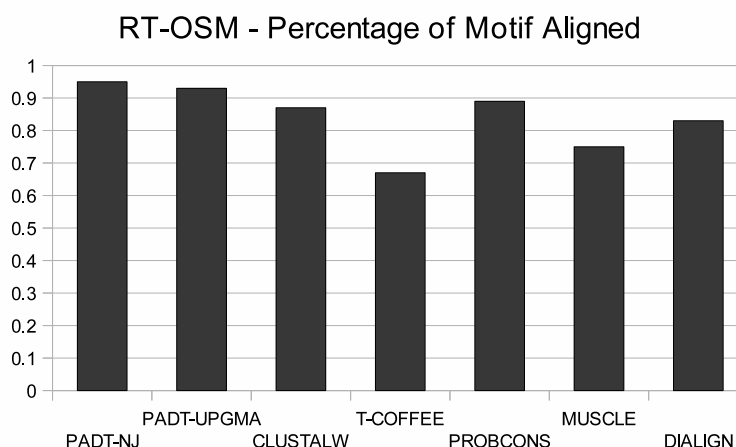


Figure 5.25. Percentage of motifs aligned by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on RT-OSM sequence set PADT-NJ and PADT-UPGMA surpass PROBCONS by 5% and 7% respectively

**BALiBASE Results** Besides the RT-OSM test, all tests on the first five reference sets (Ref #1 to Ref#5) of BALiBASE benchmark are performed. These reference sets have been confirmed and verified and contain more than 200 reference alignments. Ref#4 and Ref#5 contain alignments with large N/C insertions and extensions. These are reference sets that many global alignment algorithms do not perform very well, while local alignment algorithms do best. The results of BALiBASE tests are shown in Figure 5.26 and Figure 5.27. The new algorithm using neighbor-joining method (PADT-NJ) achieves a significant improvement in Ref#2 (about 9%) and slightly better than other algorithms in two other categories. It seems that the new algorithm performs relatively well on datasets with large insertions and extensions (BALiBASE Ref#4 and Ref#5). The average total column score of PADT-UPGMA is comparable with the DIALIGN - an alignment based on local strategy, while the PADT-NJ achieves 6% improvement in Ref#5 datasets.

**PREFAB Results** Similarly, 60 reference tests on PREFAB benchmark are selected and categorize into three different groups. The first group contains sequences with less than 20% identity, the second group contains sequences with 20-40% identity, and the third group contains sequences with 40-60% identity. The same evaluations techniques done on BALiBBASE benchmarks are performed on these datasets. The results are shown in Figure 5.28.

Since PREFAB reference sets are automatically generated, these sequences do not have any biological information neither they have any actual reference alignment. Thus, total column score cannot be used.



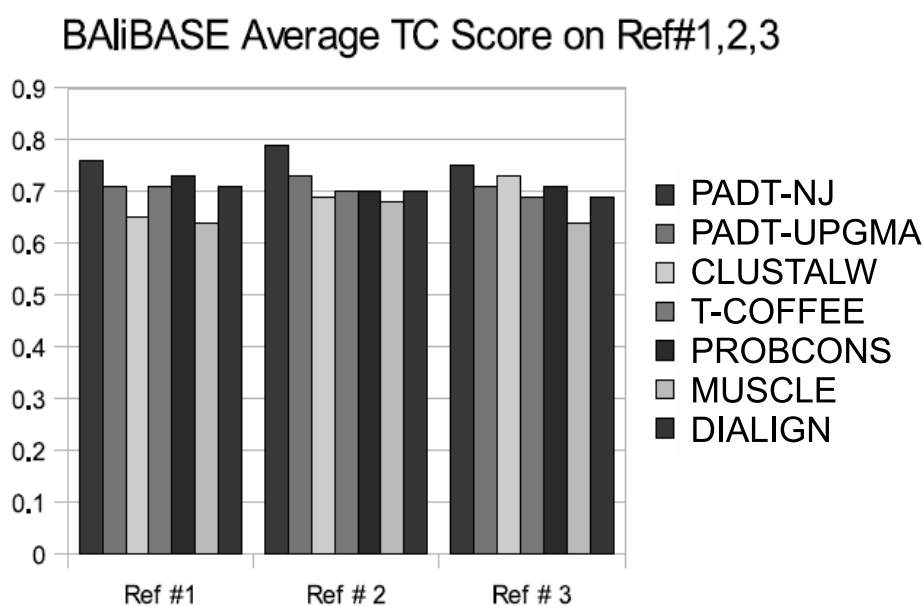


Figure 5.26. Total column (TC) score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on BAlIbASE Ref#1, Ref #2, and Ref#3. PADT-NJ gains at least 9% improvement on Ref#2.

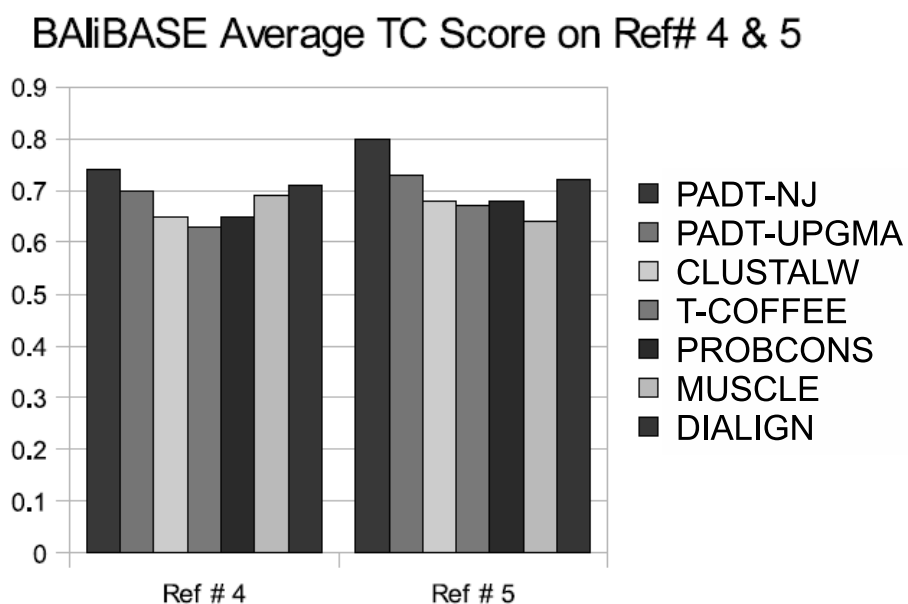


Figure 5.27. Total column (TC) score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on BAlIbASE Ref#4 and Ref#5. PADT-NJ gains 6% on Ref#5 over DIALIGN.

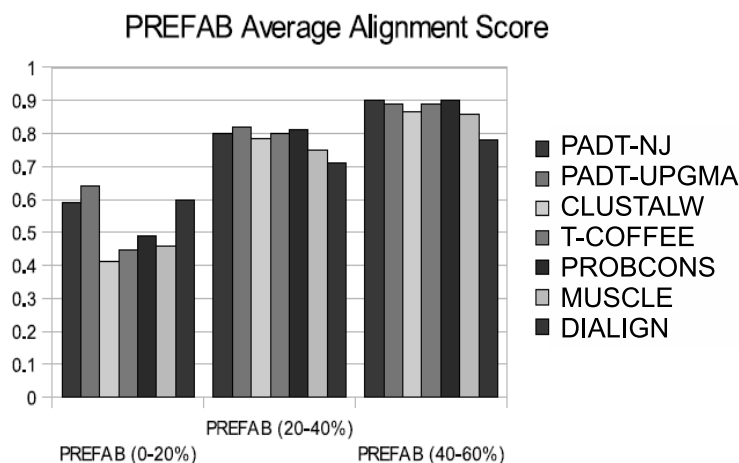


Figure 5.28. Quality score by PADT-NJ, PADT-UPGMA, CLUSTALW, T-COFFEE, PROBCONS, MUSCLE, and DIALIGN on PREFAB datasets. On the first group, PADT-UPGMA gains 4% improvement over DIALIGN, and more than 10% improvement over others.

To measure the accuracy of the new alignments, the quality score (Q score) that is provided along with PREFAB benchmark is used. Q score is applied only to the initial pair of sequences in each reference alignment of PREFAB. Q score is the number of correctly aligned residue pairs from the new alignment divided by the number of aligned residue pairs in the reference alignment.

In these tests, the new alignment algorithm (PADT) relies only on the local alignment consistency between the sequences. Surprisingly, both versions of PADT perform comparable to DIALIGN and better than all others on sequences with less than 20% identity. For sequences with higher identity, PADT is comparable to PROBCONS and are among the most accurate alignment algorithms.

Overall, the new method shows a significant improvement in datasets with low percent identity, especially, when extra biological information is available. The phylogeny estimation methods and the datasets also play important roles in the alignment accuracy. On the RT-OSM and BALiBASE benchmarks, estimating the sequence phylogeny by neighbor-joining method seems to give better alignment results. However, on the PREFAB datasets, UPGMA method is more favorable.

There are many other multiple sequence alignment algorithms have been developed, however, most of them do not get much attention due to their practical limitations and their source code availability and accessibility.

In the next chapter, we will explore alignment models and techniques that have been designed and developed to improve the run-time complexity of general multiple sequence alignment algorithms.

## Chapter 6

### MULTIPLE SEQUENCE ALIGNMENT ON HIGH-PERFORMANCE COMPUTING MODELS

The current advancement of sequencing technology has created a vast amount of sequences to be categorized, stored and analyzed. However, multiple sequence alignment, one of the most used tools for sequence analysis, is already passing its practical threshold due to large number of sequences to be aligned and the NP-Completeness nature of the alignment problem. A practical solution to this problem is to perform independent subtasks of the alignment problem simultaneously on parallel or distributed computers. In this chapter, we will start by describing the basics of parallel computing models and then investigate multiple sequence alignment algorithms that have been parallelized to speed up multiple sequence alignment process.

#### 6.1 Parallel Systems

There are many types of parallel computing systems that have been deployed to solve computationally intensive problems like sequence alignment and classification. The most models are described below.

##### 6.1.1 Multiprocessor

Multiprocessor systems consist of multiple processing nodes connected through a network. Each processing node contains one or more processors or multi-core chips that share a limited on-board memory. This memory is called local memory or distributed memory. The processing nodes may share a common memory. With these systems, each individual processing node can perform different task on different set of data. Communication among processing nodes relies primary the network connections. These systems are classified as Multiple-Instruction, Multiple-Data (MIMD) systems.

##### 6.1.2 Vector

Vector processors have special instructions that can perform the same instruction on several data elements. For example, a vector instruction can add one array of scalar to another simultaneously, while a regular scalar instruction has to iterate through the elements in the arrays individually. These systems are classified as Single-Instruction, Multiple-Data (SIMD) systems.

### 6.1.3 GPU

Graphics Processing Unit (GPU) is very popular in portable and desktop computers for rendering computer graphics. GPU instructions are specialized for graphics manipulation and processing. Currently, it is common to utilize GPU for general purpose processing. GPUs must work in conjunction with a host system as co-processors.

### 6.1.4 FPGA

Field-Programmable Gate Arrays (FPGAs) are a type of reconfigurable computing systems. Hardware logic on FPGAs is configurable, which allows custom operations and data types to be programmed into the chip for each processing element on board. FPGAs act as co-processors and are utilized as hardware accelerators. FPGA processing elements are locally interconnected, thus providing a very low overhead communication environment.

### 6.1.5 Reconfigurable Mesh

Reconfigurable Mesh (R-Mesh) computing system is a  $k$ -dimension grid of processing elements. Communication between the processing elements is primary through a grid network, which is either electrical or optical. The difference in the network medium leads to different communication protocols for this computing model. Generally, R-Mesh processing elements can perform a few simple operations and contain minimal amount of local memory.

## 6.2 Exiting Parallel Multiple Sequence Alignment

In general, independent operations can be performed simultaneously on different computing elements to speedup the time it takes to complete a task. The trade-off between time and processing hardware may not be feasible if the overhead communication between the elements exceeds the computing saving time. Similarly, if the task is not deterministic, parallelizing them may not be effective; and it is the major issue of most multiple sequence alignment paradigms, except progressive multiple sequence alignment as described in Section 5.2.

In general, progressive multiple sequence alignment algorithm follows three steps:

- (i) Perform all pair-wise alignments of  $n$  input sequences.
- (ii) Compute a dendrogram indicating the order in which the sequences are aligned.

(iii) Pair-wise align two sequences (or two pre-aligned groups of sequences) following the dendrogram starting from the leaves to the root of the dendrogram.

A quick analysis of these three steps reveals that:

- Step (i) has  $O(n^4)$  run-time complexity for performing  $\frac{n(n-1)}{2}$  pair-wise alignment by dynamic programming, which is of order  $O(n^2)$ .
- Step (ii) yields an order of  $O(n^3)$  run-time complexity for either UPGMA or neighbor-joining (NJ) methods [see Sections 4.2 and 4.1].
- Step (iii) performs  $(n - 1)$  pair-wise alignments of pre-aligned groups of sequences, in the worse case. Each of these pair-wise alignments yields an order of  $O(n^4)$  for utilizing dynamic programming ( $O(n^2)$ ) and sum-of-pair scoring ( $O(n^2)$ ) [see Section 3.4.1]. Thus, the run-time complexity of this step is  $O(n^5)$ .

Progressive multiple sequence alignment algorithms are widely parallelized, mostly because they perform  $\frac{n(n-1)}{2}$  independent pair-wise alignments as in step (i). These individual pair-wise alignments can be designated to different processing units for computation as in [47, 60, 63, 68, 85, 86, 94, 95, 101, 112].

These parallel progressive multiple sequence alignment implementations are across many computing architectures and platforms. For example, Lima et al. [63] implemented a DP algorithm on Field-Programmable Gate Array (FPGA). Similarly, Oliver et al. [85, 86] distributed the pair-wise alignment of the first step in the progressive alignment, where all pair-wise alignments are computed, on FPGA. Liu et al. [68] computed DP via Graphic Processing Units (GPUs) using CUDA platform, [112] used CRCW PRAM neural-networks, [47] used Clusters, [60] used 2D R-Mesh, [94] used Network Mesh, or [95] used 2D PR-Mesh computing model.

The two most notable parallel versions of dynamic programming algorithm are proposed by Huang [49] and Chun-Shi and Aluru [4, 47]. Huang's algorithm exploits the independency between the cells on the anti-diagonals of the DP matrix, where they can be calculated simultaneously. There are  $m + n$  anti-diagonals on a matrix of size  $(m \times n)$ . Thus, this parallel DP algorithm takes  $O(n)$  processing units and completes in  $O(m + n)$  time.

Independently, Chun-Hsi et. al [47] and Aluru [4] propose similar algorithms to partition the DP matrix column-wise and assign each partition to a processor. All processors are synchronized to calculate their partitions one row at a time. For this algorithm to perform properly, each processor must hold a copy of the sequence that is mapped to the rows of the matrix. Since these calculations are performed

row-wise, the values from cells  $c_{i-1,j-1}$  and  $c_{i-1,j}$  are available before the calculation of cell  $c_{i,j}$ . The value of  $c_{i,j-1}$  can be obtained by performing prefix-sum across all cells in row  $i^{th}$ . Thus, with  $n$  processors, the computation time of each row is dominated by the prefix-sum calculations, which is  $O(\log n)$  time on PRAM models. Therefore, the DP matrix can be completed in  $O((m+n)\log n)$ , or  $O(n\log n)$  for  $n \geq m$ , time using  $O(n)$  processors.

Recently, Sarkar, et al. [101] implement both of these parallel DP algorithms [4, 49] on a Network-on-Chip computing platform [22].

In addition, the construction of a dendrogram can be parallelized as in [68] using  $n$  Graphics Processing Units (GPUs) and completing in  $O(n^3)$  time.

Furthermore, there are attempts to parallelize the progressive alignment step, step (iii)] as in [113] and [70]. In [113], the independent pre-aligned pairs along the dendrogram are aligned simultaneously. This technique gains some speed-up, however, the time complexity of the algorithm remains unchanged since all the pair-wise alignments eventually must be merged together. Another attempt is seen in [70], where Huang's algorithm [49] is used. When an anti-diagonal of a DP alignment matrix in lower tree level in step (iii) is completed, it is distributed immediately to other processors for computing the pair-wise alignment of a higher tree level. This technique can lead to an incorrect result since the actual pair-wise alignment of the lower branch is still uncertain.

Overall, the major speedup achieved from these implementations comes from two parallel tasks: performing  $\frac{n(n-1)}{2}$  pair-wise alignments simultaneously and calculating the dynamic programming matrix anti-diagonally (or in blocks). These two tasks can only reduce the overall run time complexity of the original algorithm by an order to  $O(n^4)$ , regardless of how many processing units are used. The bottleneck resides in step (iii), where the pair-wise group alignments are done sequentially following the guiding tree, and each alignment requires all the column pair-wise scores be calculated.

The remaining of this chapter is devoted to our newly developed parallel sequence alignment algorithms. These algorithms depend on a reconfigurable computing model, so we start by describing the model before going into details of the algorithms

### 6.3 Reconfigurable-Mesh Computing Models - (R-Mesh)

A Reconfigurable Mesh (R-Mesh) computing is a two-dimensional grid of processing units (PUs), in which each processing unit contains 4 ports: North (N), South (S), East (E), and West (W). These ports can be fused or defused in any order to connect one node of the grid to its neighboring nodes. These

configurations are shown in Figure 6.1. Each processing unit has its own local memory, can perform simple arithmetic operations, and can configure its ports in  $O(1)$  time.

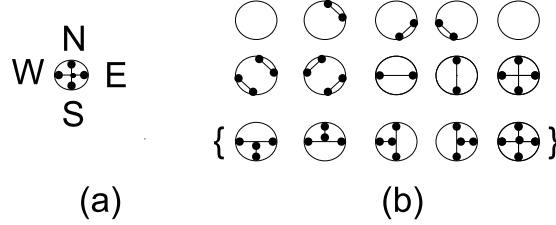


Figure 6.1. Allowable configurations on 4 port processing units; (a) shows the ports directions; (b) shows the 15 possible port connections, where the last five port configurations in curly braces are not allowed in Linear R-Mesh (LR-Mesh) models.

There are many reconfigurable computing models such as Linear R-Mesh (LR-Mesh), Processor Array with Reconfigurable Bus System (PARBS), Pipelined R-Mesh (PR-Mesh), Field-programmable Gate Array (FPGA), etc. These models are different in many ways from construction to operational run-time complexities. For example, the PR-Mesh model does not function properly with configurations containing cycles, while many other models do. However, there are many algorithms to simulate the operations of one reconfigurable model onto another in constant time as seen in [12, 19, 81, 90, 104, 119].

In the scope of this study, we will use a simple electrical R-Mesh system, where each processing unit, or processing element (PU or PE), contains four ports and can perform basic routing and arithmetic operations. Most reconfiguration computing models utilize the representation of the data to parallelize their operations; and there are various data presentation formats [120]. Commonly, data in one format can be converted to another in  $O(1)$  time [120]. For clarity, we will describe the unary representation format being used in this study, which is denoted as 1UN. The 1UN format is defined as:

**Definition 3** *Given an integer  $x \in [0, n - 1]$ , the unary 1UN presentation of  $x$  in  $n$ -bit is:  $x = (b_0 b_1, \dots, b_{n-1})$ , where  $b_i = 1$  for all  $i \leq x$  and  $b_i = 0$  for all  $i > x$  [120].*

For example, a number 3 is represented as 11110000 in 8-bit 1UN representation.

In addition to the 1UN unary format, we will be utilizing the following theorem for some of the operations:

**Theorem 1** *The prefix-sum of  $n$  values in range  $[0, n^c]$  can be found in  $O(c)$  time on an  $n \times n$  R-Mesh [120].*

In terms of multiple sequence alignment, the number of bits used in the 1UN notation is correlated to the maximum length of the input sequences. In the following section, we will describe the designs of R-Mesh components to use in dynamic programming algorithms.

## 6.4 Pair-wise Dynamic Programming Algorithms

This section begins with the description of several configurations of R-Mesh needed to compute various operations in pair-wise dynamic programming algorithm. Following the R-Mesh constructions are new constant-time parallel dynamic programming algorithms for Needleman-Wunsch's, Smith-Waterman, and the Longest Common Subsequence (LCS) algorithms.

### 6.4.1 R-Mesh Max Switches

One of the operations in dynamic programming algorithm requires the capability to select the largest value from a set of input numbers. Following is the design of an R-Mesh switch that can select the maximum value from an input triplet in the same broadcasting step. For 1-bit data, the switch can be built as in Figure 6.2(a) using one processing unit, (adapted from [10]). The unit configures its ports as {NSEW}, where the North and West are input ports and the others are output ports. When a signal (or 1) comes through the switch, the max bit will propagate through the output ports. Similarly, a switch for finding a maximum value of four input bits can be built using 4 processing units with configurations: {NSW,E}, {NSE,W}, {NE,S,W}, and {NSW,E} as in Figure 6.2(b). To simulate a 3-input max switch on positive numbers, one of the input ports loads in a zero value. These switches can be combined together to handle the max of three n-bit values as in Figure 6.3. This n-bit max switch takes  $4 \times n$ , (i.e.  $O(n)$ ) processing units and can handle 3 to 4 n-bit input numbers. All of these max switches allow data to flow directly through them in exactly one broadcasting step, and they will be used in the design of our algorithm, described latter.

### 6.4.2 R-Mesh Adder/Subtractor

Similarly, to get a constant time dynamic programming algorithm we have to be able to perform a series of additions and subtractions in one broadcasting step. Exploiting the properties of 1UN representation, we are presenting an adder/subtractor that can perform an addition or a subtraction of two n-bit numbers in 1UN representation in one broadcasting time. The adder/subtractor is a  $k \times n$  R-Mesh, where  $k$  is the smaller magnitude of the two numbers. The R-Mesh adder/subtractor is shown in Figure 6.4.



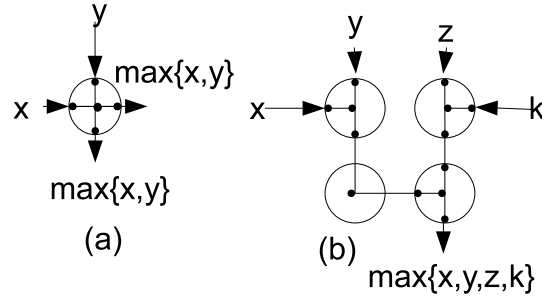


Figure 6.2. Two 1-bit max switches. (a) - fusing  $\{NSEW\}$  to find the max of two inputs from North and West ports; (b) - construction of a 1-bit 4-input max switch.

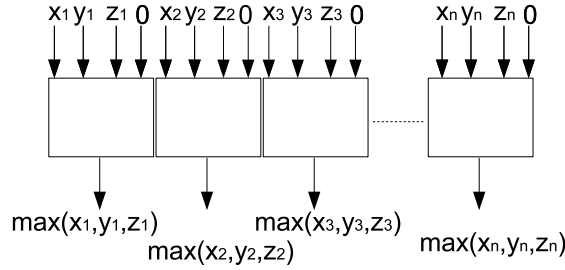


Figure 6.3. An  $n$ -bit 3-input max switch, where the rectangle represents a 1-bit 4-input max switch from Figure 6.2.

To perform addition, one addend is fed into the North-bound of the R-Mesh, and another addend is left-shifted one bit and fed into the West-bound. The left-bit shifting operation removes the bit that represents a zero, which in turn reduces one row of the R-Mesh. Similarly, there is no need to have extra rows in the R-Mesh to perform additions on the right trailing zeros of the second addend. Therefore, the number of rows in the R-Mesh adder/subtractor can be reduced to  $k$ , where  $k + 1$  is the number of 1-bits in the second addend. Each processing unit in the adder/subtractor fuses  $\{NE, SW\}$  if the West input is 1, otherwise, it will fuse  $\{NS, E, W\}$ . The first configuration allows the number to be incremented if there is a 1-bit coming from the West, and the second configuration maps the result directly to the output ports. Figure 6.4 shows the addition of 3 and 3 represented in  $n$ -bit 1UN. In this case, the R-Mesh needs only 3 rows to compute the result. Similarly, for subtractions, the minuend is fed into the South bound (bottom) of the R-Mesh, the subtrahend is 1-bit left-shifted and fed into the R-Mesh from the West bound (left), the East bound (right) is fed with zeros, or no signals. The output is obtained from the North border (top).

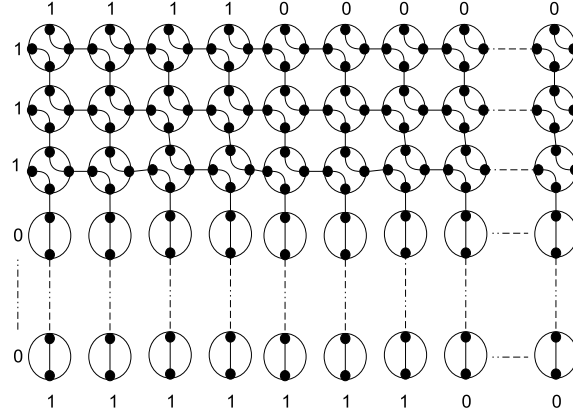


Figure 6.4. An  $n$ -bit adder/subtractor that can perform addition or subtraction between two 1UN numbers during a broadcasting time. For additions the inputs are on the North and West borders and the output is on the South border. For subtractions, the inputs are on the West and South borders and the output is on the North border. The number on the West bound is 1-bit left-shifted. The dotted lines represent the omitted processing units that are the same as the ones in the last rows. This figure shows the addition of 3 and 3. Note: the leading 1 bit of input number on the West-bound (left) has been shifted off. The right border is fed with zero (or no signal) during the subtract operation.

This adder/subtractor can only handle numbers in 1UN representation, i.e. positive values. Thus, any operation that yields a negative result will be represented as a pattern of all zeros. When this adder/subtractor is used in a DP algorithm, one of the two inputs is already known. For example, to calculate the value at cell  $c_{i,j}$ , three binary arithmetic operations must be performed:  $c_{i-1,j-1} + s(x_i, y_j)$ ,  $c_{i-1,j} + g$ , and  $c_{i,j-1} + g$ , where both the gap  $g$  and the symbol matching score  $s(x_i, y_j)$  between any two residue symbols are predefined. Thus, we can store these predefined values to the West ports of the adder/subtractor units and have them configured accordingly before the actual operations.

For biological sequence alignments, symbol matching scores are commonly obtained from substitution matrices such as PAM [23], BLOSUM [45], or similar matrices, and gap cost is a small constant in the same range of the values in these matrices. These values are one or two digits. Thus,  $k$  is very likely is a 2-digit constant or smaller. Therefore, the size of the adder/subtractor unit is bounded by  $O(n)$ , in this scenario.

#### 6.4.3 Constant-Time Dynamic Programming on R-Mesh

The dynamic programming techniques used in the Longest Common Subsequence (LCS), Smith-Waterman's and Needle-Wunsch's algorithms are very similar. Thus, a DP R-Mesh designed to solve one

problem can be modified to solve another problem with minimal configuration. We are presenting the solution for the latter cases first, and then show a simple modification of the solution to solve the first case.

**Smith-Waterman's and Needle-Wunsch's Algorithms** Although the number representation can be converted from one format to another in constant time [120], the DP R-Mesh run-time grows proportionally with the number of operations being done. These operations could be as many as  $O(n^2)$ . To eliminate this format conversion all the possible symbol matching scores, or scoring matrix, (4x4 for RNA/DNA sequences and 20x20 for protein sequences) are pre-scaled up to positive values. Thus, an alignment of any pair of residue symbols will yield a positive score; and gap matching (or insert/delete) is the only operation that can reduce the alignment score in preceding cells. Nevertheless, if the value in cell  $c_{i-1,j}$  (or  $c_{i,j-1}$ ) is smaller than the magnitude of the gap cost ( $g$ ), a subtraction will produce a bit pattern of all zeros (an indication of an underflow or negative value). This value will not appear in cell  $c_{i,j}$  since the addition of the positive value in cell  $c_{i-1,j-1}$  and the positive symbol matching score  $s(x_i, y_i)$  is always greater than or equal to zero.

In general, we do not have to perform this scale-up operation for DNA since DNA/RNA scoring schemes traditionally use only two values: a positive integer value for match and the same cost for both mismatch and gap.

Unlike DNA, scoring protein residue alignment is often based on scoring scoring/substitution/mutation matrices such as that in [23, 45], etc. These matrices are log-odd values of the probabilities of residues being mutated (or substituted) into other residues. The difference between the matrices are the way the probabilities being derived. The smaller the probability, the less likely a mutation happens. Thus, the smallest alignment value between any two residues, including the gap is at least zero. To avoid the complication of small positive fractional numbers in calculations, log-odd is applied on these probabilities. The log-odd score or substitution score in [23] is calculated as  $s(i, j) = \frac{1}{\lambda} \log(\frac{Q_{ij}}{P_i P_j})$ , where  $s(i, j)$  is the substitution score between residues  $i$  and  $j$ ,  $\lambda$  is a positive scaling factor,  $Q_{ij}$  is the frequency or the percentage of residue  $i$  correspond to residue  $j$  in an accurate alignment, and  $P_i$  and  $P_j$  are background probabilities which residues  $i$  and  $j$  occur. These probabilities and the log-odd function to generate the matrices are publicly available via The National Center for Biotechnology Information's web-site <sup>1</sup> along with the substitution matrices themselves. With any given gap cost, the probability of a residue aligned with a gap can be calculated proportionally from a given gap cost and other values from the un-scaled scoring matrices by solving the log-odd function. Thus, when a positive number  $\beta$  is added to the scores in

---

<sup>1</sup><http://www.ncbi.nlm.nih.gov>

these scoring matrices, it is equivalent to multiply the original probabilities by  $a^\beta$ , where  $a$  is the log-based used in the log-odd function.

A simple mechanism to obtain a scaled-up version of a scoring matrix is: (a) taking the antilog of the scoring matrix and  $-g$ , where  $g$  is the gap cost (a positive value) and  $-g$  is the equivalent log-odd of a gap matching probability; (b) multiplying these antilog values by  $\beta$  factor such that their minimum log-odd value should be greater than or equal to zero; (c) performing log-odd operation on these scaled-up values.

When these scaled-up scoring matrices are used, the Smith-Waterman's algorithm must be modified. Instead of setting sub-alignment scores to zeros when they become negative, the scores are set to  $\beta$  if they fall below this scaled-up factor ( $\beta$ ).

We will use the scaled-up scoring matrices to eliminate the representation of signed numbers in our following algorithm designs. However, if there is a need to obtain the alignment score based on the original scoring matrices, all we have to do is score the alignment result using the original scoring matrices.

Having the adder/subtractor units and the switches ready, the dynamic programming R-Mesh, (DP R-Mesh), can be constructed with each cell  $c_{i,j}$  in the DP matrix containing 3 adder/subtractor units and a 3-input max switch allowing it to propagate the max value of cells  $c_{i-1,j-1}$ ,  $c_{i-1,j}$  and  $c_{i,j-1}$  to cell  $c_{i,j}$  in the same broadcasting step. Figure 6.5 shows the dynamic programming R-Mesh construction. The adder/subtractor units are represented as "+" or "-" corresponding to their functions.

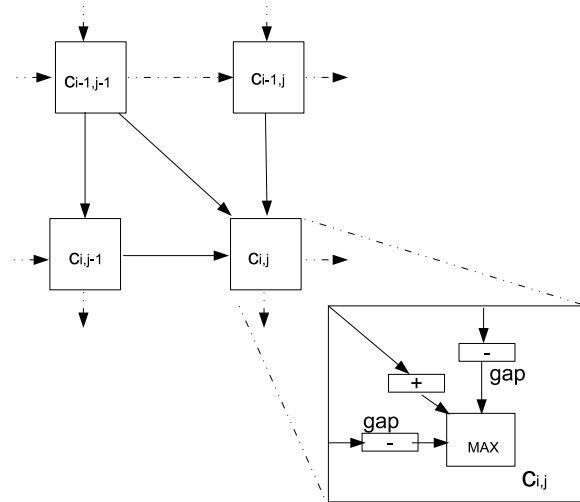


Figure 6.5. A dynamic programming R-Mesh. Each cell  $c_{i,j}$  is a combination of a 3-input max switch and three adder/subtractor units. The "+" and "-" represent the actual functions of the adder/subtractor units in the configuration.

A  $1 \times n$  adder/subtractor unit can perform increments and decrements in the range of  $[-1, 0, 1]$ . As a result, a DP R-Mesh can be built with 1-bit input components to handle all pair-wise alignments using constant scoring schemes that can be converted to  $[-1, 0, 1]$  range. For instance, the scoring scheme for the longest common subsequence rewards 1 for a match and zero for mismatch and gap extension.

To align two sequences,  $c_{i,j}$  loads or computes its symbol matching score for the symbol pair at row  $i$  column  $j$ , initially. The next step is to configure all the adder/subtractor units based on the loaded values and the gap cost  $g$ . Finally, a signal is broadcasted from  $c_{0,0}$  to its neighboring cells  $c_{0,1}$ ,  $c_{1,0}$ , and  $c_{1,1}$  to activate the DP algorithm on the R-Mesh. The values coming from cells  $c_{i-1,j}$  and  $c_{i,j-1}$  are subtracted with the gap costs. The value coming from  $c_{i-1,j-1}$  is added with the initial symbol matching score in  $c_{i,j}$ . These values will flow through the DP R-Mesh in one broadcasting step, and cell  $c_{n,n}$  will receive the correct value of the alignment.

In term of time complexity, this dynamic programming R-Mesh takes a constant time to initialize the DP R-Mesh and one broadcasting time to compute the alignment. Thus, its run-time complexity is  $O(1)$ .

Each cell uses  $10n$  processing units ( $4n$  for the 1-bit max switch and  $2n$  for each of the three adder/subtrator units). These processing units are bounded by  $O(n)$ . Therefore, the  $n \times n$  dynamic programming R-Mesh uses  $O(n^3)$  processing units.

To handle all other scoring schemes,  $k \times n$  adder/subtractor R-Meshes and  $n \times n$  max switches must be used. In addition, to avoid overflow (or underflow) all pre-defined pair-wise symbol matching scores may have to be scaled up (or down) so that the possible smallest (or largest) number can fit in the 1UN representation. With this configuration, the dynamic programming R-Mesh takes  $O(n^4)$  processing units.

**Longest Common Subsequence (LCS)** The complication of signed numbers does not exist in the longest common subsequence problem. The arithmetic operation in LCS is a simple addition of 1 if there is a match. The same dynamic programming R-Mesh as seen in figure 6.5 can be used, where the two subtractors units are removed or the gap cost is set to zero ( $g = 0$ ).

To find the longest common subsequence between two sequences  $x$  and  $y$ , each max switch in the DP R-Mesh is configured as in Figure 6.6. The value from cell  $c_{i-1,j-1}$  is fed into the North-West processing unit, and the other values are fed into the North-East unit. Then,  $c_{i,j}$  loads in its symbols and fuses the South-East processing unit (in bold) as NS, E, W if the symbols at row  $i$  and column  $j$  are different; otherwise, it loads 1 into the adder unit and fuses N,E,SW. These configurations allow either the value from cell  $c_{i-1,j-1}$  or the max value of cells  $c_{i-1,j}$  and  $c_{i,j-1}$  to pass through. These are the only changes for the DP R-Mesh to solve the LCS problems.

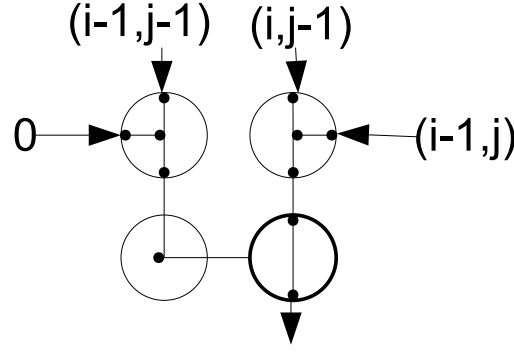


Figure 6.6. A configuration of a 4-way max switch to solve the longest common subsequence (lcs). The South-East processing unit (in bold) configures NS, E, W if the symbols at row  $i$  and column  $j$  are different; otherwise, it configures N,E,SW. This figure shows a configuration when the two symbols are different.

This modified constant-time DP R-Mesh used  $O(n^3)$  processing units. However, this is an order of reduction comparing the current best constant parallel DP algorithm that uses an R-Mesh of size  $O(n^2) \times O(n^2)$  [10] to solve the same problem.

#### 6.4.4 Affine Gap Cost

Affine gap cost (or penalty) is a technique where the opening gap has different cost from an extending gap [41]. This technique discourages multiple and disjoint gap insertion blocks unless their inclusion greatly improves the pair-wise alignment score. The gap cost is calculated as  $p = o + g(l - 1)$ , where  $o$  is the opening gap cost,  $g$  is the extending gap cost, and  $l$  is the length of the gap block. To handle affine gap cost, we need to extend the representation of the number by 1 bit (right most bit). This bit indicates whether a value coming from  $c_{i-1,j}$  or  $c_{i,j-1}$  to  $c_{i,j}$  is an opening gap or not. If the incoming value has been gap-penalized, its right most bit is 1, and it will not be charged with an opening gap again; otherwise, an opening gap will be applied. The original "-" units must be modified to accommodate affine gaps. Figure 6.7 shows the modification of the "-" unit. The output from the original "-" unit is piped into an  $n \times n + 1$  R-Mesh on/off switch (described in Section 6.4.5), an adder/subtractor, and a max switch. When a number flows through the "-" unit, an extending gap is applied. If the incoming value has not been charged with gap to begin with, its right most bit (i.e. selector bit denoted as "s") remains zero, which keeps the switch in off position. Therefore, the value with extra charge on the adder/subtractor is allowed to flow through; otherwise, the switch will be on, and the larger value will be selected by the max

switch. A value is not from diagonal cells must have its selector bit set to 1 (right most bit) after a gap cost is applied to prevent multiple charges of an opening gap.

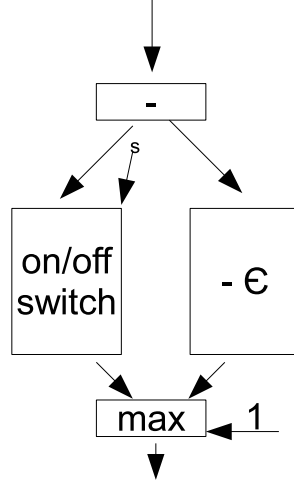


Figure 6.7. A configuration to select one of the two inputs in 1UN notation using the right most bit as a selector  $s$ . When  $s = 1$  the switch is turned on to allow the data to flow through and get selected by the max switch. When the selector  $s = 0$ , the on/off switch produces zeros and the other data flow will be selected.  $\epsilon = o - g, o \geq g$ , is the difference between opening gap cost  $o$  and extending gap cost  $g$ .

The modification of the dynamic programming R-Mesh to handle affine gap cost requires additional 2 adder/subtractor units, 2 on/off switches, and one 2-input max switch. Asymptotically, the amount of processing units used is still bounded by  $O(n^4)$  and the run-time complexity remains  $O(1)$ .

#### 6.4.5 R-Mesh On/Off Switches

To handle affine gap cost in dynamic programming, we need a switch that can select, i.e. turn on or off, the output ports of a data flow. The on/off R-Mesh switch can be configured as in Figure 6.8, where the input value is mapped into the North-bound (top). The right most bit of the input is served as a selector bit. The R-Mesh size is  $n \times n + 1$ , where column  $i$  fuses with row  $n - i$  to form an L-shape path that allows the input data from the Northbound to flow to the output port on the Eastbound. The processors on the last column will fuse the East-West ports allowing the input to flow through only if they receive a value of 1 from the input (Northern ports). Since the selector bit travels a shorter distance than all the other input bits, it will arrive in time to activate the opening or closing of the output ports.

This R-Mesh configuration uses  $(n \times n + 1)$ , i.e.,  $O(n^2)$ , processing units to turn off the flow of an  $n$ -bit input in a broadcasting time.

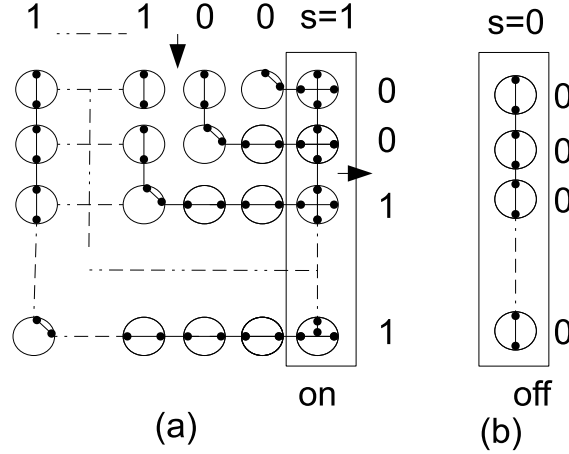


Figure 6.8. An  $n \times n + 1$  R-Mesh represents an  $n$ -bit on/off switches. By default, all processing units on the last column (column  $n + 1$ ) configure  $\{NS, E, W\}$ , and fuse  $\{NSEW\}$  when a signal (i.e. 1) travels through them. All cells on the main anti-diagonal cells of the first  $n$  rows and columns fuse  $\{NE, S, W\}$ , cells above the main anti-diagonal fuse  $\{NS, E, W\}$ , and cells below the main anti-diagonal fuse  $\{N, S, EW\}$ . Figure (a) shows the R-Mesh configuration on a selector bit of 1 ( $s=1$ ) and Figure (b) shows the R-Mesh configuration on a selector bit of 0 ( $s=0$ ).

#### 6.4.6 Dynamic Programming Back-tracking on R-Mesh

The pair-wise alignment is obtained by following the path leading to the overall optimal alignment score, or the end of the alignment. In the case of the Needleman-Wunsch's algorithm, cell  $c_{n,n}$  holds this value; and in the case of the Smith-Waterman's algorithm, cell  $c_{i,j}$  with the maximum alignment score is the end point. The cell with the largest value can be located in  $O(1)$  time on a 3-dimension  $n \times n \times n$  R-Mesh through these steps:

1. Initially, the DP matrix with calculated values are stored in the first slice of the R-Mesh cube, i.e. in cells  $c_{i,j,0}$ ,  $0 < i, j \leq n$ .
2.  $c_{i,j,0}$  sends its value to  $c_{i,j,i}$ ,  $0 \leq i, j \leq n$ , to propagate each column of the matrix to the 2D R-Meshes on the third dimension.
3.  $c_{i,j,i}$  sends its value to  $c_{0,j,k}$ , i.e. to move the solution values to the first row of each 2D R-Mesh slice.
4. Each 2D R-Mesh slice finds its max value  $c_{0,m,k}$  where  $m$  is the column of the max value in slice  $k$
5.  $c_{0,m,k}$  sends  $m$  to  $c_{k,0,0}$ , i.e. each 2D R-Mesh slice sends its max value column number  $m$  to the first 2D R-Mesh slice. This value is the column index of the max value on row  $k$  in the first slice.
6. The first 2D R-Mesh slice,  $c_{i,j,0}$ , finds the max value of  $n$  DP R-Mesh cells whose row index is  $i$  and



column index is  $c_{i,0,0}$  (i.e. value  $m$  received from the previous step). The row and column indices of the max value found in this step are the coordinates of the max value in the original DP R-Mesh.

These above steps rely on the capability to find the max value from  $n$  given numbers on an  $n \times n$  R-Mesh. This operation can be done in  $O(1)$  time as follows:

1. initially, the values are stored in the first row of the R-Mesh.
2.  $c_{0,j}$  broadcasts its value, namely  $a_j$ , to  $c_{i,j}$ , (column-wise broadcasting).
3.  $c_{i,i}$  broadcasts its value, namely  $a_i$ , to  $c_{i,j}$  (row-wise broadcasting).
4.  $c_{i,j}$  sets a flag bit  $f(i,j)$  to 1 if and only if  $a_i > a_j$ ; otherwise sets  $f(i,j)$  to 0.
5.  $c_{0,j}$  is holding the max value if  $f(0,j), f(1,j), \dots, f(n-1,j)$  are 0. This step can be performed in  $O(1)$  time by ORing the flag bits in each column.

The location of the max value in the DP R-Mesh can be obtained in  $O(1)$  time because each step in the process takes  $O(1)$  time to complete.

To trace back the path leading to the optimal alignment, we start with the end point cell  $c_{e,d}$  found above and following these steps:

1.  $c_{i,j}$ , ( $i \leq e, i \leq d$ ), sends its value to  $c_{i,j+1}, c_{i+1,j}, c_{i+1,j+1}$ . Thus, each cell can receive up to three values coming from its Noth, West, and Northwest borders.
2.  $c_{i,j}$  finds the max of the inputs and fuses its port to the neighbor cell that sent the max value in the previous step. If there are more than one port to be fused, (this happens when there are multiple optimal alignments),  $c_{i,j}$  randomly selects one.
3.  $c_{e,d}$  sends a signal to its fused port. The optimal pair-wise alignment is the ordered list of cells where this signal travels through.

Each operation in the back-tracking process takes  $O(1)$  time to complete, and there are no iterative operations. Therefore, the back-tracking steps requires  $n^3$  processing units and takes  $O(1)$  time.

## 6.5 Progressive Multiple Sequence Alignment on R-Mesh

In this section, we start by describing a parallel algorithm to generate a dendrogram, or guiding tree, representing the order in which the input sequences should be aligned. Then we will show a reworked version of sum-of-pair scoring method that can be performed in constant time on a 2D R-Mesh. Finally, we will describe our parallel progressive multiple sequence alignment algorithm on R-Mesh along with its complexity analysis.

### 6.5.1 Hierarchical Clustering on R-Mesh

The parallel neighbor-joining (NJ) [98] clustering method on R-Mesh is described here; other hierarchical clustering mechanisms can be done in a similar fashion. The neighbor-joining takes a distance matrix between all the pairs of sequences and represents it as a star-like connected tree, where each sequence is an external node (leaf) on the tree. NJ then finds the shortest distance pair of nodes and replaces it with a new node. This process is repeated until all the nodes are merged.

Followings are the actual steps to build the dendrogram:

1. Initially, all the pair-wise distances are given in form of a matrix  $D$  of size  $n \times n$ , where  $n$  is the number of nodes (or input sequences).
2. Calculate the average distance from node  $i$  to all the other nodes by  $r_i = \frac{\sum_1^n D_{ij}}{n-2}$ .
3. The pair of nodes with the shortest distance  $(i, j)$  is the pairs that gives minimal value of  $M_{ij}$ , where  $M_{ij} = D_{ij} - r_i - r_j$ .
4. A new node  $u$  is created for shortest pair  $(i, j)$ , and the distances from  $u$  to  $i$  and  $j$  are:  $d_{iu} = \frac{D_{ij}}{2} + \frac{(r_i - r_j)}{2}$ , and  $d_{ju} = d_{ij} - d_{iu}$ .
5. The distance matrix  $D$  is updated with the new node  $u$  to replace the shortest distance pair  $(i, j)$ , and the distances from all the other nodes to  $u$  is calculated as  $D_{vu} = D_{iv} + d_{ju} - D_{ij}$ .

These steps are repeated for  $n - 1$  iterations to reduce distance matrix  $D$  to one pair of nodes. The last pair does not have to be merged, unless the actual location of the root node is needed.

Step 1 and 4 are constant time operations on an  $n \times n$  R-Mesh, where each processing unit stores a corresponding value from the distance matrix. Since the progressive multiple sequence alignment algorithm only uses the dendrogram as a guiding tree to select the closest pair of sequences (or two groups of sequences), the actual distance values between the nodes on the final dendrogram mostly are insignificant. Consequently, the values in distance matrix  $D$  can be scaled down without changing the order of the nodes in the dendrogram. In addition, if these values are not to be preserved, the calculations in step 4 can be skipped.

Before proceeding to step 2, we should reexamine some facts. First, the maximum alignment score from all the pair-wise DP sequence alignments are bounded by  $b^2$ , where  $b$  is the max pair-wise score between any two residue symbols. An alignment score of  $b^2$  occurs only if we align a sequence of these symbols to itself.  $b + 1 \leq n$  is the number of bits being used to represent this value in 1UN. Similarly, the maximum value in distance matrix  $D$  generated from these alignment scores are also bounded by  $b^2$ .

Thus, the sum of  $n$  of these distance values is bounded by  $b^4$ . These facts allow us to calculate the sums in step 2 in  $O(c)$  time using an  $n \times n$  R-Mesh as in Theorem 1. In this case,  $c$  is constant, ( $c = 4$ ). There are  $n$  summations to calculate, so the entire calculation requires  $n$  such R-Meshes, or  $n^3$  processing units, to complete in  $O(1)$  time.

In step 3, each processing unit computes value  $M_{ij}$  locally. The max value can be found using the same technique described in Section 6.4.6 in constant time.

Similarly, step 5 is performed locally by the processing units in the R-Mesh in  $O(1)$  time. Since this procedure terminates after  $n - 1$  iterations, the overall run-time complexity to build a dendrogram, (or guiding tree), for any given pair-wise distance matrix of size  $n \times n$  is  $O(n)$ , using  $O(n^3)$  processing units.

### 6.5.2 Constant Run-time Sum-of-Pair Scoring Method

The third step [step (iii)] of the progressive MSA algorithm is following the dendrogram, built in the earlier step, to perform pair-wise dynamic programming alignment on two pre-aligned groups of sequences. The dynamic programming alignment algorithm in this step is exactly the same as the one in step (i); however, quantifying a match between two columns of residues are no longer a simple constant look-up, unless the hierarchical expected probability (HEP) matching scoring scheme is used [82]. Most multiple sequence alignment algorithms use the popular sum-of-pair (SP) scoring method [15]. This quantification is the sum of all pair-wise matching scores between the residue symbols, where each paired-score is obtained either from a substitution matrix or from any scoring scheme discussed earlier. The alignment at the root of the tree gets  $n$  residues for every pair of columns to be quantified. Thus, there are  $\frac{n(n-1)}{2}$  lookups per column quantification, i.e.  $\frac{n(n-1)}{2}$  lookups for each DP matrix cell calculation. The sum-of-pair is formally defined as:

$$sp(f, g) = \sum_{i=1}^{|f|} \sum_{j=i+1}^{|g|} s(f_i, g_j) \quad (6.1)$$

where  $f$  is a column from one pre-aligned group of sequences and  $g$  is a column from the other pre-aligned group of sequences.  $f_i$  and  $g_j$  are residue symbols from columns  $f$  and  $g$ , respectively, and  $s(f_i, g_j)$  is the matching score between these two symbols  $f_i$  and  $g_j$ .

For example, to calculate the sum-of-pair of the following two columns  $f$  and  $g$ :

$$\text{Column } f: \begin{Bmatrix} A \\ C \\ T \end{Bmatrix}$$

and

$$\text{Column } g: \begin{Bmatrix} G \\ T \\ T \end{Bmatrix}$$

we will have to score 15 residue pairs: (A,C), (A,T), (A,G), (A,T), (A,T), (C,T), (C,G), (C,T), (C,T), (T,G), (T,T), (T,T), (G,T), (G,T), (T,T). Since the matching between residue  $a$  to residue  $b$  is the same as the matching between residue  $b$  to residue  $a$ , these pairs become (A,C), 3(A,T), (A,G), 3(C,T), (C,G), 3(G,T), 3(T,T). These redundancies occur since the set of symbols representing the residues is small (1 for gap plus 20 for protein [or 4 for DNA/RNA]). Thus, if we combine the two column symbols with their number of occurrences, the sum-of-pair method can be transformed into a counting problem and can be defined as:

$$sp(f, g) = \sum_{i=1}^T \frac{n_i(n_i - 1)}{2} s(i, i) + n_i \sum_{j=i+1}^T n_j \times s(i, j) \quad (6.2)$$

where  $f, g$  are the two columns,  $T$  is the number of different residue symbols ( $T=4$  for DNA/RNA and  $T=20$  for proteins),  $s(i, j)$  is the pair-wise matching score, or substitution score, between two residue symbols  $i$  and  $j$ , and  $n_i$  and  $n_j$  are the total count of symbols  $i$  and  $j$  (i.e. the occurrences of residue symbols  $i$  and  $j$ ), respectively. Since  $T$  is constant, the summations in Equation 6.5.2 remain constant, regardless how many sequences are involved.

Thus, the sum-of-pair score of the two columns given above will be:

$$\frac{3(3-1)}{2} s(T, T) + [s(A, C) + s(A, G) + 3s(A, T) + s(C, G) + 3s(C, T) + 3s(G, T)]$$

This scoring function can be implemented on an array of  $n$  processing units as follows. First, map each residue symbol into a numeric value from 1 to  $T$ . Next,  $n$  residues from any two aligning columns are assigned to  $n$  processing units. Any processing unit holding a residue sends a 1 to processing unit  $p_k$ , where  $k$  is the number represents the residue symbol it is holding.  $p_k$  sums the 1's it receives. The sum-of-pair score can be computed between the pairs of processing units containing a sum larger than 0 calculated from previous steps. All of these steps are carried out in constant time. There are  $n^2$  possible pair-wise column arrangements of two pre-aligned groups of sequences of length  $n$ . Thus, the sum-of-pair

column pair-wise matching scores for two pre-aligned groups of sequences can be done in  $O(1)$  using  $n^3$  processing units.

### 6.5.3 Parallel Progressive MSA Algorithm and Its Complexity Analysis

Progressive multiple sequence alignment algorithm is a heuristic alignment technique that builds up a final multiple sequence alignment by combining pair-wise alignments starting with the most similar pair and progressing to the most distant pair. The distance between the sequences can be calculated by dynamic programming algorithms such as Smith-Waterman's or Needle-Wunsch's algorithms (step i). The order in which the sequences should be aligned are represented as a guiding and can be calculated via hierarchical clustering algorithms similar to the one described in Section 6.5.1 (step ii). After the guiding tree is completed, the input sequences can be pair-wise aligned following the order specified in the tree (step iii).

In the previous Sections, we have described and designed several R-Meshes to handle individual operations in the progressive multiple alignment algorithm. Finally, a progressive multiple sequence alignment R-Mesh configuration can be constructed. First, the input sequences are pair-wise aligned using the dynamic programming R-Mesh described previously in Section 6.4. These  $\frac{n(n-1)}{2}$  pair-wise alignments can be done in  $O(1)$  using  $\frac{n(n-1)}{2}$  dynamic programming R-Meshes, or in  $O(n)$  time using  $O(n)$  R-Meshes. The latter is preferred since the dendrogram [step (ii)] and the progressive alignment [step (iii)] steps each takes  $O(n)$  time to complete. Then, a dendrogram is built, using the parallel neighbor-joining clustering algorithm described earlier, from all the pair-wise DP alignment scores from step (i). Lastly, [step (iii)], for each pair of pre-aligned groups of sequences along the dendrogram, the sum-of-pair column matching scores must be pre-calculated to initialize the DP R-Mesh before proceeding with the dynamic programming alignment. There are  $n - 1$  branches in the dendrogram leading to  $n - 1$  pair-wise group alignments to be performed.

In terms of complexity, the progressive multiple sequence alignment takes  $O(n)$  time using  $O(n)$  DP R-Meshes to complete all the pair-wise sequence alignments [step (i)] - (or  $O(1)$  time using  $\frac{n(n-1)}{2}$  DP R-Meshes). Its consequence step, [step (ii)], to build the sequence dendrogram takes  $O(n)$  time using  $O(n^3)$  processing units. Finally, the progressive step, [step (iii)], takes  $O(n)$  time using a DP R-Mesh. Therefore, the overall run-time complexity of this parallel progressive multiple sequence alignment is  $O(n)$ .

The number of processing units utilized in this parallel algorithm is bounded by the number of DP R-Meshes used and their sizes. The general DP R-Mesh uses  $O(n^4)$  processing units to handle all scoring schemes with affine gap cost. And step (i) needs  $n$  of such DP R-Meshes resulting in  $O(n^5)$  processing units used.

For alignment problems that use constant scoring schemes without affine gap cost, this parallel progressive multiple sequence alignment algorithm only needs  $O(n^4)$  processing units to complete in  $O(n)$  time.

Table 6.1 summarizes the R-Mesh size and the run-time complexity of various components in this study, where the components with "broadcast" run-time can finish their operations in one broadcasting time. The "DP" R-Mesh is designed to handle all the Needleman-wunsch's [77], Smith-Waterman's [107], and Longest Common Subsequence algorithms.

Table 6.1. Summary of Progressive Multiple Sequence Alignment Components along with their time and CPU complexity

Component	input size	processors	run-time
2-input max switch	$1 - bit$	1	broadcast
4-input max switch	$1 - bit$	4	broadcast
2-input max switch	$n - bit$	$n$	broadcast
4-input max switch	$n - bit$	$4n$	broadcast
on/off switch	$n - bit$	$n \times n + 1$	broadcast
adder/subtractor	$n$	$k \times n, k \leq n$	broadcast
DP(const. scoring)	2 sequences, max length = $n$	$O(n^3)$	broadcast
DP	2 sequences, max length = $n$	$O(kn^3), k \leq n$	broadcast
DP back-tracking	$n \times n$	$n \times n \times n$	$O(1)$
Neighbor-Joining	$n \times n$	$O(n^3)$	$O(n)$
Sum-of-pair	2 pre-aligned groups of sequences	$n^3$	$O(1)$
MSA(const. scoring)	$n$ sequences, max length = $n$	$O(n^4)$	$O(n)$
MSA	$n$ sequences, max length = $n$	$O(n^5)$	$O(n)$

## Chapter 7

### WEB-SERVER FOR SEQUENCE ANALYSIS IMPLEMENTATION

In the current information age, it is beneficial to have web-services available for public and private use, especially services in biological sequence analysis. There are many web-servers around the world have been built both from government sponsored and private organizations and individuals. In this chapter we are describing a new biological sequence analysis that has been built during this research. Figure 7 shows the front page of the web-server. Most of the sequence tools are obtained from BioPHP.org and modified to work on SeqAna server. The multiple sequence alignment software is from Ubuntu public depository and in-house development. These services are self-explanatory.

#### 7.1 SeqAna: A New Sequence Analysis Web-Server

SeqAna is a web-server that provides many state of the arts protein/DNA/RNA sequence analysis tools such as sequence search, (Blast, FASTA), pairwise sequence alignment (Needle-Wunsch, Smith-Waterman), multiple sequence alignment (MSA) such as ClustalW, MAFFT, ProbCons, Poa, etc. SeqAna provides many unique and helpful features that none of the existing sequence analysis servers ever provide. For example, SeqAna provides a multiple sequence alignment scoring and ranking service. This service is the only of its kind that allows users to perform multiple sequence alignment (MSA) via many alignment tools and rank the results of these alignments. With this service, users are able to identify which alignment tools are more appropriate for their specific set of sequences. SeqAna's users can provide a reference alignment in many format such as Fasta, MSF, PILEUP, etc for ranking and scoring their alignment results. If

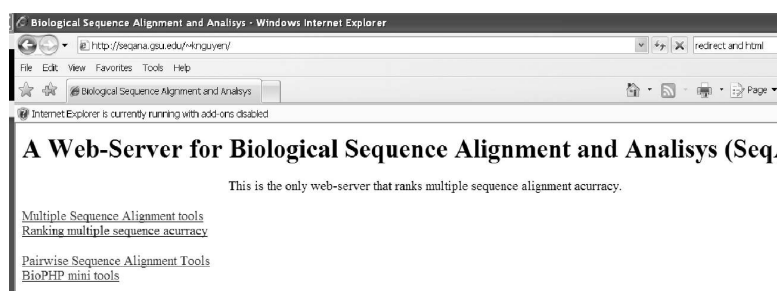


Figure 7.1. Front page of SeqAna Web-Server



an alignment reference is not available, the users can obtain an alignment by using one of the alignment services provided by SeqAna web-server, edit the alignment result to fit their expectation, and use it as a reference alignment. SeqAna also has a built-in feature to automatically select the best applicable MSA algorithms to align any given set of input sequences. With this service, scientists are guaranteed to get the best multiple sequence alignment results without spending days, or even weeks, to try many different MSA tools. SeqAna also has many other sequence analysis services. SeqAna is publicly available at <http://seqana.gsu.edu>.

## 7.2 Multiple Sequence Alignment Ranking

SeqAna provides a multiple sequence alignment scoring and ranking services. SeqAna multiple sequence alignment ranking web-service is very unique, where its users can perform multiple sequence alignment (MSA) using many alignment tools and rank the results of these alignments. With this service, the users are able to identify which alignment tools are the most appropriate for their specific set of sequences. By default, the ranking is either automatic or by comparing the results with a reference alignment. The closest service to SeqAna's multiple sequence alignment ranking is provided by the National Institutes of Health (NIH) on a super-computer at <http://helixweb.nih.gov/multi-align/>. NIH web-service that allows its users to align a set of sequences by varies multiple alignment methods at once; however, the users have to sort through the outputs to identify the best one. In contrary, SeqAna will rank these results and show the users the best alignment result among those that are returned from many different alignment tools. Figure 7.2 shows a web-interface of SeqAna multiple sequence alignment ranking service. The ranking service has two modes: (a) user-selected and (b) best-predicted. In the user-selected mode, the users can select the alignment tools they want to perform on their input sequences. The system will invoke these tools one by one and then score and rank their result based on best performance first. In the best-predicted mode, the server will randomly choose a small sample set of input sequences and invoke all available multiple sequence alignment tools on the sample sequences. For alignment jobs with a few input sequences, the entire input data set will be used. The alignment results will be ranked to identify the best result. The alignment tool with corresponding best result will be chosen to perform alignment service on the entire input sequences.

By default, the ranking on peptide sequences utilizes Hierarchical Expected matching Probability (HEP) scoring function as described in Section 3.9 and Sum-of-Pair (SP) as described in Section 3.4.1. If

## Multiple Sequence Alignment Ranking

**Input Sequences**  

```
>seq1
GGAGTGAGGGGAGCAGTTGGGAACAGATGG
>seq2
AGCTGTGGCAGACCTGGCTTCTTAACACG
>seq3
GACGCTCAGATCCAAACGAAGCTGAGAAACAGCTTCTTC
```

**Or**  
 Upload your sequence file:   Max size = 10 MB

Reference Alignment (optional):   Max size = 10 MB

Select the MSA tools you want to include:  
☐ Amap ☐ Clustal ☐ DiAlign ☐ Mafft ☐ Muscle  
☐ Poa ☐ ProbCons ☐ Tcoffee

Please select output format (default is Fasta):

Figure 7.2. Multiple Sequence Alignment Ranking Service

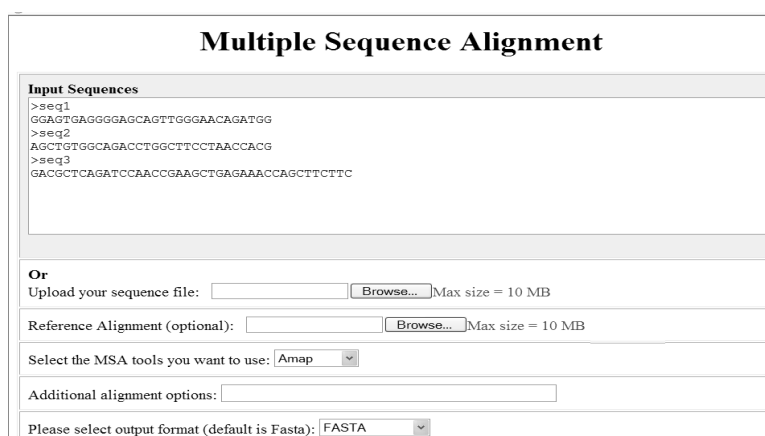
the user provides a reference alignment, the server will use the Total Column score (TC), that is provided along with BaliBASE [115] and PREFAB [30] benchmarks, to rank the alignment results.

### 7.3 Multiple Sequence Alignment Analysis

Currently the following multiple sequence alignment tools are available:

- AMAP
- CLUSTALW
- DIALIGN
- MAFFT
- MUSCLE
- PADT
- POA
- PROBCONS

More alignment tools will be added as more wrappers are developed. The scoring of these alignment service is similar to those in Section 7.2. Figure 7.3 shows the web-interface of this alignment service.



**Multiple Sequence Alignment**

**Input Sequences**

```
>seq1
GGAGTGAGGGGAGCAGTTGGGAACAGATGG
>seq2
AGCTGTGGCAGACCTGGCTTCTTAACCACG
>seq3
GACGTCAGATCCCAACGAAAGCTGAGAAACAGCTTCTTC
```

**Or**

Upload your sequence file:   Max size = 10 MB

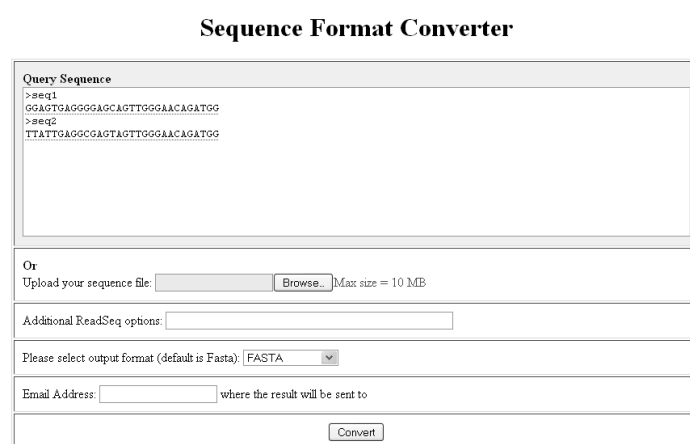
Reference Alignment (optional):   Max size = 10 MB

Select the MSA tools you want to use:

Additional alignment options:

Please select output format (default is Fasta):

Figure 7.3. Multiple Sequence Alignment Service



**Sequence Format Converter**

**Query Sequence**

```
>seq1
GGAGTGAGGGGAGCAGTTGGGAACAGATGG
>seq2
TTATTAGGCCAGTAGTTGGGAACAGATGG
```

**Or**

Upload your sequence file:   Max size = 10 MB

Additional ReadSeq options:

Please select output format (default is Fasta):

Email Address:  where the result will be sent to

Figure 7.4. Sequence Format Converting Service

## 7.4 Multiple Sequence Format Converter

There are many reasons to convert the sequence from one format to another. For example, one may want to have the conservation blocks between the sequences to be clearly annotated, to use the sequences as input for sequence analysis tools that require a specific input format, or to store the sequences with minimal space. SeqAna format converting web-service utilizes **readseq** package written by D.G. Gilbert and publicly available <http://iubio.bio.indiana.edu/soft/molbio/readseq/>. This web-service allows users to either paste in or uploaded a file containing nucleic or peptide sequences in many popular formats and convert them to any other available format as listed on the site. Figure 7.4 show the web-interface of this sequence formatting services.

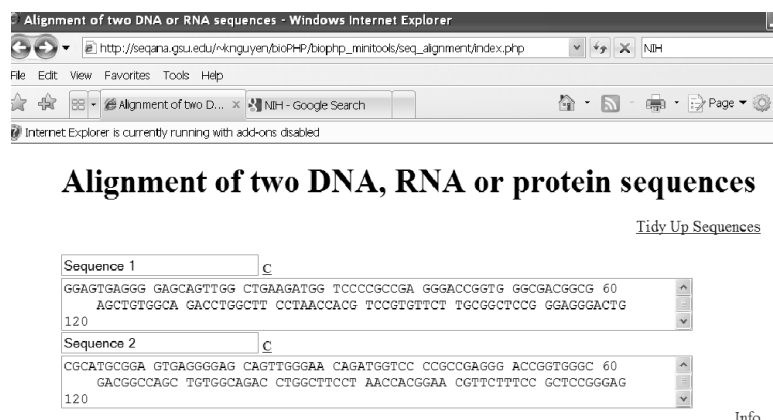


Figure 7.5. Pair-wise Alignment Service

## 7.5 Pairwise Sequence Analysis

To perform pairwise alignment, the users just cut and paste their two sequences in the two given boxes and press the alignment button. Both Needleman-Wunsch global alignment and Smith-Waterman's local alignment are available.

## 7.6 Sequence Retrieval and other services

For sequence homologous search, we installed NCBI BLAST2 on SeqAna web-server. A full BLAST service with sequence databases of different species is available at <http://blast.ncbi.nlm.nih.gov/Blast.cgi>. SeqAna's BLAST service allows its users to perform BLAST on their own specific two sequences: the query sequence and the database sequence without the need of setting up the database and the BLAST installation. SeqAna's BLAST web-interface is shown in Figure 7.6.

Due to SeqAna physical constraints, biological sequence databases are not made available to public until the SeqAna is moved onto a more capable machine.

SeqAna will send an email containing the output of the job to its user's email, if provided. This is a preferred method for jobs with large input sequences.

## BLAST - Between Two sequences in FASTA format

*Click [HERE](#) To BLAST against NCBI's databases*

<b>Query Sequence</b>	
<pre>&gt;seq1 GGAATTAAGGGGACCAATTTGGGAACTAGATGG</pre>	
<b>Target Sequence (i.e database)</b>	
<pre>&gt;seq2 GGAATTAAGGGGACCAATTTGGGAACTAGATGG</pre>	
<b>Or</b>	
Upload your sequence file:	<input type="text"/> <input type="button" value="Browse..."/> Max size = 10 MB <b>Note:</b> if your file contain > 2 sequences, the first sequence will be "BLASTed" against the second
Target Sequence(database):	<input type="text"/> <input type="button" value="Browse..."/> Max size = 10 MB
Additional BLAST options:   <input type="text"/>	
Email Address:	<input type="text"/> where the result will be sent to
<input type="button" value="BLAST"/>	

Figure 7.6. BLAST Service

## Chapter 8

### CONCLUSIONS AND FUTURE RESEARCH DIRECTION

Multiple sequence alignment is a very fundamental activity in sequence analysis. With the current advancement of sequencing techniques and available biological sequence information, it is expected that multiple sequence alignment can handle large number of sequences and produce reliable and biologically meaningful results. However, multiple sequence alignment is an NP-Complete problem thus it cannot handle large amount of data and resolve the problem in practical amount of time. In this study, we have investigated many techniques that have been designed and developed to overcome these issues. As a result, we are able to contribute a significant improvement to the biological sequence analysis, especially in the aspect of multiple sequence alignment. In summary, our contributions include:

- A rigorous scoring function for multiple sequence alignment based on residue matching probability and amino acid class covering hierarchy (see Section 3.9).
- An overlapping clustering algorithm that can be utilized in multiple sequence alignment (see Section 4.3).
- Three new multiple sequence alignment algorithms: dynamic guiding tree, knowledge-based, and consistency-based algorithms (see Sections 5.5.1 and 5.5.2).
- Three different pair-wise alignment algorithms (Smith-Waterman's, Needle-Wunsch's, and LSC) via dynamic programming on reconfigurable mesh (see Section 6.4).
- A parallel progressive multiple sequence alignment on reconfigurable mesh (see Section 6.5).
- An implementation of a web-server for sequence analysis (see Chapter 7).

Throughout many years we have worked on this research, we have noticed that multiple biological sequence alignment problem have been well-studied and developed. The current and future tasks would be to improve and apply multiple sequence alignment techniques to large datasets. Since most of the practical solutions to alignment problem are heuristic, it is best to incorporate sequence knowledge database into the algorithms to derive more biological meaningful results. And that is the direction we are heading. We are going to develop a complete sequence analysis framework that utilized multiple sequence alignment and

sequence clustering to characterize and combine current sequence databases into a homologous hierarchical structure. This structure will allow us to perform homologous searches, multiple sequence alignments and phylogeny tree constructions effectively and efficiently by targeting groups of sequences that are homologous and meaningful to the sequences being analyzed.

We are planning to define a new scoring function based on biological structure and motif for multiple sequence alignment. Using this biological scoring function we can assess the actual biological reliability of an alignment result.

The development of this biological structure and motif scoring function will lead to different multiple sequence alignment models and techniques, where sequence structures and motifs are primary factors in sequence alignment.

## REFERENCES

- [1] Y Akiyama, T Hosoya, AM Poole, and Y Hotta. The gcm-motif: a novel dna-binding motif conserved in drosophila and mammals. *Proc Natl Acad Sci.*, 25:14912–14916, 1996.
- [2] S.F. Altschul. Amino Acid Substitution Matrices from an Information Theoretic Perspective. *J. Mol. Bd*, 219:555–565, 1991.
- [3] Stephen F. Altschul<sup>1</sup>, Warren Gish<sup>1</sup>, and et. al. Basic local alignment search tool. *J. Mol. Biol*, 215:403–410, 1990.
- [4] Srinivas Aluru, Natsuhiko Futamura, and Kishan Mehrotra. Parallel biological sequence comparison using prefix computations. *Journal of Parallel and Distributed Computing*, 63(3):264 – 272, 2003.
- [5] A. Armon, D. Graur, and N. Ben-Tal. ConSurf: an algorithmic tool for the identification of functional regions in proteins by surface mapping of phylogenetic information. *J. Mol. Biol*, 307(1):447–463, 2001.
- [6] A Bairoch, R Apweiler, and et al. The universal protein resource (uniprot). *Nucleic Acids Res.*, 33:154–159, 2005.
- [7] Pierre Baldi and Yves Chauvin. Smooth on-line learning algorithms for hidden markov models. *Neural Comput.*, 6(2):307–318, 1994.
- [8] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucl. Acids Res.*, 34(suppl 1):D16–20, 2006.
- [9] H. M. Berman and et al. The protein data bank - (pdp). *Nucleic Acids Res.*, 28:235–242, 2000.
- [10] Alan A. Bertossi and Alessandro Mei. Constant time dynamic programming on directed reconfigurable networks. *IEEE Transactions on Parallel and Distributed Systems*, 11:529–536, 2000.
- [11] B Boeckmann, A Bairoch, R Apweiler, and et al. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Res.*, 31:365–70, 2003.
- [12] Anu G. Bourgeois and Jerry L. Trahan. Relating two-dimensional reconfigurable meshes with optically pipelined buses. *Parallel and Distributed Processing Symposium, International*, 0:747, 2000.



- [13] N. Bray, I. Dubchak, and L. Pachter. Avid: a global alignment program. *Cambridge University Press, Cambridge, UK*, 13:97–102, 2003.
- [14] L Cai, D Juedes, and E. Liakhovitch. Evolutionary computation techniques for multiple sequence alignment. *Proceedings of the Congress on Evolutionary Computation*, 2:829–835, 2000.
- [15] H Carillo and D. Lipman. The Multiple Sequence Alignment Problem in Biology. *SIAM Journal of Applied Math.*, 48(5):1073–1082, 1988.
- [16] Bhardwaj-N. Anand A. P. Chakrabarti, S. and R. Sowdhamini. Improvement of alignment accuracy utilizing sequentially conserved motifs (falign). *BMC Bioinformatics*, 5:167–, 2004.
- [17] K. Chellapilla and G. B. Fogel. Multiple sequence alignment using evolutionary programming. *Congress on Evolutionary Computation*, pages 445–452, 1999.
- [18] C. Chothia and A.M. Lesk. The relation between the divergence of sequence and structure in proteins. *EMBO J.*, 5:823–826, 1986.
- [19] Carlos Alberto Cordova-Flores, Jose Alberto Fernandez-Zepeda, and Anu G. Bourgeois. Constant time simulation of an r-mesh on an lr-mesh. *Parallel and Distributed Processing Symposium, International*, 0:269, 2007.
- [20] J Cornfield. A applications to cancer of the lung, breast, and cervix. *Journal of the National Cancer Institute*, 11:1269–1275, 1951.
- [21] F. CORPET. Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.*, 16(22):10881–10890, 1988.
- [22] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. pages 684 – 689, 2001.
- [23] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. Matrices for detecting distant relationships. *Atlas of Protein Sequence and Structure*, 5(Suppl 3):345–358, 1978.
- [24] Mount DM. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2 edition, 2004.

- [25] C.B. Do, M.S.P. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.
- [26] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italie, 1992.
- [27] J. B. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, pages 32–57, 1973.
- [28] J. B. Dunn. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1–38), 1977.
- [29] S.R. Eddy. Where did the BLOSUM 62 alignment score matrix come from? *Nature Biotechnology*, 22(8):1035–1036, 2004.
- [30] R.C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.
- [31] A.W.F. Edwards. The measure of association in a 2x2 table. *Journal of the Royal Statistical Society*, 1:109–114, 1963.
- [32] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol*, 60:351–360, 1987.
- [33] M. Gerstein and R. Altman. Average core structures and variability measures for protein families: application to the immunoglobulins. *J. Mol. Biol*, 251:161–175, 1995.
- [34] A. J. Gibbs and McIntyre. G. A. The diagram, a method for comparing sequences. its use with amino acid and nucleotide sequences. *Eur. J Biochem*, pages 1–11, 1970.
- [35] Taillard-E. Glover, F. and D. de Werra. A user’s guide to tabu search. *Ann. Oper. Res*, 41:3–28, 1993.
- [36] Gaston H. Gnnnet, Mark A. Cohen, and Steven A. Benner. Exhaustive matching of the entire. sequence database. *Journal of Science*, 256:1443–1445, 1992.
- [37] D.E Golberg. Genetic Algorithms in Search, Optimisation and Machine Learning. *Addison-Wesley*, 1989.
- [38] F. Golver and M. Laguna. Tabu search. *Kluwer Academic Publishers*, pages –, 1997.

- [39] Gaston H. Gonnet, Chantal Korostensky, and Steve Benner. Evaluation Measures of Multiple Sequence Alignment. *Journal of Computational biology*, 7(1):261–276, 2000.
- [40] O Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol*, 264:823–838, 1996.
- [41] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705 – 708, 1982.
- [42] R. L. Graham and L. R. Foulds. Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Mathematical Biosciences*, 60:133–142, 1982.
- [43] C. Grasso and C. Lee. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *J. Mol. Biol*, 20(10):1546–1556, 2004.
- [44] D. Gusfield. Algorithms on strings, trees, and sequences: science and computational biology. *Cambridge University Press, Cambridge, UK*, 1997.
- [45] S. Henikoff and J. G. Henikoff. Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.
- [46] J.H. Holland. Adaptation in Natural and Artificial Systems. *University of Michigan Press*, 1975.
- [47] Chun-Hsi Huang and Ratnabali Biswas. Parallel pattern identification in biological sequences on clusters. *Cluster Computing, IEEE International Conference on*, 0:127, 2002.
- [48] X. Huang and W. Miller. A time-efficient, linear space local similarity algorithm. *Adv. Appl. Math*, 12:337–357, 1991.
- [49] Xiaoqui Huang. A space-efficient parallel sequence comparison algorithm for a message-passing multiprocessor. *Int. J. Parallel Program.*, 18(3):223–239, 1990.
- [50] T.J.P. Hubbard and T.L. Blundell. Comparison of solvent-inaccessible cores of homologous proteins: definitions useful for protein modelling. *Protein Engineering*, 1:159–171, 1987.
- [51] J. Hudak and M. A. McClure. A comparative analysis of computational motif-detection methods. *Pacific Symposium on Biocomputing*, 4:138–149, 1999.

- [52] Greer J. Comparative modeling methods: application to the family of the mammalian serine proteases.
- [53] Mannu Jayakanthan, Gulshan Wadhwa, Thangavel Madhan Mohan, Loganathan Arul, Ponnusamy Balasubramanian, and Durai Sundar.
- [54] R. Jores, PM Alzari, and T. Meo. Resolution of Hypervariable Regions in T-Cell Receptor ss Chains by a Modified Wu-Kabat Index of Amino Acid Diversity. *Proceedings of the National Academy of Sciences*, 87(23):9138–9142, 1990.
- [55] TH Jukes and CR. Cantor. Evolution of protein molecules. *Academic Press, New York*, pages 21–123, 1969.
- [56] S. Karlin and L. Brocchieri. Evolutionary conservation of RecA genes in relation to protein structure and function. *Journal of Bacteriology*, 178(7):1881–1894, 1996.
- [57] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.
- [58] A Krogh, I S Mian, and D Haussler. A hidden markov model that finds genes in e. coli dna. *Nucleic Acids*, 22(2):307–318, 1994.
- [59] Timo Lassmann and Erik LL Sonnhammer. Kalign an accurate and fast multiple sequence alignment algorithm. *Communications of the ACM*, 6:298, 2005.
- [60] Hsi-Chieh Lee and F. Ercal. Rmesh algorithms for parallel string matching. *Third International Symposium on Parallel Architectures, Algorithms, and Networks, I-SPAN97 Proceedings*, pages 223–226, 1997.
- [61] Zne-Jung Lee, Shun-Feng Su, Chen-Chia Chuang, and Kuan-Hung Liu. Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. *Applied Soft Computing*, 8:55–78, 2008.
- [62] A.M. Lesk and C. Chothia. How different amino acid sequences determine similar protein structures: the structure and evolutionary dynamics of the globins. *J. Mol. Biol.*, 136:225–270, 1980.
- [63] Carlos R. Erig Lima, Heitor S. Lopes, Maiko R. Moroz, and Ramon M. Menezes. Multiple sequence alignment using reconfigurable computing. In *ARC'07: Proceedings of the 3rd international conference on Reconfigurable computing*, pages 379–384, Berlin, Heidelberg, 2007. Springer-Verlag.

- [64] D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. A Tool for Multiple Sequence Alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415, 1989.
- [65] DJ Lipman and WR Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.
- [66] D. Liu, X. Xiong, Z. Hou, and B DasGupta. Identification of motifs with insertions and deletions in protein sequences using self-organizing neural networks. *Neural Networks*, 18:835–842, 2005.
- [67] Li-fang Liu, Hong-wei Huo, and Bao-shu. Wang. Aligning multiple sequences by genetic algorithm. *International Conference on Communications, Circuits and Systems (ICCCAS 2004)*, pages 994–998, 2004.
- [68] Yongchao Liu, Bertil Schmidt, and Douglas L. Maskell. Msa-cuda: Multiple sequence alignment on graphics processing units with cuda. *Application-Specific Systems, Architectures and Processors, IEEE International Conference on*, 0:121–128, 2009.
- [69] SW Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–9, 1999.
- [70] Jiancong Luo, Ishfaq Ahmad, Munib Ahmed, and Raymond Paul. Parallel multiple sequence alignment with dynamic scheduling. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I*, pages 8–13, Washington, DC, USA, 2005. IEEE Computer Society.
- [71] AM Maxam and W Gilbert. A new method for sequencing dna. *Proc. Natl. Acad. Sci. U.S.A.*, 74:560–564, 1977.
- [72] J. B. McQueen. Some methods for classification and analysis of multivariate observations. *Proceeding of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- [73] K Mizuguchi, CM Deane, TL Blundell, and JP. Overington. Homstrad: a database of protein structure alignments for homologous families. *protein Science*, 7:2469–2471, 1998.
- [74] B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.
- [75] Frederick Mosteller. Association and estimation in contingency tables. *Journal of the American Statistical Association*, 321:1–28, 1968.

- [76] S. Muthukrishnan and Sleyman Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. *proceeding of 32nd ACM on Theory Computing*, pages 416–424, 2000.
- [77] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–53, 1970.
- [78] HD Nguyen, K Yamamori, I. Yoshihara, and M. Yasunaga. Improved ga-based method for multiple protein sequence alignment. *The 2003 Congress on Evolutionary Computation (CEC '03)*, 3:1826–1832, 2003.
- [79] Ken Nguyen and Yi Pan. Kb-msa: Knowledgebase multiple sequence alignment. *Preceeding of ISBRA 2010*, 6:68–71, 2010.
- [80] Ken Nguyen and Yi Pan. Multiple sequence alignment based on dynamic weighted guidance tree. *International Journal of Bioinformatics Research and Applications*, 7(2):168–182, 2011.
- [81] Ken D. Nguyen and Anu G. Bourgeois. Ant colony optimal algorithm: Fast ants on the optical pipelined r-mesh. *International Conference on Parallel Processing (ICPP'06)*, pages 347–354, 2006.
- [82] Ken D. Nguyen and Yi Pan. A reliable metric for quantifying multiple sequence alignment. *BIBE*, pages 788–795, 2007.
- [83] C. Notredame, D.G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol*, 302(1):205–217, 2000.
- [84] C. Notredame, DG Higgins, and O. Journals. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524, 1996.
- [85] Tim Oliver, Bertil Schmidt, Douglas Maskell, Darran Nathan, and Ralf Clemens. High-speed multiple sequence alignment on a reconfigurable platform. *Int. J. Bioinformatics Res. Appl.*, 2:394–406, October 2006.
- [86] Tim Oliver, Bertil Schmidt, Darran Nathan, Ralf Clemens, and Douglas Maskell. Using reconfigurable hardware to accelerate multiple sequence alignment with clustalw. *Bioinformatics*, 21(16):3431–3432, 2005.
- [87] E. Ollivier, H. Soldano, and A. Viari. 'multifrequency'location and clustering of sequence patterns from proteins. *Oxford University Press*, 7:31–38, 1991.

- [88] O. Olsvik, J. Wahlberg, B. Petterson, and et al. Use of automated sequencing of polymerase chain reaction-generated amplicons to identify three types of cholera toxin subunit b in vibrio cholerae o1 strains. *J. Clin. Microbiol*, 31:22–25, 1993.
- [89] O. O’Sullivan, K. Suhre, C. Abergel, D. G. Higgins, and C. Notredame. 3dcoffee: combining protein sequences and structures within multiple sequence alignments. *Journal of Molecular Biology*, 340:385–395, 2004.
- [90] Yi Pan, Keqin Li, and M. Hamdi. An improved constant-time algorithm for computing the radon and hough transforms on a reconfigurable mesh. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 29(4):417–421, jul. 1999.
- [91] S. W. Perrey and J. Stoye. FDCA: Fast and Accurate Approximation to Sum-of-Pairs Score Optimal Multiple Sequence Alignment . *Universitt Bielefeld*, 114, 1997.
- [92] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. An eulerian path approach to dna fragment assembly. *PNAS*, 98:9748–9753, 2001.
- [93] KD Pruitt, T Tatusova, and DR Maglott. Ncbi reference sequence (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, 33:501–504, 2005.
- [94] Viswanadha S. Raju and A. Vinayababu. Optimal parallel algorithm for string matching on mesh network structure. *International Journal of Applied Mathematical Sciences.*, 3:167–175, 2006.
- [95] Viswanadha S. Raju and A. Vinayababu. Parallel algorithms for string matching problem on single and two dimensional reconfigurable pipelined bus systems. *Journal of Computer Science*, 3:754–759, 2007.
- [96] Kiira Ratia, Scott Pegana, Jun Takayamab, Katrina Sleemanc, Melissa Coughlind, Rima Bali-jic, Surendranath Chaudhuria, Wentao Fua, Bellur S. Prabhakard, Susan C. Johnsona, Michael E. Bakerc, Arun K. Ghoshc, and Andrew D. Mesecarc. A noncovalent class of papain-like protease/deubiquitinase inhibitors blocks sars virus replication. *PNAS*, 42:16119–16124, 2008.
- [97] Tariq Riaz, Yi Wang, and Kuo-Bin Li. Multiple sequence alignment using tabu search. *Conferences in Research and Practice in Information Technology*, 29:223–232, 2004.
- [98] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Oxford University Press*, 4:406–425, 1987.

- [99] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function & Genetics*, 9(1):56–68, 1991.
- [100] F. Sanger and AR Coulson. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *J. Mol. Biol.*, 3:441–448, 1975.
- [101] Souradip Sarkar, Gaurav Ramesh Kulkarni, Partha Pratim Pande, and Ananth Kalyanaraman. Network-on-chip hardware accelerators for biological sequence alignment. *IEEE Transactions on Computers*, 59:29–41, 2010.
- [102] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
- [103] P.S. Shenkin, B. Erman, and L.D. Mastrandrea. Information-theoretical entropy as a measure of sequence variability. *Proteins: Struct. Funct. Genet*, 11:297–313, 1991.
- [104] Hongchi Shi, Gerhard X. Ritter, and Joseph N. Wilson. Simulations between two reconfigurable mesh models. *Information Processing Letters*, 55(3):137 – 142, 1995.
- [105] R. F. Smith and T. F. Smith. Automatic Generation of Primary Sequence Patterns from Sets of Related Protein Sequences. *Proceedings of the National Academy of Sciences*, 87(1):118–122, 1990.
- [106] R. F. Smith and T. F. Smith. Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling. *Protein Engineering Design and Selection*, 5:35–41, 1992.
- [107] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
- [108] P. H. A. Sneath and R. R. Sokal. Numerical taxonomy. the principles and practice of numerical classification. *Freeman, San Francisco*, page 573, 1973.
- [109] J. Stoye. Multiple sequence alignment with the Divide-and-Conquer method. *Gene*, 211(2):56, 1998.
- [110] S.-H. Sze, Y. Lu, and Q. Yang. A polynomial time solvable formulation of multiple sequence alignment. *Journal of Computational Biology*, 13:309–319, 2006.
- [111] J. Taheri and A. Y. Zomaya. Rbt-ga: a novel metaheuristic for solving the multiple sequence alignment problem. *BMC Genomics*, 7, 2009.



- [112] Y. Takefuji, T. Tanaka, and K.C. Lee. A parallel string search algorithm. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):332–336, mar. 1992.
- [113] Guangming Tan, Shengzhong Feng, and Ninghui Sun. Parallel multiple sequences alignment in smp cluster. In *HPCASIA '05: Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, page 426. IEEE Computer Society, 2005.
- [114] W. R. Taylor. The classification of amino acid conservation. *J Theor Biol*, 119(2):205–18, 1986.
- [115] J. D. Thompson, P. Koehl, R. Riip, and O. Poch. Balibase 3.0: latest development of multiple alignment benchmark. *protein*, 61:127–136, 2005.
- [116] J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27:2682–2690, 1999.
- [117] JD Thompson, TJ Gibson, F. Plewniak, F. Jeanmougin, and DG Higgins. The CLUSTAL\_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 25(24):4876–4882, 1997.
- [118] J.D. Thompson, D.G. Higgins, T.J. Gibson, et al. 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(22):4673–4680, 1994.
- [119] Jerry L. Trahan, Anu G. Bourgeois, Yi Pan, and Ramachandran Vaidyanathan. Optimally scaling permutation routing on reconfigurable linear arrays with optical buses. *Journal of Parallel and Distributed Computing*, 60(9):1125 – 1136, 2000.
- [120] R. Vaidyanathan and J. L. Trahan. *Dynamic Reconfiguration: Architectures and Algorithms*. Kluwer Academic/Plenum Publishers, January 2004.
- [121] W.S.J. Valdar. *Residue conservation in the prediction of protein-protein interfaces*. PhD thesis, University College London, 2001.
- [122] Ivo Van Walle, Ignace Lasters, and Lode Wyns. Sabmarka benchmark for sequence alignment that covers the entire known fold space. *bioinformatics*, 21:1267–1268, 2005.
- [123] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J Comput Biol*, 1(4):337–48, 1994.

- [124] CH Wu, LS Yeh, H Huang, Arminski, and et al. The protein information resource. *Nucleic Acids Res.*, 31:345–347, 2003.
- [125] S Wu and U Manber. Fast text searching allowing errors. *Communications of the ACM*, 35:83–91, 1992.
- [126] T.T. Wu and E.A. Kabat. An Analysis of the Sequences of the Variable Regions of Bence Jones Proteins and Myeloma Light Chains and Their Implications for Antibody Complementarity. *Journal of Experimental Medicine*, 132(2):211–250, 1970.
- [127] Shinsuke Yamada, Osamu Gotoh, and Hayato Yamana. Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinformatics*, 7:254–, 2006.
- [128] C Zhang and AKC. Wong. Toward efficient multiple molecular sequence alignment: a system of genetic algorithm and dynamic programming. *IEEE Transactions on Systems,, Man and Cybernetics*, 27:918–932, 1997.
- [129] Y. Zhang and M. S. Waterman. An eulerian path approach to local multiple alignment for dna sequences. *Proc. Natl. Acad. Sci*, 102:1285–1290, 2005.
- [130] J.M. Zvelibil, Barton G.J., Taylor W.R., and Sternberg M.J. Prediction of Protein Secondary Structure and Active Sites using the Alignment of Homologous Sequences. *J. Mol. Biol*, 195:957–961, 1987.

## APPENDICES

### APPENDIX A: PUBLICATIONS RELATED TO THIS RESEARCH

#### REFEREED JOURNALS AND CONFERENCE PROCEEDINGS

- Nguyen, K. D. and Yi Pan, "Multiple Sequence Alignment Based on Dynamic Weighted Guidance Tree", International Journal of Bioinformatics Research and Applications. Vol 7, No 2, pp. 168-182, 2011
- Nguyen, K. D. and Yi Pan. "A Reliable Metric for Quantifying Multiple Sequence Alignment", proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, pp. 788-795, 2007
- Nguyen, K. D. and Anu G. Bourgeois. "Ant colony optimal algorithm: Fast ants on the optical 2659 pipelined r-mesh", International Conference on Parallel Processing (ICPP'06), pp. 347-354, 2006

#### POSTERS, ABSTRACTS,PRESENTATIONS AND INVITED TALKS

- Nguyen, K. D.. "BLAST Toward The Future", Oral Presentation, MBD Research Day at GSU, June 17, 2011. [BEST PRESENTATION AWARD]
- Nguyen, K. D.. "Reliability in Multiple Molecular Sequence Alignments", Invited TALK, MBD Distinguish Lecture at GSU, Nov. 18, 2010
- Nguyen, K. D. and Yi Pan. "KB-MSA: Knowledge-based Multiple Sequence Alignment", POSTER presentation at the 6th International Symposium on Bioinformatics Research and Applications (IS-BRA), 2010
- Nguyen, K. D. and Yi Pan. "KB-MSA: Knowledge-based Multiple Sequence Alignment", POSTER presentation at GSU Sixth Annual Molecular Basis of Disease Research Day - May, 2010 .[BEST POSTER AWARD]
- Nguyen, K. D. and Yi Pan. "Reliable Multiple Sequence Alignment", POSTER presentation at GSU Computer Science Department 1st Research Poster day - February, 2010

**PAPERS UNDER REVIEW**

- Nguyen, K. D. and Yi Pan. "An Improved Scoring Method for Multiple Sequence Alignment"
- Nguyen, K. D. and Yi Pan. "A Consistency-based and Knowledge-based Multiple Sequence Alignment Algorithm"
- Nguyen, K. D. and Yi Pan. "Parallel Progressive Multiple Sequence Alignment on R-Mesh"