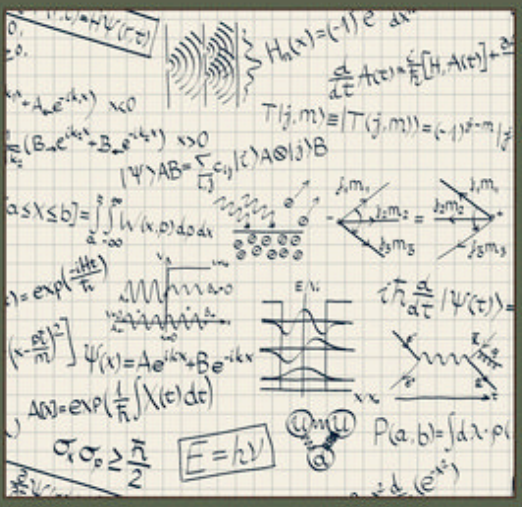
Wiley Series on Bioinformatics:
Computational Techniques and Engineering
Yi Pan and Albert Y. Zomaya, Series Editors

Multiple Biological Sequence Alignment

SCORING FUNCTIONS, ALGORITHMS AND APPLICATIONS



KEN NGUYEN • XUAN GUO • YI PAN

WILEY

MULTIPLE BIOLOGICAL SEQUENCE ALIGNMENT

Wiley Series on

Bioinformatics: Computational Techniques and Engineering

A complete list of the titles in this series appears at the end of this volume.

MULTIPLE BIOLOGICAL SEQUENCE ALIGNMENT

Scoring Functions, Algorithms and Applications

KEN NGUYEN
XUAN GUO
YI PAN

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Names: Nguyen, Ken, 1975- author. | Guo, Xuan, 1987- author. | Pan, Yi, 1960- author.
Title: Multiple biological sequence alignment : scoring functions, algorithms and applications / Ken Nguyen, Xuan Guo, Yi Pan.
Description: Hoboken, New Jersey : John Wiley & Sons, 2016. | Includes bibliographical references and index.
Identifiers: LCCN 2016004186| ISBN 9781118229040 (cloth) | ISBN 9781119273752 (epub)
Subjects: LCSH: Sequence alignment (Bioinformatics)
Classification: LCC QH441 .N48 2016 | DDC 572.8–dc23 LC record available at http://lccn.loc.gov/2016004186

Cover image courtesy of GettyImages/OktalStudio

Typeset in 10/12pt TimesLTStd by SPi Global, Chennai, India

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

The advancement in sequencing technology has resulted in enormous amount of data that must be analyzed and categorized. Recently, the US National Center for Biotechnology Information (NCBI) announced the availability of whole genome sequences for more than 1000 species. The number of sequenced individual organisms is growing very fast around the world. However, the availability of sequence information is only the first step in understanding how cells survive, reproduce, and adjust their behavior. The natural next step is to analyze the data, extract knowledge from them, and get a better understanding of the functions, and control mechanisms of these species. Since sequence alignment is the fundamental operation in dealing with the sequencing data, it is gaining a lot of interest from researchers who are mining biological information from massive sequence databases. Multiple sequence alignment algorithms are used to align three or more biological sequences. For example, multiple biological sequence alignment is used in motif finding, sequence clustering, and sequence mining. The current advancement in sequencing technology has created a massive number of biological sequences (DNA/RNA/protein, etc.), thus expanding the sequence databases at a rate exceeding the capability of existing sequence analysis tools and facility. As a result, many traditional sequence alignment models and methods become obsolete, along with the books covering only these topics.

Exhaustive dynamic programming is a straightforward way to compute optimal multiple sequence alignments. However, this approach is prohibitive in terms of both time and space. To overcome these constraints, heuristics such as progressive alignment have been suggested. Other new heuristic algorithms based on information extracted from sequences themselves are described in the book. Another issue is how to evaluate the quality of the alignments obtained. Clearly, a simple numerical score for matching or mismatching of two biological symbols is not always the

right thing to do. We need to take into account the biological meanings of these symbols. Furthermore, the validations should be done by biological experiments or by biological experts on real biological sequences based on their experiences. The aim of this book is to cover as much topics as possible arising recently in this field apart from traditional multiple sequence alignment algorithms. This book provides detailed descriptions of the traditional and modern approaches in biological sequence alignments and homology search. It covers the full spectrum of the field from alignment algorithms to scoring methods, practical techniques, alignment tools, and their evaluations and applications. It provides the details of the state of the computational methodologies for challenging tasks in multiple sequence alignment including the following topics:

- Background of protein/DNA/RNA pairwise and multiple sequence alignment
- Sequence clustering methods
- Theories and developments of scoring functions and scoring matrices
- Cutting-edge development in multiple sequence alignment algorithms
- Knowledgebase-assisted sequence alignment
- Evaluation of existing sequence alignment algorithms
- Homologous sequence classification and organization
- Phylogeny estimation
- Large-scale homology search
- Parallel and distributed sequence alignment algorithms
- Popular sequence alignment servers
- Alignment algorithms for next-generation sequences
- Variants detection based on multiple sequence alignment
- RNA/protein secondary structure prediction using multiple sequence alignment

Multiple sequence alignment is often included as a chapter in most of bioinformatics and sequence analysis books. It is the first book in the field that provides a comprehensive view into recently developed large-scale alignment models and techniques that promise to handle the current and future challenges of aligning enormous amount of sequences. It is also the first book specifically written for multiple biological sequence alignment.

Readers will be informed with the fundamentals in multiple biological sequence alignment and analysis to understand the state-of-the-art techniques in sequence alignment and be able to choose appropriate sequence analysis tools for their tasks. The intended readers of the book are researchers, engineers, graduate and postgraduate students in bioinformatics and systems biology, and molecular biologists. It provides the readers with a comprehensive book in multiple biological sequence alignment theory, methodology, evaluation, applications, and technologies.

Finally, we would like to express our sincere thanks to Brett Kurzman (editor), Allison McGinniss (project editor), Simone Taylor (senior editor), and Diana Gialo

(editorial assistant) from Wiley for their guidance and help in finalizing this book. We would also like to thank our families for their support, patience, and love. We hope that our readers will enjoy reading this timely-produced book and give us feedback for future improvements.

Ken Nguyen, PhD
Department of Computer Science and Information Technology
Clayton State University
2000 Clayton State Blvd. Morrow, GA 30260, USA
Email: KenNguyen@clayton.edu

Xuan Guo, PhD
School of Genome Science and Technology
University of Tennessee Knoxville
F337 Walters Life Science, Knoxville, TN 37996-0840, USA
Email: gxuan@utk.edu
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6420
Email: guox@ornl.gov

Yi Pan, PhD
Department of Computer Science
Georgia State University
25 Park Place, Room 744, Atlanta, GA 30302-5060, USA
Email: yipan@gsu.edu

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 1970s, deoxyribonucleic acid (DNA) sequences were obtained using laborious methods based on 2D chromatography. Thus, the number of sequences is limited and often being studied and annotated individually by scientists. By the late 1970s, Gilbert [1] and Sanger and Coulson [2] 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 [3], microarrays, mass spectrometry, X-ray, ultracentrifugation, and so on. 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, and so on. The main step to solve this problem is finding the best way to start with the sequence fundamentals, thus leading the readers to the most

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

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 that bioinformatics scientists are dealing with. The first trend relates to prediction of the sequence structures and homology, evolution of species, or determination of the relationship between sequences in order to categorize and organize 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 resemble each other. Thus, the second trend can be seen in evolution analysis, in 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 [4, 5] . 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 BOOK

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 pairwise sequence alignment.

Chapter 3 describes popular existing quantitative models that have been designed to quantify multiple sequence alignment along with their analysis and evaluations.

Chapter 4 describes practical clustering techniques that have been used in multiple sequence alignment.

Chapter 5 describes, characterizes, and relates many multiple sequence alignment models.

Chapter 6 describes how traditional phylogenetic trees have been constructed and how available sequence knowledge bases can be used to improve the accuracy of reconstructing phylogeny trees.

Chapter 7 describes the latest methods developed to improve the run-time efficiency of multiple sequence alignment. A large section of this chapter is devoted to parallel alignment model on reconfigurable networks.

Chapter 8 describes several popular existing multiple sequence alignment server and services.

Chapter 9 describe several multiple sequence alignment techniques that have been developed to handle short sequences (reads) produced by the next-generation sequencing technique (NSG).

Chapter 10 describes a bioinformatics application, genetic variant detection, using multiple sequence alignment of short reads or whole genomes as input.

Lastly, Chapter 11 provides a review of ribonucleic acid (RNA) and protein secondary structure prediction using the evolution information inferred from multiple sequence alignments.

1.3 SEQUENCE FUNDAMENTALS

DNA is the fundamental unit that characterizes a living organism and its genome, that is, its genetic information set. DNA contains thousands of genes that 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 noncoding 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 (see Figure 1.1). Thus, DNA sequence determines protein sequence and its structure;

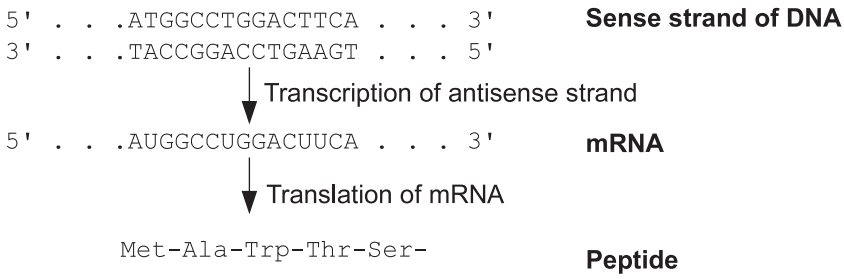
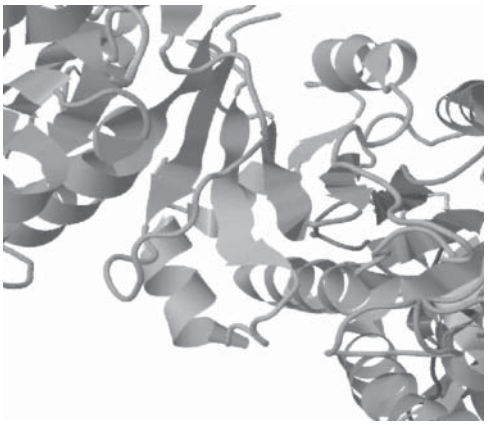


Figure 1.1 Transcription and translation processes: DNA → RNA → protein (the "central dogma" of biology).

(a)



(b) . . . PICNSRNPLELQSTTIEYITNKPVTVPPIFFFVVDLTSE . . .

Figure 1.2 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

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	–1	5																		
Asn	–2	0	6																	
Asp	–2	–2	1	6																
Cys	0	–3	–3	–3	9															
Gln	–1	1	0	0	–3	5														
Glu	–1	0	0	2	–4	2	5													
Gly	0	–2	0	–1	–3	–2	–2	6												
His	–2	0	1	–1	–3	0	0	–2	8											
Ile	–1	–3	–3	–3	–1	–3	–3	–4	–3	4										
Leu	–1	–2	–3	–4	–1	–2	–3	–4	–3	2	4									
Lys	–1	2	0	–1	–3	1	1	–2	–1	–3	–2	5								
Met	–1	–1	–2	–3	–1	0	–2	–3	–2	1	2	–1	5							
Phe	–2	–3	–3	–3	–2	–3	–3	–3	–1	0	0	–3	0	6						
Pro	–1	–2	–2	–1	–3	–1	–1	–2	–2	–3	–3	–1	–2	–4	7					
Ser	1	–1	1	0	–1	0	0	0	–1	–2	–2	0	–1	–2	–1	4				
Thr	0	–1	0	–1	–1	–1	–1	–2	–2	–1	–1	–1	–1	–2	–1	1	5			
Trp	–3	–3	–4	–4	–2	–2	–3	–2	–2	–3	–2	–3	–1	1	–4	–3	–2	11		
Tyr	–2	–2	–2	–3	–2	–1	–2	–3	2	–1	–1	–2	–1	3	–3	–2	–2	2	7	
Val	0	–3	–3	–3	–1	–2	–2	–3	–3	3	1	–2	1	–1	–2	–2	0	–3	–1	4

Figure 1.3 BLOSUM62 substitution matrix.

and the protein structure dictates the protein function. In this section, we 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 α-helices, β-turns, or β-strands. The size of a protein sequence ranges from a few to several thousand residues. Figure 1.2 shows both the structure and the primary sequence of the yeast 1M2V protein. Tables 1.1 and 1.2 list the amino acids and their common abbreviations. Figure 1.3 shows a substitution matrix to quantify the probable substitution between two amino acids in a protein molecule.

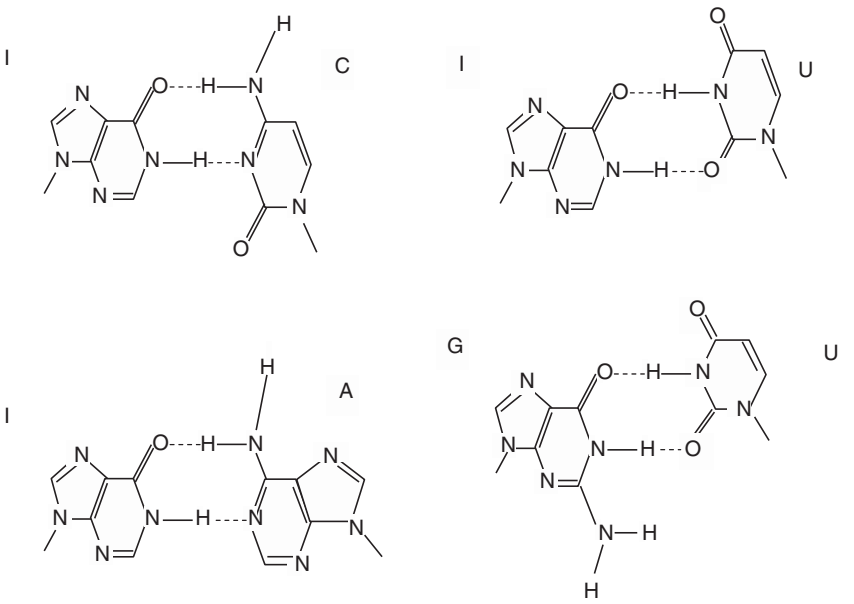


Figure 1.4 Four fundamental wobble base pairs.

1.3.2 DNA/RNA

Both DNA and 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. DNA has four bases namely adenine (A), thymine (T), guanine (G), and cytosine (C), while RNA has adenine (A), uracil (U), guanine (G), and cytosine (C) bases. Adenine and guanine are double-ring molecules known as purine, while other bases are single-ring molecules known as pyrimidine (see Figure 1.4 for an example).

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 structure 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 process, and mRNA is a template for protein synthesis. During the process of producing an amino acid chain, the nucleotide sequence of an 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 an amino acid or a translation termination signal. There are 64 ($4 \times 4 \times 4$) possible codons for the bases.

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, Swiss-Prot, and so on. 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 that 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 starts with > character, and an asterisk "*" marks the end of a sequence and can be omitted. Each sequence should be separated by a new line. The following is an example of sequences in FASTA format:

Example 1.1 *>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken*
ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEIDGDGQVNYEEFVQMMTAK

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIEWIWG
GFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTI

In the early 1990s, the National Center for Biotechnology Information (NCBI), which houses the GenBank and genome sequencing data, defined a standard to uniquely identify the sequence. The header of the sequence that contains the sequence name should include a unique sequence identification. Figure 1.5 shows some sequence headers being used by different groups.

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, and so on.

The alignment sequence formats such as GCG, MSF, Clustal, Phylis, and so on are used to represent sequence alignments. These formats preserve the actual alignment of the sequences making it easier for visual inspections. For example, Figure 1.7 shows two 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 substructure 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

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

Figure 1.5 NCBI's sequence formats.

```
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="MNRWVEKWLRVYLKCYINLILFYRNVYPPQSFDYTTYQSFNLPQ
                     FVPINRHPALIDYIEELILDVLSKLTHVYRFSICIINKKNDLCIEKYVLDFSELQHVD
                     KDDQIITETEVFDEFRSSLNSLIMHLEKLPKVNDDTITFEAVINAIELELGHKLDRNR
ORIGIN
        1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
       61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
```

Figure 1.6 Illustration of GenBank sequence format.

motifs is the GCM (glial cells missing) motif, found in humans, mice, and fruit flies, which is identified by Akiyama et al. [6]. 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 [7] and active sites [8], homologous sequences can have low overall

```
MCHU       ---------- ---------- ---------- ---------- ----------
gi|5524211 LCLYTHIGRN IYYGSYLYSE TWNTGIMLLL ITMATAFMGY VLPWGQMSFW

           ---------- ---------- ---------- ADQLTEEQIA EFKEAFSLFD
           GATVITNLFS AIPYIGTNLV EWIWGGFSVD KATLNRFFAF HFILPFTMVA

           KDGDGTITTK ELGTVMRSLG QNPTEAELQD MINEVDADGN GTIDFPEFLT
           LAGVHLTFLH ETGSNN-PLG LTSDSDKIPF HPYYTIKDFL GLLILILLLL

           MMARKMKDTD SEEEIREAFR VFDK------ DGNGYISAAE LRHVMTNLG-
           LLALLSPDML GDPDNHMPAD PLNTPLHIKP EWYFLFAYAI LRSVPNKLGG

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

           GQVNYEEFVQ MMTAK----- ---------- -----
           QPVEYPYTII GQMASILYFS IILAFLPIAG XIENY
```

(a)

```
MCHU                        --------------------------------------------------
gi|5524211|gb|AAD44166.1|   LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFW 50

MCHU                        ----------------------------ADQLTEEQIAEFKEAFSLFD 20
gi|5524211|gb|AAD44166.1|   GATVITNLFSAIPYIGTNLVEWIWGGFSVDKATLNRFFAFHFILPFTMVA 100
                                                        *..    .*  .*::.

MCHU                        KDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTIDFPEFLT 70
gi|5524211|gb|AAD44166.1|   LAGVHLTFLHETGSNN-PLGLTSDSDKIPFHPYYTIKDFLGLLILILLLL 149
                              *      :* *:   .** .. . ::      .  *  * : :  :*

MCHU                        MMARKMKDTDSEEEIREAFRVFDK------DGNGYISAAELRHVMTNLG- 113
gi|5524211|gb|AAD44166.1|   LLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGG 199
                            ::*    *  .: : : .   ::.     :    :: * ** * .:**

MCHU                        ---------------EKLTDEEVDEMIR---------------EADIDGD 133
gi|5524211|gb|AAD44166.1|   VLALFLSIVILGLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGS 249
                                         *.:. . *:*                    *.

MCHU                        GQVNYEEFVQMMTAK------------------- 148
gi|5524211|gb|AAD44166.1|   QPVEYPYTIIGQMASILYFSIILAFLPIAGXIENY 284
                              *:*   :    *.
```

(b)

Figure 1.7 (a) Phylis format and (b) Clustal format.

sequence similarity [9]. In sequence alignment studies, the terms motif and motifs have broader spectrum, they also refer to conserved segments of 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. Protein sequences can be found in Swiss-Prot [10], TrEMBL [10], UniProt [11], PIR [12], and PDP [13] databases. Similarly, DNA and RNA sequences are in GenBank [14], PDB, HOMSTRAD [15], and RefSeq [16] databases. As of June 16, 2011, 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 four 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, of which 2.7% of the entries are predicted and about 0.4% of the entries are uncertain. The shortest sequence in Swiss-Prot is GWASEPOF (P83570) obtained from cuttlefish, which contains two amino acids (G and W), and the longest sequence is TITINMOUSE (A2ASS6) containing 35,213 amino acids. Swiss-Prot's average sequence length is 355 residues.

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 data 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, this 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 designed 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.

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

2.1 SEQUENCE ALIGNMENT FUNDAMENTALS

Sequence alignment problems can be generalized as follows: Let $W = \{\alpha_1, \ldots, \alpha_i, \ldots, \alpha_n\}, i > 0$, 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 $\{s_1, s_2, \ldots, s_k\}$ into $\{s'_1, s'_2, \ldots, s'_k\}$, where s'_i is the ith sequence s_i with GAP (-) symbols inserted to maximize 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 pairwise alignment techniques are dot matrix [17], dynamic programming [18, 19], and word method [20]. Each method provides a host of advantages. For example, the dot-matrix method 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 explore the details of these algorithms.

2.2 DOT-PLOT MATRIX

The dot-matrix method [17] 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 onto the two perpendicular edges of an $m \times n$ matrix.

(ii) A scoring scheme is used to mark the similarity 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 2.1 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

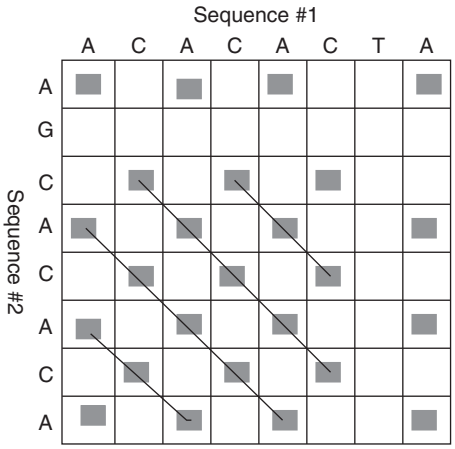


Figure 2.1 Dot-plot matrix of two sequences ACACACTA and AGCACACA. The connected diagonal lines indicate matching segments.

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.

Apart from 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 known as hairpin loops) or inverted repeats. These are subsequences that are the reverse of others subsequences downstream. For example: $5'$ … **GCCGCGGGCCG**AAAAAA CCCCCC**CGGCCCGCGGC** … $3'$ 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 two 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 TMMARKM KDT DSEEEIREAF RVFDKDGNGY ISAAELRHVM TNLGEKLTDE EVDEMIR EAD IDGDGQVNYE EFVQMMTAK"

Another advantage, as seen in [21], 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

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

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 form a double helix that ends in unpaired loop. In terms of complexity, the dot-plot method takes $O(n \times m)$ for plotting the matrix, $O(n \times m)$ for filtering the noise, and $O(n \times m)$ space for the matrix. Thus, the overall time and space complexity for dot plot is $O(n \times m)$. Despite the fact that the dot-plot method is great to identify features being shared between two sequences, it does not provide 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. Figure 2.3 shows an example output of the dot-plot method

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 that was first introduced in the Needleman–Wunsch's algorithm [19], and a local alignment version is in the Smith–Waterman's algorithm [18]. Similar to the dot-plot technique, two sequences are mapped onto the two perpendicular sides 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

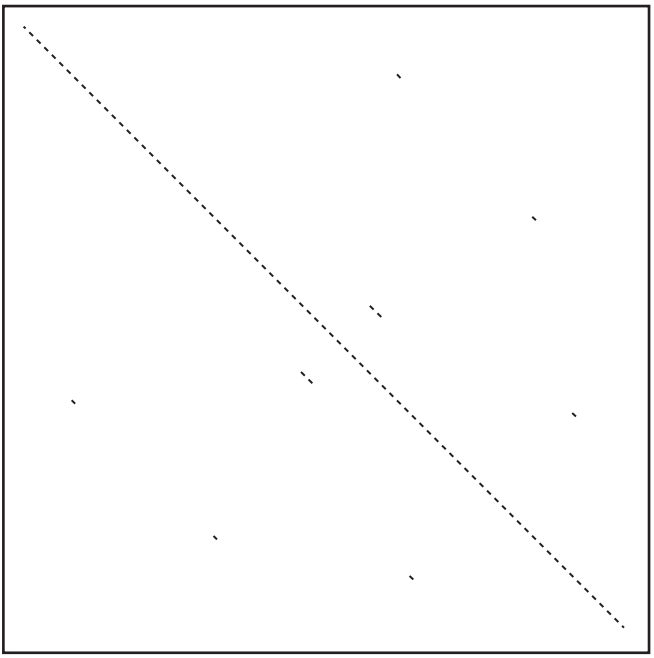


Figure 2.3 Dot-plot matrix of human CalM against itself to reveal motifs.

until the scores of all slots are computed. The DP technique takes $O(n \times m)$ for space and time complexity if the scoring scheme takes constant time to compute.

2.3.1 Needleman–Wunsch's Algorithm

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

$$H(i,0) = i \times g, 0 \leq i \leq m,$$
$$H(0,j) = j \times g, 0 \leq j \leq n,$$

$$H(i,j) = \max \begin{Bmatrix} H(i-1,j-1) + w(a_i, b_i) & Match/Mismatch \\ H(i-1,j) + g & Deletion \\ H(i,j-1) + g & Insertion \end{Bmatrix},$$
$$1 \leq i \leq m, 1 \leq j \leq n,$$

where

g is the gap penalty, a_i is the ith 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.

Moreover, the alignment is the trace-back route from the rightmost end of highest score slot to the starting slot.

2.3.2 Example

Given two sequences:

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

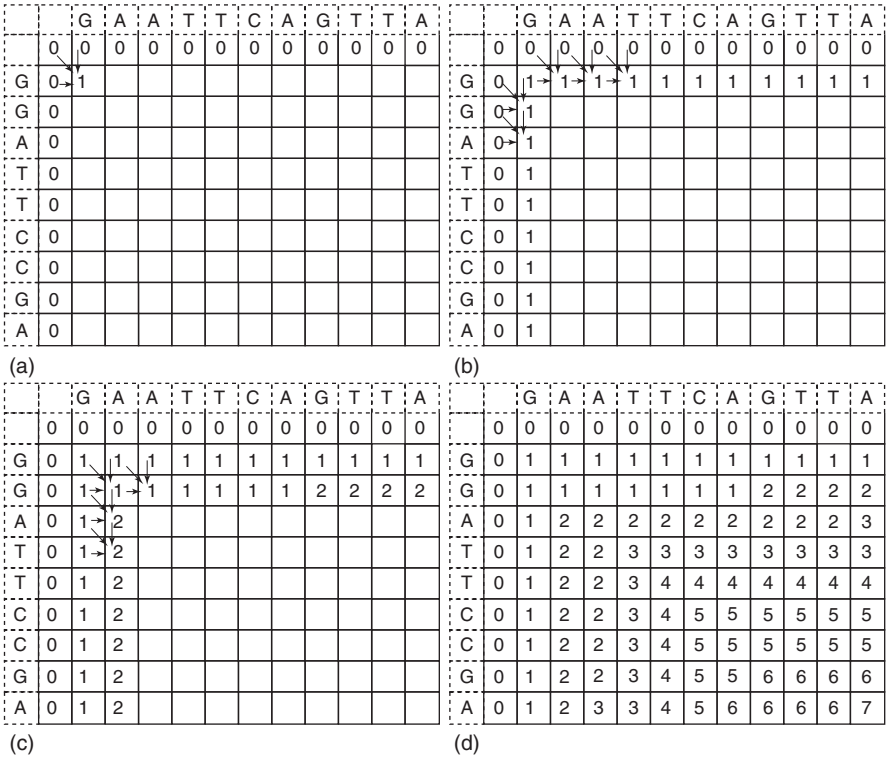


Figure 2.4 Generating a scoring matrix for sequence "GAATTCAGTTA" and sequence "GGATTCCGA."

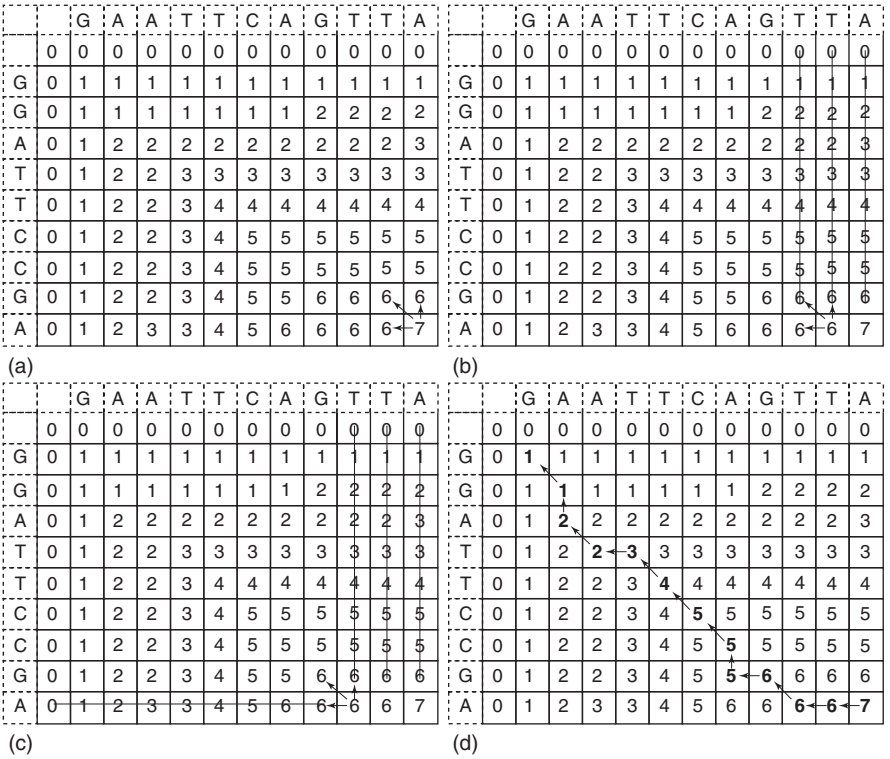


Figure 2.5 Backtracking, (a–c), to obtain the optimal alignment (d).

```
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.

and a scoring scheme where a matching pair of symbols get a score of 1, mismatch gets a score of 0 and gap penalty is 0. Figures 2.4 and 2.5 show the steps of Needleman–Wunsch's algorithm. And one of the optimal alignment results is shown in Figure 2.6.

2.3.3 Smith–Waterman's Algorithm

From the evolution perspective, two related sequences could evolve independently of 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 is proposed by Smith and Waterman, identifies the longest segment pair that yields the best alignment score that is more preferable. In the Smith–Waterman's algorithm, the longest segment pair between 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.

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

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

where

a_i is the ith 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 backtracking steps in Smith–Waterman is similar to those of Needleman–Wunsch's; however, it starts from the matrix cell with a 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 get a local alignment of ATTC as in Figure 2.7. It is important to note that the backtracking 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 2.7) 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 nonaligned segments until all residues in the sequences are aligned. However, this technique is time consuming and does not guarantee an overall optimal alignment.

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0
G	0	1	**0**	0	0	0	0	0	1	0	0	0
A	0	0	**2**	**1**	0	0	0	1	0	0	0	1
T	0	0	0	0	**2**	1	0	0	0	1	1	0
T	0	0	0	0	1	**3**	1	0	0	1	2	0
C	0	0	0	0	0	1	**4**	2	0	0	0	0
C	0	0	0	0	0	0	2	2	0	0	0	0
G	0	1	0	0	0	0	0	0	3	1	0	0
A	0	0	2	1	0	0	0	1	1	1	0	1

(a)

```
A T T C (sequence #1)
| | | |
A T T C (sequence #2)
```

(b)

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/deletion of a segment, more than consecutive insertions/deletion. This is called the affine gap penalty [22] and it is calculated as

$$w(g) = o + (l - 1)e, \tag{2.1}$$

where $w(g)$ is the weight or penalty of gap g, o is the 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 [20], are heuristic methods that find a series of short and nonoverlapping 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 as a time, to obtain all k-length words in a 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 λ are considered highly homologous. Figure 2.8 and the following example illustrates how to use the word method in sequence alignments.

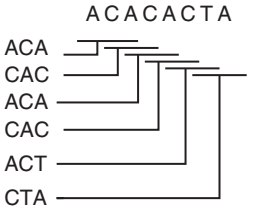


Figure 2.8 Displays all available words of size 3.

2.4.1 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.

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

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 word can then be extended until there is no more matching symbol. Considering the "hit and extend" scheme, the first longest sharing subsegment 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 yield $O(1)$ order.

The exact word matching scheme may not be able to detect significant similarity such as wobble base pairs as seen in Figure 2.9. 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 I-C). 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 score-matching scheme would be preferable. In this scheme, a word that yields the highest score by summing

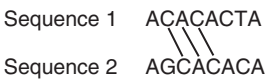


Figure 2.9 Displays an exact word matched.

each pair of matching symbol scores is selected. Effectively, if the following scoring scheme is used:

Score = 2 for X-X, that is, conserved nucleotide exact match.
Score = 1 for Y-X or X-X, that is, any pair of nonconserved nucleotide.
Score = 0 for X-Y, that is, any conserved nucleotide mismatched.

Then GGuUCuACgAAg and GGcUCcACaAAA are identical.

2.5 SEARCHING SEQUENCE DATABASES

Most application of pairwise 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 [23], which is a fast version of dot-plot method. Recently, it has been replaced by BLAST [24], a more efficient algorithm.

2.5.1 FASTA

Application of the dot-plot method is seen in FASTA [23] (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 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 [25] for protein and identity matrix for nucleotide sequences. Retain d diagonals with highest scores (in the original design, $d = 10$). Trim the ends of those diagonals to include only those contributing to the highest score. These are identified as 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 speed up the processing time. The lookup table is similar to a hash table, where each symbol is mapped onto a location in the table such as "A" is 1, "R" is 2, and so on. And the values in the table represent the location in the sequence where the symbol occurs.

Figure 2.10 illustrates these steps of FASTA.

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 [24]. BLAST is a heuristic search method that applies the word method

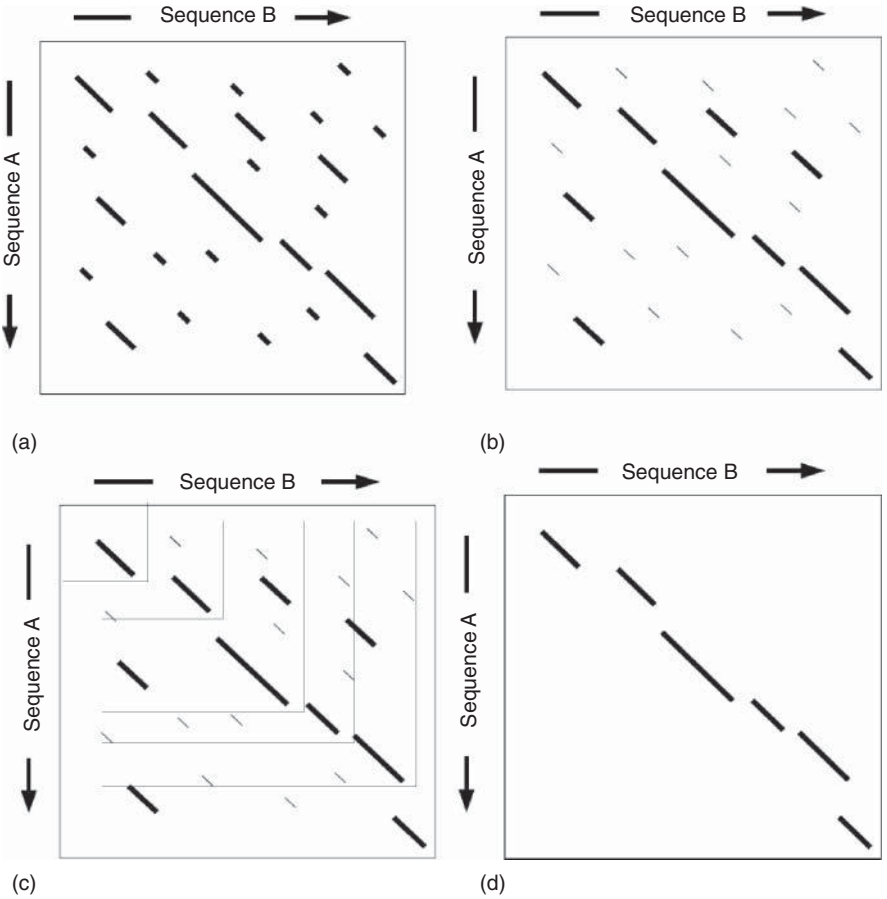


Figure 2.10 (a) Filtering dot plot; (b) being reevaluated by score matrix; (c) filtering highest scored diagonals starting from the core region – identified by an arrow; (d) assembling the diagonals.

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 MSP to be the highest scoring pair of identical length segments chosen from two 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 $4 \geq w \leq 11$ 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). MSP is an HSP with the highest matching score. A low threshold T results in more hits and slows down the search. When aligning two random sequences of lengths m and n (n, in this case, is 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

$$E = 1 - e^{-y}, \tag{2.2}$$

where $y = Kmne^{-\lambda S}$, and K and λ are statistical parameters that represent the natural scales of the search space and the scoring system. K and λ depend on the substitution matrix, gap penalties, and 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 of at least S is

$$P = 1 - e^{-y} \sum_{i=0}^{c-1} \frac{y^i}{i!}. \tag{2.3}$$

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

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

In later 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.

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 evolutionary distances and similarities.

3.1 EVOLUTION AND MEASURING EVOLUTION

In terms of sequence evolution, whether the sequence is DNA, RNA, or protein, the measurement often relies on the information of the primary sequence. In general, the measurement is focused on four features: sequence similarity, homology, divergence, and convergence. These features involve the sequences and 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 used for the randomness involved. For example, finding a high

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

similarity between a nonfunctional sequence segment and 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 leads to the 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. In practice, quantification of 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 the available biological information and the information derived from the other three measuring techniques. For example, Calmodulin (CalM: CALcium MODULated proteIN), a calcium-binding protein expressed in all eukaryotic cells to meditate inflammation, metabolism, short-term and long-term memory, nerve growth, and immune response, is well known to have approximately 148 amino acids length with four EF-hand motifs, each of which binds a 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 a human CalM 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 [26] 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 to amino acids) as in Figure 3.2, that is, 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

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, 1's represent an incident of mutation. There are 12 substitutions between the two sequences of length 27, which is about 44% (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 because C could change to T, then change to G, and so on.

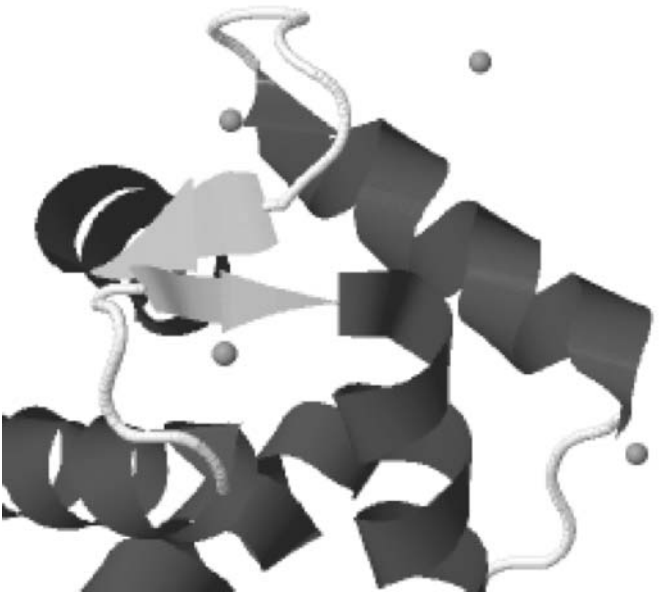


Figure 3.1 A human Calmodulin wraps around its binding domain in the plasma membrane. Ca^{2+} atoms are depicted as circles.

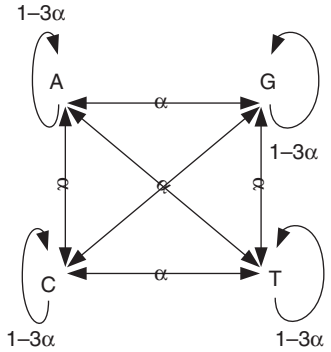


Figure 3.2 Displays the possible paths a nucleotide can be mutated to with the same probability α in Jukes and Cantor's model.

Jukes and Cantor's method estimates that the total actual number of mutations is about 18 ($t = 66\%$), not 44%.

Jukes and Cantor's equation is defined as

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

where t is the distance between two sequences, and p is the fractional dissimilarity defined as the ratio of the 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 is 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 nonidentical or matched and unmatched. The matched pairs are given a positive score (usually 1) and the unmatched pairs get 0 score.

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

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

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. While the hydrophobic isoleucine (I) will get a negative score with hydrophilic cysteine (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 [25, 27] 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, that is, 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 2% 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 database searches. The log-odds [28–30] 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 are 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

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

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:

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

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

$$M(a,b) = \lceil \lambda \, logR(a,b) \rceil, \tag{3.4}$$

where λ is a scaling factor. Table 3.2 shows the PAM250 matrix.

Similarly, Blocks Substitution Matrix (BLOSUM) [31, 32] is also designed for scoring alignments between divergent protein sequences used in database search developed along with BLAST. BLOSUM was first mentioned in [25]. 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 represents 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:

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

where p_{ij} is the probability of 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

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	17
	C	G	P	S	A	T	D	E	N	Q	H	K	R	V	M	I	L	F	Y	W

the i and j in a sequence. λ 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 represents the BLOSUM45 derived from sequence blocks clustered at the 45% identity level.

GONNET matrix [33] is another matrix developed by Gonnet, Cohen, and Benner in 1992 using exhaustive protein pairwise Needleman–Wunsch's alignment algorithm with PAM matrix on entire existing protein sequence database at the time. GONNET matrix is shown in Table 3.4.

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 is 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 [34] showed that deletions/insertions are observed primarily in unstructured coils or surface regions between secondary structure units of homologous protein sequences (i.e., α-helices and β-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.

TABLE 3.3 BLOSUM45 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	12
	G	P	D	E	N	H	Q	K	R	S	T	A	M	V	I	L	F	Y	W	C

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	−1
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	X

The gap penalty is an 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 above-mentioned matrices serve only one purpose, that 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, that is, 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, that is, the number of substitutions must be done to transform one sequence into the other. The main drawback of this simple method is that 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) [35]. Mathematically, sum-of-pair score is define as follows:

Definition 1 *The pairwise alignment score of two induced sequences* s'_i *and* s'_j *where* $s'_i, s'_j \in A$ *is* $S(A) = S(s'_i, s'_j) = \sum_{k=1}^{|A|} S(s'_i[k], s'_j[k])$.

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 pairwise 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 using BLOSUM62 matrix give a score of 8 (assuming the gap-mismatch is -9). Figure 3.3 illustrates these scores.

```
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
(a)               (b)
```

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

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.

3.4 SCORING MULTIPLE SEQUENCE ALIGNMENTS

Aligning and scoring two sequences are quite primitive with these available substitution matrices and pairwise alignment algorithms. However, when more than two sequences are 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 (MSA).

3.4.1 Sum-of-Pair Score

One of the most popular quantitative measurement for MSA is the sum of sum-of-pair score, or simply sum-of-pair (SP). It comes with many variations as in [35–37]. To calculate the SP score of n sequences, the scores of all $\binom{n}{2}$ Pairwise alignments

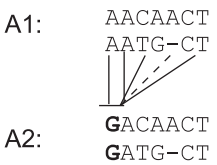


Figure 3.4 Illustrates the steps in assembling an MSA.

within the MSA are added. This sum-of-pair method is the extension of sum-of-pair method mentioned in scoring pairwise alignment to cover all pairwise scores. Formally, the SP score is defined as

Definition 2 *The multiple alignment score of n induced sequences* $s'_1, s'_2, \cdots, s'_n$, *where* $s'_i, s'_j \in A$, *is* $S(A) = S(s'_1, s'_2, \cdots, s'_n) = \sum_{k=1}^{|A|} \sum_{i,j,j\neq i}^{n} S(s'_i[k], s'_j[k])$.

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

3.4.1.1 Example 1 Given the following sequences

S1: MTEITA
S2: MILIA
S3: TITIA

and their alignment A:

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

The SP score of this alignment is the sum of

$$\begin{aligned} \mathrm{S(A)} = {} & [\mathrm{S(M,M)} + \mathrm{S(M,T)} + \mathrm{S(M,T)}] + [\mathrm{S(T,I)} + \mathrm{S(T,I)} + \mathrm{S(I,I)}] \\ & + [\mathrm{S(E,L)} + \mathrm{S(E,T)} + \mathrm{S(L,T)}] + [\mathrm{S(I,I)} + \mathrm{S(I,I)} + \mathrm{S(I,I)}] \\ & + [\mathrm{S(T,-)} + \mathrm{S(T,-)} + \mathrm{S(-,-)}] + [\mathrm{S(A,A)} + \mathrm{S(A,A)} + \mathrm{S(A,A)}] \end{aligned}$$

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

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

$$\mathrm{A}(s_i, s_j, s_k) = \begin{cases} s'_j: & \text{L} & \text{L} & \text{L} \\ s'_k: & \text{I} & \text{M} & \text{I} \\ s'_i: & \text{M} & - & \text{I} \end{cases}$$

yields a score of -5, which is higher than the alignment of

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

yielding a score of -8.

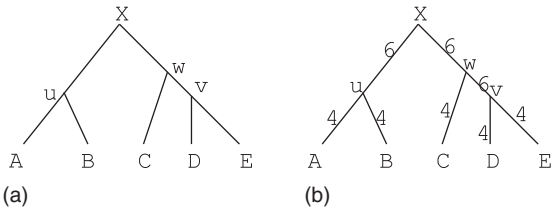


Figure 3.5 Traversal of a tree by an SP measurement. The total visits on each edge is shown. Some edges are visited more often than others.

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. In Figure 3.5, (a) represents the evolution tree of the sequences (the leaves) and (b) represents the SP accounting method for scoring the MSA. The score of a pairwise alignment (A, B) evaluates the likelihood evolutionary events on edges (A, u) and (u, B) of the tree. Similarly, the score of a pairwise 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 is 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 run-time 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.

3.5 CIRCULAR SUM SCORE

Circular sum (CS) score [38] 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, $S = s_1, \ldots, s_n$, 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 as seen in Figure 3.6.

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.

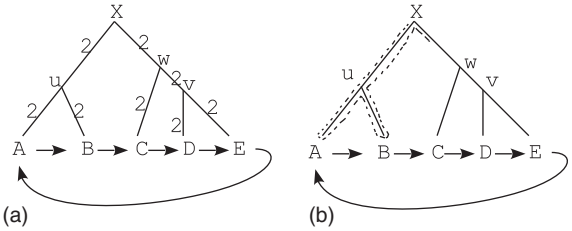


Figure 3.6 Circular traversal of a tree. The dotted arrow indicates the traversal path.

3.6 CONSERVATION SCORE SCHEMES

When multiple residues are grouped together, scoring them are no longer trivial. Each residue has its own stereochemical 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 [39] proposed a method to measure sequence conservation. The method is as follows:

$$S = \frac{k}{nl} \times N, \tag{3.6}$$

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 is

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

where $k = 3$ for three symbol types, $nl = 2$ since I occurs twice in that column, and $N = 3$ as three sequences involved. Thus, the score is $\frac{3}{2} \times 3 = 4.5$.

This method simply ignores the probability that some other nonconserved 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 ignore the other conserved half.

3.6.2 Jores's Method

In 1990, Jores *et al.* [40] proposed a different scoring scheme to address the problem in Wu and Kabat's method. Their scoring method is defined as

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

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 occur. 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's method does not take gap penalty into account.

3.6.3 Lockless and Ranganathan's Method

Lockless and Ranganathan [41] propose a different scoring method where they measure the conservation of an amino acid in a position that 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 $X \sim Bin(N, q_a)$, in which $Bin(N, q_a)$ is the binomial probability of a. For example, if one half of the residues in SWISS-PROT [10] database were all As, then the probability of A occurring eight times in a column with 10 residues is the same as the probability of tossing a coin 10 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

$$d(n_a, \overline{n_a}) = ln\left(\frac{P(X = n_a)}{P(X = \overline{n_a}}\right), \tag{3.8}$$

where $\overline{n_a}$ 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 of overall protein amino acids, such as

$$S = \sqrt{\sum_a d(n_a, \overline{n_a})^2}. \tag{3.9}$$

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

3.7 DIVERSITY SCORING SCHEMES

Shannon's entropy, proposed in 1948 [42], is commonly used to quantify the residue 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: Consider 10 colored balls of 5 reds, 2 greens, and 3 yellows. The total distinct arrangement is $10!/(5!2!3!) = 2520$. In general, with n symbols of k types, the total distinct permutation is given as

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

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

$$lnW = -N \sum_{i}^{k} p_i lnp_i, \tag{3.11}$$

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

$$S = -\sum_{i}^{k} p_i log_2 p_i. \tag{3.12}$$

3.7.2 Methods

In 1991, Sander and Schneider [43] defined a conservation score based on a normalized Shannon's entropy as

$$S = -\sum_{i}^{k} p_i lnp_i \times \frac{1}{lnk}, \tag{3.13}$$

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

Similarly, Shenkin et al. [44] proposed their score as follows:

$$S = 2^S \times 6, \tag{3.14}$$

where S is the Shannon's entropy and k is the number of symbol types. These two equations are transformations of each other; thus, they are very similar. One problem with these functions is that the maximal diversity only occurs when all symbol types are represented evenly. Second, these functions will get their maximum 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 method [45]. Their function is designed to compare sequence conservation against structural conservation in multiple alignments of protein sequence structures. The function is defined as

$$S = \sum_{i}^{k} \overline{p_i}\ log_2\ \overline{p_i} - \sum_{i}^{k} p_i\ log_2\ p_i, \tag{3.15}$$

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

Similar to 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 stereochemical properties of residue symbols in a column. The stereochemical property is first classified by Taylor in 1986 [46]. The classification is based on the residue conservation from the database and the Dayhoff's mutation matrix [25]. The classification is represented in a Venn diagram as depicted in Figure 3.7.

In 1987, Zvelibil et al. [47] 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

$$S = n_{const} \times \frac{1}{10}, \tag{3.16}$$

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 nine properties, $n_{const} = 9$, in the table except D is small and E is not.

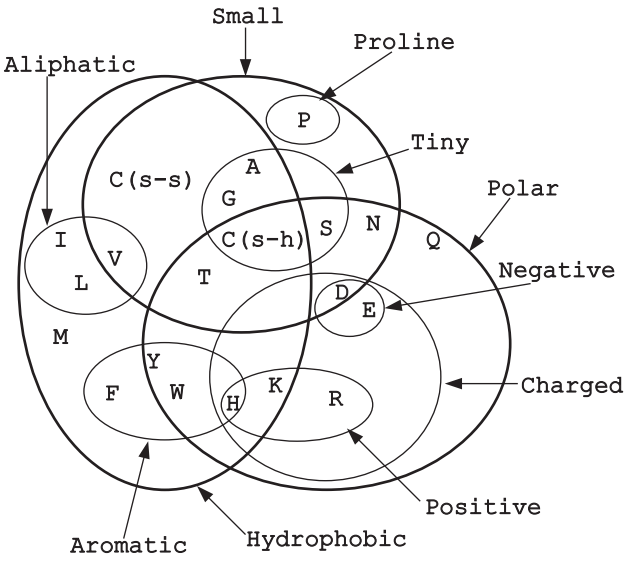


Figure 3.7 Taylor's Venn diagram of amino acid properties.

Properties: Amino acid	Hydro-phobic	Polar	Small	Charged	Positive	Negative	Tiny	Aliph-atic	Aroma-tic	Proline
I	√							√		
L	√							√		
V	√		√					√		
V	√		√							
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.

3.8.1 Valdar's Method

In 2001, Valdar and Thornton [48] propose a scoring method to measure the conservation and the frequency of residues in a column as follows:

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

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

$$\lambda = \frac{1}{\sum_{i}^{n} \sum_{j>i}^{n} w_i w_j}, \tag{3.18}$$

where w_i is the weight of sequence s_i such that

$$w_i = \frac{1}{n} \sum_{i}^{n} d(s_i, s_j), \tag{3.19}$$

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

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

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

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

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

$$S(x) = (1 - t(x))^{\alpha}(1 - r(x))^{\beta}(1 - g(x))^{\gamma}, \tag{3.22}$$

where t is a function of normalized symbol diversity relating to Shannan's entropy [42]; r is a function of stereochemical diversity, g is a function of gap cost; and α, β, and γ 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, we will investigate a different model of scoring.

3.9 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 a single column k is $P(k) = \prod_1^n p(a|b), P(k) \to 0$. 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) [49] 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 as seen in Figure 3.9. Thus, a scoring mechanism for measuring group-to-group matches is provided. 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 is 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

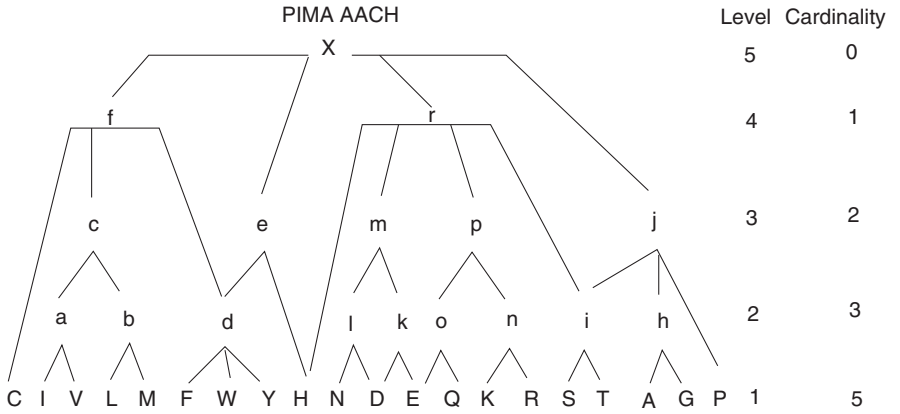


Figure 3.9 Amino Acid Class Hierarchy (AACH) used in PIMA [50]. Uppercase characters are amino acids; lowercase characters are amino acid classes. X is the wild-card character of any type, including a gap.

with E, or F with P, and so on; thus, adding 4 to every entry in the substitution matrix makes 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

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
$M_{i,j} \leftarrow (M_{i,j} + abs(min(M))$
endfor
endif
for all $c \in T$
$c_{cardinality} \leftarrow 0$
endfor
for all $c \in T$
if $size(c) = 1$
$c_{cardinality} \leftarrow M(c, c)$
else
for all $(i, j) \in c, i \neq j$
$c_{cardinality} \leftarrow (c_{cardinality} + M(i, j))$
endfor

$c_{cardinality} \leftarrow \left(c_{cadinality} / \frac{(size(c) \times (size(c)-1)}{2 \times c_{level}} \right)$
endif
endfor
if $X_{cardinality} > 0$! adjusting the root
for all $c \in Sdo$
$c_{cardinality} \leftarrow (c_{cardinality} - X_{cardinality})$
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 pairwise sum of the 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 column scores 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 decrease proportionally in the logarithmic length extension of the alignment. Thus,

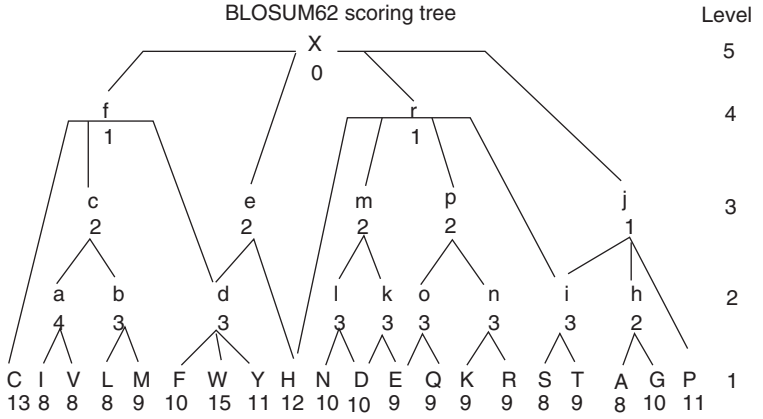


Figure 3.10 HEP scoring tree generated from BLOSUM62 substitution matrix. The matching cardinality of each amino acid class is shown at the bottom of the corresponding class symbol.

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.

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

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:

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

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

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

$\forall w_l > 0$, where

$|T_i|$: the size of score tree for column i, that is, 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,
Γ_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, compared to sum-of-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

$$\frac{1}{(n-1)^{\Gamma}} \times \left[(n-1-a_1)^{c_l} + (n-1-a_2)^{c_1} + \cdots + (n-1-a_{\frac{n}{2}})^{c_1} + \cdots + \frac{(a_i + a_j - 1)^{c_t}}{t} + \cdots + \frac{(n-1)^0}{5}\right],$$

$$\forall c_t \geq 1 \text{ and } a \geq 1, i \neq j, 1 \geq i, j \leq \frac{n}{2}, \text{ and } 1 \leq t \leq logn.$$

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 homogeneous 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 makes it a d degree divergent tree.

3.9.4 Examples

Consider the following sequences: $s_1 = NNN$, $s_2 = NND$, and $s_3 = NDE$. A possible alignment is

$$A_3 = \begin{cases} s_1 & NNN \\ s_2 & NND \\ s_3 & NDE \end{cases}.$$

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:

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

(Figure 3.11 shows the active scoring tree using PIMA scoring cardinal scheme.)

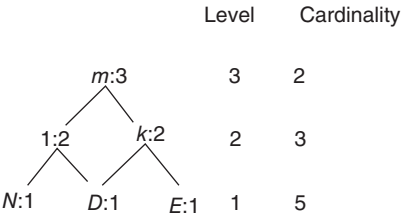


Figure 3.11 PIMA active scoring tree for the third column.

If we use 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

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

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 the known sequences. Normalizing and factorizing out the redundancy in this case are not useful and may lead to unwanted results. Therefore, the weight factor should be from a function that measures the significance

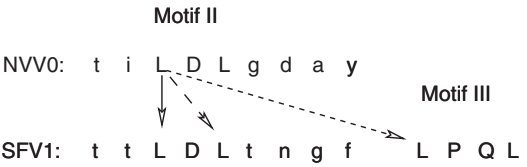


Figure 3.12 Matching Leucine from NNV0 with a similar amino acid in sequence SFV1 [51]. The dotted arrow shows a mismatch, the dashed arrow shows a better match, and the solid arrow shows the best match.

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

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

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

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

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

3.9.6 Evaluation Data Sets

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 an MSA. For this test, we utilize the theoretical sequence data set created by Valdar and Thornton [48]. These theoretical residue conservation sets are 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 MSA benchmarks: RT-OSM [51], BAliBASE3.0 (Benchmark Alignment dataBASE) [52], and PREFAB4.0 [53]. Along with these benchmarks, the following MSA tools: ClustalW, PIMA, DCA, DALIGN2, and MAFFT2, T-COFFEE [50, 54–58] are used to generate alignments for the test. And the third test is to detect whether HEP scoring function can lead to better MSA 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 MSA program. These three scoring functions are chosen for various reasons: SP is the most popular scoring function 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.

3.9.6.1 Theoretical Residue Conservation Measurement In 2001, Valdar developed a theoretical data set [48] to evaluate scoring function's capability on measuring residue conservation correctly. This data set 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 MSA perspective, column (*c*) indicates that phenylalanine (F) is conserved and shared by 9 out of 10 sequences. On the other hand, column (*d*) fails to indicate which amino acid is conserved among all the sequences. Analytically, the chance that aspartic acid (D) will

TABLE 3.5 Each Label Column Represents a Residue Position in a Multiple Sequence Alignment

	Columns										
Sequences	(*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		-

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*). *Source*: Adapted from Valdar and Thornton 2001 [48]).

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 $(p_{\mathrm{DE}})^5 = \prod_1^5 p_{\mathrm{DE}}$ [where p_{DE} is the probability that D is mutated to 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 [32]

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

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 [32]. With the background frequencies $f_{\mathrm{D}} = 0.0539$,$f_{\mathrm{E}} = 0.0539$, and $f_{\mathrm{F}} = 0.469$ obtained from [32], we found that $p_{\mathrm{DE}} = 5.8\mathrm{E}-3$ and $f_{\mathrm{DF}} = 9.0\mathrm{E}-4$. Thus, $\prod_1^5 p_{\mathrm{DE}} = 6.56357\mathrm{E}-12 \ll p_{\mathrm{DF}} = 9.0\mathrm{E}-4$. 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.

3.9.6.2 RT-OSM Data Set It is best to measure the reliability of a scoring method based on actual facts. Therefore, it is realistic to use well-characterized known motifs to validate scoring functions. In 1999, Hudak and McClure [51] prepared the RT-OSM data set for such purpose. The data set is shown in Figure 3.13. These

	I	II	III	IV	V	VI
HT13	pvk**K**a--	t-**IDL**kdaf	-**LPQG**-fk	q**YMDDI**ll	sh**GL**--	k**FLG**qii
NVV0	ikk**K**---	ti**LDI**gday	-**LPQG**-wk	-**YMDDI**yi	qy**GFM**-	k**WLG**fel
SFV1	pvp**K**p--	tt**LDL**tngf	-**LPQG**-fl	a**YVDDI**yi	na**GYV**v	e**FLG**fni
HERVC	pvp**K**p--	tc**LDL**kdaf	-**LPQR**-fk	q**YVDDL**ll	tv**GIR**c	c**YLG**fti
GMG1	mvr**K**a--	tk**VDV**raaf	-**CPFG**-la	a**YLDDI**li	--**GLN**-	k**YLG**fiv
GM17	v-p**K**kqd	tt**IDL**akgf	-**MPFG**-lk	v**YLDDI**iv	--**NLK**-	t**FLG**-hv
MDG1	lvp**K**ksl	sc**LDL**msgf	-**LPFG**-lk	l**YMDDL**vv	--**NLK**-	t**YLG**-hk
MORG	vvr**K**k--	tt**MDL**qngf	-**APFG**-fk	l**YMDDI**iv	--**GLK**-	h**FLG**-hi
CAT1	lvd**K**pkd	eq**MDV**ktaf	k**SLYG**-lk	l**YVDDM**li	--**EMK**-	r**ILG**idi
CMC1	tit**K**rpe	hq**MDV**ktaf	k**AIYG**-lk	l**YVDDV**vi	---**KR**-	h**FIG**iri
CST4	ftk**K**rng	t-**LDI**nhaf	k**ALYG**-lk	v**YVDDC**vi	in**KLK**-	d**ILG**mdl
C1095	fnr**K**rdg	tq**LDI**ssay	k**SLYG**-lk	l**FVDDM**il	it**TLK**k	d**ILG**lei
NDM0	mih**K**t--	af**LDI**qqaf	g**VPQG**svl	t**YADDT**av	ts**GL**--	k**YLG**itl
NL13	lip**K**p--	s-**IDA**ekaf	g**TRQG**cpl	l**FADDM**iv	vs**GYK**-	k**YLG**iql
NLOA	fip**K**a--	af**LDI**egaf	g**CPQG**gvl	g**YADDI**vi	ev**GLN**-	k**YLG**vi-
NTC0	vlr**K**p--	am**LDG**rnay	g**VRQG**mvl	a**YLDDV**tv	al**GIE**-	r**VLG**agv
ICD0	eip**K**p--	vd**IDI**k-gf	g**TPQG**gil	r**YADDF**ki	rl**DLD**i	d**FLG**fkl
IAG0	fkk**K**t--	ie**GDI**ks-f	g**VPQG**gii	r**YADDW**lv	el**KIT**l	-**FLG**vnl
ICS0	wip**K**p--	ld**ADI**sk-c	g**TPQG**gvi	r**YADDF**vi	em**GLE**l	n**FLG**fnv
IPL0	yip**K**s--	le**ADI**r-gf	g**VPQG**gpi	r**YADDF**vv	sr**GLV**l	d**FVG**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. (*Source*: Adapted from Hudak and McClure 1999 [51].)

are order-specific-motif (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 functions to be evaluated are SP [35], Valdar's [48], Trident [48], Smith's PIMA [50], and HEP scoring functions.

3.9.6.3 BAliBASE3.0 and PREFAB4.0 Data Sets The BAliBASE3.0 (Protein REFerence Alignment Benchmark) [52] is a reference benchmark of sequences that are manually aligned with detailed annotations. The alignment of BAliBASE sequences is based on 3D structural superposition, except the transmembrane sequences. BAliBASE MSA reference benchmark consists of 247 reference alignments, containing over 1000 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 of few short and distanced sequences. Similarly, PREFAB4.0 [53] benchmark contains 1932 alignments averaging 49 sequences of length 240. 150 reference alignments are randomly selected to use as test cases. Both BAliBASE3.0 and PREFAB4.0 use a 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 a specific purpose, so each evaluation must be designed for each data set. Applying to the theoretical sequence set, the sequence columns are ranked using HEP and the results are compared to the predefined ranks. For the

RT-OSM sequence set, the residues in the sequences are randomly shifted and the gaps are removed 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 an MSA result indicates the functional importance of that aligned position. Thus, when the residue in the RT-OSM data set is 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.

Moreover, for the BAliBASE3.0 and PREFAB4.0 benchmarks, we define a reliability score to evaluate the scoring functions.

3.9.7.1 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 the measurements, where HEP-PIMA is HEP score using Smith's PIMA cardinality, HEP-P250 is HEP score using the generated PAM250 scoring tree, and HEP-BL62 is HEP score using the generated BLOSUM62 scoring tree. The correct rankings 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) is 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.

3.9.7.2 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 six different alignments, each containing one to six of the motifs aligned. These alignments are scored by the following scoring methods: Sum-of-pair (SP) [35], PIMA [50], Valdar and Thornton [48], Trident [48], HEP-PIMA, HEP-P250, and HEP-BL62 (as described 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% 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, that is,

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

TABLE 3.6 Each Label Column Represents a Residue Position in a Multiple Sequence Alignment

Columns											
Sequences	(*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

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 four sequences (no gaps); and the table cannot show all significant digits.

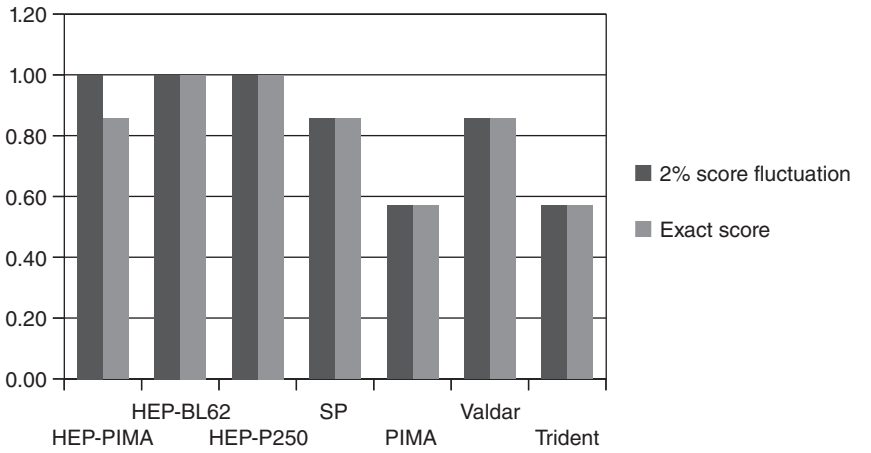


Figure 3.14 Motif detection accuracy on the RT-OSM data set. Score of 1.0 means a method correctly detects and ranks all six motifs in the data set.

Figure 3.14 shows the motif detection accuracy of these scoring methods. Both versions 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 [50] and Valdar's method [48]. Overall, all three versions of HEP scoring method consistently outperform other scoring methods.

3.9.7.3 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, that is, 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, that is,

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

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 us further assume that the references of total column scores of these alignments are $\text{TC(A)} = 1$, $\text{TC(B)} = 0.8$, and $\text{TC(C)} = 0.3$. These TC scores indicate that 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 functions get zero point. For n different sets of sequences, the reliability score is the number of points that each scoring function accumulates over n.

Since random alignments are often meaningless, we use MAFFT2 [57], ClustalW [59], T-COFFEE [58], PIMA [50], and DALIGN2 [56] MSA tools to generate 5 MSAs 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 sets 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.

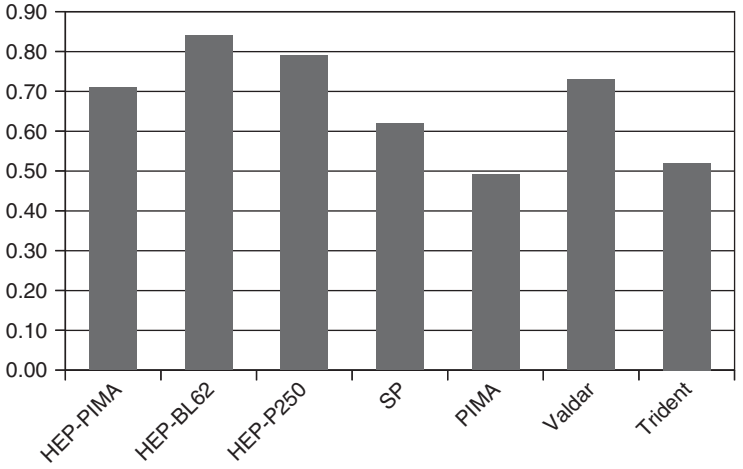


Figure 3.15 Reliability scores on the BAliBASE3.0 benchmark.

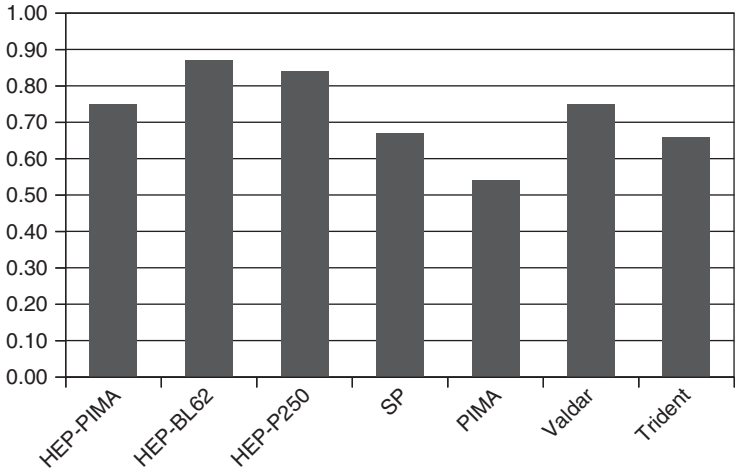


Figure 3.16 Reliability scores on 150 references of the PREFAB4.0 benchmark.

Both BAliBASE3.0 and PREFAB4.0 benchmark tests indicate that HEP scoring method with BLOSUM62 and PAM250 is more likely to identify and rank the MSAs correctly. This feature allows biologists to identify which alignment tools are more accurate and reliable for their MSAs. From these results, it seems that HEP scoring with BLOSUM62 is the most reliable method.

3.9.7.4 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 MSA results, we have implemented HEP in a progressive MSA, which is the same alignment technique employed in ClustalW, PIMA, T-COFFEE, and so on. 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 [60] or NJ [61] hierarchical clustering methods.
(iii) Pairwise align the sequences (or two groups of sequences) following the order of the dendrogram from the leaves to the root.

In this 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 [19] algorithm are implemented: one uses SP [35] scoring method, one uses Smith's Pima [50] method, one uses Valdar's method [48], 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.

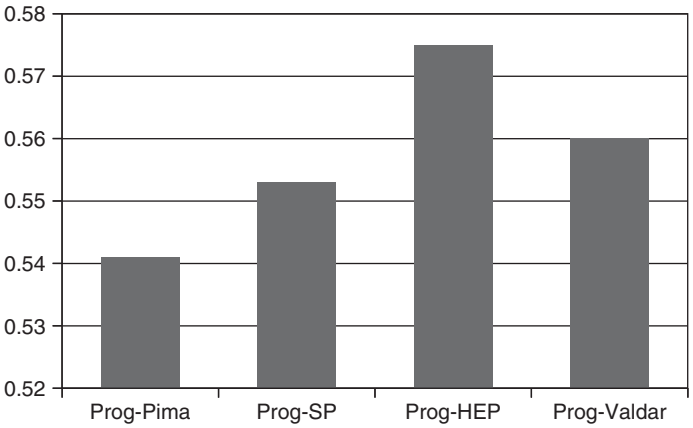


Figure 3.17 BAliBASE total column (TC) scores.

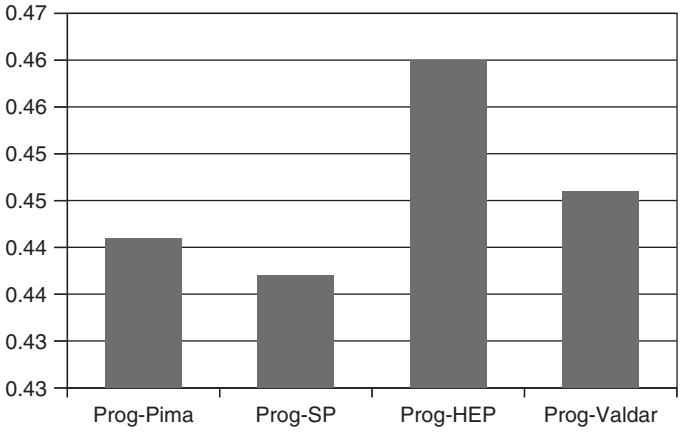


Figure 3.18 Total column (TC) scores of 150 random sequence sets from PREFAB4.0.

When executing this 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/deletion 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. Of 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 [50], SP [35], and Valdar and Thornton [48] 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 Figures 3.17 and 3.18. Thus, it is a strong indication that HEP scoring method will improve MSA accuracy.

As a conclusion to this chapter, each scoring scheme is designed for a specific purpose and may work on a 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.

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, that is, the homologous features and functional similarities between sequences. Figure 4.1a 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. Figure 4.1b 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 in Figure 4.1c.

There are many algorithms for clustering such as K-means [62], fuzzy C-means [63], hierarchical clustering, and probabilistic clustering [64]. 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 cluster. 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

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

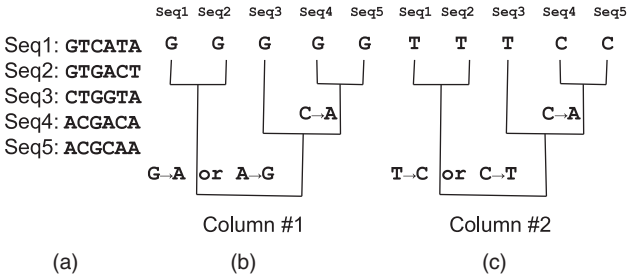


Figure 4.1 (a) is the input sequences, (b) substitutions on first column, and (c) substitutions on second column.

and very time consuming, for example, K-means is NP-hard for d dimensions, and it yields a run-time 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 [61, 65] and unweighted pair group method with arithmetic mean (UPGMA), also known as average linkage method. The weighted version of UPGMA, (WPGMA) [60] also known as maximum linkage, is not popular.

4.1 UNWEIGHTED PAIR GROUP METHOD WITH ARITHMETIC MEAN – UPGMA

Similar to 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 and all the members of the new OTU. This process is repeated until all OTUs are merged.

Example: Consider four sequences A, B, C, D and their distance matrix d.

$d =$

	A	B	C	D
A	0			
B	**7**	0		
C	11	6	0	
D	14	9	7	0

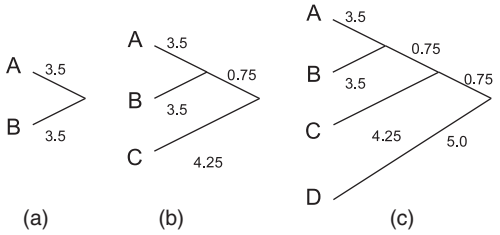


Figure 4.2 Illustrates each step of building the UPGMA phylogeny.

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

$$d_{C(AB)} = (d_{AC} + d_{BC})/2 = (11 + 6)/2) = 8.5,$$
$$d_{D(AB)} = (d_{AD} + d_{BD})/2 = (14 + 9)/2 = 11.5.$$

Thus, the new distance matrix will be

$d =$

	AB	C	D
AB	0		
C	**8.5**	0	
D	11.5	9	0

Next, C is grouped with AB to build OTU ABC and the distance between D and 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 Figure 4.2c. One of the weaknesses of UPGMA is its constant rate of evolution. UPGMA has run-time 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 and all other OTUs. NJ combines a pair of OTUs that are close to each other and far from all other OTUs into a new OTU. The distance between all other OTUs to the new OTU is calculated. This process is repeated until only two

OTUs remain. This algorithm is detailed as follows: Consider 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, that is, 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: Consider four sequences A, B, C, D and their distance matrix d.

$d =$

	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

$$M_{AB} = d_{AB} - u_A - u_B = 1 - 3.5 - 3.55 = -6$$
$$M_{AC} = d_{AC} - u_A - u_C = 3 - 3.5 - 5 = -5.5$$
$$M_{AD} = d_{AD} - u_A - u_D = 3 - 3.5 - 5 = -5.5$$
$$M_{BC} = -5.5$$
$$M_{BD} = -5.5$$
$$M_{CD} = -6.$$

Therefore,

$M =$

	A	B	C	D
A				
B	−6			
C	−5.5	−5.5		
D	−5.5	−5.5	−6	

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 us 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

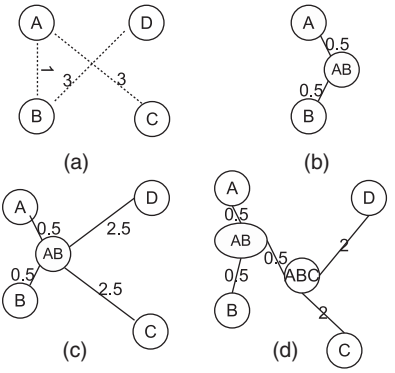


Figure 4.3 Producing NJ tree – not drawn to scale. (a) is initial star tree; (b) shows the combined OTU of A and B; (c) is the NJ tree with the new OTU AB; and (d) is the final NJ evolution tree.

The second iteration of the algorithm will produce

Step 1:

$$u_{\mathrm{AB}} = (d_{(\mathrm{AB})\mathrm{C}} + d_{(\mathrm{AB})\mathrm{D}})/1 = (2.5 + 2.5)/1 = 5$$
$$u_C = (2.5 + 4)/1 = 6.5$$
$$u_D = (2.5 + 4)/1 = 6.5.$$

And step 2 gives

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

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

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

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

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

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 farthest leaf OTUs.

4.3 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, there is a clustering algorithm designed specifically for multiple sequence alignment called Overlapping Sequence Clustering (OSC) [66]. 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 local alignment scores, 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

$$R^+ = \bigcup_{i \in \mathcal{N}} R^i, \tag{4.1}$$

Figure 4.4 Illustration of 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.

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

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

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

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

where $A(s_i, s_j)$ is the local alignment of two sequences s_i and s_j.

The Overlapped Sequence Clustering algorithm is defined 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 kth 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 are 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 nonclustered 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 run-time complexity of $O(kn)$, where n is the number of sequences to be clustered and k is the number of clusters.

Example: Consider sequence 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	**11**								
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	**13**	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}.

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, that is, 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 $O(n^4)$.

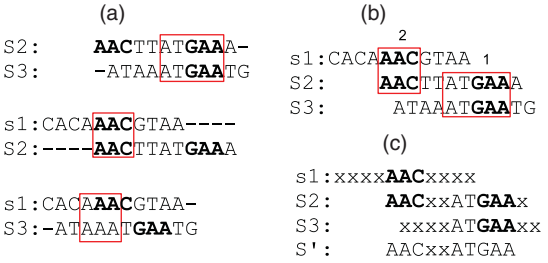


Figure 4.5 Illustration of the clustering of the sequences to generate a sequence pattern. (a) is the all-pairwise alignments, (b) is sequence clusterization, and (c) is sequence pattern identification, where S' is the cluster pattern. The boxes represent the DP local alignment results.

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.

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 [67]. 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.

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 [52] reference 1 ubiquitin set.) With this extension, the MSA problem becomes intractable, in fact, finding the optimal solution for MSA is an NP-complete problem as proved by Wang and Jiang in 1994 [68]. 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 MSA 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, and similarity. In 1999, Thompson et al. [69] performed a comprehensive comparison of the 10 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, was 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, that is, sequences that share less than 25% identity.

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

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

```
1ubi      MQIFVKTLTGKTITLEVEPSDTIENVKAKIQDKEGIPPD--------QQR
1guaB     NTIRVFLPNKQRTVVNVRNGMSLHDCLMKALKVRGLQPEC--------CAV
1alo      MIQKVITVNGIEQNLFVDAEALLSDVLRQQLGLTGVKVGCEQGQCGACSV
1awd      YKVTLKTPSG-EETIECPEDTYILDAAEEA-GLD-LPYSCRAGACSSCAG
```

Figure 5.1 Sample alignment of BAliBASE reference 1 [52] ubiquitin set. Each block represents a secondary feature such as alpha helices, beta strands, or unimportant structures.

5.1 DYNAMIC PROGRAMMING

MSA algorithms in this group are extensions of dynamic programming.

5.1.1 DCA

Divide and Conquer multiple sequence Alignment (DCA) algorithm is designed by Stoye [54] for his dissertation. The basic idea of DCA is rather simple: each sequence is split into two subsequences 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, that is, shorter than a predefined threshold *L*. These submultiple sequence alignments are then optimally aligned by dynamic programming method. The alignments of these subproblems are then concatenated yielding an MSA for the original sequences. This algorithm is illustrated in Figure 5.2.

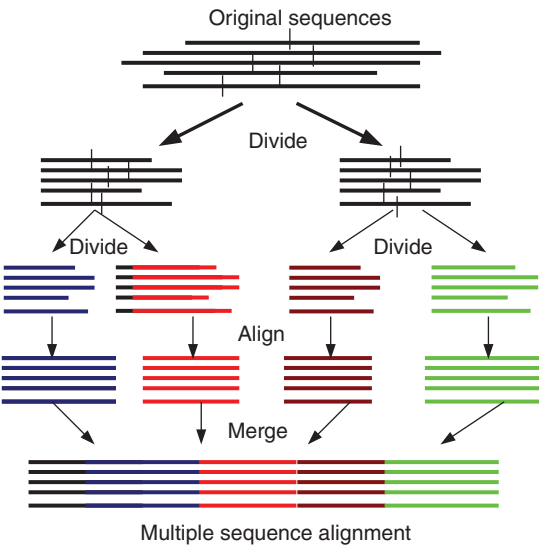


Figure 5.2 Illustration of the divide and conquer multiple sequence alignment (DCA).

5.1.1.1 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

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

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:

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

where $\alpha_{p,q}$ is the sequence-dependent weight factor. The dependent weight factor is used with the sum-of-pair scores [70] to increase the weight (or score) of a matching pair of residues. It is defined as

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

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, that is, 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 [71], 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 implements dynamic program technique.

5.2 PROGRESSIVE ALIGNMENT

Progressive MSA is introduced by Feng and Doolittle in 1987 [72]. The main idea is that a pair of sequences with minimum edit distance is most likely to originate from

a recently diverged species. Thus, these optimally aligned 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 are based on this technique, such as ClustalW [55, 59], MULTALIN [73], PILEUP [created by Genetics Computer Groups(GCG), which is commercially known as Accelrys], or PIMA. The difference between these programs is minimal. For example, PIMA [50] 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 [74] uses Wu-Manber [75] 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.

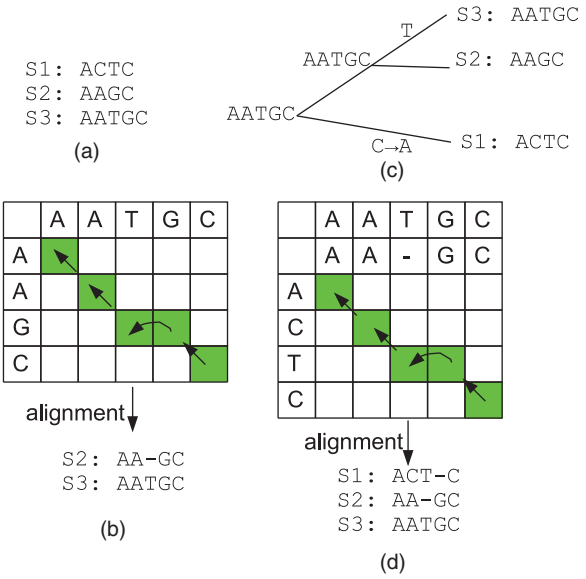


Figure 5.3 An example of progressive multiple sequence alignment. (a) is the input sequences, (b) is an alignment guiding tree, (c) is external nodes alignment, and (d) is internal nodes alignment.

5.2.1 Clustal Family

The most popular program in this group is the Clustal family (including ClustalX and ClustalW) [55]. It is fast and relatively reliable. Similar to 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 that 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 others are the gap penalties. For example, Clustal assumes that short stretches of hydrophilic residues (e.g., 5 or more) is an indication loop or random coil regions so that it reduces the opening gap penalty for these stretches. Similarly, it increases the opening gap penalty for any gap that are less than 8 residues apart based on the observation of alignments between sequences of known structures, which rarely has a gap within 8-residue segments. The initial opening gap penalty is

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

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 number of mismatched pairs over the total number of residue pairs.

The extended gap penalty is defined as

$$GEP \times \left(1.0 + \left|log\left(\frac{n}{m}\right)\right|\right), \tag{5.5}$$

where GEP is the user-defined extended gap cost.

5.2.2 PIMA: Pattern-Induced Multisequence Alignment

PIMA [50, 76] 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 [60] (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,

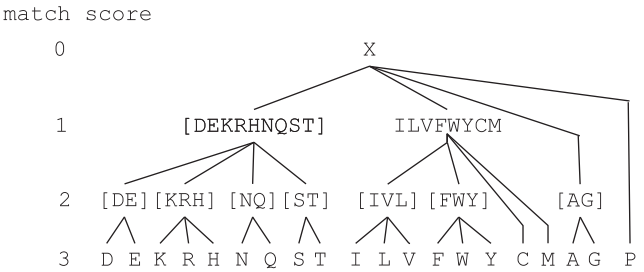


Figure 5.4 The Amino Acid Class Hierarchy (AACCH) used in PIMA family; X represent a wildcard for matching any symbol, including gap.

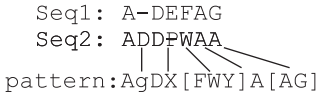


Figure 5.5 A pattern generated from a pair-wise alignment.

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 wildcard that matches any symbol, and (DE) represents either D or E. 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 [77] is an extension of progressive alignment 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, that is, no better alignment score found. In its earlier version, PRRN [78], 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

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

where a_i and b_j are the column ith of group a and column jth of group b; $w_{q,p}$ is the weighted sum-of-pair score of two sequences represented as rows p and q; m and n are the number 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 pth and qth 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 [56] combines both global pair-wise alignments and local pair-wise alignment features. MSAs 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 sequences, 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.

5.2.4.1 Tabu Search Tabu search [79, 80] 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 MSA [81] 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 removed after x number of iterations so that they can be back in the solution space. The moves are repeated until convergent, that is, 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.

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

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 model of the sequence family primary structure consensus as in [82, 83]. The probability that a region being a codon is calculated as

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

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. HMM is a finite state machine with the triple $(A, B, \prod)$, where A is the transition probabilities, B is the output probabilities, and $\prod$ is the initial state probabilities. For any given time $t \geq 1$, $A, B, and \prod$ are defined as follows:

$A = \{a_{ij} = P(q_j \text{ at } t+1 | q_i att)\}$, 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 $o_k \in \Sigma$, that is, B is the probability that the output is o_k with given state q_i. And π is calculated as

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

An example of HMM being used in multiple sequence alignment is shown in Figure 5.6.

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

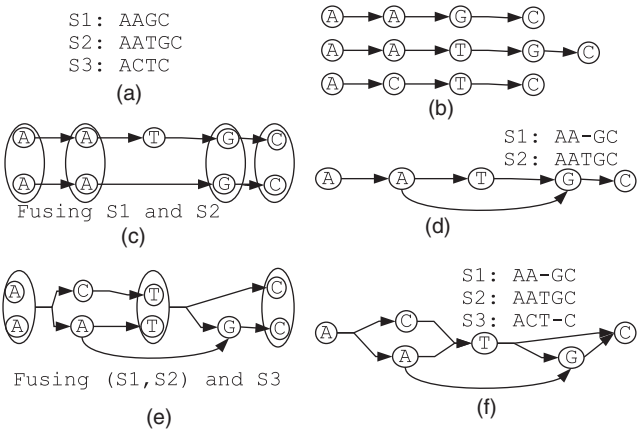


Figure 5.6 POA algorithm: (a) input sequences, (b) directed graphs for the sequences, (c) graph fusion of sequences S1 and S2, (d) alignment of S1 and S2, (e) graph fusion of S3 to alignment (S1,S2), (f) final partial order graph and final multiple sequence alignment.

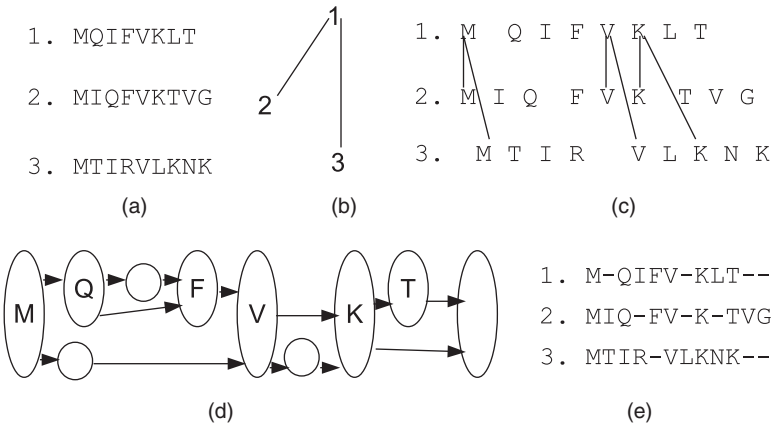


Figure 5.7 PSAlign algorithm: (a) input sequences, (b) minimum spanning tree, (c) undirected graph constructed from pair-wise alignments and spanning three, (d) partial order graph, and (e) final alignment result.

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 [84, 85] are developed based on this technique. In POA, each sequence is converted into a directed graph. The graphs are then combined to identical residue symbols. This process is repeated until all graphs are merged together. Figure 5.7 illustrates each step of POA.

5.3.2 PSAlign

Similarly, PSAlign [85] is a polynomial time solvable MSA program developed based on finding the shortest preserving alignment. This method starts by pair-wise alignment of the sequences similar to those in T-Coffee [58] and ProbCons [86] 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 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 5.8 illustrates this algorithm.

The main difference 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 [84].

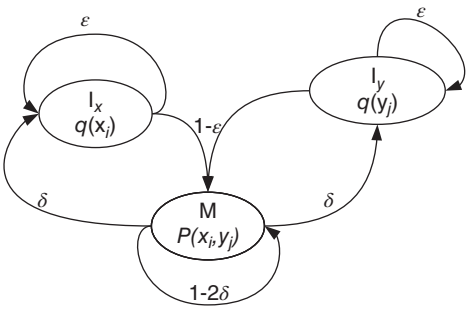


Figure 5.8 Basic pair-HMM for aligning two sequences x and y. State M emits two letters that are being aligned from each sequence. States I_x and I_y emit a letter in sequence x and y, respectively, that are aligned to a gap.

5.3.3 ProbCons: Probabilistic Consistency-Based Multiple Sequence Alignment

ProbCons [86] algorithm is based on the probabilistic consistency of the sequences. It is a pair-HMM-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) [64] 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, is paired in an alignment, and is calculated as follows:

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

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:

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

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

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

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

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

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 [58] is a progressive multiple sequence alignment. It starts by using ClustalW and LALIGN [87] (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 are 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 MSA. T-Coffee algorithm is illustrated in Figure 5.10. 3D-Coffee [88] is an extended version of T-Coffee in which the 3D structural 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 residue pairs in the extended library.

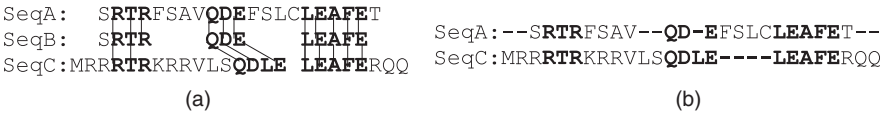


Figure 5.9 Generating extended pair-wise alignment in T-Coffee. (a) sequence SeqA is aligned to sequence SeqC via immediate sequence SeqB. (b) new pair-wise alignment for sequence SeqA and sequence SeqC.

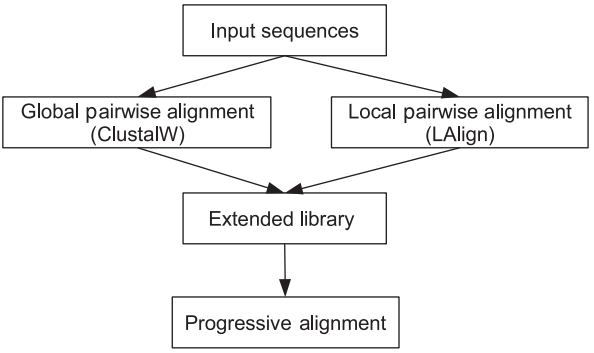


Figure 5.10 T-Coffee multiple sequence alignment schema.

5.3.4.1 Example Using the same sequences in Figure 5.9:

sequence a: S**R**TRFSAVQDEFSLCLEAFET
sequence b: S**R**TRQDELEAFE
sequence c: MRR**R**TRKRRVLSQDLE LEAFERQQ

Assume that the alignment score of {a,b} is 88, {a,c} is 100, and {b,c} is 77. The triplet weight is calculated as aligning of $a_2 = \text{R}$ to $c_4 = \text{R}$ gets a weight of 100, and aligning of $a_2 = \text{R}$ to $c_4 = \text{R}$ through $b_2 = \text{R}$ 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 FFT 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 MSA problem, fast Fourier transform greatly reduces the solution space and speeds up 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 $\hat{v}(a) = [v(a) - \overline{v}]/\delta_v$ and $\hat{p}(a) = [p(a) - \overline{p}]/\delta_p$, where overbar represents the average over the number of different symbols (20 for protein and 4 for DNA/RNA), δ_v and δ_p are the standard derivation of polarity and volume.

The correlation between two sequences is defined as

$$c(k) = c_v(k) + c_p(k) \tag{5.11}$$

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

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

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

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

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

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

and

$$\hat{p}_{group1(n)} = \sum_{i \in group1} w_i.\hat{p}_i(n), \tag{5.15}$$

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

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

where M_{ab} is PAM200, $average1 = \sum_a f_a M_{aa}$, $average2 = \sum_{a,b} f_a \cdot f_b.M_{ab}$, 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 opening gap penalty.

5.3.6 AVID

AVID [89] 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 [90]. These segments are then used as anchors. This technique is repeated for segments around the anchors.

5.3.7 Eulerian Path MSA

The Eulerian path method [91] maps the segments of sequences into nodes of a connected graph and finds the Eulerian path through the graph nodes. In this method, a windows 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

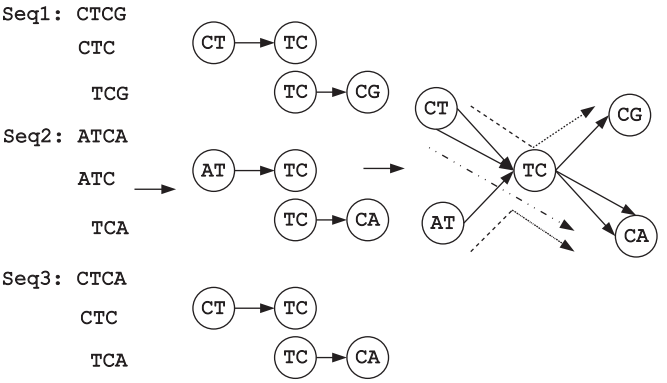


Figure 5.11 Representing three sequences in de Bruijn graph.

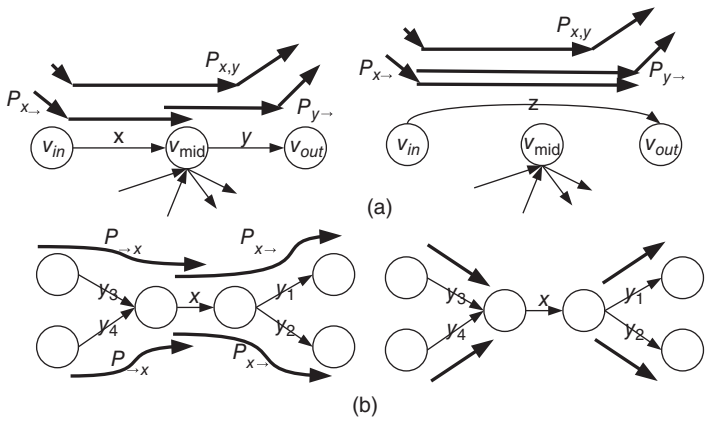


Figure 5.12 Superpath transformation: (a) detachment and (b) *x*-cut.

into a new graph with superpaths [92]. A superpath is a path equivalent to all subpaths going from vertex *x* to vertex *y*.

Figure 5.12 illustrates the transformation of subpaths 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 [93, 94] to solve the MSA 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 [95], GA-DP [96], SAGA [97], RBT-GA [98], GA-ACO [99], and in [100–102], etc.

5.4.1 SAGA: Sequence Alignment by Genetic Algorithm

SAGA [97] optimizes its multiple sequence alignment by applying genetic algorithm (GA) [93, 94]. 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 nth 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 5.13; 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 parameters by users before executing the algorithm. This algorithm is depicted in Figure 5.14.

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 backtracking matrix similar to the one created in dynamic programming. The optimal rubber band is exactly the same as the DP backtracking 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.

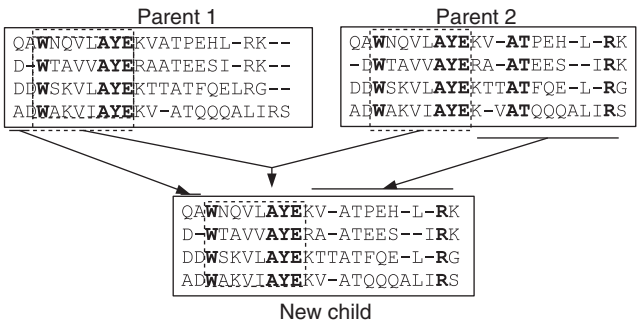


Figure 5.13 Illustration of SAGA, where the two parents are crossed breed. The dotted boxes represent the consistent alignment between the two parents.

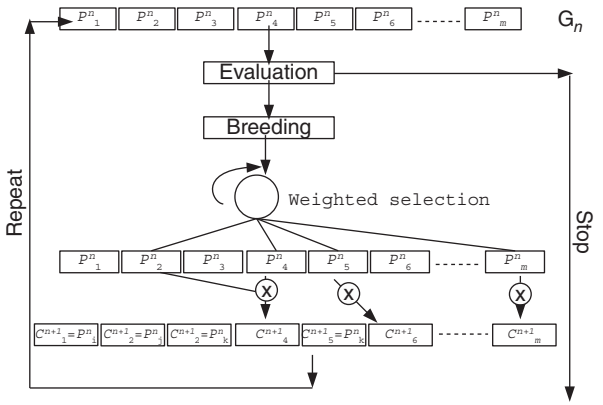


Figure 5.14 SAGA alignment scheme. At any generation G_i, the parents P^i are crossed breed or mutated by a random operation X to generate a new set of children P^{i+1} .

Genetic algorithm with ant colony optimization (GA-ACO) combines the GA technique with ant colony optimization (ACO) [103] 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 [104] 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) generate a list of words using window-sliding method with length 3 for each input sequence; (ii) feed the words into the input nodes of the neural networks; (iii) classify the words that emerge from the third subnetwork as motifs and give them more weights.

The decision of which word is allowed to emerge from the third subnetwork 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 classify them into groups at the top level. Each word in these groups are passed down to the next

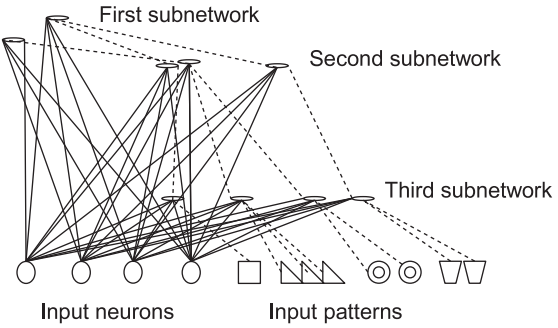


Figure 5.15 Neural networks.

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 similar to the technique done at the top-level subnetwork. A pattern arrived 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.

5.4.3 FAlign

FAlign [105] 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 nonmotif regions to improve the alignment score.

5.5 NEW DEVELOPMENT IN MULTIPLE SEQUENCE ALIGNMENT ALGORITHMS

In this section, we will explore some details of new sequence alignment algorithms and their reliability against popular existing algorithms.

5.5.1 KB-MSA: Knowledge-Based Multiple Sequence Alignment

KB-MSA method [66] utilizes the existing biological sequence knowledge databases such as Swiss-Prot, UniProt, or Homstrad 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 [19] 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 the best hit scores, or scores that are higher than some predefined threshold α. The biological annotated blocks from these sequences are extracted to generate a set of metasequences.

Step 4: Locally pair-wise align the semi-related sequences in each group to their metasequences by Smith and Waterman's algorithm [18]. Sequences that yield low alignment scores will be aligned with other metasequences and regrouped to their highest alignment scored group where the metasequences come from. Subsequences that are identical, or similar, to the annotated blocks of the

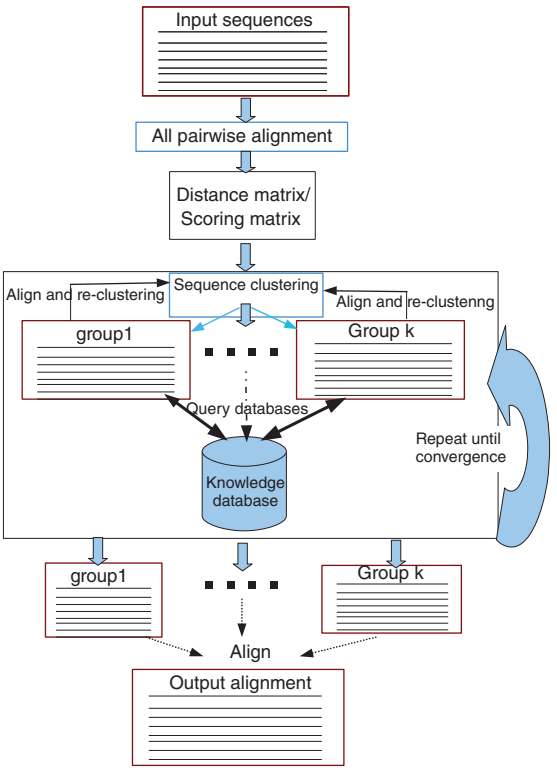


Figure 5.16 Knowledge-based multiple sequence alignment scheme. Initially, the input sequences are partitioned into semi-related groups. The groups' representative sequences are used to search for any available biological information from sequence knowledge bases. The sequences are given different weights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

metasequences are marked as significant blocks and given extra weights for further alignments. Gap insertions in significant blocks are kept intact.

Step 5: Remove metasequences, 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 metasequence.

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) [60].

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

Step 11: Use MAFFT [57] to align nonsignificant blocks to improve alignment score.

5.5.1.1 Assembling the Sequence Alignment Steps 8–10 of the KB-MSA algorithm deserve more details, while other steps are straightforward. 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 [84] where each node represents a single residue.

For example, assuming the input sequences are (obtained from BAliBASE [52])

1*ABOA* *WCEAQTKNGQGWVPSNYITPVN*
1*YCSB* *WWWARLNDKEGYVPRNLLGLYP*
1*PHT* *WLNGYNETTGERGDFPGTYVEYIGRKKISP*
1*IHVA* *AVVIQDNSDIKVVPRRKAKIIRD*
1*VIE* *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 [19] is modified to use a node

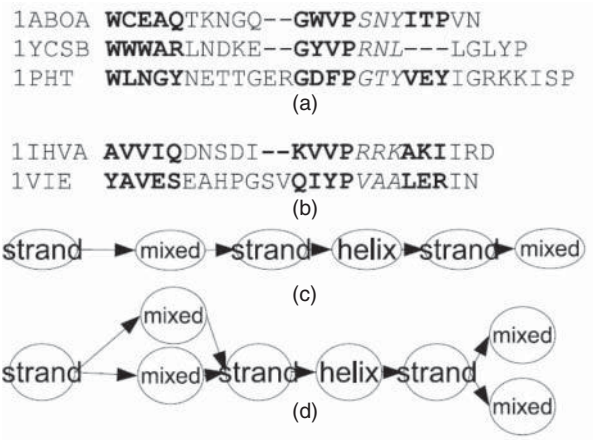


Figure 5.17 Clusters alignment by partial order graph, where bold texts represent beta-strand blocks, italics represents alpha-helix blocks, and others are nonsignificant blocks. (a) and (b) show the two clusters to be aligned, each sequence in the cluster is preceding by its name; (c) is the partial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

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

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, that is, 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 nonoverlapping columns, the Hierarchical Expected matching Probability (HEP) [49] is used to calculate the matching score between columns of two nodes. The sliding technique allows the arrangement of residues in each graph node to remain intact throughout the alignment process.

5.5.1.2 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 *ith* 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

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

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

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

$$A = \begin{cases} L & F & V & A \\ L & F & - & A \end{cases}$$

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

5.5.1.3 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 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 divided by 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 fundamental 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.

5.5.1.4 Alignment Benchmarks To validate the accuracy and reliability of the alignment generated by knowledge-based multiple sequence alignment algorithm (KB-MSA), the BAliBASE [52], PREFAB [53], HOMSTRAD [15], and SABmark [106] 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%

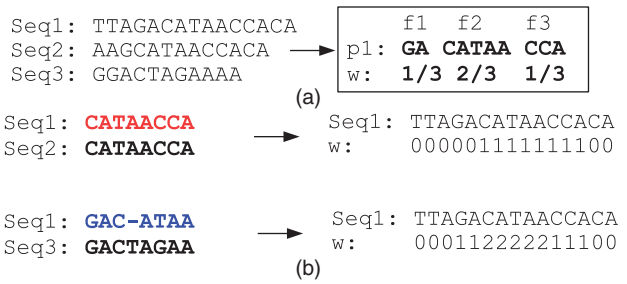


Figure 5.19 Creation of an annotated sequence for KB-MSA consistency database. (a) shows the input sequences and the final annotated pattern for seq1 having three featured blocks f1, f2, and f3 with weights 1/3, 2/3, and 1/3, respectively; (b) shows the immediate pair-wise local alignments of seq1 and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

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 [53] that contains 1682 alignments. These alignments are generated by pair-wise alignment of 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 [106] contains two subsets: the superfamily and the twilight. Each superfamily group represents SCOP superfamily with 25–50% identity. Each group in the twilight subset contains sequences with 0–25% identity. These two subsets are extended to include nonhomologous 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) [15], which contains 130 protein families and 590 aligned structures that are selected from the X-ray structure analysis. Despite that HOMSTRAD is not designed as a benchmark, its sequences are clustered and aligned by families making it a good candidate for MSA.

5.5.1.5 Results and Discussions In these evaluation tests, default parameters are used on ClustalW [55], MAFFT [57], ProbCons [86], and T-Coffee [58]. 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 Swiss-Prot [10] (built on PDP), sequence knowledge database is used in these benchmark tests. It is possible that the knowledge database may contain incomplete sequence information, that is, biological knowledge of some sequences are not available. To simulate this situation, benchmark tests are performed with 10% (KB-MSA 10%) sequence knowledge where maximum of 10% of the sequences being used to query the sequence knowledge database are used. The 10% is obtained from these experiments (ranging from 50%, 30%, 20%, 10%, 5%) on HOMSTRAD random data sets at which majority of alignment accuracy drops significantly.

The first five fully verified references (Ref1, Ref2, Ref3, Ref4, and Ref5) from the BAliBASE [52] are used. Figure 5.20 shows the average BAliBASE benchmark alignment scores. The KB-MSA tested with full support of Swiss-Prot 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.

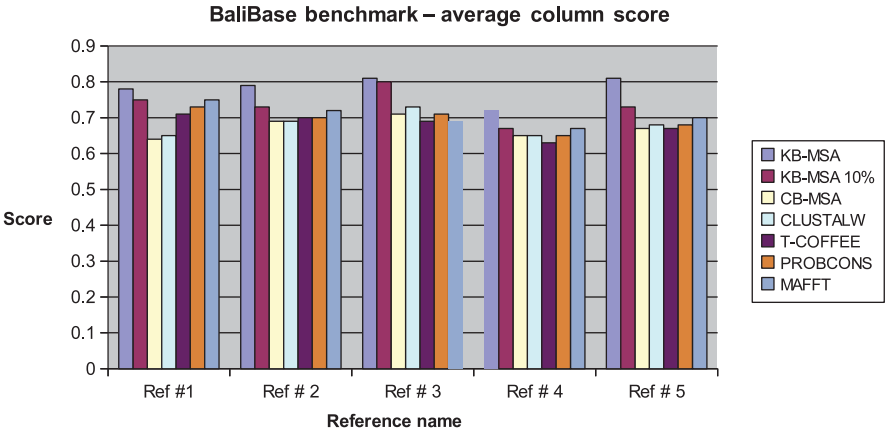


Figure 5.20 KB-MSA tested against ClustalW, T-Coffee, ProbCons, and MAFFT on BAliBASE benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by more than 9%. With 10% sequence knowledge, KB-MSA yields about 4% improvement.

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% 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 Swiss-Prot knowledgebase, the new algorithm (KB-MSA) outperforms other algorithm by a margin between 8% and 15%. With the simulation of 10% 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.

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% identity range. Figures 5.22 and 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 data sets 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 data sets with sequence percent identity greater than 40%, the new algorithm accuracy

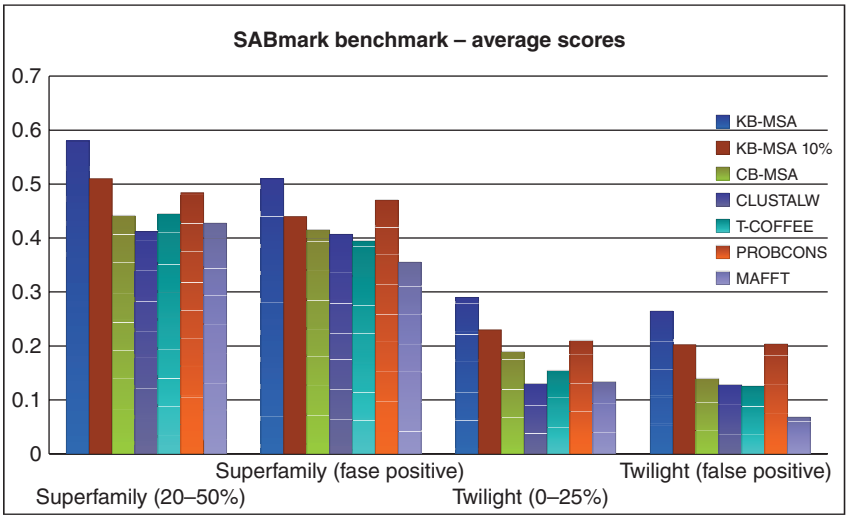


Figure 5.21 KB-MSA testing against ClustalW, T-Coffee, ProbCons, and MAFFT on SABmark benchmark. With complete sequence knowledge, KB-MSA increases average alignment score by at least 8% on the original data sets and more than 6% on false-positive data sets.

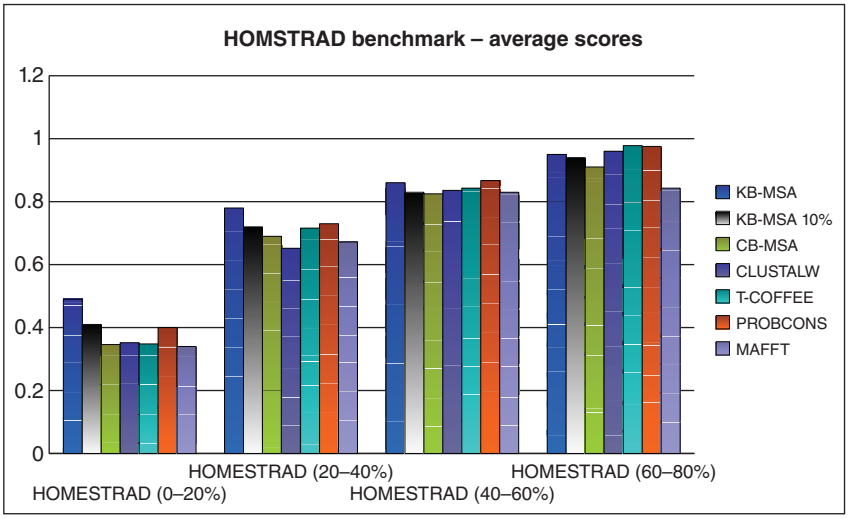


Figure 5.22 KB-MSA testing against ClustalW, T-Coffee, ProbCons, and MAFFT on HOMSTRAD. With complete sequence knowledge, KB-MSA increases average alignment score about 10% on data sets with less than 20% sequence identity.

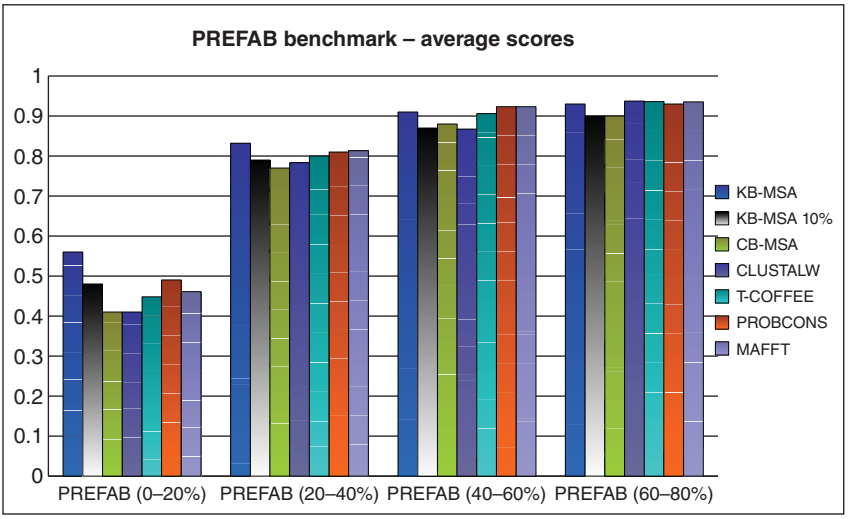


Figure 5.23 KB-MSA testing against ClustalW, T-Coffee, ProbCons, and MAFFT. With complete sequence knowledge, KB-MSA increases average alignment score at least 7% on data sets with less than 20% sequence identity.

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 data sets, especially when the aligning sequences share at most 20% identity. In the PREFAB benchmark tests, it is reasonable to expect that the new algorithm will perform much better on data sets in the twilight zone since scoring is done only on the core-pair of each data set. 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 data sets 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 difference between CB-MSA and T-Coffee is 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]tree (PADT) [107] is a special variation of progressive MSA that has been employed in

T-Coffee [58] and ClustalW [55]. What makes this algorithm distinctively different from others 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 [10]. To combine this information, each residue in the sequences will be represented as a triplet, (r, α, β), containing the residue r, the local alignment score α, and the biological score β. This information is utilized to enhance the global pair-wise alignments, estimating the sequence alignment guiding tree. Moreover, the guiding tree in this algorithm can dynamically reorder its branches depending on the result of alignments on lower branches.

5.5.2.1 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 ith triplet of sequence j represents the residue ith of sequence and two values α and β. 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 [18] to identify the longest common subsequence between any pair of sequences. Residues included in the common subsequence are given scores of α. These α 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 ϵ 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.

5.5.2.2 Scoring Method Traditionally, Needleman–Wunsch's algorithm [19] 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, that is, $H[0,i] = H[i,0] = i \times gap$, where gap is the gap-aligned penalty (a negative value). A pair of residues from row ith and column jth is the maximum of three values:

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

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, α_i, β_i) and (y_j, α_i, β_j), representing residues y_j and x_i, is defined as

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

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 prealigned 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 (β) and local alignment scores (α) are used as weight control parameters. They can be increased or decreased to adjust the weights of these two features.

5.5.2.3 Dynamic Alignment Guiding Tree Similar to traditional method, an alignment guiding tree is built from either UPGMA [60] or neighbor-joining (NJ) [61] method from all-pair-wise sequence distances. However, in this method, the branches of the alignment guiding tree that are less than ϵ 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 distances will change depending on gap insertions and the arrangement of the columns of 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

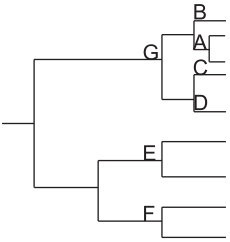


Figure 5.24 This figure illustrates the changeability between branches of an alignment guiding tree. Assume that the distances from A, B, C, and D to G are all less than ϵ, then A, B, C, and D are interchangeable.

is done by performing all pair-wise alignments at the tree level where there are more than two interchangeable branches. The best 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. ϵ 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 ϵ. 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, rearrangement 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 β, or their local alignment scores α, 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 [55], DIAlign [56], T-Coffee [58], MUSCLE [53], and ProbCons [86], 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 data sets from three different reference benchmarks: the Reverse-Transcriptase Order-Specific-Motifs

(RT-OSM) sequences [51] (as seen in Figure 3.13), BAliBASE3.0 [52], and PREFAB (Protein REFerence Alignment Benchmark version 4.0) [53]. Details of these benchmarks are described earlier in Chapter 3.

In the next section, we will describe the experimental results of the new MSA 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 α is set to $\frac{1}{n}$, where n is the number of input sequences, β is set to 1 for residues with biological significance, gap penalty is -10, and the substitution scores are from BLOSUM62 [27].

The new algorithm PADT (Progressive MSA based on Dynamic weighted Tree) is implemented to generate two alignments from the same input sequences: one is constructed by following the phylogeny tree created by UPGMA method [60], and the other is constructed by following the tree created by neighbor-joining (NJ) method [61].

5.7.1 Measuring Alignment Quality

The two popular quantitative methods for measuring such conformity are Total Columns (TC) [52] and percentage of conserved regions, or motifs, correctly aligned. TC score is the most 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 that 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 MSA algorithm utilizing the UPGMA phylogeny estimation method) yield highest alignment scores. The result is shown in Figure 5.25.

5.7.2.1 BAliBASE Results Apart from 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 Figures 5.26 and 5.27. The new algorithm using neighbor-joining method

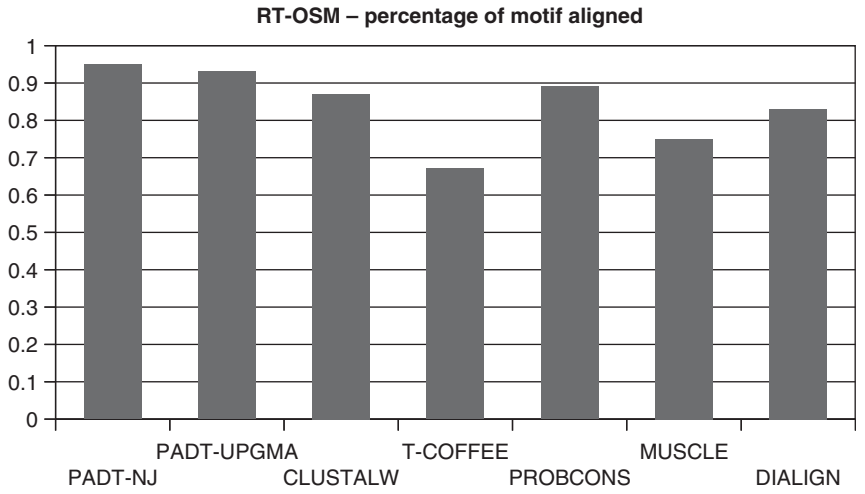


Figure 5.25 Percentage of motifs aligned by PADT-NJ, PADT-UPGMA, ClustalW, T-Coffee, ProbCons, MUSCLE, and DIAlign on RT-OSM sequence set PADT-NJ and PADT-UPGMA surpass ProbCons by 5% and 7%, respectively.

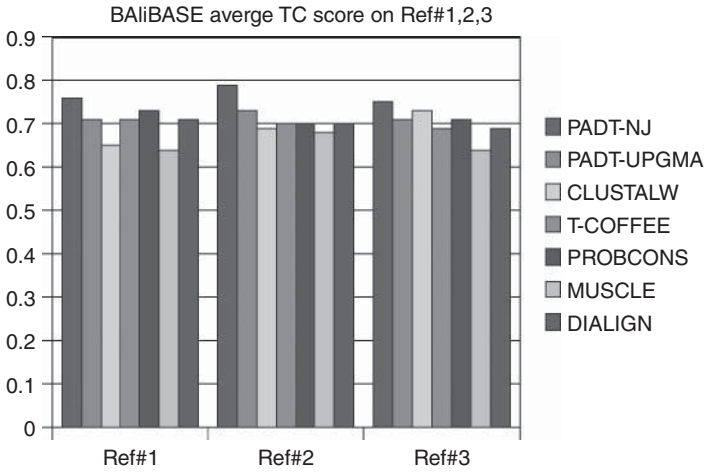


Figure 5.26 Total column (TC) score by PADT-NJ, PADT-UPGMA, ClustalW, T-Coffee, ProbCons, MUSCLE, and DIAlign on BAliBASE Ref#1, Ref#2, and Ref#3. PADT-NJ gains at least 9% improvement on Ref#2.

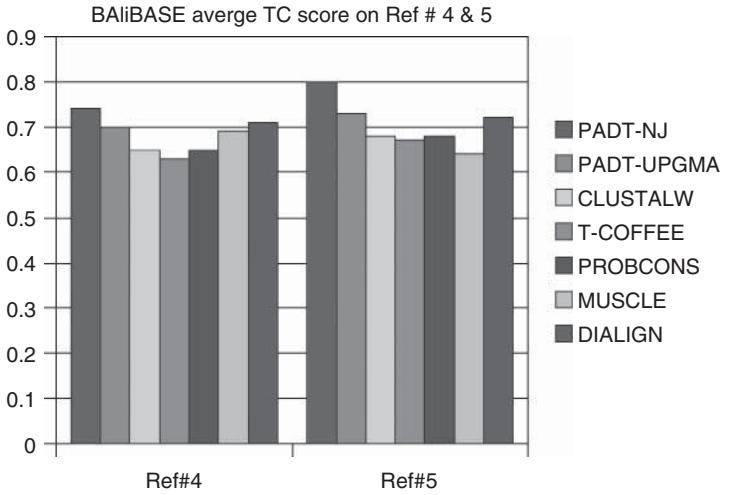


Figure 5.27 Total column (TC) score by PADT-NJ, PADT-UPGMA, ClustalW, T-Coffee, ProbCons, MUSCLE, and DIAlign on BAliBASE Ref#4 and Ref#5. PADT-NJ gains 6% on Ref#5 over DIAlign.

(PADT-NJ) achieves a significant improvement in Ref#2 (about 9%) and is slightly better than other algorithms in two other categories. It seems that the new algorithm performs relatively well on data sets 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 data sets.

5.7.2.2 PREFAB Results Similarly, 60 reference tests on PREFAB benchmark are selected and categorized 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 evaluation techniques done on BAliBASE benchmarks are performed on these data sets. The results are shown in Figure 5.28.

Since PREFAB reference sets are automatically generated, these sequences do not have any biological information nor have any actual reference alignment. Thus, total column score cannot be used. 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

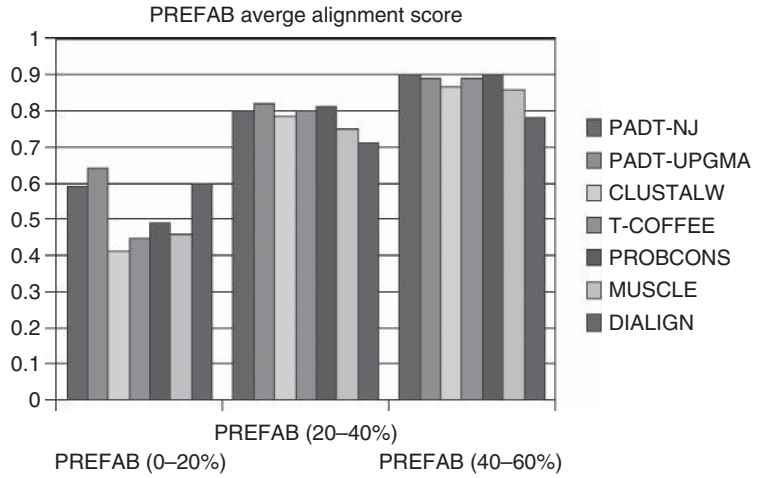


Figure 5.28 Quality score by PADT-NJ, PADT-UPGMA, ClustalW, T-Coffee, ProbCons, MUSCLE, and DIAlign on PREFAB data sets. On the first group, PADT-UPGMA gains 4% improvement over DIAlign and more than 10% improvement over others.

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 data sets with low percent identity, especially, when extra biological information is available. The phylogeny estimation methods and the data sets 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 data sets, UPGMA method is more favorable.

There are many other MSA algorithms that 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 MSA algorithms.

6

PHYLOGENY IN MULTIPLE SEQUENCE ALIGNMENTS

Multiple Sequence Alignment (MSA) is a common tool in phylogenetic analysis, where the evolutionary tree of different organisms are identified and organized in a hierarchical structure in which closely related species are physically placed near each other. This kind of structure is a phylogeny – a conjecture of evolution of taxonomic groups. It is not to confuse that many MSA algorithms also use pairwise alignments to construct a phylogeny as a reference for further alignment refinement. These two types of phylogenetic structures may or may not represent the same information because the first one is built from pairwise alignment and the latter is derived from the final assembled MSA. We should view the phylogeny constructed by calculating the distance of any sequences as a specific phylogeny for these sequence being aligned.

6.1 THE TREE OF LIFE

MSAs are often used to create a phylogenetic tree – tree of life – due to the two distinct features that MSAs possess. First, most functional domains are known and well annotated in sequences; thus, MSAs help identify these functional domains in non-annotated sequences. This way, sequences can be categorized by similar functional domains. The second feature is the capability to find conserved regions that are known to be functionally important. With the flexibility in MSA techniques that

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

allow mutations and indels, MSAs make it possible to analyze and find evolution relations through homology between sequences. Traditionally, biologists use the standard classical model of Linnaean taxonomy, put forward by Carl Linnaeus in his *Systema Naturae* in 1735 [108], in which every organism is assigned a kingdom, phylum, class, order, family, genus, and species. This system classified organisms into even smaller and smaller groups. In this model, organisms are abstractly organized by their context, not by their genetic composition. For example, the classification of plants has 24 classes based on sexual system such as class 1 – Monandria is for flower with 1 stamen, and class 24 – Cryptogamia for flowerless plants. Similarly, animals are categorized into classes such as Mammalia, Aves, Amphibia, Pisces, Insecta, and Vermes. The phylogenetic classification system only named clades – groups of organisms that genetically descended from a common ancestor. This classification system relies on the evolution process by which modern organisms have descended from ancient ancestors. Thus, the modern organisms will carry some genetic variations from their ancestral species. The variations are the result of selective forces that may lead to mutation, migration, genetic drift, fitness, or natural selection. Some common causes are: changes in the environment, supplies of foods, diets, and cross-breeding.

Since the phylogenetic classification relies on the genetic information such as DNA, RNA, protein structures, and so on, we can genetically identify and categorize the species systematically and precisely. Illustratively, reptiles and birds are genetically classified in the same clade as seen in Figure 6.1.

Phylogenetic classification system allows newly identified species be placed into its family, genetically. For example, dinosaurs and birds belong to the same family of dinosauria clade because they have greater DNA similarity than others. With this system, each branch of the tree can be given an estimate time that had taken for a species to evolve into the next species.

In addition, the phylogenetic model can identify the extinct lineage based on the missing functional domains currently seen in its descendants. There are concrete methods that allow biologists to identify what happened and when it happened in the evolution such as radiometric dating, stratigraphy, and molecular clock. For example,

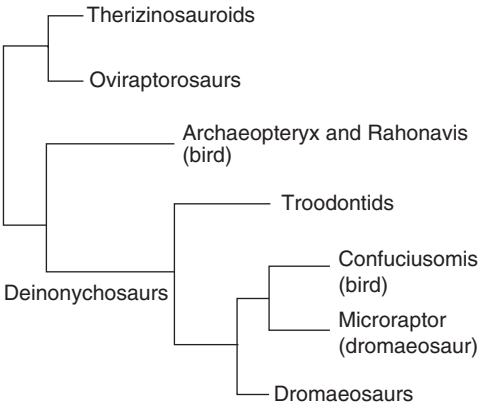


Figure 6.1 A small clade of the phylogeny. (*Source*: Adapted from Mayr et al. 2005 [109].)

fossils can be dated directly by measuring the half-life decay of radioactive elements. Similarly, fossil constraints and rate of molecular change can be used to deduce the time when two species diverged. These techniques, while time consuming and expensive, give a more precise and detailed information about the evolution of an organism. Fortunately, all organisms can be represented as biological sequences. Those that have been identified, analyzed, and annotated allow us to utilize computer tool such as MSA to apply known knowledge and existing annotated sequences to extrapolate, predict, and identify similar functional regions and domain in other sequences.

6.2 PHYLOGENY CONSTRUCTION

The main purpose of a phylogenetic tree is to depict the evolution of the organisms. Thus, correctly reconstructing the evolution and representing it as a phylogenetic tree is a critical task. As seen in Chapter 4, a phylogeny can be constructed from pairwise alignments via identity and similarity scores. A phylogeny is represented as a root-tree or an un-rooted tree. In the rooted tree, the root is the common ancestor of all Operational taxonomic units (OTUs) that are being studied. A path from the root to a node defines an evolutionary path.

In the un-rooted tree, the relationships between OTUs are depicted, but the evolutionary paths are not specified. Both the rooted and un-rooted trees can be transformed from one to the other. For example, the un-rooted tree (b) in Figure 6.2 can be transformed into a rooted tree by making node *Seq*2 as the root. It is a common practice that each branch in the phylogenetic tree has two descendants if it is rooted or three neighbors if it is un-rooted. This hierarchical structure is the result of the rarity of the same and synchronous evolution rate observed in organisms, (otherwise, there is no divergence). Phylogeny reconstruction is well supported by the scientific communities. There are large numbers of available software packages that are designed to create phylogeny trees. Most MSA packages support some types of phylogenetic tree construction. A comprehensive list of phylogeny package is maintained by the Department of Genome Sequences at the Washington University http://evolution.genetics.washington.edu/phylip/software.html. Phylogenetic trees are usually generated by the distance methods or character-based methods

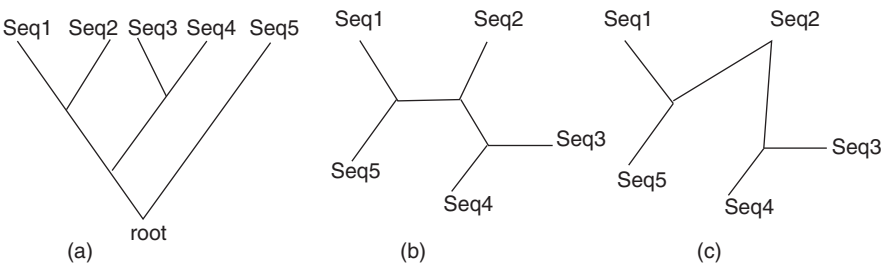


Figure 6.2 (a) Rooted phylogeny, (b) un-rooted phylogeny, and (c) transformation of (b) into a rooted phylogeny.

6.2.1 Distance Methods

In this model, an $n \times n$ matrix M, where $M_{ij} \geq 0$ and M_{ij} is the distance between OTU i and OTU j, is given. An edge-weighted tree is built where each leaf corresponds to an OTU, and the distance between the ith leaf and the jth leaf is the value of M_{ij}. For example, Table 6.1 represents a distance matrix between OTUs.

With this construction, the distances in M are additive, where the sum of all edge weights between any two OTUs are exactly the distance between that two OTUs as depicted in matrix M. The tree in Figure 6.3 is the additive tree representing the matrix M. For all additive distance matrices, there is $O(n^2)$ algorithm for finding the phylogenetic tree. In reality, the distance matrix is rarely additive; thus, finding a tree that best fits the data is often the case. There are two popular methods for this purpose:

1. Cavalli-Sforza and Edwards's method finds a distance d_{ij} that minimizes

$$\sum_{i,j} (M_{ij} - d_{ij})^2. \tag{6.1}$$

2. Fitch and Margoliash's method finds a distance d_{ij} such that

$$\sum_{i,j} \frac{(M_{ij} - d_{ij})^2}{M_{ij}^2}. \tag{6.2}$$

Both of these models lead to computationally intractable problems. One simple illustration is to build a tree from 3 OTUs, there is only one way to build a 3-node tree. Adding another OTU to the 3-node tree, we will have three more trees to consider,

TABLE 6.1 A Distance Matrix

	A	B	C	D	E
A	0				
B	9	0			
C	9	8	0		
D	8	7	3	0	
E	6	11	11	10	

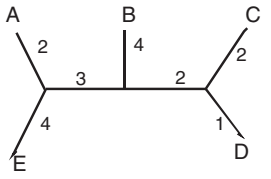


Figure 6.3 A tree generated from the additive distance matrix of Table 6.1.

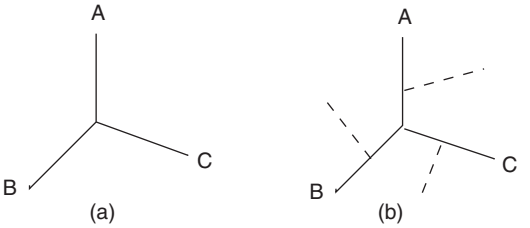


Figure 6.4 Adding a new node to a tree – the dashed lines represent all possible placements of a new edge.

each generated by adding the 4th node to one of the existing edge of the 3-node tree, illustrated as dashed edges. Figure 6.4. And with n OTUs, there will be $\prod_{i=3}^{n}(2i-5)$ different trees to consider. Thus, finding a tree that best fits the given distance M is not computationally expensive. Therefore, heuristic tree construction such as UPGMA or neighbor-joining methods, as described in Chapter 4, are often used.

6.2.2 Character-Based Methods

In this model, each OTU is represented as sequence of characters, in which each character represents a distinct feature (such have wings or not having wings), or biochemical structure of the OTU (such as amino acids in protein, or nucleotides in DNA). These character-based methods assume that each different character is independent of the other. There are two popular methods: maximum parsimony method and maximum likelihood method.

6.2.2.1 Maximum Parsimony The Maximum Parsimony (MP) method uses an optimal criterion under which the preferred phylogenetic tree is the one that minimizes the total number of character-state changes. For example, Figure 6.5 shows a rooted phylogeny for the DNA sequences listed in Table 6.2, where *Seq*1 and *Seq*2 differ by the mutations between character A and C, and *Seq*4 and *Seq*5 differ by a mutation between C and T. Finally, *Seq*3 diverges from the ancestor of the others by the mutation between T and G.

TABLE 6.2 An Example Set of Five Sequences Each Containing Five Residue Symbols

	1	2	3	4	5
Seq1	A	A	C	C	G
Seq2	C	A	A	C	G
Seq3	C	C	T	A	T
Seq4	C	T	G	G	G
Seq5	T	T	G	G	G

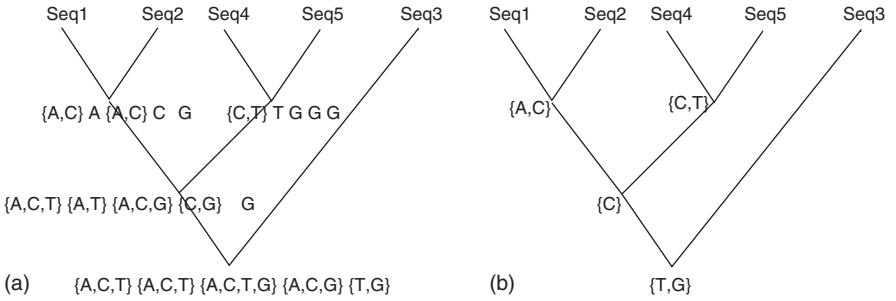


Figure 6.5 (a) A rooted phylogeny created for sequences in Table 6.2 with the mutations are represented in the internal nodes. (b) A Fitch's representation of the tree in (a).

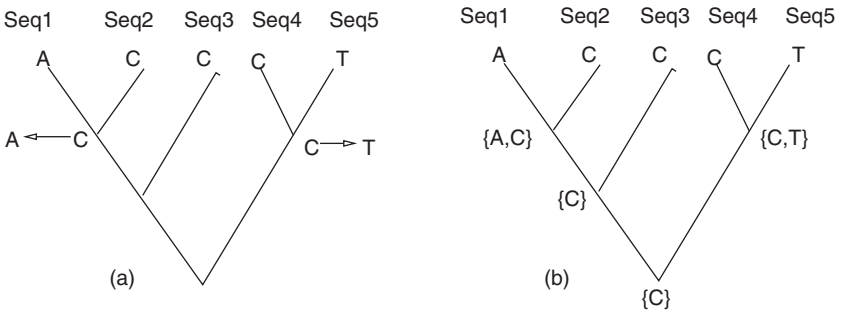


Figure 6.6 Illustration of character mutations. Figure (b) represents the construction of internal nodes with Fitch's algorithm.

The mutations at each character site of the OTUs can also be presented as a tree. This tree can be constructed inward from the external nodes (children) to the root using Fitch's algorithm. For each position of the sequence (column), a leaf node is labeled by the character symbol occurred at that position, in this example, it is the nucleotide at that position. For each internal node, its label is based on the mutation of its child node. For example, if the child nodes carry the same symbol, then the parent node has no label; if the child nodes carry distinct symbols, then the parent node will carry all the symbols exist at the child level; finally, if the child nodes carry some overleaped symbols, then the parent node will carry only the overlapped symbols. This technique is illustrated in Figure 6.6, in which a parsimony tree is built for the first position (column) of the sequences. Closely analyze the tree (b) of Figure 6.6, it is intuitive that the ancestor character is more probable to be C, which gets diverged after two generation (after the first generation, it remains the same as seen in *Seq*3). To find the maximum parsimony tree, all the tree search space must be examines, which is an infeasible problem. Thus, a heuristic technique such as branch and bound or nearest-neighbor interchange technique is used.

In the branch and bound, the initial tree is first built by the neighbor-joining technique or similar techniques as described in Chapter 4. Then a new tree is built from

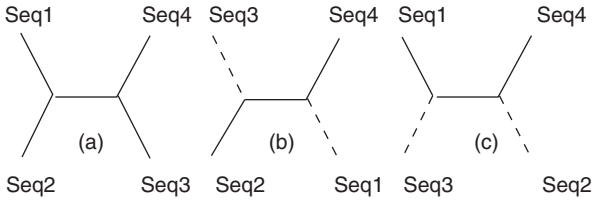


Figure 6.7 Illustration of the nearest-neighbor interchange technique. (a) the original tree with four OTUs in which *Seq*1 is closest neighbor of *Seq*2 and *Seq*3 is closest neighbor of *Seq*4; (b) the first interchange where the nearest neighbor of *Seq*2 (which is *Seq*1) is interchanged with the nearest neighbor of *Seq*4 (which is *Seq*3), and (c) the second interchange of the neighbors.

any OTU, level by level. A kth level tree represents a partial tree with k OTUs. If a partial tree has more mutations (cost) than the initial tree, the search along that path is abandoned. There is no guarantee on the amount of search space the branch and bound method can eliminate; thus, for a phylogeny tree with a large number of OTUs, a stopping criteria such as a timer/counter is set to stop the process and return the best found phylogeny tree.

The nearest-neighbor interchange method is a heuristic procedure that exchanges neighboring OTUs starting with the nearest neighboring one. For example, in Figure 6.7, the tree has four subtrees. The nearest-neighbor interchange method will exchange these subtrees to obtain a new tree. With four subtrees, there are only two exchanges that lead to two new trees. For a larger tree, this procedure is repeated on each internal branch until no further better improvement can be found. A binary un-rooted tree with n OTUs, $n > 2$, will have $n - 3$ internal edges, thus requiring $2(n - 3)$ exchanges. The character-based methods for constructing phylogenetic trees are rather simple and they do not consider the weight of the edges in the tree construction; thus, it is very probable that the resulting phylogeny is incorrect. This is one of the reasons that lead to development of many scoring schemes as seen in Chapter 3.

6.2.3 Maximum Likelihood Methods

The Maximum Likelihood (ML) methods rely on the probabilistic model for residue substitution (such as in Jukes and Cantor's model) to find the branch lengths and the topology of the tree that give the highest probability of observing the sequence data in a given model. The residue could be any character-based feature representation, or amino acids or nucleotides. In these methods, if we call the set of characters being considered at a position in the sequence is D, we need to find a tree T that maximizes the probability P in expression $P(D|T, M)$, where M is the evolutionary model under consideration (substitution matrix such as PAM or BLOSUM matrix is often used). And

$$P(D|T, M) = \prod_i P(D_i|T, M). \tag{6.3}$$

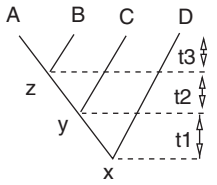


Figure 6.8 Illustration of calculations of maximum likelihood of a tree.

The theoretical model of maximum likelihood is complicated and involved a large set of parameters. In practice, the probability of characters in D is known by observing the frequency of the characters in the sequences and the ancestral sequences are most likely unknown. In addition, different characters in the considered OTU/sequence data are assumed to evolve independently, and these OTUs/sequences are also assumed to evolve independently after they have diverged. Thus, the method is often simplified to the summation of ancestor mutations in a tree. Figure 6.8 shows an example tree for maximum likelihood, where we also assume the mutation among the branch is constant.

Thus, the maximum likelihood of the tree, expressed in Equation 6.3, can be expressed as

$$P((A,C,T,G)|T,M) = \sum_x \sum_y \sum_z p_x(0).p_{xG}(t_1+t_2+t_3).p_{xy}(t_1).$$
$$p_{yT}(t_2+t_3).p_{yz}(t_2).p_{zC}(T_3).p_{zA}(t_3) \tag{6.4}$$

where t_i is the evolution time at level ith of the tree, and p_{ij} is the probability of a character symbol i being mutated to symbol j. Note that the symbol representing an ancestor can be a set of characters. If the branch lengths are known, dynamic programming can be deployed to compute the maximum likelihood. To reconstruct the phylogenetic tree, the optimal branch length for each tree topology must be calculated. Similar to the distance methods, there is a large amount of tree topologies to measure. The normal practice is to use an iterative estimating the local maximum such as expectation maximization (EM) algorithm to reduce the search space.

6.2.4 Bootstrapping

The large search space of finding the optimal phylogenetic tree for a set of OTUs is computation prohibited; thus, bootstrapping is often employed. Bootstrapping is a resampling method to estimate the properties of an estimator by measuring those properties when sampling from an approximation distribution. In practice, when a phylogeny tree is constructed, regardless of method used or whether the tree is complete or partially complete, a bootstrapping is employed to estimate the confidence of the arrangement of the OTUs in that tree (or subtree). For example, if we have four sequences $Seq1, Seq2, Seq3, Seq4$ representing four different OTUs, after performing

TABLE 6.3 Illustration of Bootstrapping Sampling and Replacement

	1	2	3	4	5			**1**	2	**3**	4	5
Seq1	A	A	C	C	G		Seq1	**C**	A	**C**	C	G
Seq2	C	A	A	C	G	⇒	Seq2	**A**	A	**A**	C	G
Seq3	T	C	T	A	T		Seq3	**T**	C	**T**	A	T
Seq4	G	T	G	G	G		Seq4	**G**	T	**G**	G	G

Note: The first column is randomly replaced by another column of characters.

some sequence alignments, we determine that sequences *Seq*1 and *Seq*2 should be grouped together, and sequences *Seq*3 and *Seq*4 should be grouped together. To gain confidence that *Seq*1 belongs to *Seq*2 and *Seq*3 belongs to *Seq*4, we align all columns of these four sequences *Seq*1, *Seq*2, *Seq*3, and *Seq*4 together, make a copy of the alignment, and replace the first column of characters of the copied alignment by a random column obtained from the original alignment. Thus, we have a new set of sequences, in which the first column differing from the original sequences. We repeat this sampling and replacement process by copying the original alignment and replacing the second column of the copied alignment by a random column obtained from the original alignment. This process is repeated for k times, where k is the number of characters in the original sequences. Thus, each newly generated sample omits a different column from the original alignment, and we should have k new samples. For each of these generated samples, we create a tree and keep track of how many times *Seq*1 and *Seq*2 belong to the same group and *Seq*3 and *Seq*4 stay together. The confidence is then calculated as the fraction of the each pair appear together over the total tree generated. Table 6.3 illustrates the process of generating the first bootstrapping sample. The left sequences are the original sequences and the sequences on the right are the sample, in which the first column being replaced by the third column – the random selected column.

All examples illustrated in this chapter are assumed to have the same length, which are not common in practice. To handle sequences with different lengths, multiple sequence alignment must be employed to induce indels into the sequences before any phylogenetic tree construction.

6.2.5 Subtree Pruning and Re-grafting

Subtree pruning and re-grafting is another heuristic tree topology search, in which a new tree is generated by disconnecting a subtree and reattaching it into a different branch. Each time a new tree is generated, the tree will be evaluated for possible improvement. With a tree with n leaf OTUs, there are $O(n)$ possible reattaching points and there are $O(n)$ possible subtrees for each iteration. A popular use of this technique is to maintain the hierarchy structure of subtrees that are closely related as units, and the pruning and re-grafting are at the level of these new units rather than at random subtrees.

6.3 INFERRING PHYLOGENY FROM MULTIPLE SEQUENCE ALIGNMENTS

Most MSA algorithms rely on the phylogeny tree constructed by pairwise alignment to guide the assembling of the alignment. However, the phylogeny is also benefited from the result alignment.

Majority of alignment algorithms employ certain levels of refinement to improve the overall alignment score. These refinement steps may change the order of the sequences in the pairwise alignment generated guiding tree. Moreover, most biologists align multiple sequences to systematically identify some common regions in the sequences that are tedious and time consuming for human to do manually. When reviewing the multiple alignment result where these regions are lined up, the experts often intervene to analyze, correct, and annotate these regions, thus changing the weight of each OTU. Additionally, existing large sequence knowledge databases such as GenBank, European Nucleotide Archive, DNA Data Bank of Japan, and secondary databases such as UniProt, Swiss-Prot, and TrEMBL allow sequences to be cross-referenced easily to identify their family classification and possibly primary and secondary structures.

When a new sequence of a species is assembled and ready to add into the sequence databases, BLAST is often employed to search across existing sequence databases to identify the set of sequences that share the most similarity with the new sequence. These matching sequences are aligned with the new sequence to identify similar functional regions. Information from secondary structural databases can be used to annotate the sequence further, and expert will verify the classification of the sequence before having it indexed within the appropriate family in the database. With the vast available sequence knowledge bases, MSA and phylogeny reconstruction are often more biologically precise with algorithms that utilize these knowledge bases in their process.

7

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.

7.1 PARALLEL SYSTEMS

There are many types of parallel computing systems that have been deployed to solve computationally intensive problems such as sequence alignment and classification. Following are the most models that are described.

7.1.1 Multiprocessor

Multiprocessor systems consist of multiple processing nodes connected through a network. Each processing node contains one or more processors or multicore

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

chips that share a limited onboard 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 on the network connections. These systems are classified as Multiple-Instruction, Multiple-Data (MIMD) systems.

7.1.2 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.

7.1.3 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 coprocessors.

7.1.4 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. FGPAs act as coprocessors and are utilized as hardware accelerators. FPGA processing elements are locally interconnected, thus providing a very low overhead communication environment.

7.1.5 Reconfigurable Mesh

Reconfigurable Mesh (R-Mesh) computing system is a k-dimension grid of processing elements. Communication between the processing elements is primarily 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.

7.2 EXITING PARALLEL MULTIPLE SEQUENCE ALIGNMENT

In general, independent operations can be performed simultaneously on different computing elements to speed-up 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 pairwise alignments of n input sequences.
(ii) Compute a dendrogram indicating the order in which the sequences are aligned.
(iii) Pairwise align two sequences (or two prealigned groups of sequences) following the dendrogram starting from the leaves to the root of the dendrogram.

A quick analysis of these three steps reveals the following:

- Step (i) has $O(n^4)$ run-time complexity for performing $\frac{n(n-1)}{2}$ pairwise 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 $(n-1)$ pairwise alignments of prealigned groups of sequences in the worse case. Each of these pairwise 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 pairwise alignments as in step (i). These individual pairwise alignments can be designated to different processing units for computation as in [110–119].

These parallel progressive multiple sequence alignment implementations are across many computing architectures and platforms. For example, Lima et al. [112] implemented a DP algorithm on FPGA. Similarly, Oliver et al. [118, 119] distributed the pairwise alignment of the first step in the progressive alignment, where all pairwise alignments are computed, on FPGA. Liu et al. [113] computed DP via GPUs using CUDA platform [117], used concurrent read concurrent write parallel random-access machine (CRCW PRAM) neural networks [110], used Clusters [111], used 2D R-Mesh [115], used Network Mesh, or [116] used 2D pipelined R-Mesh (PR-Mesh) computing model.

The two most notable parallel versions of dynamic programming algorithm are proposed by Huang [120], Huang and Biswas [110], and Aluru et al. [121]. Huang's algorithm exploits the independency between the cells on the antidiagonals of the DP matrix, where they can be calculated simultaneously. There are $m+n$ antidiagonals 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, Huang and Biswas [110] and Aluru et al. [121] 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 in 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 ith. 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 $O(nlog\ n)$ for $n \geq m$, time using $O(n)$ processors.

Recently, Sarkar et al. [114] implement both of these parallel DP algorithms [120, 121] on a network-on-chip computing platform [122].

In addition, the construction of a dendrogram can be parallelized as in [113] using n GPUs and completing in $O(n^3)$ time.

Furthermore, there are attempts to parallelize the progressive alignment step, step (iii) as in [123] and [124]. In [123], the independent prealigned 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 pairwise alignments eventually must be merged together. Another attempt is seen in [124] where Huang's algorithm [120] is used. When an antidiagonal of a DP alignment matrix in lower tree level in step (iii) is completed, it is distributed immediately to other processors for computing the pairwise alignment of a higher tree level. This technique can lead to an incorrect result since the actual pairwise alignment of the lower branch is still uncertain.

Overall, the major speed-up achieved from these implementations comes from two parallel tasks: performing $\frac{n(n-1)}{2}$ pairwise alignments simultaneously and calculating the dynamic programming matrix antidiagonally (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 pairwise group alignments are done sequentially following the guiding tree, and each alignment requires all the column pairwise scores be calculated.

The remaining of this chapter is devoted to 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.

7.3 RECONFIGURABLE-MESH COMPUTING MODELS – (R-MESH)

A Reconfigurable Mesh (R-Mesh) computing is a 2D grid of processing units (PUs), in which each processing unit contains four 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

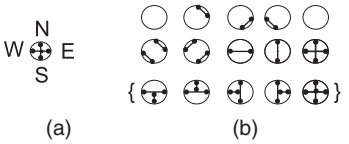


Figure 7.1 Allowable configurations on four port processing units; (a) shows the port's directions; (b) shows the 15 possible port connections, where the last five port configurations in curly braces are not allowed in linear R-Mesh (LR-Mesh) models.

the grid to its neighboring nodes. These configurations are shown in Figure 7.1. Each processing unit has its own local memory, can perform simple arithmetic operations, and can configure its ports in $O(1)$ time.

There are many reconfigurable computing models such as linear R-Mesh (LR-Mesh), processor array with reconfigurable bus system (PARBS), PR-Mesh, and FPGA. 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 [125–130].

In the scope of this book, 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 [131]. Commonly, data in one format can be converted to another in $O(1)$ time [131]. 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 follows:

Definition 3 *Given an integer $x \in [0, n-1]$, the unary 1UN presentation of x in n-bit is $x = (b_0 b_1, \cdots, b_{n-1})$, where $b_i = 1$ for all $i \leq x$ and $b_i = 0$ for all $i > x$ [131].*

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 [131].*

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.

7.4 PAIRWISE DYNAMIC PROGRAMMING ALGORITHMS

This section begins with the description of several configurations of R-Mesh needed to compute various operations in pairwise 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.

7.4.1 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 7.2a using one processing unit (adapted from [132]). 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 four processing units with configurations: {NSW,E}, {NSE,W}, {NE,S,W}, and {NSW,E} as in Figure 7.2b. To simulate a three-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 7.3. This n-bit max switch takes $4 \times n$ (i.e., $O(n)$) processing units and can handle three to four 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 the algorithm, described later.

7.4.2 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

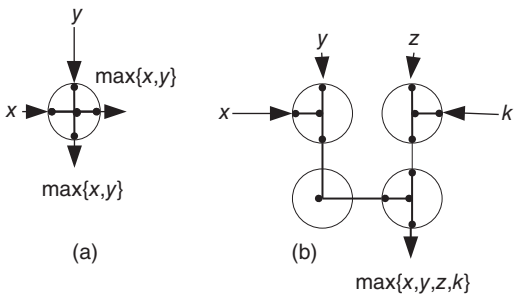


Figure 7.2 Two 1-bit max switches. (a) Fusing {NSEW} to find the max of two inputs from North and West ports (*Source*: Adapted from Bertossi and Mei 2000 [132]); (b) construction of a 1-bit 4-input max switch.

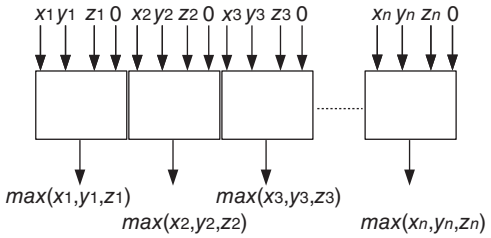


Figure 7.3 An n-bit three-input max switch, where the rectangle represents a 1-bit four-input max switch from Figure 7.2.

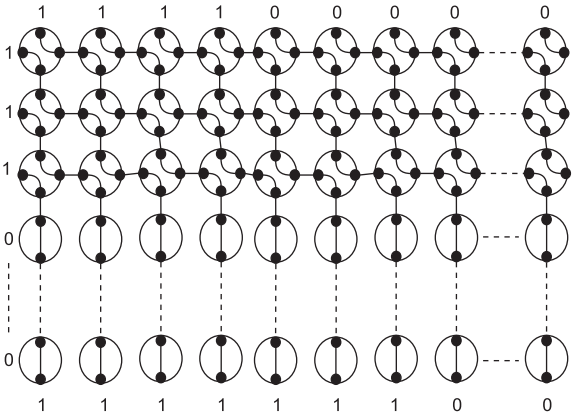


Figure 7.4 An n-bit adder/subtractor that can perform addition or subtraction between two 1UN numbers during a broadcasting time. For additions, the inputs are on the North and West borders and the output is on the South border. For subtractions, the inputs are on the West and South borders and the output is on the North border. The number on the Westbound 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 Westbound (left) has been shifted off. The right border is fed with zero (or no signal) during the subtract operation.

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 7.4.

To perform addition, one addend is fed into the Northbound of the R-Mesh, and another addend is left-shifted one bit and fed into the Westbound. 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 onto the output ports. Figure 7.4 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 Southbound (bottom) of the R-Mesh, the subtrahend is 1-bit left-shifted and fed into the R-Mesh from the Westbound (left), the Eastbound (right) is fed with zeros, or no signals. The output is obtained from the North border (top).

This adder/subtractor can only handle numbers in 1UN representation, that is, 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 [25], BLOSUM [27], 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 a 2-digit constant or smaller. Therefore, the size of the adder/subtractor unit is bounded by $O(n)$ in this scenario.

7.4.3 Constant-Time Dynamic Programming on R-Mesh

The dynamic programming techniques used in the 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.

7.4.3.1 Smith–Waterman's and Needle–Wunsch's Algorithms Although the number representation can be converted from one format to another in constant time [131], 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 (4×4 for RNA/DNA sequences and 20×20 for protein sequences), are prescaled 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 [25, 27], and so on. 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 [25] is calculated as $s(i, j) = \frac{1}{\lambda} \log\left(\frac{Q_{ij}}{P_i P_j}\right)$, where $s(i, j)$ is the substitution score between residues i and j, λ is a positive scaling factor, Q_{ij} is the frequency or the percentage of residue i corresponding to residue j in an accurate alignment, and P_i and P_j are background probabilities in 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 website[1] 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 unscaled scoring matrices by solving the log-odd function. Thus, when a positive number β is added to the scores in these scoring matrices, it is equivalent to multiply the original probabilities by a^{β}, where a is the log-base that is 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 β 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 subalignment scores to zeros when they become negative, the scores are set to β if they fall below this scaled-up factor (β).

We will use the scaled-up scoring matrices to eliminate the representation of signed numbers in the 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/subtrator 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 three adder/subtractor units and a three-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 7.5 shows the dynamic programming R-Mesh construction. The adder/subtrator units are represented as "+" or "−" corresponding to their functions.

[1] http://www.ncbi.nlm.nih.gov.

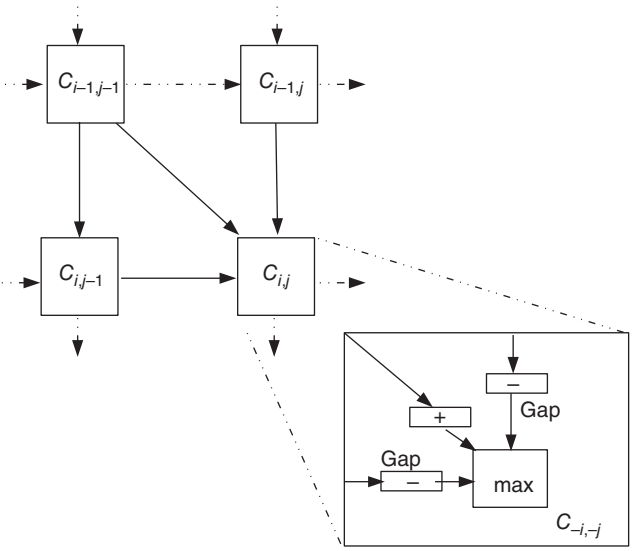


Figure 7.5 A dynamic programming R-Mesh. Each cell $c_{i,j}$ is a combination of a three-input max switch and three adder/subtractor units. The "+" and "−" represent the actual functions of the adder/subtractor units in the configuration.

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 pairwise 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 terms 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/subtractor 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 predefined pairwise symbol-matching scores may have to be scaled up (or down) so that

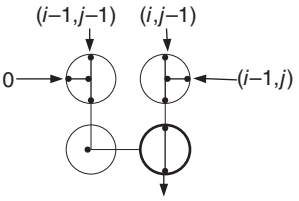


Figure 7.6 A configuration of a four-way max switch to solve the longest common subsequence (LCS). The South-East processing unit (in bold) configures NS, E, W if the symbols at row i and column j are different; otherwise, it configures N, E, SW. This figure shows a configuration when the two symbols are different.

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.

7.4.3.2 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 7.5 can be used, where the two subtractor units are removed or the gap cost is set to zero ($g = 0$).

To find the LCS between two sequences x and y, each max switch in the DP R-Mesh is configured as in Figure 7.6. 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.

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 $O(n^2) \times O(n^2)$ [132] to solve the same problem.

7.4.4 Affine Gap Cost

Affine gap cost (or penalty) is a technique where the opening gap has different cost from an extending gap [22]. This technique discourages multiple and disjoined gap insertion blocks unless their inclusion greatly improves the pairwise 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 rightmost 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 7.7 shows the modification of the "−" unit. The output from the original "−"

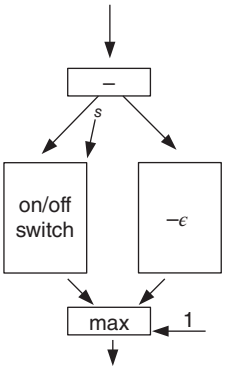


Figure 7.7 A configuration to select one of the two inputs in 1UN notation using the rightmost bit as a selector s. When $s = 1$, the switch is turned on to allow the data to flow through and get selected by the max switch. When the selector $s = 0$, the on/off switch produces zeros and the other data flow will be selected. $\epsilon = o - g, o \geq g$, is the difference between opening gap cost o and extending gap cost g.

unit is piped into an $n \times n + 1$ R-Mesh on/off switch (described in Section 7.4.5), 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 rightmost 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. After a gap cost is applied, all non-diagonal cells will have their selector bits, (the right most bit), set to 1 to prevent further opening gap penalty.

The modification of the dynamic programming R-Mesh to handle affine gap cost requires additional two adder/subtractor units, two 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 $O(1)$.

7.4.5 R-Mesh On/Off Switches

To handle affine gap cost in dynamic programming, we need a switch that can select, that is, turn on or off, the output ports of a data flow. The on/off R-Mesh switch can be configured as in Figure 7.8, where the input value is mapped onto the Northbound (top). The rightmost 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

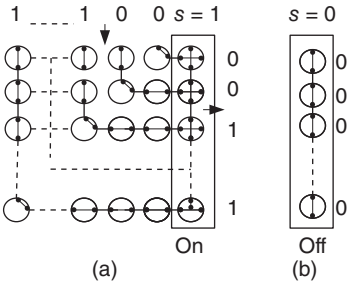


Figure 7.8 An $n \times n + 1$ R-Mesh represents an n-bit on/off switches. By default, all processing units on the last column (column $n + 1$) configure {NS,E,W} and fuse {NSEW} when a signal (i.e., 1) travels through them. All cells on the main antidiagonal cells of the first n rows and columns fuse {NE,S,W}, cells above the main antidiagonal fuse {NS,E,W}, and cells below the main antidiagonal 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).

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 $(n \times n + 1)$, that is, $O(n^2)$, processing units to turn off the flow of an n-bit input in a broadcasting time.

7.4.6 Dynamic Programming Backtracking on R-Mesh

The pairwise 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 3D $n \times n \times n$ R-Mesh through these steps:

1. Initially, the DP matrix with calculated values are stored in the first slice of the R-Mesh cube, that is, 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 $c_{0,j,k}$, that is, 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 $c_{k,0,0}$, that is, 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 $f(0, j), f(1, j), \cdots, f(n-1, j)$ are 0. This step can be performed in $O(1)$ time by O Ring 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 then 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 North, 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 pairwise alignment is the ordered list of cells where this signal travels through.

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

7.5 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 parallel progressive multiple sequence alignment algorithm on R-Mesh along with its complexity analysis.

7.5.1 Hierarchical Clustering on R-Mesh

The parallel neighbor-joining (NJ) [61] clustering method on R-Mesh is described here; other hierarchical clustering mechanisms can be done in a similar manner. 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 pairwise distances are given in the 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 pair 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 $d_{j,u} = d_{ij} - d_{iu}$.
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 $D_{vu} = D_{iv} + d_{jv} - D_{ij}$.

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.

Steps 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 are mostly 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 pairwise DP sequence alignments are bounded by b^2, where b is the max pairwise 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 is 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 that 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 7.4.6 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 pairwise distance matrix of size $n \times n$ is $O(n)$, using $O(n^3)$ processing units.

7.5.2 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 pairwise dynamic programming alignment on two prealigned 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 is no longer a simple constant lookup, unless the hierarchical expected probability (HEP) matching scoring scheme is used [49]. Most multiple sequence alignment algorithms use the popular sum-of-pair (SP) scoring method [35]. This quantification is the sum of all pairwise 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, that is, $\frac{n(n-1)}{2}$ lookups for each DP matrix cell calculation. The sum-of-pair is formally defined as

$$sp(f, g) = \sum_{i=1}^{|f|} \sum_{j=i+1}^{|g|} s(f_i, g_j), \tag{7.1}$$

where f is a column from one prealigned group of sequences and g is a column from the other prealigned 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:

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

and

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

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 and residue b is the same as the matching between residue b and 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

$$sp(f, g) = \sum_{i=1}^{T} \frac{n_i(n_i - 1)}{2} s(i, i) + n_i \sum_{j=i+1}^{T} n_j \times s(i, j), \tag{7.2}$$

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 pairwise 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 7.2 remain constant, regardless how many sequences are involved.

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

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

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 1 to a processing unit p_k, where k is the number representing 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 pairwise column arrangements of two prealigned groups of sequences of length n. Thus, the sum-of-pair column pairwise matching scores for two prealigned groups of sequences can be done in $O(1)$ using n^3 processing units.

7.5.3 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 pairwise 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 Needleman–Wunsch's algorithms (step i). The order in which the sequences should be aligned is represented as a guiding tree and can be calculated via hierarchical clustering algorithms similar to the one described in

TABLE 7.1 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 backtracking	$n \times n$	$n \times n \times n$	$O(1)$
Neighbor-joining	$n \times n$	$O(n^3)$	$O(n)$
Sum-of-pair	2 prealigned 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 $= n$	$O(n^5)$	$O(n)$

Section 7.5.1 (step ii). After the guiding tree is completed, the input sequences can be pairwise 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 pairwise aligned using the dynamic programming R-Mesh described previously in Section 7.4. These $\frac{n(n-1)}{2}$ pairwise 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 pairwise DP alignment scores from step (i). Finally, [step (iii)], for each pair of prealigned groups of sequences along the dendrogram, the sum-of-pair column matching scores must be precalculated to initialize the DP R-Mesh before proceeding with the dynamic programming alignment. There are $n - 1$ branches in the dendrogram leading to $n - 1$ pairwise 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 pairwise 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 7.1 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 [19], Smith–Waterman's [18], and LCS algorithms [133]

8

SEQUENCE ANALYSIS SERVICES

Multiple sequence alignment (MSA) is a common operation in biological sequence analysis; however, MSA is an intensive and expensive process that requires tremendously amount of computing power, time, and resources [68]. Thus, majority of sequence alignments are done locally on dedicated computing server setup at each individual's lab. There are inherent issues with this model such as costs and efforts for setting up and maintaining the sequence alignment servers, service utilization, and scalability. Alternatively, the MSA service can be rendered via a shared resource. This section will give a survey of existing major continental MSA web-based services.

8.1 EMBL-EBI: EUROPEAN BIOINFORMATICS INSTITUTE

The European Bioinformatic Institute [134] hosts two large and popular databases: the European Nucleotide Archive and the protein sequence UniProt. With these two homologous databases with hundreds of thousands of users and collaborators worldwide, the EMBL-EBI has to provide a set of tools to support its collaborators and users in accessing and utilizing the databases. With its large-scale databases and concurrent servers, EMBL-EBI provides the following MSA services.

Clustal alignment family includes ClustalW2, ClustalOmega [135], and DBClustal [136]. Clustal alignment is a tree-based progressive MSA, in which a

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

guide tree (an estimated phylogeny) is built from estimated distance between the sequences; and the sequences are pairwise aligned following the order specified by the tree from the leaves inward to the root. ClutalW2 is a newer release of the popular ClustalW [55] program, written by Thompson et al. that is suitable for medium alignments. DBClustal is a utility tool that takes a protein basic local alignment search tool (BLAST) search result and aligns the sequences in the result by ClustalW2. ClustalOmega is the latest addition to the Clustal alignment family and is one of the two existing alignment tools (the other is MAFFT-PartTree [137]) with the capability to align more than 10,000 sequences. In order to align such a large number of sequences, ClustalOmega utilizes a modified version of mBed algorithm developed by Blackshields et al. [138] to estimate the guiding tree for its sequence alignment. The tree is constructed by embedding each sequence in an n-dimensional space and then clustering them using standard clustering technique such as K-means [139] or UPGMA [60].

Kalign is a fast sequence alignment that employs a string matching algorithm, developed by Wu-Manber [75], to estimate the guiding tree using Levenshtein's edit distance.

MAFFT [57] is a time-efficient progressive multiple sequences based on fast Fourier transform (FFT) method. The general operational scheme of FFT is to find peaks in data and 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, FFT greatly reduces the solution space and speeds up the process. MAFFT method when combined with PartTree [137] estimation can align more than 10,000 sequences at a time.

MUSCLE [53] is a progressive alignment method. The distinction between MUSCLE and other alignment methods is the technique it uses to estimate the distance between the sequences. MUSCLE utilizes a sliding window to scan across all the sequences to identify the most abundant residues and uses this statistic to calculate the log-expectation score between the matches of the sequences.

T-Coffee [58] is a progressive multiple sequence alignment. It starts by using ClustalW [55] and LALIGN [87] (LALIGN is a fast version of Smith and Waterman's algorithm [18] – LALIGN has two versions: NAlign and PAlign, one for protein sequences and one for nucleotide sequences) to generate a primary pairwise sequence alignment library that combines both global pairwise alignment (ClustalW) and local pairwise alignment (LALIGN). Duplicate pairs are removed and the remaining pairs, of the removed duplicates, get double weights. An extended library is built based on triplets of the primary library where a new pairwise alignment is created if two sequences are aligned via the third sequence. A guiding tree is created from the library, and the sequences are aligned and refined similar to other progressive alignment methods.

T-Coffee has many extended versions such as 3DCOFFEE/Expresso [88] (utilizing 3D structure in alignments), M-Coffee (a package in T-Coffee that merges and refines alignment results of other alignment tools), R-Coffee [140] (for RNA sequences), and so on. There are many T-Coffee web servers currently available such

as http://www.tcoffee.org/, http://tcoffee.crg.cat/, http://tcoffee.vital-it.ch/, http://www.igs.cnrs-mrs.fr/, http://toolkit.tuebingen.mpg.de/, http://cbsuapps.tc.cornell.edu/, and http://www.es.embnet.org/.

WebPRANK [141] provides an interactive web interface enabling its users to customize their MSAs by modifying the sequence-estimated phylogeny. After the sequences are uploaded to the server, WebPRANK performs a progressive alignment [142] and displays the alignment result along with its estimated phylogeny. WebPRANK's users can reorder the branches in the tree, and the alignment will be recalculated. WebPRANK works well with small alignments.

MVIEW is a utility tool that takes an alignment result and adds HTML markups for online representations.

Overall, the European Bioinformatics Institute provides the most alignment web-based services.

8.2 NCBI: NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION

The National Center for Biotechnology Information (NBCI) [143] is the most comprehensive facility in the United States that maintains and hosts many major sequence databases and biological related research information. Representative sequence databases at NCBI are GenBank, EST, GSS, HomoloGen, HTG, SNP, RefSeq, STS, UniSTS, and UniGen.

Since NCBI contains so many large sequence databases, to its best interest, NCBI provides tools mainly for database searches. Moreover, NCBI is known for its most complete and efficient BLAST service.

BLAST [24] is the most popular pairwise sequence alignment being used to search sequence databases. BLAST is a heuristic search method that splits an input query sequence into small segments (or words) of size w and scans a sequence database for matching segments with a score greater than a predetermined cut-off score S. (BLAST treats the sequence database as a long sequence concatenating all the sequences in the database.) When such a segment is found, it is anchored, on both the query and the database, and extended until the total score of the segment begins to decrease. These pairs of segments are called high scoring segment pairs (HSP). BLAST returns any sequences from the database that contain the highest scored segment pairs (i.e., the maximum segment pairs, MSP). 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

$$E = 1 - \mathrm{e}^{-y}, \tag{8.1}$$

where $y = Kmn\mathrm{e}^{-\lambda S}$, and K and λ are statistical parameters representing the natural scales of the search space and the scoring system. K and λ depend on substitution matrix, gap penalties, and 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 BLAST result is highly unique and not due to errors.

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

$$P = 1 - \mathrm{e}^{-y} \sum_{i=0}^{c-1} \frac{y^i}{i!}. \tag{8.2}$$

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

There are multiple versions of BLAST developed for specific type of sequence search such that BLASTn and mega-BLAST are for nucleotide sequences; BLASTtp, psi-BLAST, Phi-BLAST, and delta-BLAST are for protein sequences; and tBLAST and tBLASTn are for translated protein sequences and translated nucleotide sequences, respectively.

In addition to BLAST, NCBI also provides COBALT [144] as its primary multiple sequence alignment tool. COBALT is a progressive alignment tool that generates its alignment result by pairwise-combining sequence constraints derived from domain and motif databases and sequence similarity using BLAST.

8.3 GENOMENET AND DATA BANK OF JAPAN

Similar to NCBI and EMBL-EBI, the DNA Data Bank of Japan (DDBJ) [145] mirrors most of the sequence databases hosted by NCBI and EMBL-EBI and provides similar access mechanism to these protein, genome, and DNA sequence databases. Among many tools for screening and annotating sequences, DDBJ provides BLAST [24] for database sequence search and ClustalW [55] for multiple sequence alignment.

GenomeNet [146] is a Japanese network of databases and computational services for research in genome and related areas. For pairwise sequence analysis, GenomeNet provides BLAST, FASTA [23], and MOTIF [146]. While BLAST and FASTA perform pairwise alignment against a database for the best sequence matches, MOTIF [146] tool searches against a database of motifs to identify possible motifs embedded in any given input sequence. For MSA, GenomeNet provides ClustalW [55], MAFFT [57], and PRRN [78] alignment tools.

FASTA [23](stands for FAST-All) is a sequence search algorithm that utilizes a lookup table, for efficient access, to organize residue symbols of any two sequences – one is the query sequence and the other is the sequence database as a long sequence connecting all other sequences. FASTA performs its search by mapping these two sequences onto two connecting sides of a 2D matrix. A slot in the matrix is scored when there are similar residues between its column and row using scoring matrix such as PAM matrix [25], BLOSUM [27] matrix, or a scoring scheme. FASTA then eliminates the matrix short and low scored diagonals and extends the remaining diagonals for the maximum combined score. The max scored diagonals represent the best marching between the query sequence and the

corresponding sequences in the database. FASTA is rather outdated comparing to BLAST, a more preferable sequence search tool.

PRRN [78] is a progressive MSA that employs an iterative step to refine its alignment result based on the sequence 3D structural information. Similar to most progressive alignment methods, PRRN builds its phylogeny/guiding tree from all pairwise sequence alignments and aligns the sequences in the following order dictated by the tree. After the alignment is assembled, PRRN assigns weights to residues that are parts of similar 3D structures and then refines the alignment result to shift these residues closer together. The weights assigning and residue shifting process is repeated until the alignment converged with no more changes. The incorporation of 3D structure information in its iterative refinement step boosts PRRN's alignment quality.

8.4 OTHER SEQUENCE ANALYSIS AND ALIGNMENT WEB SERVERS

Apart from those previously listed facilities, there are many other organizations that provide servers with web-based services for sequence analysis and alignment. We are not trying to list all the existing servers, but only those that are popular, unique, or those that provide practical computing power for their services.

Top of this list would be the France's Pasteur Institute [147] that provides the largest set of web-based sequence analysis and alignment tools (more than 30 tools as of August, 2012). However, more modern alignment tools such as ProbCons [86] or PADT [107] are not available. Likewise, the Japan's Computational Biology Research Center (CBRC) [148] provides a host of web-based services for protein and DNA analyses. CBRC's chosen tool for MSA is MAFFT [57].

Welcome Trust Sanger Institute [149] is another top contender in this list. It is one of the primary leaders in the Human Genome Project. Since the institute deals with genomic data, SSAHA [150], a pairwise alignment tool, is provided for mapping sequencing reads onto genomic reference sequences. Similar to BLAST and FASTA, SSAHA (Sequence Search and Alignment by Hashing Algorithm) is a sequence search that partitions DNA sequences in a database into segments and organizes them in a hash table for efficient substring searches.

Some other worth-mentioning organizations that provide web-based sequence analysis and alignment tools are the Center for Biological Sequence Analysis at Denmark technical University [151], the Swiss Institute of Bioinformatics [152], the Bielefeld University Bioinformatics [153], the France Institute of Genetics and Microbiology (IGM) [154], and the Pole Bioinformatique Lyonnais (PBIL) [155]. Among these, PBIL uniquely provides a web server for 3D modeling protein sequence based on multiple alignment of complete sequences, and IGM uses Multalin [156], one of the first implemented progressive MSA tool, for its multiple sequence alignment.

The Helix Systems group at National Institutes of Health [157] provides HelixWeb, a web-based server for MSA on a Biowulf supercomputer with 12,000+ processor Linux cluster. HelixWeb can perform several multiple sequence alignment

tools on the same set of sequences. This unique service is unseen from any other providers.

Many sequence analysis alignment software packages can be downloaded from servers such as Grishin Lab [158] at Southwestern Medical Center, the Finland IT center for Science [159], or the Center for Bioinformatics at Peking University in China [160].

8.5 SEQANA: MULTIPLE SEQUENCE ALIGNMENT WITH QUALITY RANKING

SeqAna is a web-based sequence analysis server that provides MSA scoring and ranking services. SeqAna can process sequences with all known formats; thus, its users do not have to deal with the tedious and cumbersome task of format conversion. SeqAna's users can perform multiple sequence alignment (MSA) using many available alignment tools, and the results are ranked by the order of alignment quality. With this unique service, SeqAna's 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 service is the combined services of HelixWeb [157] and Mumsa [161]. Figure 8.1 shows a web interface of SeqAna multiple sequence alignment with ranking service. The ranking service has two modes: (a) user-selected and (b) best-predicted. In the user-selected mode, the users can select

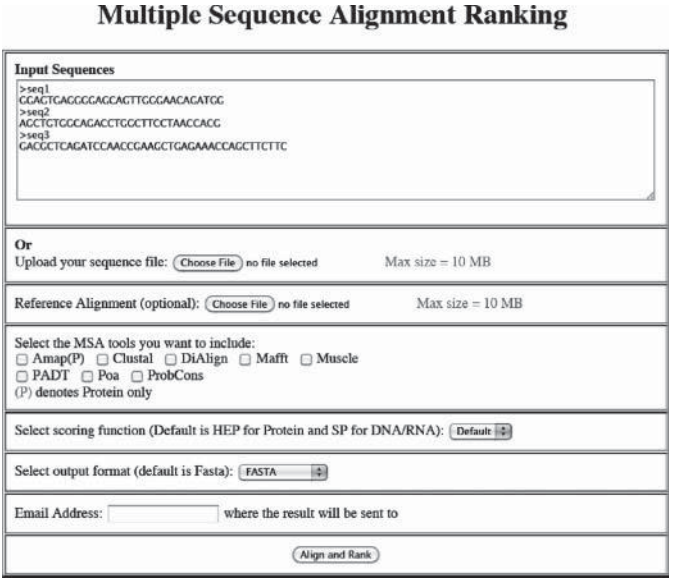


Figure 8.1 SeqAna's multiple sequence alignment and quality ranking service.

the alignment tools they want to perform on their input sequences. The system will invoke these tools one by one and then score and sort the results by the order of alignment quality. SeqAna's users can choose one of the provided scoring methods to rank the result. Alternatively, they can upload a reference alignment for ranking. The latter is very useful for evaluating the capability of existing alignment and new alignment tools.

In the best-predicted mode, SeqAna randomly chooses a small sample set of input sequences and invokes all available multiple sequence alignment tools on these sample sequences. (For alignment jobs with a few input sequences, the entire input data set will be used.) Alignment tools that yield the highest alignment quality on the sample set will be used to align the entire input sequence set. SeqAna's users can also pick a small set of sequences, align them using one of SeqAna alignment tools, correct and modify the alignment result to their likes, and submit this small alignment as a reference for aligning other sequences. SeqAna will use this reference to select alignment tools that give the most similar result to the reference for the entire sequence set alignment. As a result, SeqAna users will probably get highest alignment quality results.

By default, SeqAna ranks protein sequences via Hierarchical Expected matching Probability (HEP) scoring function [162] and the standard sum-of-pair (SP) [35] method. If a reference alignment is given, SeqAna will use the Total Column score (TC), commonly used with alignment benchmarks such as BAliBASE [52] or PREFAB [53], to rank its alignments.

The Total Column (TC) score calculates the percentage of residue symbol similarity between the alignment columns and their corresponding column in the reference alignment. The sum-of-pair scoring method measure the overall quality of each alignment by summing all pairwise scores of the residue symbols in each column, and the alignment score is the sum of all column scores. Some commonly used scoring matrix series are PAM [25] and BLOSUM [27]. These scoring matrices are derived from the probability a symbol being substituted by another based on either the residue physiochemical properties or their occurring frequency in sequence databases.

Unlike these scoring methods, the HEP [162] score builds a set covering hierarchy for all residue symbols from a pairwise scoring matrix, where each overlapping set is considered a tree node in the scoring tree as in Figure 8.2.

To calculate the score of an alignment column, the HEP scoring method sums all matching scores of the scoring tree covering all residues in the column. Each tree node score is 1 fewer than the number of residues matching at that tree node raised to the power of the node cardinality and then factored out by the node level. The HEP column score is mathematically defined as

$$cScore(i) = \begin{cases} 0, \text{if } i \text{ matches only at tree root,} \\ \frac{1}{(k-1)_i^{\Gamma}} \sum_{l=1}^{log\ |T_{col_i}|} \sum_{j=1}^{|T_l|} \frac{(count(node_j)-1)^{c_j}}{l}. \end{cases} \tag{8.3}$$

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 $q[j]$.

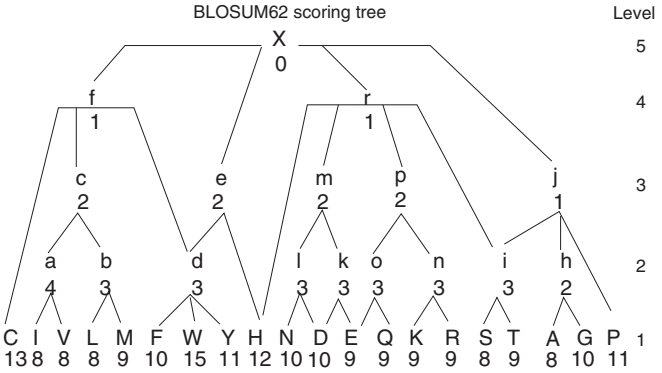


Figure 8.2 The hierarchical expected matching scoring tree built from BLOSUM62.

The overall alignment score is the sum of all HEP column score. While the HEP score is more complicated to calculate than other methods, it has shown to be more biologically sensitive and reliable than others.

The MSA tools implemented on SeqAna are from both Ubuntu public depository and in-house development. Some of the sequence analysis utility tools provided by SeqAna are obtained from BioPHP.org and modified for SeqAna. The utility services are self-explanatory.

Currently, the following MSA tools are available on SeqAna: AMAP [163] ClustalW [59], DIAlign [56], MAFFT [57], MUSCLE [53], PADT [107], POA [84], and PROBCONS [86].

8.6 PAIRWISE SEQUENCE ALIGNMENT AND OTHER ANALYSIS TOOLS

To perform pairwise alignments, SeqAna's users can paste their two sequences in the two given boxes and press the alignment button, as shown in Figure 8.3. Both Needleman–Wunsch's global alignment [19] and Smith–Waterman's local alignment [18] are available.

For sequence retrieval and homologous search, SeqAna provides NCBI's BLAST tool. However, it is not our intention to replace NCBI's BLAST service but rather provide a different type of service that is not available anywhere. SeqAna's BLAST service allows its users to perform BLAST on their own specific two sequences, the query sequence and the database of sequences, without the need of setting up the database and installing the BLAST. This service is very useful for performing BLAST on a small sequence database consisting of sequences that are simulated, unknown, or have not been uploaded to NCBI. SeqAna's BLAST web interface is shown in Figure 8.4.

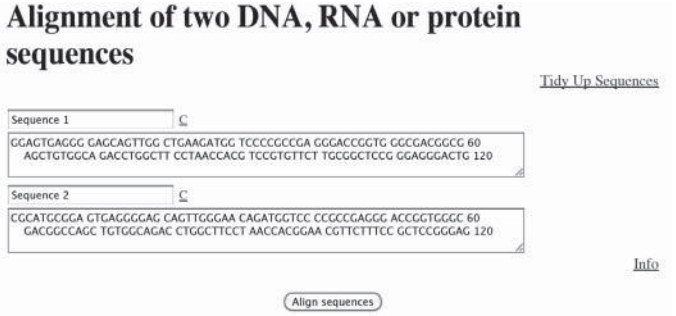


Figure 8.3 Pairwise alignment service.

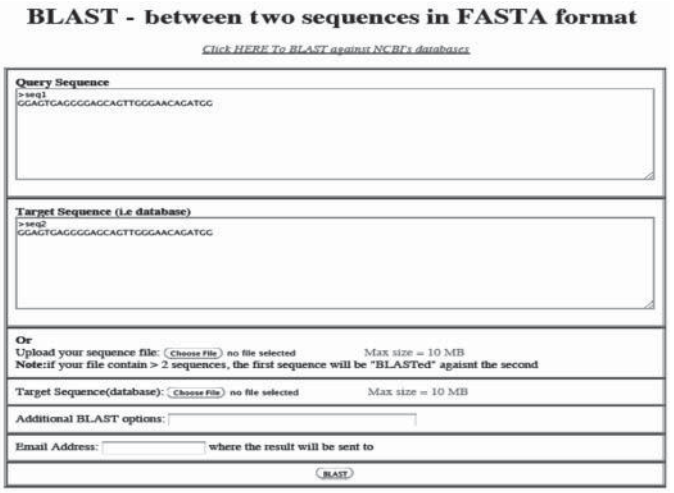


Figure 8.4 SeqAna's Unique BLAST Service.

SeqAna also provides a format conversion service allowing sequences to be converted from one format to another. This is a very useful service when SeqAna's users want to have their alignment conservation blocks be annotated with markups, to use their sequences as input for some sequence analysis tools that require a specific input format, or to store the sequences with minimal space. SeqAna's format converting web service utilizes **readseq** package written by Gilbert and publicly available [164]. Figure 8.5 shows the web interface of this sequence formatting service.

To use SeqAna's services, the user can either paste their sequences or upload files containing the sequences. For small jobs, SeqAna will return the result immediately; otherwise, it will send an email containing the output of the job to the user's emails, if provided. The latter is a preferred method for jobs with large input sequences.

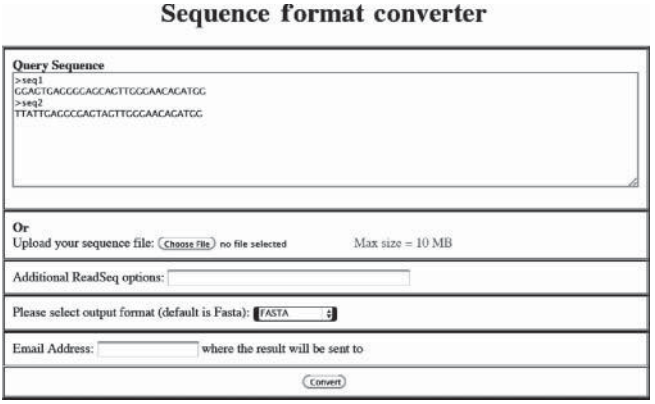


Figure 8.5 Sequence format converting service.

8.7 TOOL EVALUATION

Sequence search and alignment tools rely on benchmark data sets to evaluate their effectiveness. And each benchmark data set has its own criteria for selecting the sequences and measuring the effectiveness of the result. Some contains real biological sequences, while other contains computer-generated sequences. There are many sequence benchmark sets, however, the following six benchmark data sets are popular. BAliBASE [52] has eight reference sets, one for each type of alignment problem. Ref1 is specific for test cases containing small numbers of equidistant sequences and is subcategorized by percent identity. Ref2 alignments contain unrelated sequences. Ref3 contains two divergent subfamilies with less than 25% identity between the groups. Ref4 contains tests with long terminal extensions. Ref5 is for testing large internal insertions and deletions. References 6–8 deal with transmembrane regions, inverted domains, and repeat sequences. Some of the BAliBASE test cases are confined to homologous regions, which are difficult to detect because these region boundaries may be unknown.

OxBench [165] contains three related data sets: MASTER, FULL, and EXTENDED. The MASTER set is used to test isolated domains derived exclusively from sequences of known structure. The FULL set is a full-length sequence data that was generated from the suitable MASTER set. And the EXTENDED set is the MASTER set with additional high-scoring homologous sequences.

PREFAB [53] set is generated by pairwise alignment of sequences with known 3D structure and added up to 24 high-scoring homologues for each sequence. The effective assessment of an alignment is based on the structural alignment of original pair alone.

SABmark [166] contains two subsets: SUPERFAMILY and TWILIGHT. The tests in the first set contain sequence with 20–50% similarity, and the second set contain sequences with 25% similarity or less. False-positive homologous blocks are also

included in the sequences. The assessment is based on average pairwise alignment of all sequences in a test.

IRMBASE [167] contains test cases where random sequences are induced with simulated motifs. The assessment is based on the accuracy of locally aligning thee motifs.

HOMSTRAD [15] contains homologous protein families, which is derived exclusively from the protein structures from Protein Data Bank (PDB).

In 2006, Gordon Blackshields et al. [168]. performed benchmark tests on Align_m [169], ClustalW [55], DIAlign2 [56], DIAlign-t [167], MAFFT [57] with various versions, MUSCLE [53], ProbCons [86], PCMA [170], POA [84], and T-Coffee [171] alignment packages.

ProbCons is more accurate than others on the BAliBASE, OxBench, and HOMESTRAD data sets. In the PREFAB data set test, ClustalW, T-Coffee, and MAFFT are more accurate than others. With the SABmark tests, MUSCLE, MAFFT, and ProbCons are the most accurate. And with the IRMBASE benchmark, DIAlign-t, DIAlign2, POA, T-Coffee, and PCMA are more accurate because these tools incorporate local alignment strategies.

In the next chapter, we will explore other multiple sequence alignment algorithms and tools being used in assembling and aligning Next Generation Sequencing sequence reads.

9

MULTIPLE SEQUENCE FOR NEXT-GENERATION SEQUENCES

9.1 INTRODUCTION

Traditional sequencing techniques, such as capillary electrophoresis-based Sanger sequencing and microarray, are expensive and have limitations such as throughput, scalability, speed, and resolution. The new approach is called next-generation sequencing (NGS), which takes a slightly different approach to improve the sequencing throughput. The NGS technique utilized the same concept as in capillary electrophoresis-based Sanger technique on small fragments of DNA. From a DNA template strand, the bases of a small fragment of DNA sample are identified from signals emitted as the fragment is resynthesized. This process is performed sequentially for each base in the fragment. The NSG can perform millions of DNA fragments in parallel, thus producing hundreds of gigabases of data in each sequencing run. Consider a single genomic DNA (gDNA) is being fragmented into a library of small segments. The segments are flushed through the DNA template multiple times with different synthesizing agents for each base, and the signals emitted from the binding of the bases and the template are identified and recorded. Hence, a string of bases for each segment is identified as a sequence read. The next step in the process is to assemble these small sequence reads onto to a reference genome. Differences between the reads and the reference are indications

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

of polymorphisms or of sequencing errors. If they are polymorphisms, the samples would have high alignment repetitions on the reference. With NGS techniques, researchers can specify the resolution of a particular region of the sample. Higher resolution means there are more sequence reads covering each based in the sample. For example, a 30× coverage means that there are 30 sequence reads covering each base of the sample on the average. There are many platforms for next sequencing, some of the popular platforms are Roche/454, Illumina, SOLiD, and Helicos, which are able to produce data of the order of giga base-pairs (Gbp) per machine day [172]. For example, the Roche/454 can generate 10^6 reads with 450–500 base-pairs in a 8-h run or the Illumina/Solexa can generate 50×10^6 reads with 35–70 base-pairs in about 48 h.

While the NGS technique can produce a massive throughput with great resolution, an inherent challenge with NSG technologies is how to correctly and effectively map and align a large amount of sequence reads. These problems arise because sequence reads are short and must be mapped onto unique positions in the reference, the reads may contain sequencing errors, the orientation of the read relative to the reference is not known, and the original sample may have diverged from the reference. Moreover, there are large amount of redundancy genome references, for example, there are less than 85% of 30 base-pairs (bp) sequences in the human reference genome that are unique [173], and there are no unique 9 bp sequences. Figure 9.1 shows the assembling of reads, where continuous regions of overlapping reads are called contigs and the groups of contigs connected by gaps of known length are called scaffolds. Since the length of each read is known, the distance between two end read is known so as the gap between the contigs is a scaffold.

In this chapter, we will look at various multiple sequence alignment techniques that have been developed for NGS reads.

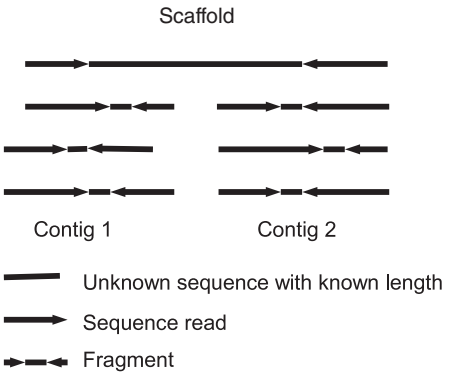


Figure 9.1 Assembling sequence reads.

9.2 OVERVIEW OF NEXT GENERATION SEQUENCE ALIGNMENT ALGORITHMS

Most of the alignment problems in NGS are assembling the reads and aligning them to their reference. With the magnitude of sequencing technologies throughput, most alignment algorithms utilize data structures that provide effective indexing and searching operations. In general, the alignment problem is treated as a string searching, where the reads are substrings of the reference, and the alignments are done by identifying the location of the read in the reference. One of the most effective techniques to handle this problem is indexing the sequences, both the reads and the reference. The sequences can be indexed by keywords and stored in hash tables where the words can be identified efficiently. An alternative is creating a suffix tree, where all possible suffix in the reads and the reference can be identified optimally.

9.2.1 Alignment Algorithms Based on Seeding and Hash Tables

The most simple form to find the occurrences of a read on the reference genome is to slide the read from one end of the reference to the other end, one symbol as a time. At every step, if the read and the corresponding subsequence from the reference genome are identical, we have a match – a location where the read should be mapped onto the reference. Obviously, this technique yields a prohibitively large time complexity for a task with hundreds of thousands of reads. To reduce the computation, hashing is a technique to convert a string, that is, a subsequence into a key for fast search – the key usually is a number representing a location on a table in which we would place the string. Thus, similar string will get the same key and will be stored as the same location.

The concept of using a hash table indexing is seen earlier in BLAST. Similar to BLAST, hash table based algorithms fundamentally follow the same seed-and-extend paradigm, where the position of each k-mer ($k = 11$ by default) subsequence of the query is stored in a hash table and the k-mer sequence is the key. The algorithm scans the database sequences for k-mer exact matches by looking up the hash table. These matches become the seeds or the anchor points, where the algorithm extends its matching and joining all the seeds without gaps as a maximum sequence pairs (MSP). The Smith–Waterman [18] pairwise alignment is performed to find the optimal matching between the MSP and the sequence database, in NGS reads, it is between the reference genome and the reads. Nevertheless, it is the same technique to search for the best alignment of the short sequence reads onto a long reference genome. To handle inexact matches between reads, Needleman–Wunsch [19] is utilized.

In particular, these alignment algorithms using seeding and hash tables: SeqMap [174], MAQ [175], RMAP [176], ZOOM [177], MOM [178], BFAST

[179], PASS [180], PerM [181], CloudBurst [182], SHRiMP [183] and RazerS [184], GNUMAP [185], NovoAlign (http://www.novocraft.com), ELAND (http://www.illumina.com/), and FANse [186]. Majority of these algorithms hash sequence reads, except GNUMAP [185], MOM [178], NovoAlign (http://www.novocraft.com), PASS [180], and BFAST [179] hash the reference genome. Majority of the differences are in seeding techniques or result in fine-tuning techniques.

Most of the early developed algorithms for read mapping such as SeqMap [174] and MAQ [175] utilize the same strategy as seen in BLAST. To handle mismatches in the reads, SeqMap [174] and MAQ [175] were modified to allow k-mismatches. The concept is that if a read of length n is fragmented into more than $n = k + j$ parts, there would be at least j parts that are perfectly matching the reference. These perfect matched parts are combined as a key to index all possible matchings of the parts. This is the problem of n choose j, thus reducing the total number of possible outcomes in the solution space. Asymptotically, the solution space remains the same; however, in practice, both SeqMap [174] and MAQ [175] restrict the number of mismatches to a few. For example, MAQ only allows two mismatches in the first 28 base-pairs; thus, it can reduce the search space significantly.

RMAP [176] partitions each read into $k + 1$ segments of the same length (seeds), where k is the maximum number of mismatches allowed. Thus, there must be at least one seed that matches exactly with the reference, and RMAP will search and starts with this seed. With the exact seed location found, RMAP utilizes Baeza-Yates–Perleberg algorithm [22] to identify $k + 1$ seed combinations as templates and scan the template for the best match. RMAP algorithm does not consider insertions and deletions in their reads.

ZOOM [177] is an improvement of RMAP, where the filtering is improved by applying spaced-seed mechanism. The idea of spaced seed is that for each seed, some of the regions must match exactly while others are not, which are considered "don't care." A binary string of 1's and 0's is used to represent the order of the exact matches and "don't cares," respectively. The length of the seed is the length of the string and the weight of the seed is the total number of 1's in the string. For example, a spaced seed represented as a string of "0011101000" has a weight of 4 and a length of 10. Analyzing the spaced seed representation, Lin et al. realized that the zeros in the string can be removed without compromising the capability to filter out seed alignments matching with low scores. Thus, the string "00011101000" can be reduced to "11101." The shorter the string, the smaller the solution space. Lin et al. attempted to prove case by case that it is possible to construct a minimal set of seeds with certain weights. They suggest to build the seeds manually to achieve 100% sensitivity. From their result, nine spaced seeds of weight 14 are sufficient to detect four mismatches out of 50 base-pairs.

Spaced seed is one of the most popular techniques in reducing solution space for aligners of NGS reads. For example, PerM [181] utilizes a periodic spaced seeds. For example, the following spaced-seed pattern: (1110100)(1110100) (1110100)(1110100) can be slid six times to generate 7 spaced seed patterns that provide full sensitivity up to two mismatches for 34 base-pairs reads.

It is intuitive that for any read position i (except the boundaries of the reads), there exists a position j, $8 \leq j \leq 14$, where $i\text{mod}7 = j\text{mod}7$, such that when j is aligned to -0-, so is i. Therefore, if every pair of positions within a pattern are covered with "don't care" "0" will have a correspondence pair of positions in the read with the same coverage. The following spaced seed pattern: (1110100)(1110100)(1110100), (1110100)(1110100)(1110100)(1110100)11 is used to generate the periodic spaced seeds to cover read length from 25 to 36 base-pairs.

CloudBurst [182] and GNUMAP [185] implement a similar algorithm as in RMAP. The differences are small, for example, CloudBurst is an implementation of RMAP to run on cloud platform, and GNUMAP utilizes position weight matrix (PWM) with Needleman–Wunsch's algorithm to align the read to the reference genome.

The most recent development of mapping algorithm utilizing seeding is FANse [186] by Zhang et al. This algorithm splits the reads into seeds of fixed size k. The only overlapping is allowed on the last seed in case the read length is not multiple of seed length k. The seeds are then aligned to the reference genome. Adjacent seeds from the same read will be combined to extend the alignment if the order of the seeds is the same in the original read. Locations on the reference genome where multiple seeds overlapped are given priority in the extending step based on the number of overlaps.

Another newly developed algorithm is VICUNA [187]. This algorithm performs its mapping by clustering the reads onto similar groups, progressively aligning the groups into contigs and mapping the contig into the referenced genome. While these steps are very common in VICUNA, clustering uses min-hashed technique developed by Broder et al. [188] for web clustering. The clustering is done by generating sub-reads (fragments) with at most k-residues – kmers using sliding window techniques. These segments are uniformly hashed and represented as a min-hashed value. Thus, sub-reads with similar min-hashed values are considered closed since they are hashed into the same location, and reads that share multiple min-hashed values are clustered together. For a given variation rate of p, reads that have at most $p \times |r|$ bases, $|r|$ is the length of the read, are allowed to stay in their cluster. Finally, the reads in the cluster are progressively aligned by Needleman–Wunsch algorithm to generate a contig. The clusters are then used to build a de Bruijn graph, and a superpath for the graph is calculated. By aligning contigs to the reference, VICUNA we can effectively detect and provide very accurate mapping of reads.

The biggest disadvantage of seeding technique is that gaps are not allowed in the seeds; thus, sequence querying could be biased toward unfavorable results. A common way to resolve this problem is to use dynamic programming in latter steps to insert gaps into the seeding regions. SHRiMP [183] and RazerS [184] are alignment packages attempted to resolve seeding gap issues by applying q-gram filters with spaced seed in their alignment models.

In the q-gram filter technique, all matching substrings (α, β) between a target string A and a query string B contain at most ε errors, that is, difference symbols or gaps; where α is the substring from sequence A and β is the substring sequence B. It is also

required that the edit distance between α and β – the number of symbols to be changed or inserted to make the two substrings identical – is less the error rate dividing the max length of the substring, and the length of β is greater than a predefined threshold. This technique is similar to the dot-plot method in 2.2; however, the band of diagonals that contain the most matching substrings are identified.

Apart from spaced-seed and q-gram filtering techniques, SHRiMP [183] algorithm implements vectorized Smith–Waterman dynamic programming. The alignment concept in SHRiMP is rather simple, the user will determine how many seeds are needed for each read, and local pairwise alignment by Smith–Waterman dynamic programming will be performed on each fragment generated by q-gram filter. Since SHRiMP needs to identify the matching location of the fragment to the reference genome, multiple fragments can be aligned simultaneously using vector instruction on computer with multiple CPUs.

Similarly, RazerS [184] uses q-grams and dot-plot techniques [17] to map all the possible matching between a read and its reference genome. The dot-plot matrix is then filtered to keep diagonals with length greater than a predefined threshold. These diagonals are then joined to find the highest pairwise alignment. Similar to SHRiMP, RazerS transforms these diagonals into bit-vectors and have them computed simultaneously on multiple CPUs.

An earlier attempt to speed up the mapping process would be BFAST [179]. In this algorithm, the reference genome broken into seeds and multiple indexes of the seeds are stored into table. Thus, a read lookup could result in multiple hits, which indicate a possible correct alignment location. A small list of highest hit locations is tallied, and each read is then aligned to the reference genome at these locations by Smith–Waterman's dynamic programming. Using this technique, BFAST can quickly align the reads to the reference genome. While the quality of BFAST result is not very high, its rapid turnaround time is suitable to filter and generate initial result for generation sequencing machines.

Another early attempt to speed up the mapping process is PASS [180]. This simple algorithm fragments the reference genome into seeds and hashes them into a table. To speed up the process, PASS builds another index table containing short words (a selected segment of the seed) with corresponding pairwise alignment scores and its location on the reference genome. All seeds returning as a result of a read lookup will be used to identify all the corresponding prescored words in the index table. The read will be mapped onto the same location of the word it resembles most. Since the locations of the seeds and their words are in the index table, the mapping is very efficient.

Maximum oligonucleotide mapping algorithm(MOM) [178] is the simplest algorithm in this group. It builds a hash table for the reference genome. When a read is looked up against the table, all the matching seeds will be used to pairwise align with the read. The read is then mapped on the the reference genome onto the seeds that yield high alignment scores. The alignment is extended both ends until a mismatch is found.

9.2.2 Alignment Algorithms Based on Suffix Tries

Hashing-based algorithms can rapidly identify exact matches between the read and the reference; however, it is a challenge when mismatches must be allowed. An alternative to hashing is using a suffix trie – introduced by Weiner [189], allowing to search for all occurrences of identical substrings in $O(n)$ time. The suffix trie, also known as a PAT tree or a position tree, is a tree data structure whose edges are labeled with a string (usually a symbol) and a suffix of a string S corresponds to exactly one path from the root to a leaf of the three. Each internal node of the tree contains at most two child nodes (except the root node). A suffix trie of n nodes can be built in $\Theta(n)$ time, and depending on the underline data structures used, the time complexity for lookups, insertions, and traversals can be done in linear time. The suffix trie can be implemented as an array on mass storage, called suffix array, to overcome the limitation of computer memory. Figure 9.2 shows the suffix trie for "GATTATTACCA."

To achieve a practical performance, all algorithm must reduce all searches into operations similar to hash table lookup. This requires all inexact matches must be transformed into exact matches. With exact matching search, the suffix trie provides linear time lookup. For inexact matches, some nodes on the suffix tree lookup must be bypassed – the number of bypassing nodes corresponds to the number of mismatched symbols allowed in the search. One of the earliest algorithm using suffix trie for NGS reads is MUMmer [190]. This algorithm builds the suffix trie for the reference genome and search the reads on the suffix trie for the longest matching string. MUMmer [190], such as OASIS [191], originally develop to align general biological sequences. When the the NGS technology emerges in the 2000s, it has been updated to handle the new type of alignment. Similarly, VMatch [192] and Segemehl [193] implement reads mapping using suffix search on suffix arrays to handle a large number of reads.

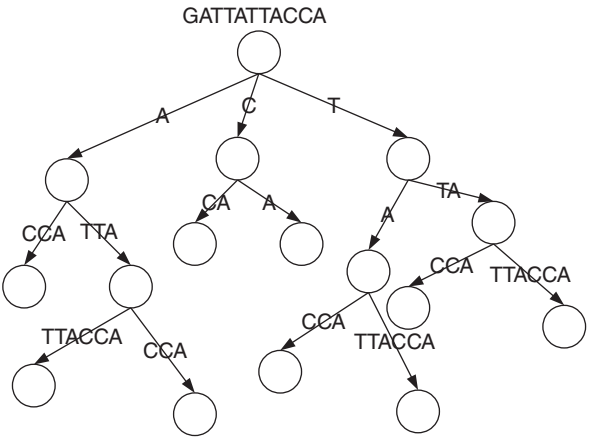


Figure 9.2 "GATTATTACCA" suffix tree.

While suffix tree is an effective data structure for subsequence searches, it requires a large amount of computer memory to operate. Since the alphabet set used in the reads and the genome is small, many alignment algorithms try to reduce symbol representation redundancy in the sequence to speed up the computation. For example, Bowtie [194], BWA [195], and SOAP [196] utilize Burrows–Wheeler transform (BWT) [197] – a space efficient text compressing algorithm.

Bowtie [194] is another well-known a memory-efficient and fast indexing algorithm. Bowtie uses BWT [197] and FM-index [198] to index reads. BWT [197] algorithm was invented by Burrows and Wheeler in 1994 to compress large texts for efficient search. The transformation permutes the order of the symbols in the text so that several substrings that occur often will be signified by similar single character appearing multiple times in a row. In general, BWT takes in an input sequence of length n, generates n sequences of length n by rotating the input sequence, sorts all the sequences by lexicographical order, and then outputs the last column of the sorted sequences. Table 9.1 illustrates the BWT of the input sequence "^GTGTAC$," where "^" and "$" denote the beginning and the end of the string, and they are considered larger than all the alphabets, ("^" > "$").

The BWT is reversible. If we take the output, that is, the last column of the sorted sequences, then sort it by lexicographical order, precede the sorted column with the input, and repeat sorting the result and combine it with the input until each row has exactly the same number of symbols as in the input. The original input sequence is the one that is preceded by "^" and ended by "$." This process is shown in Table 9.2, in which each step is proceeded from left to right.

The advantage of BWT is that repeated symbols and substring are coming together in the transformed sequence. For example, the sequence "^GTGTAC$" produces "TAT^GG$C", where two symbols "G" become adjacent. This phenomenon is very beneficial in practice because there a only four symbols being used for the bases in sequence reads, and BWT will produce transformed reads containing long runs of identical symbols. Thus, multiple subregions of the sequence will end up having the same representation that greatly reduce storage and searching space.

The FM-index [198] method exploits BWT property where generated rotating sequences are sorted. Each generated rotated sequence is inherently the suffix of the

TABLE 9.1 Burrows–Wheeler Transformation

Input	Rotate	Sort	Output
	^GTGTAC$	AC$^GTGT	
	GTGTAC$^	C$^GTGTA	
	TGTAC$^G	GTAC$^GT	
^GTGTAC$	GTAC$^GT	GTGTAC$^	TAT^GG$C
	TAC$^GTG	TAC$^GTG	
	AC$^GTGT	TGTAC$^G	
	C$^GTGTA	^GTGTAC$	
	$^GTGTAC	$^GTGTAC	

TABLE 9.2 Reversing Burrows–Wheeler Transformation

Input	Sort	Append	Sort	Append	Sort	Append	Sort
T	A	TA	AC	TAC	AC$	TAC$	AC$^
A	C	AC	C$	AC$	C$^	AC$^	C$^G
T	G	TG	GT	TGT	GTA	TGTA	GTAC
^	G	^G	GT	^GT	GTG	^GTG	GTGT
G	T	GT	TA	GTA	TAC	GTAC	TAC$
G	T	GT	TG	GTG	TGT	GTGT	TGTA
$	^	$^	^G	$^G	^GT	$^GT	^GTG
C	$	C$	$^	C$^	$^G	C$^G	$^GT

Append	Sort	Append	Sort	Append	Sort	Append	Output
TAC$^	AC$^G	TAC$^G	AC$^GT	TAC$^GT	AC$^GTG	TAC$^GTG	
AC$^G	C$^GT	AC$^GT	C$^GTG	AC$^GTG	C$^GTGT	AC$^GTGT	
TGTAC	GTAC$	TGTAC$	GTAC$^	TGTAC$^	GTAC$Ĝ	TGTAC$^G	
^GTGT	GTGTA	^GTGTA	GTGTAC	^GTGTAC	GTGTAC$	^GTGTAC$	^GTGTAC$
GTAC$	TAC$^	GTAC$^	TAC$^G	GTAC$^G	TAC$^GT	GTAC$^GT	
GTGTA	TGTAC	GTGTAC	TGTAC$	GTGTAC$	TGTAC$^	GTGTAC$^	
$^GTG	^GTGT	$^GTGT	^GTGTA	$^GTGTA	^GTGTAC	$^GTGTAC	
C$^GT	$^GTG	C$^GTG	$^GTGT	C$^GTGT	$^GTGTA	C$^GTGTA	

original sequence. Thus, when the transformed sequence is compressed and hashed, FM- indexing method indexes only these compressed substring as the preceding subsequence of all suffices being compressed. The exact locations of the suffices on the original sequence and the exact suffices themselves can be reconstructed easily via the BWT reverse transformation. In this concept, FM-indexing is a hashing of the prefixes of all suffixes generated by BWT. And this combination further reduces the searching space.

Bowtie [194] utilizes FM-indexing to search for possible matching of the read and use BWT to trace back the actual mapping of the reads on the reference genome. With this combined techniques, Bowtie needs only about 1.3 gigabytes (GB) to index the human genome, which makes it practical for most desktop computers. With such small-memory foot print, Bowtie can perform much faster than other alignment methods. However, the trade-off is that Bowtie may not find inexact matches in reads.

Similar to Bowtie, SOAP [196, 199] packages, and BWA [195] use FM-index method to build their indexing tables. To handle inexact matches (both mismatch and indel), BWA using a backtracking technique traversing a suffix tree implemented the indexed symbols generated by the FM-indexing. BWA utilizes breadth-first search (BFS) to search for the maximum paths among all the matches iteratively, some nodes on the path are systematically bypassed to allow inexact matches.

Unlike BWA, SOAP [196, 199] searches for inexact matches by splitting the reads into segments, a split for each allowable mismatch – the same splitting technique as seen in SeqMap [174] and MAQ [175]. For example, for one mismatch, each read is

split into two segments, which means the mismatch must be in one segment exactly and the other segment must match exactly with the reference genome. Similarly, for two mismatches, each read is broken into three segments, and at least one of these segment must match exactly to the reference genome.

Like all alignment algorithms, these algorithms are fine-tuned to align a specific type of reads and to identify a specific feature. Thus, they are incomparable by nature. In recent surveys [200, 201] of algorithms for NGS, most of these algorithms are performed comparable to each other as long as the reads are trimmed and unreliable reads are removed. Among these algorithms, NovoAlign and ELAND – both are proprietary products – seem to be the fastest of all. In any event, algorithms that are more sensitive to low-quality reads are the one implementing dynamic programming techniques; however, the trade-off is a big surge in performance time.

9.3 NEXT-GENERATION SEQUENCING TOOLS

Majority of these discussed mapping algorithms are implemented as console applications, where users have to understand how the algorithm works with appropriate commands. This section provides details information about where these tools can be obtained and their general constraints such as allowable read lengths, sequencing platforms, etc. This information is tabulated in Tables 9.3 and 9.4. While these tools are related to NGS mapping and aligning, many of them are designed for quality refinement.

Most of these tools come with detailed user manuals; however, it is beneficial to have some insights into how some of these tools behave.

BWA execution should be done in three successively commands: “bwa index,” “bwa aln,” and “bwa samse.” The first command indexes the reference genome, the second command (bwa aln) searches for all the coordinates of the hits of each individual read in the suffix array, and the last command (bwa samse) converts the suffix array coordinates to reference genome coordinates and creates the alignments in the SAM format. It is possible to set the maximum number of allowable mismatches per hit and the max number of hits to report; however, random hits will be reported if there are more hits encountered.

NovoAlign should be done in two successive commands: “novoindex” and “novoalign.” The first command (novoindex) indexes the reference genome and the second command (novoalign) aligns the reads against the indexed reference. In the free academic version, NovoAlign does not allow the user to set the maximum number of exact matches/mismatches between the read and the reference genome.

Similarly, Bowtie execution is done in two successively commands: bowtie-build and bowtie. The first command (bowtie-build) indexes the reference genome and the second command (bowtie) generates a list of alignments from a given input index and a set of reads.

SOAP2 execution also done in two successively commands: “2bwt-builder” and “soap.” The first command (2bwt-builder) builds the Burrows–Wheeler index of the

TABLE 9.3 Next-Generation Sequencing Alignment Tools – Download

Algorithm	Download
ABI SOLiD mapreads	http://solidsoftwaretools.com/gf/project/mapreads/frs/
AGILE	http://users.eecs.northwestern.edu/smi539/agile_x86_64_0.3.0
BFAST	http://sourceforge.net/projects/bfast/files/bfast/0.6.4/bfast-0.6.4d.tar.gz/download
BOAT	http://boat.cbi.pku.edu.cn/
Bowtie	https://sourceforge.net/projects/bowtie-bio/files/bowtie/0.12.7
BS Seeker	http://sourceforge.net/projects/bsseeker/files/BS Seeker 04-23-2010/examples.tgz/download
BWA	http://sourceforge.net/projects/bio-bwa/files/
BWT-SW	http://i.cs.hku.hk/ckwong3/bwtsw/bwtsw-20070916.tar.gz
CloudBurst	http://sourceforge.net/projects/cloudburst-bio/files/cloudburst/
ERANGE	http://woldlab.caltech.edu/erange/ERANGE3.2.tgz
Exonerate	http://www.ebi.ac.uk/guy/exonerate/exonerate-2.2.0.tar.gz
FusionHunter	http://bioen-compbio.bioen.illinois.edu/FusionHunter/
FusionMap	http://www.omicsoft.com/fusionmap/
GASSST	http://www.irisa.fr/symbiose/projects/gassst/Gassst_v1.23.tar.gz
GMAP	http://www.gene.com/share/gmap/src/gmap-2007-09-28.tar.gz
GNUMAP	http://dna.cs.byu.edu/gnumap/download.cgi
GSNAP	http://research-pub.gene.com/gmap/src/gmap-gsnap-2011-03-28.tar.gz
HMMSplicer	http://derisilab.ucsf.edu/index.php?software=105
MapSplice	http://www.netlab.uky.edu/p/bioinfo/MapSpliceDownload
Maq	http://sourceforge.net/projects/maq/files/maq/0.7.1/
MOM	http://mom.csbc.vcu.edu/node/14
Mosaik	http:/code.google.com/p/mosaik-aligner/downloads/list
mr(s)FAST	http://sourceforge.net/projects/mrfast/files/mrfast/mrfast-2.0.0.2/
MUMmer	http://sourceforge.net/projects/mummer/files/mummer/3.22/MUMmer3.22.tar.gz/download

TABLE 9.3 (*Continued*)

Algorithm	Download
MuMRescueLite	http://genome.gsc.riken.jp/osc/english/software/src/MuMRescueLite_090522.tar.gz
NovoAlign	http://www.novocraft.com/download.html
NSMAP	https://sites.google.com/site/nsmapforrnaseq/home/nsmap
PALMapper	http://ftp.tuebingen.mpg.de/fml/raetsch-lab/software/palmapper/palmapper-0.4-rc4.tar.gz
PASS	http://pass.cribi.unipd.it/cgi-bin/pass.pl?action=Download
PASSion	https://trac.nbic.nl/passion/wiki/Download
PERM	http://code.google.com/p/perm/downloads/detail?name=PerMSource0.2.9.9.zip&can=2&q=
ProbeMatch	http://pages.cs.wisc.edu/jignesh/probematch/
QPALMA	ftp://ftp.tuebingen.mpg.de/pub/fml/raetsch-lab/software/qpalma/qpalma-0.9.2-rc1.tar.gz
RazerS	http://www.seqan.de/downloads/projects.html
RMAP	http://www.cmb.usc.edu/people/andrewds/rmap/rmap_v2.05.tbz2
Segemehl	http://www.bioinf.uni-leipzig.de/Software/segemehl/segemehl_0_0_9_3.tar.gz
SeqMap	http://biogibbs.stanford.edu/jiangh/seqmap/download/seqmap-1.0.13-src.zip
SHRiMP	http://compbio.cs.toronto.edu/shrimp/releases/
SliderII	http://www.bcgsc.ca/downloads/slider/SliderII.tar.gz
SOAPaligner/soap2	http://soap.genomics.org.cn/down/SOAPaligner-v2.20-Linux-x86_64.tar.bz2
SOCS	http://socs.biology.gatech.edu/Download.html
SpliceMap	http://www.stanford.edu/group/wonglab/SpliceMap/download.html
SplitSeek	http://solidsoftwaretools.com/gf/project/splitseek/frs/
SSAHA2	http://www.sanger.ac.uk/resources/software/ssaha2.html
Stampy	http://www.well.ox.ac.uk/stampy-registration
SWIFT Suit	http://bibiserv.techfak.uni-bielefeld.de/download/tools/swift.html
Tophat	http://tophat.cbcb.umd.edu/downloads/tophat-1.3.3.Linux_x86_64.tar.gz
Vmatch	http://www.zbh.uni-hamburg.de/vmatch/Vmatchlic.pdf
Zoom	http://www.bioinformaticssolutions.com/products/zoom/download.php
ZOOM Lite	http://www.bioinfor.com/zoom/lite

TABLE 9.4 Next-Generation Sequencing Alignment Tools – Constraints

Algorithm	Max Read Length	Sequencing Platform	Methodology
ABI SOLiD mapreads		ABI SOLiD	
AGILE		Roche 454 Life Sciences	Hash-based alignment
BFAST	100 bp	Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	Hash-based
BOAT		Roche 454 Life Sciences, Illumina/Solexa	
Bowtie	1024 bp	Roche 454 Life Sciences, Illumina/Solexa	Burrows–Wheeler transform
BS Seeker	1024 bp	Illumina/Solexa	
BWA	bwa-short: 200 bp–100 kbp	Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	Burrows–Wheeler transform
BWT-SW		N/A	Burrows–Wheeler transform
CloudBurst		Illumina/Solexa	Seed-and-extend
ERANGE		Illumina/Solexa	
Exonerate			
FusionHunter		N/A	
FusionMap		Illumina/Solexa	Spanning fusion junctions
GASSST	500 bp	Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	Hash-based
GMAP			
GNUMAP		Illumina/Solexa	
GSNAP		Roche 454 Life Sciences, Illumina/Solexa	Uses GMAP
HMMSplicer	>45 bp	Roche 454 Life Sciences	
MapSplice	any	Illumina/Solexa	
Maq		Illumina/Solexa, ABI SOLiD	Hash-based
MOM		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	
Mosaik		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	
mr(s)FAST		Illumina/Solexa	
MUMmer			Suffix trees
MuMRescueLite		ABI SOLiD	
NovoAlign		Illumina/Solexa	
NSMAP		Illumina/Solexa	

TABLE 9.4 (*Continued*)

Algorithm	Max Read Length	Sequencing Platform	Methodology
PALMapper	75 bp	Illumina/Solexa	Burrows–Wheeler transform
PASS		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	Modified Smith–Waterman
PASSion		Illumina/Solexa	Pattern growth
PERM	128 bp	Illumina/Solexa, ABI SOLiD	
ProbeMatch		Illumina/Solexa	*q*-grams
QPALMA		Roche 454 Life Sciences, Illumina/Solexa	Modified Smith–Waterman
RazerS		Roche 454 Life Sciences, Illumina/Solexa	Hash
RMAP		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	
Segemehl		Roche 454 Life Sciences, Illumina/Solexa	Suffix-arrays
SeqMap		Illumina/Solexa	Indexing
SHRiMP		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	Hash-based
SliderII		Illumina/Solexa	Merge sorting
SOAPaligner/ SOAP2	1024bp	Illumina/Solexa	Burrows–Wheeler transform
SOCS		ABI SOLiD	
SpliceMap		Illumina/Solexa	
SplitSeek	min 50 bp	ABI SOLiD	Match and extend
SSAHA2		Roche 454 Life Sciences, Illumina/Solexa, ABI SOLiD	
Stampy	4500 bp	Roche 454 Life Sciences, Illumina/Solexa	Hash-based
SWIFT Suit			
Tophat	min 75 bp	Illumina/Solexa	
Vmatch			Enhanced suffix arrays
Zoom	240 bp	Illumina/Solexa, ABI SOLiD	Hash-based
ZOOM Lite	240 bp	Illumina/Solexa, ABI SOLiD	

reference genome, and the second command (soap) performs the alignments. The maximum number of mismatches allowed is 2.

BFAST execution should be done in five commands: “bfast fasta2brg,” “bfast index,” “bfast match,” “bfast localalign,” and “bfast postprocess.” The first command (bfast fasta2brg) converts the reference genome into BFAST format, the next three commands (bfast index, bfast match, bfast localalign) build the seeds for the reference genome, use the reads to find location of matches, and then align the reads. The last command (bfast postprocess) refines, formats, and outputs the data.

Most of tools listed in Table 9.4 follow similar logic as these described tools.

10

MULTIPLE SEQUENCE ALIGNMENT FOR VARIATIONS DETECTION

10.1 INTRODUCTION

Human genetic variations are generally defined as the differences in DNA sequence within the individual genomes in populations [202]. They serve as the footprints of errors or mistakes that occur in DNA replication during cell division. Although external factors, such as viruses and chemical mutagens, can induce changes in the DNA sequence, genetic variations happen naturally in the human genome. Variants in the human genomes have often been related to the disease. In some cases, relatively rare variants involve with large effect in rare disorders [203]; in others, relatively common variants have been shown to have smaller effects in complex diseases [204, 205]. Genetic variations have been widely utilized as genetic markers in the area of disease gene mapping. For example, in family linkage and genetic association studies, genetic variations are the clues to identify the susceptibility loci or genes for simple and complex diseases [206–208]. Genetic variations are also useful as genetic markers for many other applications, including forensic investigations [209], drug response prediction [210], clinical tests [211], personal medicine [212], and population genetics [213]. The ability to obtain all sorts of variants on the entire genome will make possible comprehensive searches for genetic factors in common disease and mutations underlying linkages in Mendelian disease, that is, simple disease. This ability will also enable us to find out spontaneously variation for which no gene-mapping shortcuts are available. Variation detection, also known as variant calling, which is largely

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

Figure 10.1 Types of single-nucleotide changes and short tandem repeats.

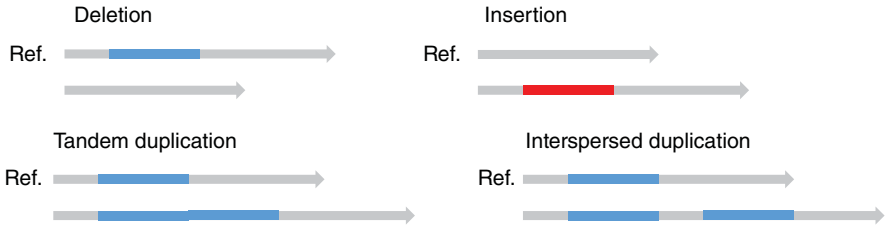


Figure 10.2 Types of copy number variations, including insertion and deletion.

statistical inference, is the process of inferring a sample genome from the noisy short-read data [214]. Most of the detection methods rely on a reference genome, and the goal is to find variants between the sample and the reference. Genetic variations in the human genome have many types of formations. If classifying variants according to the complexity and their segment size, we can have single-nucleotide variants (SNV) (changes and substations) (Figure 10.1); tandem repeats, insertions, and deletions (also known as indels); copy number variations (CNVs) that gain or lose multiple copies of a large segment of DNA sequence (Figure 10.2); and chromosomal rearrangements including inversions, translocations, and copy neutral loss of homozygosity or heterozygosity (Figure 10.3). The above genetic variations span a range of sizes from single nucleotides to megabases. It is worth noting that the

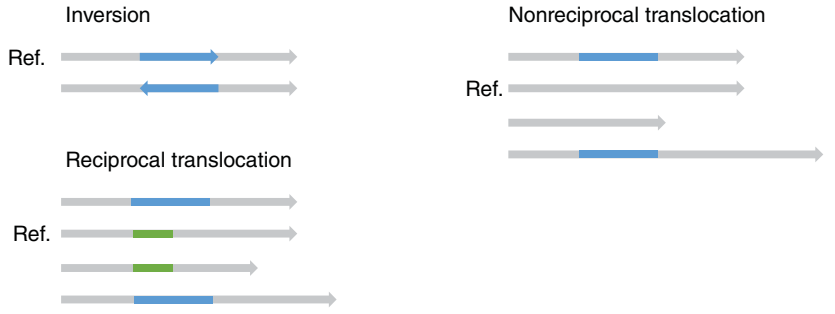


Figure 10.3 Types of chromosomal rearrangements.

categorizing of genetic variations into distinct groups is vague, and no clear agreement of genetic variation classification has been reached in the academy.

Recent advances in the high-throughput next-generation sequencing (NGS) technology [215] now provide scientists a cost-effective way to resequence human samples in a large scale for medical and population genetics [216]. Projects, such as the 1000 Genomes Project [217], the Cancer Genome Atlas [218], and numerous large medically focused exome sequencing projects [219], are currently in process with an attempt to elucidate the full spectrum of human genetic diversity and to complete the genetic architectures of human diseases [220]. Many bioinformatics downstream applications, such as read-quality assessment, mapping onto the reference genome, sequence assembly, variant identification, and individual genotyping, are required to obtain a complete and accurate record of the variation from NGS sequencing data, which holds incredible promise for the study of DNA sequence variations [221]. However, the massive NGS reads would have limited biological benefit when the performance of downstream processing and analysis is not reliable [222]. Although tools for performing these tasks have improved in recent years, they are still expensive and time consuming in the sequencing studies, including generating and assessing the quality of the reads on the whole genome, and calling and genotyping variants among multiple sets of individuals. With the sequencing errors minimized by quality control steps and high-coverage sequencing reads [223], the genotyping from sequence data becomes more reliable since the input data come with less noise, but the genotyping itself is not trivial due to the complicated variant types. Therefore, in this chapter, we introduce the current progress in variant calling making use of MSA technologies. This chapter is organized as follows: we first give a brief introduction about four typical variants, including SNVs, tandem repeats, CNVs, and chromosomal rearrangements; then we elaborate six variant callers that use MSA as a guidance; we next provide the common evaluation strategies for comparing and assessing genotyping results by different variant calling tools; at last, we summarize and point out future directions in this crossing realm of MSA and variant detection.

10.2 GENETIC VARIANTS

SNV, implied by the name, is defined as a change of a single nucleotide at a unique locus in the DNA sequence. SNV, as shown in Figure 10.1, contains single-nucleotide polymorphisms (SNPs) and single-nucleotide indels, including the transition or transversion. Based on the population frequencies, SNV may be classified as point mutations and polymorphisms; while the former includes both single-nucleotide substitutions and single-nucleotide indels with population frequencies less than 1%, and the latter is indicating those genetic variations with population frequencies higher than 1%.

The continuous multiple replicas involving more than one nucleotide are called tandem repeats. Based on the length of the repeat DNA sequence, tandem repeat can be broadly categorized into two classes: short tandem repeats and longer tandem repeats (Figure 10.1). The difference between the former and the latter is whether

the replica segment is less than eight base-pairs (bp) or not. It is apparent that the eight-base criteria are arbitrary. More specifically, repeats of the identical nucleotide of several bases or longer in the length are known as homopolymer sequences, for example, AAAAA or CCCCC. Compared to other more complex changes of DNA sequence, the rearrangement of the tandem repeat is simple, but the repeat of replica segments can range from tens to hundreds of times, thus making up a high allelic diversity or heterozygosity.

Similar to the definition of tandem repeats, CNVs, one type belonging to the structural variants, are defined as a DNA segment of 1 kb or larger with variable copy number, either tandem or interspersed duplicates, compared to the reference genome [224] (Figure 10.2). Although there is a low bound in the length for CNVs and a continuity requirement in the arrangement for tandem repeats, the distinction between them is vague. If considering indels as well, the distinction in the definitions is even more obscure. Based on the definitions by the Database of Genomic Variants [225], CNVs include the deletions and duplications/insertions larger than 1 kb, whereas indels include those deletions and insertions between 100 bp and 1 kb. As such, tandem repeats overlap with CNVs in longer DNA segments and overlap with short indels as well. In real genomes, all these variants are coexisting. For example, Bentley et al. [4] found 1400,000 indels within 1–16 bp in the African NA18507 genome, and Wang et al. [226] also detected nearly 140 000 indels within 1–3 bp in the Han Chinese genome. As indicated by some studies [202], we may need to rename those indels between 100 bp and 1 kb as "intermediate indels," and perhaps create a new category such as "short indels" for those indels within 1–100 bp. Similar to the polymorphisms of SNVs, the copy number polymorphisms, also known as common CNVs, are defined as those CNVs with population frequencies higher than 1%. However, some studies [227] also used the common CNVs to indicate the CNVs detected in two or more individuals.

The chromosomal rearrangement, as shown in Figure 10.3, is a chromosomal level abnormality involving large changes, such as deletions, duplications, inversions, and translocations in the structure of the native chromosome. Due to the propensity of misaligning during DNA repair, some chromosomal regions tend to be more easily rearranged than others and thus become sources for some cancers and genetic diseases. Apart from SNVs, all the genetic variations can be broadly grouped under the realm of structural variations (SV), including CNVs and chromosomal rearrangements. Sometimes, the same genetic variations are named as different genetic variant terms, which may confuse the beginners in the field of variant detection. For example, CNVs were used to be classified as the large-scale and intermediate-sized copy number variants separately. Some studies, using the comparative genomic hybridization array, adopted chromosomal gains and losses to indicate duplications and deletions, respectively [228]. It is worth noting that the sizes of variant segments in the definitions are capricious, even though we have created various categories of genetic variations and terminologies. Moreover, there is not much biological basis, as the mechanisms behind their occurrences, supporting the classifications. Instead, the classification is naively based on the sizes and patterns of sequence changes. Therefore, more accurate and meaningful classification of genetic variants with the consideration of

both biological inferences and sequence characteristics is needed for the downstream analysis in the future.

10.3 VARIATION DETECTION METHODS BASED ON MSA

Regarding the input and calling strategies, variant detection methods can be grouped into three categories: read mapping methods, *k*-mer matching methods, and genome alignment methods. Read-based and genome-based methods employ (multiple) sequence alignments to search genome variants. The difference between is the input data that the former use short reads while the latter use the whole genome. Variant callers based on *k*-mer matching employ alignments using subsequences with fixed length, *k* flanked by the conserved sequences among target genomes to produce high precision variants instead of aligning sequences [229, 230]. In the past, *k*-mer approaches have been heavily used to measure genome similarity and collect homologous genomic sequences [231–233]. Recently, *k*-mer matching has also been applied to SNV discovery [234–236]. Kidd et al. [236] introduced the diagnostic *k*-mer analysis to genotype sequenced novel insertions and discriminate heterozygous from homozygous insertions using the NGS data. Gardner and Hall [230] developed a software, kSNP v2, to identify SNP embedded in the center of odd-length *k*-mers possessed by multiple genomes. These matched *k*-mers can be aligned back to reference genomes if available to archive the locations of SNPs. Therefore, these approaches are suitable for both assembled genomes and NGS reads, but sensitivity is sacrificed for the improved efficiency of exact alignment [237]. Since *k*-mer approaches usually do not use MSA, we focus our attentions on the read mapping and genome alignment methods.

A common workflow of read- and genome-based method is shown in Figure 10.4. After the library preparation, samples will be sequenced on the NGS platform and filtered to control quality. Before the variant calling, the alignments are constructed by aligning short sequence reads to the existing references or the *de novo* assembled genomes. The detected variants can be further annotated and visualized using other dedicated tools, and prioritized and filtered by lab validation. In this workflow, the accuracy of sequence platform and the quality of multiple sequence alignment play a foundation role for the downstream analyses. Read-based approaches have dominated the bioinformatics landscape since the invention of high-fidelity, short-read sequencing. In these methods, reads with high quality are usually first aligned to an existing reference genome, like the human genome using a sensitive read mapper, such as Bowtie/Bowtie2 [238, 239], BWA [240], MAQ [241], and SOAP [242], or using multiple sequence aligners. No matter using which mapping or aligning tools, multiple sequence alignments are the basic input for variant calling. So far read-based methods can be divided into four broad categories regarding their inferring strategies [243]: alignment-based methods, split-read mapping methods, paired-end read mapping methods, and haplotype-based methods (Figure 10.5).

Instead of mapping short reads, high-quality genomes empower comparative genomic studies to detect variants based on multiple genome alignments. Multiple

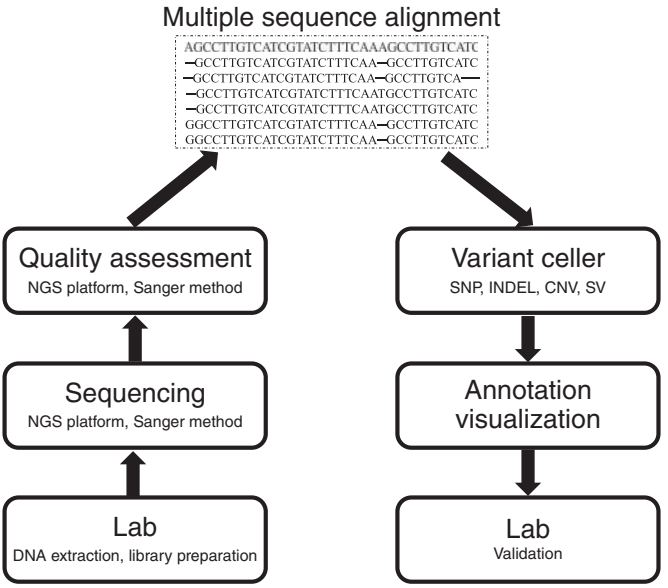


Figure 10.4 Variant calling workflow using multiple sequence alignment.

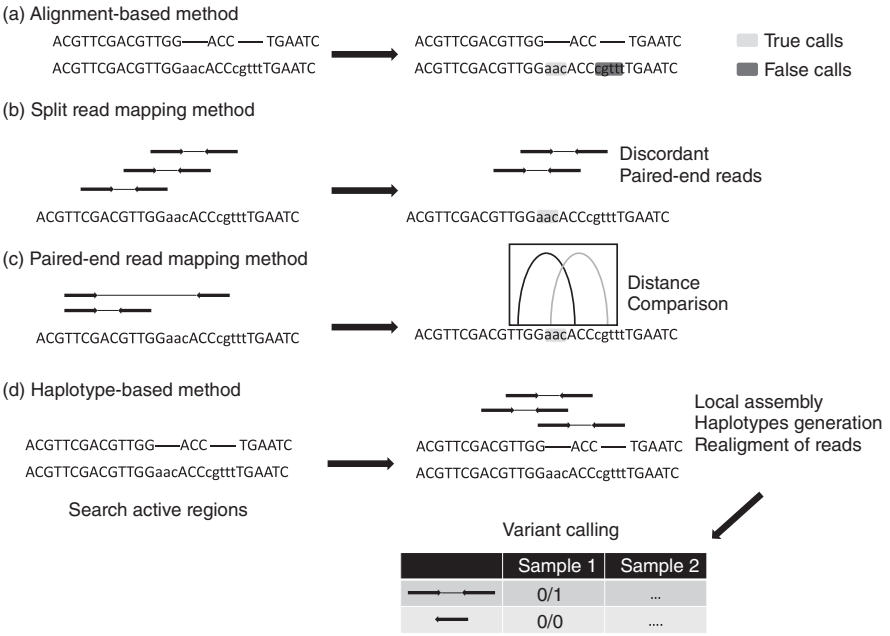


Figure 10.5 Categories of variant callers based on read mapping.

genome alignment is a fundamental tool in genomic studies, such as genome evolution tracking, identification of genomic islands, homology classification, and phylogenomic analyses. Without the requirement of the *de novo* assembly, this method has proven to be effective in practice [244]. Different from variant discovery for high-order organism, such as human, genome alignment is the original method for variant detection of closely related bacterial genomes [245, 246]. Aligning genomes is able to identify reliably all types of variants and usually serves as a gold standard for comprehensive variant identification, although it is dependent on highly accurate and continuous assemblies, which are expensive to generate sometimes. In this chapter, we focus on the variant callers based on read mapping and genome alignment, which use MSA as their main sources for variant inference. We will describe six variant callers, including three tools based on the genome alignment, that is, MUMmer [247], Mugsy [248], ParSNP [249], and three tools based on the read mapping, that is, MAQ [241], Gustaf [250], SoftSV [251], to deliver the whole picture or some insights of variant discovery.

- MUMmer
  Delcher et al. [247] described a system, named MUMer, to rapidly align sequences containing millions of bases using three techniques, that is, suffix tree [252], longest increasing subsequence (LIS) [253], and Smith–Waterman alignment [254]. With the alignment, MUMer can identify SNPs, indels, repeats, and other sophisticated variants. To locate the genetic variants, MUMer introduced a concept, maximal unique match (MUM), which is defined as a subsequence that shows exactly once in the target genomes and cannot be extended anymore. Based on the MUM, MUMer defined four classes of variations as depicted in Figure 10.6. The identifying process is comprised of four steps: (i) search all MUM subsequence in two target genomes; the principle of this action is that if a long and perfect MUM is found in genomes, it is most likely belonging to the global alignment. (ii) Find the LIS of a sequence of MUM to scan the alignment from begin to end. (iii) Locate SNPs, inserts, repeats, and tandem repeats inside the gap areas that separate the alignments. (iv) Write out the alignments and the imperfectly matched regions. The experiments using MUMer on two closely related genomes and two distant genomes revealed that a large number of MUMs are needed to be existing between highly similar genomes, which is an important assumption to ensure MUMer to be practicable.

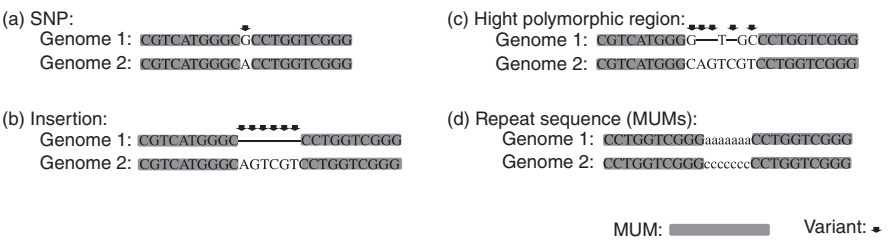


Figure 10.6 The four types of variants defined by MUM alignment.

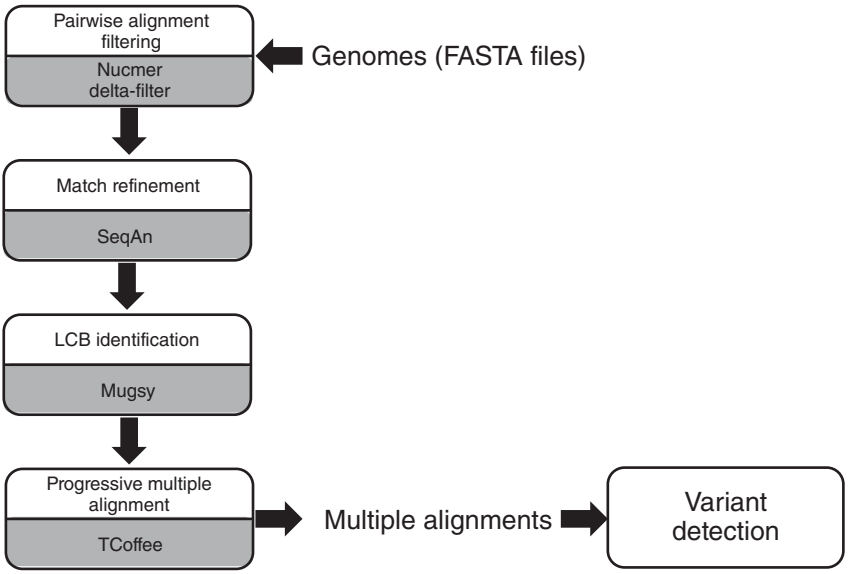


Figure 10.7 The workflow of Mugsy for variant detection. The uppers are the processes, and the lowers are employed tools.

- Mugsy
  Angiuoli and Salzberg [248] presented a multiple alignment tool, named Mugsy, which can align the mixtures of the assembled draft and the completed genome data. It can also identify genetic variations including both repeats and rearrangements. The Mugsy consists of four main steps as shown in Figure 10.7: (i) perform an all-to-all pairwise alignment with Nucmer [255] and refine the alignment using delta-filter [190]. (ii) Construct and improve the alignment graph using SeqAn [256], a sequence analysing tool. (iii) Identify Locally Collinear Blocks (LCBs) from the alignment chart. The LCBs represent aligned sections from two or more genomes without the rearrangements. (iv) Work on the multiple alignments for each LCB using T-Coffee [257]. Variants, including mutations, insertions and deletions, are extracted from the ungapped alignment columns inside the whole-genome multiple alignments produced by Mugsy. The experiments on multiple human genomes confirm that many variants reported by Mugsy are also indexed by the UCSC browser or dbSNP, which gave credits to what Mugsy found.
- ParSNP
  Treangen et al. [249] developed a suite, called Harvest, which consists a core-genome aligner, Parsnp, and a visualization tool, Gingr, for the analysis of intraspecific microbial strains. The core-genome alignment belongs to the whole-genome alignment, which focuses on identifying the set of orthologous sequence conserved in all aligned genomes. Parsnp takes MultiFASTA files

[258] as input. To do the alignment, Parsnp uses both whole-genome alignment and read mapping. Parsnp adopts the compressed suffix graph to index the reference genome for identifying the MUMs rapidly, which can be used to recruit similar genomes and anchor the multiple alignments. The compressed suffix graph maintains all intrinsic properties of the suffix tree and has the unique property to represent an optimally compressed number of nodes and edges. Parsnp requires input genomes to be highly similar to each other to generate intraspecific alignments. If genomes are inaccurate or incomplete, MUMi distance [259], the similarity index based on MUMs, is adopted to measure the similarities among genomes. Once all the MUMs are built, locally collinear blocks, the basis of the core-genome alignment, are identified. The MUMs within locally collinear blocks are employed to construct the backbone of the multiple alignments. Gaps between collinear MUMs are aligned using MUSCLE [53]. Parsnp flags all polymorphic columns in multiple alignments as variants or bases with poor qualities by FreeBayes [260]. The second tool in Harvest, Gingr, can be applied to explore interactively the alignments created by Parsnp. The experiments indicate that Parsnp is able to reconstruct and analyze hundreds of whole genomes efficiently and provides highly specific variant calls.

- MAQ
  Li et al. [241] described a tool, named MAQ, which can build assemblies by employing read mapping and estimate the error probability for the genotype calls, including short indels and SNPs, by using quality scores and mate-pair information. MAQ has two stages: the alignment stage and the SNP calling stage. At the alignment phase, MAQ first scans the reference genome sequence and indexes read sequences to identify those potential read alignments. A read is mapped to the position with the minimum sum of quality values according to the mismatched bases. To find the best read alignment, MAQ builds multiple hash tables to ensure that a sequence with no more than two mismatches will be hit. Each hash table corresponds to a noncontiguous seed templates consisting of 1 or 0, where bases at 1 will be indexed [261]. After the read indexing, the reference will be scanned and hashed base by base through the same templates used in the indexing. By looking up in the hash tables, if a potential alignment is found, then MAQ will extend out the alignment from the seed without gaps and calculate the sum of qualities of mismatched bases. MAQ then integrates the coordinate of the alignment and the read identifier as a score. If a read has multiple highest scores with different positions on the reference, only one position will be picked randomly. If the paired-end reads are available, MAQ keeps a queue to record the last two best alignments of a read on the forward strand end. When a potential alignment is found on the reverse strand, MAQ looks up the queue storing the mate reads to check whether an alignment exists on the forward strand. The mapping quality is measured as the *phred*-scaled probability [262]. MAQ finds short indels by utilizing the Smith–Waterman gapped alignment [254] and the mate-pair information. Given a read pair, if one end can be mapped to the reference but the other end cannot, a potential explanation is

that an indel interferes the alignment of the unmapped read. Smith–Waterman gapped alignment then can be used to locate these indel regions. At the SNP calling phase, MAQ generates a consensus genotype sequence from the alignments that are inferred from a Bayesian statistical model. By assuming that the sample is diploid by default, MAQ calls the genotype that maximizes the posterior probability of genotypes. The likelihood that the consensus genotype is incorrect is measured by *phred* quality. Potential SNPs can be easily detected by comparing the consensus sequence to the reference. Simulation and real data experiments suggest that MAQ is able to estimate error probabilities of alignments and consensus genotypes accurately, which provide the foundation for high-quality calls of indels and SNPs.

- Gustaf
  Trappe et al. [250] presented a generic multisplit SV detection tool, named Gustaf, for detection and classification of multiple genetic variants, including indels, inversions, repeats, and translocations. The key idea of Gustaf is to maintain all local alignments within a graph to identify the breakpoints of SV up to the base-pair resolution using standard graph algorithms. There are four primary steps in Gustaf: (i) build local alignments; (ii) split alignment; (iii) construct split-read graph; and (iv) classify SV. In the first step, Gustaf obtains the local alignment for each read using Stellar [263], a local aligner, or takes a set of local alignments as input instead. Stellar also adopts the seed and extending strategy, and it guarantees to find all local alignments between two sequences given the range of the match length and the maximal allowed error rate. A read spanning the breakpoint will be split according to its alignment, and the alignment spanning the breakpoint is confirmed according to the adjacency and relation, which indicate the particular type of SV. In the second step, two adjacent alignments overlapping at their sloppy ends are realigned to locate the breakpoint position. The implementation of realignment is similar to split alignment employed in the AGE [264]. In the third step, the information about adjacency and separated alignment is represented in a split-read graph, which is constructed in the way that the artificial vertices represent the start and end of the read. A vertex indicating each local alignment of a read is added to connect existing vertices with the edge weight calculated based on the editing score in the second step. In the last step, distinct SVs are determined by the adjacency of two alignments and their relation. For example, if two alignments are adjacent given the read and a gap exists between the reference positions, Gustaf states that the target genome has a sequence segment missing due to the deletion event. From the experiments on the benchmarked experiments, with the flexibility to allow multiple splits per read, Gustaf can detect small SVs up to 500 bp, which makes it possible to handle long contigs with mapping information.
- SoftSV
  Bartenhagen and Dugas [251] presented and evaluated a method, called SoftSV, for detecting variant breakpoints, such as deletions, inversions, tandem repeats, and interchromosomal translocations. The workflow of SoftSV is

shown in Figure 10.8. SoftSV produces two lists of the breakpoint candidates: one for relatively large SVs based on discordantly mapping paired-end reads, and another list for small SVs derived directly from the soft-clipped reads. The soft-clip is an unmapped subsequence at the 5′ and/or 3′ end(s) of a read, which is determined by the low base qualities, such as the number of mismatch or gap frequency exceeding the aligner's thresholds or this read sequenced and mapped across an SV breakpoint. Large SVs are characterized by discordantly mapped paired-end reads, which can be mapped to the reference genome under certain chromosomal changing events. For example, a deletion event happens when the insert size of a paired-end mapping is larger than the expected value, that is, greater than the mean insert size plus a specified tolerance; a tandem repeat happens when the paired-end reads have a reversed mapping order at two duplicate DNA segments. The size of SVs

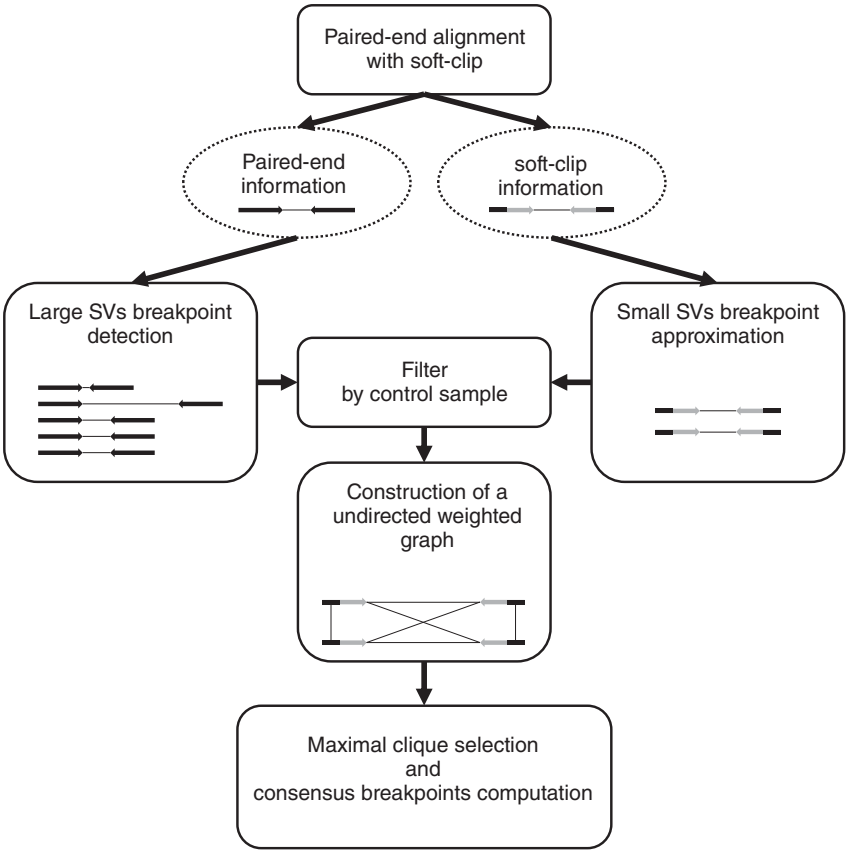


Figure 10.8 The workflow of SoftSV. SoftSV uses the paired-end reads and the soft-clip information to identify both small and large indels, inversions, tandem repeats, and translocations.

identified by using the nearly perfect match of paired-end reads is limited by the insert size and the read length, which makes short SVs with length less than the read length impossible to be detected. To identify SVs below the read length limit, SoftSV applies the start of every soft-clipped sequences. To localize the SV precisely, SoftSV needs two groups of soft-clip sequences: one located at the upstream breakpoint and the other located at the downstream breakpoint. The soft-clipped sequences at the upstream breakpoint will match the non-soft-clipped sequences located at the downstream breakpoint. With the soft-clip information, softSV builds a weighted, undirected graph where each vertex represents a soft-clip sequence, and each edge represents a match between the soft-clipped and the non-soft-clipped sequence based on the Smith–Waterman alignment between them. For every SV with at least one soft-clip sequence at both breakpoints, a maximal clique can be constructed in the graph. The edges between the soft-clip sequences at the same breakpoints are weighted zero, while edges matching with the soft-clip sequences between different breakpoints are weighted one. The SoftSV sums all the weights of the edges belonging to the same clique and assigns the sum to that clique. The clique with the highest positive value is selected as a candidate SV. The consensus interval of an SV is computed as the median of the positions of all soft-clip sequence at two breakpoints, respectively. The experimental results show that SoftSV is very efficient for data with sequencing depths falling in the range between 5× and 25× and read lengths ranging from 75 to 150 bp.

10.4 EVALUATION METHODOLOGY

Analytical requirements are the major factor for the downstream analyses when we conduct the evaluation for the variant calling. For example, in phylogenetic inferences, a small number of inaccurate variant calls may not impact the analyses of samples when there are a great number of variants. However, the more accurate results from variant callers will give us better and more detailed interpretation on the genetic modification. Before the actual evaluation, two general principles may guide us to do the evaluation: (i) The selection of benchmarking samples and reference genomes is critical to assess the variant calling process. The selected data sets must be credibly accurate and should represent the characteristics of the samples used in the studies. (ii) Multiple metrics, such as TP and FP, negative call rates, and receiver operating characteristic (ROC) curves, should be utilized to evaluate the performance of variant calling. In the rest of section, a list of metrics for performance measurement, simulation tools, real benchmark datasets, and databases are introduced to the interested readers to test and compare various variant callers with different task requirements.

10.4.1 Performance Metrics

Most of available performance metrics for variant calling are derived from contingency tables that capture the relationship between the variant labels assigned by the

target method and labels from the ground truth. There are four basic combination labels for the reported variants: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). We use TP, TN, FP, and FN to denote the numbers of variants falling into the corresponding categories. Here, we list the definitions and meanings for seven commonly used metrics:

- Accuracy
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{PN}}. \tag{10.1}$$
It is the ratio of the correct calls to the total number of positions.
- Specificity
$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{10.2}$$
It is the ratio of the nonvariant calls to the total number of nonvariant positions.
- Sensitivity
$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{10.3}$$
It is the ratio of the variant calls to the total number of variant positions. It is also known as recall, true-positive rate (TPR), or positive call rate.
- Precision
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{10.4}$$
It is the ratio of the correct variant calls to the total number of variant calls. It is also known as positive predictive value (PPV).
- $F1$-score
$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}. \tag{10.5}$$
It is interpreted as a weighted average of the precision and sensitivity, where 1 is the best score and 0 otherwise.
- ROC
It is a graphical plot created by plotting the TPR against the false-positive rate at multiple threshold. It demonstrates the performance of a binary classifier system when we alter its discrimination threshold.
- Jaccard index
$$J(A, B) = \frac{|\, A \cap B \,|}{|\, A \cup B \,|}, 0 \leq J(A, B) \leq 1. \tag{10.6}$$
It is also known as Jaccard similarity coefficient, and A and B are two finite sets of the variant calling results. Jaccard index is a measurement of the similarity between two predicted variant sets. When two sets are the same, the Jaccard index is 1; when two sets are completely different, the Jaccard index is 0.

A variety of tools is summarized in Table 10.1 for the evaluation of the performance of variant calls when the benchmarked variant calls are provided. Most

TABLE 10.1 Tools for Comparing Variant Calls

Name	Descriptions and Features	References
Bcbio.variation	Report sensitivity, specificity, and genotyping error rate by comparing to a benchmark call set	https://github.com/chapmanb/bcbio.variation
SMaSH	Report precision, recall, and uncertainty of precision and recall	https://github.com/ekg/vcflib
USeq VcfComparator	Report ROC curves	http://useq.sourceforge.net/
GATK	Combine and compare variant and genotype calls	https://github.com/broadgsa/gatk-protected/
RTGtools	Report ROC curves, TP, FP, and FN variants	http://realtimegenomics.com/products/rtg-tools/
Hap	Compare VCF files by specified haplotype and report a graph-based reference	http://github.com/sequencing/hap.py
Vgraph	Report a graph representation of genomic variants	http://github.com/bioinformed/vgraph

of them is capable of comparing the generated variant call format (VCF) files to the benchmark call sets. The VCF is a text file format only storing gene sequence variations along with the reference genome. It has been developed by many large-scale genotyping projects, such as the 1000 Genomes Project. Some of the comparison tools can provide the graph representations, such as the ROC curve, to show the performance of variant calling intuitively.

10.4.2 Simulated Sequence Data

When the reference data set is not available, or the reference data set cannot fully represent the features of the target study or the aimed variant caller, simulated sequencing data sets are the ideal ways to validate the robustness of the variant callers. An obvious advantage of using simulated data for the evaluation is that the known variants can be embedded into the target genomes. Numerous sequencing read simulators have been developed, and we list some popular ones in Table 10.2. When simulating short reads, these simulators basically use two error models, that is, the empirical and the theoretical models. Some simulators also use the base quality score (BQS) to simulate the sequence data sets.

Empirical model-based read simulator algorithms use sequence-specific errors, either uniform or position-specific error rates from the sequence data sets; while the theoretical error models are based on the sequencing chemistry, the sample preparation, and the reaction detection. For example, ART simulates the Illumina reads using an empirical quality profile, which determines quality value first, and then utilizes the quality to generate substitution error randomly. MetaSim is specially designed for simulating the metagenomic data, and it supports the empirical error profile to

TABLE 10.2 The Simulator Algorithms for the Next-Generation Sequencing

Name	Feature	References
ART	Empirically based with BQS generated for 454, Illumina, and SOLiD	[265] http://www.niehs.nih.gov/research/resources/software/art
flowsim	Empirically based with BQS generated for 454	[266] http://blog.malde.org/index.php/flowsim/
FASTQSim	Empirically based with BQS generated for 454, Illumina, and SOLiD	[267] https://sourceforge.net/projects/fastqsim
GemSIM	Empirically based with BQS generated for 454 and Illumina	[268] http://sourceforge.net/projects/gemsim/
Grinder	Theoretically based with BQS generated for 454 and Illumina	[269] http://sourceforge.net/projects/biogrinder/
pIRS	Empirically based with BQS generated for Illumina	[270] ftp://ftp.genomics.org.cn/pub/pIRS/
RSVSim	Empirically based with BQS generated for 454, Illumina, and SOLiD	[271] http://www.bioconductor.org
SInC	Empirically based with BQS generated for Illumina	[272] http://sourceforge.net/projects/sincsimulator
XS	Theoretically based with BQS generated for 454, Illumina, and SOLiD	[273] http://bioinformatics.ua.pt/software/xs/

generate reads without quality values. PIRS adopts an empirical Base-calling profile that is similar to the real Illumina data to simulate the Illumina pair-end reads. RSVSim comprises multiple functions to estimate the distribution of structural variation sizes from real data sets to simulate five types of SVs: deletions, insertions, inversions, tandem repeats, and translocations. Flowsim simulates the advanced error modeling and quality scores, although it is only able to generate Roche-454 sequencing data. SInC is capable of simulating SNPs, indels, and CNVs taking into account a distribution of BQS from Illumina instrument.

10.4.3 Real Sequence Data

Real sequence data is the ideal source of sequence data when we evaluate the variant calling results. The overlap between the report variants by the variant callers and the published variants in the databases is a fair indicator of the calling performance. Here, we list the resources to obtain the real sequence data sets.

- 1000 Genomes Project (http://www.1000genomes.org/). It is an international research effort to provide the sequence data associated with individuals and human genetic variation calls in the VCF format. In October 2012, 1092 genomes sequence data was announced in the Nature publication.
- The Cancer Genome Atlas (TCGA)(http://cancergenome.nih.gov/). It is a project begun in 2005. The data generated by the experiments are available at

the TCGA Data Portal. It offers the access to the cancer data up to 36 different types. The files including the binary version read alignments of the short DNA sequence.

- International Cancer Genome Consortium (ICGC) (https://dcc.icgc.org/). The ICGC was launched in 2008 to coordinate the studies of more than 50 different cancer types from 25,000 patient genomes.
- Sequence Read Archive (http://www.ncbi.nlm.nih.gov/sra/). It is also known as short read archive (SRA). This database provides a public repository for DNA sequence data, particularly the "short reads" generated by high-throughput sequencing, which are typically less than 1000 bp in length.

To verify the variant calls reported by variant callers, we need to compare the predicted variants to the published variants in the databases. Here, we list the resources storing the variants for human, animals, plants, and microbes.

- International HapMap Project (http://hapmap.ncbi.nlm.nih.gov/). It is proposed to develop a haplotype map of the human genome and to describe the common patterns of human genetic variation from four populations, including Yoruba trios from Ibadan, Utah residents of northern and western European ancestry, Japanese individuals, and Han Chinese individuals.
- dbSNP (http://www.ncbi.nlm.nih.gov/SNP/). It is a free public archive for genetic variation within and across various species. It is developed and hosted by the National Center for Biotechnology Information (NCBI) with the collaboration from the National Human Genome Research Institute (NHGRI). It contains a range of molecular variation, such as SNPs, short indels, and short tandem repeats. dbSNP has gathered over 184 million submissions, representing more than 64 million distinct variants for 55 organisms.
- Database of Genomic Variants (DGV; http://projects.tcag.ca/variation/). It is an online resource, which contains a curated catalog of structural variation data from the peer-reviewed scientific studies. Common types of structural variation include small indels, large CNVs, and inversions.
- COSMIC cancer database (http://www.sanger.ac.uk/cosmic). It is a modern collection of the somatic mutations in cancer. The release of COSMIC of version 71 contains about 2.1 million unique somatic variants. Currently, it contains 547 genes for which mutations have been causally implicated in cancers.
- Cancer3D database (http://cancer3d.org/). It collects about 275,000 somatic mutations mapped to ∼15,000 proteins. The Cancer3D database covers predictions from e-Drug and e-Driver, two recently developed tools for predicting sensitivity to cancer driver proteins and 24 anticancer compounds [274].

10.5 CONCLUSION AND FUTURE WORK

In this chapter, we introduced an important bioinformatics application, variant detection, based on the information extracted from the MSA by aligning similar

DNA sequences, including short sequencing reads and reference genome sequences. In the beginning, we briefly described four major genetic variant types. And then we used six variant detection tools, containing MUMer, Mugsy, ParSNP, MAQ, Gustaf, and SoftSV, as examples to illustrate the primary ideas about how to employ MSA to infer genetic variants. In the end, we presented the evaluation methodologies, including measurement metrics, available simulated and real test data sets, for assessing the performance of variant callers. Regarding the input types and calling strategies, variant detection methods can be grouped into three categories: read mapping methods, *k*-mer matching methods, and genome alignment methods. Current strategies using read mapping to identify genetic variants include *de novo* reads assembling, read splitting, coverage depth, and insert size inconsistencies analyses. Although these methods have exhibited efficient and accurate power on the variant calling, they have their strengths and weaknesses. For example, the *de novo* assembly provides the best opportunity to find variants accurately, assembling short reads in an enormous amount is a computationally challenging problem that requires significant resources. Read-split approaches achieve a high degree of accuracy for variants with short and medium sizes, but the performance drops significantly as the size of the variations increases. Coverage depth methods often fail in calling small indels due to the inaccurate breakpoint prediction. The mapping for read splitting and analyses on depth coverage requires the short sequencing read data, which is even sometimes not available or may be enormously larger than the genomes themselves and become a computational cost monster. Mapping accuracy can also be messed up due to unexpected reasons, such as contaminant, repetitive sequence, and misalign low complexity. Therefore, we still need more sophisticated inference models and more accurate and longer sequenced reads to make better variant calls.

Another challenge in variant calling development is the lack of gold standard samples or controls to benchmark calling results, especially for the somatic CNV detection [275]. Although some NGS-based benchmarks, including in silico simulated data or variant calls on the same set of tumor samples, have been applied to evaluate the published somatic CNV detection programs, the generality of these benchmarks is not clear yet. Several simulators, such as ART, GemSIM, pIRS, and RSVSim, have been proposed to simulate NGS data with consideration of reproducing known biases from experimental platform-dependent errors and sequence context, we still need a comprehensive genome simulator that is able to capture all the features of the genome, especially for the tumor genome. In the meantime, array-based platforms and calling algorithms still lack reproducibility and concordance based on the recent assessment. Without a well-controlled and well-annotated reference benchmark, it will be hard to understand the advantages and disadvantages associated with each approach. As a result, the choice of NGS-based variant callers now relies more on theoretical factors such as the technique descriptions rather than their performances on a standard benchmark. Therefore, it is necessary and urgent for the community to lay down better data sets as a benchmark, which comprehend the complexity of human genomes, for the performance estimation and the improvement development.

11

MULTIPLE SEQUENCE ALIGNMENT FOR STRUCTURE DETECTION

11.1 INTRODUCTION

In recent years, the availability of enormous volume of biological data has opened a new direction for genomic analyses and structural prediction of DNA, RNA, and proteins. Structural bioinformatics is one of the primary research areas in the field of computational biology. It concerns the analysis and prediction of 2D and 3D structures of biological macromolecules, such as DNA, RNA, and proteins. Computational predictions of RNA and protein structures have been a long-standing challenge in molecular biology for more than 40 years. The purpose of this chapter is to describe MSA-based methods for computationally predicting the secondary structure (SS) of RNA and protein. We intend to provide some insight into the field of SS prediction for the community who wish to develop new tools in their research.

In the last few decades, determining the RNA structure has gained significant attentions from the researchers, as it is one of the critical issues in understanding the genetic diseases and creating new drugs. The functional role of RNA expands widely in the cellular regulatory functions such as regulating transcription [276], gene silencing [277], and genome maintenance [278]. Functional RNA families usually have common SSs because the function of these molecules is inherently tied to their secondary structures. SSs also help the biologists to better understand the roles of the RNA and other small molecules in the cell. There are different types of RNA(s) including messenger RNA (mRNA), transfer RNA (tRNA), ribosomal RNA, viral

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

RNA, and signal recognition particle RNA (SRP RNA) [279]. In a cell, the functions of different RNAs diverge widely from each other. For instance, all organisms use the mRNA in protein synthesis; some viruses use RNA as their genetic material instead of DNA; incompletely processed mRNAs involve small nuclear RNAs (snRNAs) to become mature mRNAs. As the close connection between the RNA structure and its function, we can have more precious understanding of RNA functions with more accurate prediction of its structure. To computationally predict RNA SSs accurately is an important and difficult task. Both crystal and experimental structures confirmed that RNA SSs can be predicted accurately using merely primary sequence data since the structure of RNA is usually more evolutionarily conserved than primary sequence [280].

Due to the similar relation between the RNA function and its structure, accurate protein SS is also essential for almost all theoretical and experimental studies on protein. Genome sequencing makes a tremendous amount of amino acid sequence data available, while the corresponding structural information is much harder to obtain experimentally. The gap between the number of known protein amino acid sequences and the number of known structures in the Protein Data Bank (PDB) [281] constantly and rapidly increases. With the completion of many large-scale genome sequencing projects, the rapidly growing protein sequences require more and more structure predictions. Understanding protein structure is critical for analyzing protein function and applications such as drug design. Understanding complex dependency between protein sequence and structure is an another greatest challenge in computational biology [282]. It is widely accepted that the amino acid sequence contains sufficient information to determine the 3D structure of a protein, but it is extremely hard to predict 3D protein structure based on the primary sequence. Although the prediction of the 3D structure is one of the ultimate goals of protein science, the prediction of protein SS from sequences remains a more feasible intermediate step in this direction. Protein SS prediction is not as much difficult as 3D structure prediction, and it offers valuable information for further elucidating the 3D structure of the protein, which reduces the complex 3D problem to a much simpler 1D problem. The protein SSs define the key structural elements of the protein and the location of these elements in the protein amino acid sequence. In this chapter, we start with the definitions of essential elements in RNA and protein SSs. Next, we use some recently developed SS prediction approaches as examples to illustrate the fundamental ideas about how to use conserved information inferred from MSA as the guidance to predict SSs. We also discuss the measurements, including performance metrics and benchmark data sets, to qualify the predicted SSs. In the end, we summarize these two MSA-based applications and provide possible future directions.

11.2 RNA SECONDARY STRUCTURE PREDICTION BASED ON MSA

In RNA, the primary sequence determines the SS, which, in turn, determines its tertiary folding. RNA SSs are formed by the Watson-Crick base pairs A$\cdots$U and G$\cdots$C as well as G$\cdots$U pairs. The secondary structural elements interacting

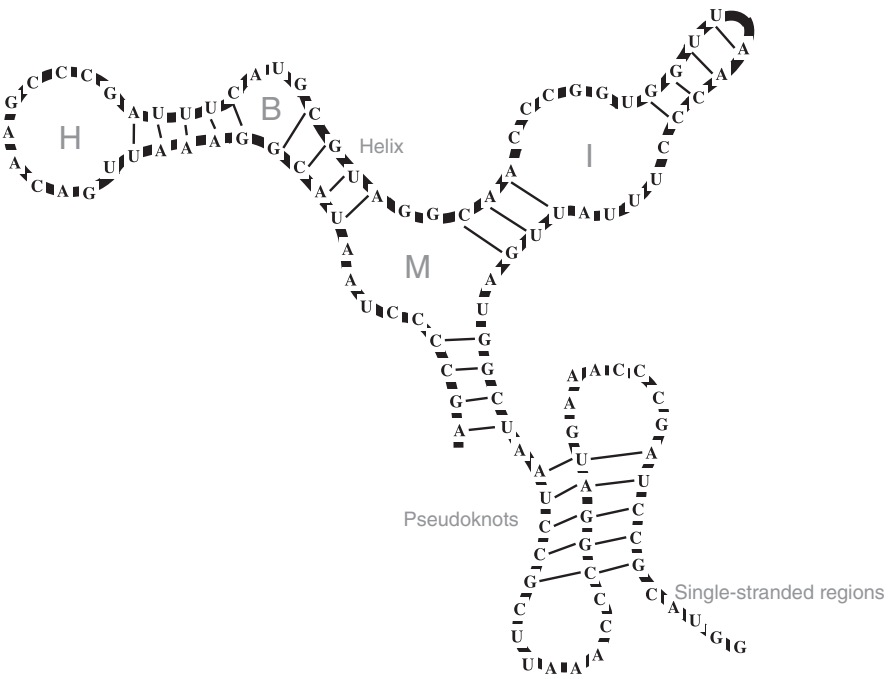


Figure 11.1 Seven RNA SS elements. Hairpin (H) loop, internal (I) loop, bulges (B) loop, multihelix (M) loop, single-stranded region, helix stem, and pseudoknot are shown in the 2D representations.

between themselves through the formation of hydrogen bonds and van der Waals interactions fold in a 3D space to the tertiary structure with the biologically active conformation. The folding interactions can take place between two helices, two unpaired regions, or one unpaired region and a double-stranded helix. Before the discussion of RNA SS predictions, it is noteworthy to have a rough description of seven well-recognized RNA secondary structural elements as shown in Figure 11.1.

- Helix stem, where base pairs are contiguously stacked;
- Single-stranded regions, where all the bases are not paired with any other bases in the RNA structure;
- Hairpin loop, where a loop locate at the end of a stem;
- Bulge loop, formed by the single-stranded bases occurring on only one side of a stem;
- Internal loop, where the single-stranded bases interrupt the both sides of a stem;
- Multibranched loop, formed by three or more stems;
- Pseudoknots, where unpaired bases of one substructure bind to unpaired bases of another substructure to form a stem.

Although the tertiary structure is the level of organization relevant for the biological function of structured RNA molecules and sometimes SSs are influenced by tertiary structures [283], the interactions that are responsible for RNA tertiary structure formation are significantly weaker than those responsible for SS formation [284]. Hence, in recent investigations, for computational simplicity, it is assumed that the influence of hydrogen bonds within the tertiary structure, on hydrogen bonds within SS, is negligible; consequentially, SSs can be predicted independently of tertiary structures [285, 286]. In this section, we first briefly introduce the information types available giving the multialigned homologous RNA sequences. Then we review three classical RNA SS prediction methods applying three different orders of folding and aligning during the prediction process. At last, we provide the standard metrics and resources to assess the performance of SS prediction results.

11.2.1 Common Information in Multiple Aligned RNA Sequences

Several types of information extracted from multialigned RNA sequence are utilized in the RNA sequence folding and aligning, and the RNA SS prediction tools and algorithms. We briefly summarized this information into three categories: mutual information, covariation information, and phylogenetic information.

11.2.1.1 Mutual Information It is well known that the mutual information can be applied to detect compensatory mutations in an alignment [287, 288]. Given the *i*th and *j*th columns in an alignment, the mutual information is defined as

$$M_{ij} = \sum_{X,Y} f_{ij}(XY)\, log\, \frac{f_{ij}(XY)}{f_i(X) f_j(Y)}, \tag{11.1}$$

where X, Y consist of the four possible bases at the positions i and j, $f_i(X)$ is the frequency of base X, $f_{ij}(XY)$ is the joint frequency of base module of X and Y. We can employ an upper triangular matrix to present the complete set of mutual information. Once the mutual information is complete, an SS can be inferred by the prediction method using a dynamic programming algorithm [289], which maximizes the sum of mutual information taken over all base pairs.

11.2.1.2 Covariation Information Because SSs of RNAs are related to their functions, mutations that preserve base pairs often occur during the evolution, especially in the RNA stem parts. The algorithm, RNAaliFold [290], which will be introduced in next section, uses the information of covariation in combination with the thermodynamic stability of common SSs. By given an input alignment, the covariation information (CI) can be obtained by evaluating the averaged number of compensatory mutations for a pair of columns. For the *i*th and *j*th columns in the alignment, the averaged number is defined as

$$C_{ij} = \frac{2}{N(N-1)} \sum_{\alpha,\beta \in A} d_{ij}^{\alpha,\beta} \theta_{ij}^{\alpha} \theta_{ij}^{\beta}, \tag{11.2}$$

where N is the number of sequences in the alignment, α, β are two RNA sequences, $\theta_{ij}^{\alpha} = 1$ if α at ith and jth positions form a base pair and $\theta_{ij}^{\alpha} = 0$ otherwise, and $d_{ij}^{\alpha,\beta} = 2 - \delta(\alpha_i, \beta_i) - \delta(\alpha_j, \beta_j)$. δ is the delta function: $\delta(a, b) = 1$ only if $a = b$, and $\delta(a, b) = 0$ otherwise.

11.2.1.3 Phylogenetic Information Phylogenetic information (PI), also known as evolutionary information, is useful for predicting RNA SSs because most SSs of functional RNAs are conserved in evolution. Several tools have employed PI in the RNA SS prediction. For example, Pfold [291] is the first tool to incorporate PI into a probabilistic model for predicting RNA SSs. A brief view of the Pfold's model can be written as

$$p^{Pfold}(\theta|A) = \frac{p(A|\theta, T)p(\theta|M)}{p(A|M, T)}, \tag{11.3}$$

where T is a phylogenetic tree, A is the alignment, M is a prior model for SSs, such as the McCaskill's model, and θ is the base-pair indicator.

11.2.2 Review of RNA SS Prediction Methods

Two main information types are used for RNA SS prediction: evolutionary information and thermodynamic information. Thus, the prediction algorithms can be categorized into two groups: comparative methods and free energy methods. The former ones make predictions use the sequence families, and the latter ones use only a single sequence to make prediction. The main assumption of comparative methods is the conserved patterns of base pairs, that is, pairs will vary at the same time during evolution and maintaining structural integrity. The comparison between the predicted results from experiments and comparative methods estimates that nearly 98% of pairings in the current comparative models will be in the crystal structures. Thus, comparative sequence analysis is able to predict almost all of the SS base pairs and even some tertiary pairs in crystal structures. Different from the comparative methods, free energy methods, also known as single sequence approaches, use one sequence to determine the structure of complementary regions that are energetically stable. The working hypothesis of free energy method is that the native SS is the one with the minimum free energy. However, the computation of free energy for a conformation is not accurate due to many factors, such as the assumption of free energy additivity, the absence of measured thermodynamic parameters of noncanonical base pairs (like the wobble pairs), and sequence-dependent loops. The single sequence approach is not as accurate as comparative approaches because the information gathered from multiple sequences would be missing if only from the single sequence [292]. In this section, we focus our topic on the comparative methods using MSA. The RNA SS prediction problem is defined as follows:

Problem (RNA SS Prediction Based on Multiple Homologous Sequences) *Given a collection of homologous sequences and a query sequence α, predict an SS y of α.*

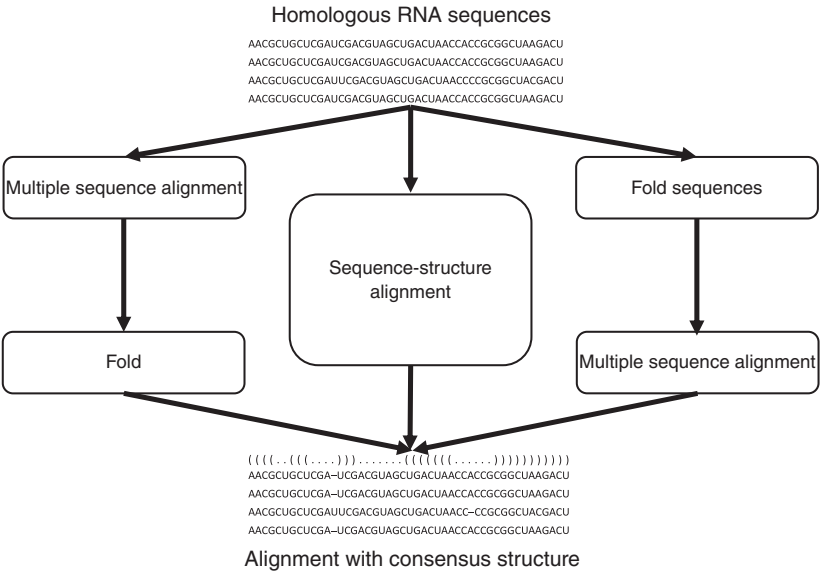


Figure 11.2 The workflows to predict RNA SS with the alignment and folding in three distinct orders.

The comparative sequence analysis assumes that molecules with similar functions and different nucleotide sequences will form similar structures. Based on the different orders to do the aligning and folding processes, Capriotii et al. [293] classified the computational approaches for predicting RNA SS into three categories: Prior Aligning Ulterior Folding Method, Simultaneously Folding and Aligning Method, and Prior Folding Ulterior Aligning Method. A consistently updated list of RNA SS software can be found in the Wikipedia (http://en.wikipedia.org/wiki/List_of_RNA_structure_prediction_software). Three workflows representing these three strategies are shown in Figure 11.2. In the rest of this section, we concisely summarize the primary ideas behind these three types of prediction tools using three specific algorithms, that is, RNAalifold [290], DAFS [294], and GA-Forest [295], to give readers a whole picture about how MSA can be a help in RNA SS prediction.

11.2.2.1 Prior Aligning Ulterior Folding Method The prior aligning ulterior folding method assumes that structure conservatism is greater than sequence conservatism. Prediction result of this idea strongly depends on the alignment effectiveness of multiple sequences. Some well-known algorithms in this categories include RNAalifold, Pfold, and SCFG [296]. Here, we utilize RNAalifold to project the vision of this method.

RNAalifold RNAalifold was first presented to fold the aligned sequences by Zuker's algorithm back in 2002 [290] and is now part of the Vienna RNA packages. It is based on the dynamic programming algorithm, which uses both phylogenetic information

(sequence covariance) and thermodynamic stability of molecules to predict a consensus structure [297]. The goal of RNAalifold is to predict the consensus structure, which is common to the homologous RNA sequences with the same or very similar structures. The idea behind phylogenetic information is that two nucleotides that form a base pair may be changed simultaneously but preserve the propensity to still be a valid base pair. These mutational patterns can help to decide whether RNA forms a functional structure. To quantitatively measure covariance, RNAalifold introduces a scoring matrix, which is derived from frequencies of the aligned and base-paired nucleotides. This scoring matrix attempts to put a high covariance score on the highly conserved base pair. To obtain these score matrices, RNAalifold calculates the log-likelihood scores from the alignments of RNA as follows:

$$R(ab, cd) = log \frac{f(ab, cd)}{f(ac)f(bd)}, \tag{11.4}$$

where $f(ab, cd)$ is the frequency when base pair ab is aligned to base pair cd, and $f(ac)$ is the frequency when base a is aligned with c. These log-likelihood scores are computed for different subsets of the RNA alignments with different conservations, and this scoring matrix is the one that most closely reflects the sequence distance in the alignment. The covariance score γ can be written as

$$\gamma(i, j) = \frac{1}{2} \sum_{\alpha, \beta \in A, \alpha \neq \beta} R(\alpha_i \alpha_j, \beta_i \beta_j), \tag{11.5}$$

where A is the alignment and α, β are two sequences. Using the covariance scores, the recursions to calculate the minimum free energy F on a subsequence from i to j can be derived as follows:

$$F_{i,j} = min(F_{i+1,j}, min_{i<k\leq j} C_{i,k} + F_{k+1,j}),$$

$$C_{i,j} = \theta\gamma(i, j) + min \begin{cases} \sum_{\alpha \in A} H(i, j, \alpha) \\ min_{i<k<l<j} \left(\sum_{\alpha \in A} I(ij, kl, \alpha) \right) \\ min_{i<k<j} (M_{i,k} + M^1_{k+1,j} + \lambda), \end{cases}$$

$$M_{i,j} = min \begin{cases} M_{i+1,j} + \delta \\ min_{i<k<j} C_{i,k} + M_{k+1,} + \eta \\ M^1_{i,j}, \end{cases}$$

$$M^1_{i,j} = min\,(M^1_{i,j-1} + \delta, C_{i,k}),$$

where F, C, M, and M^1 are four matrices corresponding to four structural components, that is, unconstrained structures, constrained structures, multiloop structures,

and multiloop with one branch, respectively. The covariance score λ is involved in the calculation of matrix C. $H(i, j, \alpha)$ is the free energy of a hairpin between bases i and j, and $I(ij, kl, \alpha)$ represents the free energy of an interior loop between base pairs i, j and k, l.

To use RNAalifold, a multiple alignment of several homologous sequences is required. Base pairs are only evaluated if γ reaches a specific threshold. This reduces the number of possible base pairs to be evaluated, which makes the speed of applying RNAalifold on multiple sequences as faster as on a single sequence.

11.2.2.2 Simultaneously Folding and Aligning Method This class of approaches simultaneously carries out sequence alignment and structure prediction and simultaneously obtains an alignment and consistent structure. In 1985, Sankoff designed the first approach using a dynamic programming algorithm to obtain a list of base groups and the maximum sum of base group weight to search the structural conformational space [298]. Sankoff's algorithm combines sequence alignment and Nussinov dynamic programming folding method, which is computationally expensive. Various algorithms have been developed to reduce the computational complexity of the Sankoff algorithm. For example, sampling-based approaches, instead of reducing the search space, have been developed [299, 300]. Some other methods use precomputed posterior probabilities of base pairs and/or aligned columns to reduce the search space of the Sankoff algorithm [301, 302]. We use a recently proposed method, named DAFS [294], to illustrate an efficient way to fold RNA sequences.

DAFS Sato et al. [294] developed a novel algorithm, named DAFS, which aligns and folds RNA sequences simultaneously by maximizing the expected accuracy for predicting the common SS and their alignments. Given a set of unaligned RNA sequences, RNA structural alignment aims to produce a reliable alignment, as well as to predict the reliable common SS. DAFS is following the maximizing expected accuracy (MEA) principle, which can be described as

$$\begin{aligned}\hat{y} &= \arg\max_{y \in Y} E_{\theta|D}[G(\theta, y)] \\ &= \arg\max_{y \in Y} \sum_{\theta \in Y} G(\theta, y) p(\theta|D),\end{aligned}$$

where D is the observation RNA sequence, θ is the corrected base pairs, y is the predicted base pairs, $p(\theta|D)$ is the base-pairing probability distribution, which can be calculated by using McCaskill model [303], and $G(\theta, y)$ is called the gain function. The goal is to maximize expected accuracy if the gain function measures the accuracy. Usually, the choice of the gain function will decide the complexity and precision of the methods. The gain function of DAFS is designed for a pairwise structural alignment. The objective function serves as the gain function in MEA, which is calculated as the expectation of the sum of the numbers of correctly predicted base pairs in a common SS and properly pairwise aligned columns. Even with the pseudoknot-free assumption, the time complexity to obtain the expected gain function requires $O(|a|^3|b|^3)$, where a and b are two RNA sequences and $|\bullet|$ denotes

the length of a sequence. To speed up the computation, DAFS uses the independence assumption in structure and alignment to approximate the original expected gain function fast, which reduces the time complexity to $O(|a|^3)$. The dual decomposition technique is used to maximize the objective function under several constraints, such as each base can be only paired with at most one base. The objective function can be decomposed into two separate SS prediction problems and becomes the pairwise alignment problem. The algorithm performs the Nussinov-style SS prediction [304] and the Needleman–Wunsch-style pairwise alignment [19] with a penalized scoring function to maintains and update the consistency of the pairwise structural alignment iteratively. To take advantages of multiple sequence alignment, DAFS can be extended by applying the standard progressive alignment technique. With the alignment information embedded into the gain function and the computational approximation, DAFS is an efficient implementation that simultaneously aligns and folds RNA sequences.

11.2.2.3 Prior Folding Ulterior Aligning Method This approach first folds the RNA sequences using the single sequence SS prediction methods and then aligns the predicted structures. It requires to obtain a large number of suboptimal structures, and this type of algorithm is very suitable when the conserved area of a sequence cannot be observed easily from the primary sequence. The common way to align the SSs is to use tree presentation [305], such as RNAforester [306] and MARNA [307] programs. In this part, we take GA-Forest [295] as an example to elucidate how to fold and align to obtain the RNA SS.

GA-Forest Lin et al. [295] developed a permutation-based genetic algorithm, which combines the minimum free energy model with the similarity information from the homologous sequences to improve the RNA SS prediction. We name their method as GA-Forest, which has three major steps: preprocessing, GA, and postprocessing. Since the genetic algorithm starts with the permutation, in the preprocessing, GA-Forest finds out all the possible stems from the RNA primary sequences and orders the stems as the permutations. Every permutation of stems represents an SS. Each permutation has the position and length information obtained by comparing to the reference. In the second step, GA-Forest iterates crossover, mutation, and selection using the free energy as the fitness function until reaching the structural stability criteria. The mutation operation is implemented by randomly switching positions of two stems. In the postprocessing, GA-Forest tries to find the most similar structures. In order to compute the similarity for RNA SSs, each SS generated by GA is represented by only one Hochsmann's forest [308]. After computing the similarity score of these forests and the reference or consensus structure, GA-Forest selects the structure, which has the highest similarity score and output it as the predicted SS.

11.2.3 Measures of Quality of RNA SS Prediction

The RNA SS prediction methods reviewed here accept either multialigned sequences or a collection of homologous sequences, and then output the SSs shared by these

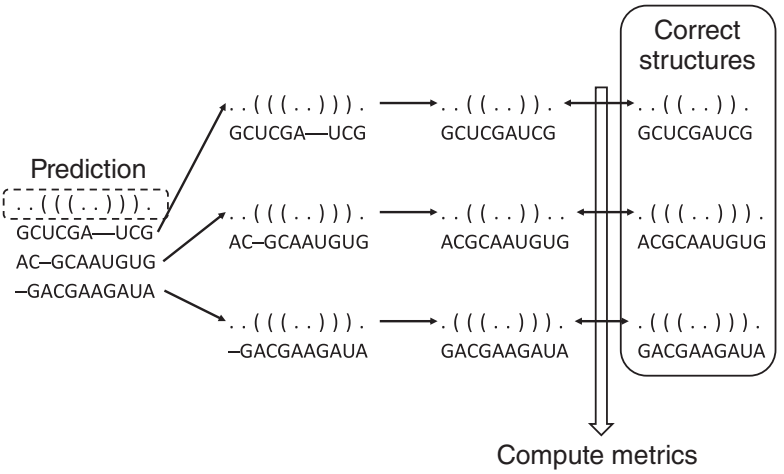


Figure 11.3 Generate the SS for each RNA sequence from the multialigned sequences.

pack of sequences. To generate the SS for a single query sequence, we need to remove the base pairs with gaps, as shown in Figure 11.3. To assess the performance of models and algorithms for RNA SS, we have to conduct a series of cross-validation experiments using known SSs taken from RNA structure databases. In the next section, we will present three popular metrics and list publicly available data sets as benchmarks.

11.2.3.1 Performance Metrics The three common ways to measure the rank of RNA prediction methods are to compute sensitivity, positive predictive value (PPV), and Matthews correlation coefficient (MCC). The calculation of these three metrics requires to classify base pairs into the following categories: TP (true positives), correctly predicted base pairs; FP (false positives), base pairs that do not exist in the reference structure; TN (true negatives), correctly predicted unpaired bases; and FN (false negatives), base pairs in the reference SS, but not in the predicted one. With the above numbers, the sensitivity value is defined as

$$Sensitivity = \frac{\text{TP}}{\text{TP}} + \text{FN} \tag{11.6}$$

Given a ϵ, the number of compatible false-positive base pairs, the formula to calculate PPV is written as

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + (\text{FP} - \epsilon)} \tag{11.7}$$

The MCC is formulated as

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}. \tag{11.8}$$

It has been shown that the MCC can be approximated by the geometric mean of sensitivity and PPV [309]. In cases when the denominator in the MCC formula equals 0, the MCC is arbitrarily set to 0, which reflects the method's inability to generate a prediction (i.e., MCC = 0 means that the prediction is no better than random).

11.2.3.2 Benchmark Database Resources We usually use the following four real databases as benchmarks when performing the evaluation of RNA SS prediction:

- PDB database [310]. The PDB is an important resource in the field of structural biology, such as structural genomics. The data, typically obtained by the X-ray diffraction, NMR spectroscopy, or electron microscopy, are freely accessible on the Internet through the websites of its member organizations, including PDBe (http://www.pdbe.org/), PDBj (http://www.pdbj.org/), RCSB (http://www.rcsb.org/pdb), and wwPDB (http://www.wwpdb.org/).
- RNA STRAND database [311]. The RNA SS and statistical ANalysis Database (RNA STRAND) is a curated database containing known SSs of any type and organism. It provides a wide collection of known RNA SSs drawn from public databases, and the data there are searchable and downloadable. It is publicly available at http://www.rnasoft.ca/strand.
- Rfam database [312]. Rfam is a database holding information about noncoding RNA (ncRNA) families and other structural RNA elements. Rfam is an annotated and open access database initially developed at the Wellcome Trust Sanger Institute collaborated with Janelia Farm, and currently it is hosted at the European Bioinformatics Institute. Rfam is designed to be similar to the Pfam database for annotating protein families. It is publicly available at http://rfam.xfam.org/.
- BRAliBase [313]. BRAliBase series contain four different benchmark data sets: BRAliBase I, BRAliBase II, BRAliBase 2.1, and BRAliBase III. BRAliBase I is used for the comparison of comparative RNA structure prediction approaches. BRAliBase II provides a benchmark of multiple sequence alignment programs on the structural RNAs. BRAliBase 2.1 provides a benchmark of multiple sequence alignment programs on the structural RNAs; BRAliBase III is for the assessment of the performance of homology search methods on noncoding RNA. All data sets are publicly available at http://projects.binf.ku.dk/pgardner/bralibase/.

11.3 PROTEIN SECONDARY STRUCTURE PREDICTION BASED ON MSA

The protein SS is defined by the patterns of hydrogen bonds between the backbone carbonyl (C=O) and N–H groups. In 1983, based mostly on the location of hydrogen bonds, Kabsch and Sander developed the Dictionary of Secondary Structure Assignments (DSSP) [314], which classifies protein SSs into eight types:

α-helix (H), extended β-strand (E), 3_{10} helix (G), π-helix (I), isolated β-bridge (B), hydrogen-bonded turn (T), bend (S), and any other structure (_). These eight types are usually reduced to three SS states: helix (H, G, and I), β-strand (E and B) and coil (T, S, and _). The protein SS prediction is proposed to be an intermediate step before the tertiary structure prediction when the homologous 3D structures are not available in the PDB. The first attempt to predict the protein SS started in the 1970s with the works of Chou and Fasman [315], which was based on a single sequence and gave the cross-validated accuracies below 60%. In the last 15 years, many methods have significantly improved the overall SS prediction using multiple sequence alignments, which contain the evolutionary information about protein structures. The main reason that information from the MSA improves the prediction accuracy is because the protein structure is more conserved than sequence during evolution, which consequently leads to the conservation of the long-range information. One may suppose that part of this long-range information is revealed by multiple alignments. Many proteins have a similar structure when sharing the sequence identities as low as 20%. Protein function is more vital for evolutionary survival than the primary sequence conservation, so random mutations to the sequence that impair its function usually let the mutated sequence be eliminated during evolution and hence do not exist [316]. In the rest of this section, we first briefly review the techniques used in protein SS prediction with the help of MSA, and then we provide available resources and empirical evaluation metrics to perform the assessment of protein SS prediction tools.

The goal of protein SS prediction is to predict SS labels correctly for each amino acid residue for any given protein sequence. An obvious difference between RNA SS prediction and protein SS prediction is that RNA folding is mainly dependent on the base pairs while protein is heavily dependent on the hydrogen bonds between the main-chain peptide groups. Some early studies concluded that the SS type of the residue tends to have the same type of SS carried by the nearby amino acid residues. This statement helps many predicting tools take advantage of MSA, which is created by using a query protein sequence against the protein database. In this section, the protein SS prediction problem using MSA information is as follows:

Problem (Protein SS Prediction Based on a Set of Aligned Sequences)

Given a collection of aligned protein sequences and a query protein with the primary sequence a, predict the protein SS y for a.

11.3.1 Review of Protein Secondary Structure Prediction Methods

Many modern methods for protein SS prediction have been extensively studied using machine learning approaches, such as support vector machines, hidden Markov model, Bayesian networks, and artificial neural networks. Support vector machines are associated with learning algorithms, such as classification and regression analysis, in its construction of a hyperplane. Hidden Markov model uses a probabilistic model and is considered as the simplest dynamic Bayesian

network. Bayesian network is a probabilistic graphical model that represents a set of random variables with their conditional dependencies. An artificial neural network models the brain-receiving inputs that are processed with the influence by weights to become outputs. A consistently updated list of protein SS software can be found in the Wikipedia (https://en.wikipedia.org/wiki/List_of_protein_secondary_structure_prediction_programs). Early methods for the prediction of protein SS utilized the propensities from a single amino acid, that is, certain residues tend to be in a particular SS state than other residues [317]. Recently, machine learning techniques are trained with sequence profiles or SS information of resolved structures to achieve better prediction accuracy. Given a query protein sequence, the most common way to collect the sequence profiles is using the position-specific score matrix (PSSM), which is generated by PSI-BLAST [318]. In the following section, we first introduce how the PSI-BLAST generate the PSSM, and then we discuss three recently proposed approaches to illustrate the strategies about how to incorporate the PSSM information into the protein SS prediction.

11.3.1.1 PSI-BLAST Altschul et al. [318] proposed the Position-Specific Iterated BLAST (PSI-BLAST) program for automatically combining statistically significant alignments produced by BLAST into a PSSM and for searching the database using this matrix. The PSSM is a popular representation of patterns in biological sequences, which has been used as an important feature in the classification of protein SS prediction. The PSSM makes multiple sequence alignments be represented easily as the mathematical entities. There are several ways to construct a PSSM, but the best way is using the log-odds method in analogous to how the substitution matrices are made. Given a set of aligned sequences, count the frequency of each type of amino acid that has occurred at each position. The frequency of amino acid a in the column u is denoted as $f_{u,a}$. The general frequency of amino acid a among sequences is denoted as p_a. Let $m_{u,a}$ be the PSSM score used for amino acid a in the column u, and then $m_{u,a} = log\left(\frac{f_{u,a}}{q_a}\right)$. It is also possible to weight the scores to compensate the bias in the original sequence selection. To use the PSSM, we need to specify a sliding window along the target sequence. At every position, a PSSM score is calculated by summing the scores of all columns, and the highest scoring position is reported.

PSI-BLAST is performed in these five steps: (i) search a query against the protein database; (ii) construct a multiple sequence alignment with the search result and then create the specialized PSSM; (iii) use the PSSM as a query against the database; (iv) estimate the statistical significance as E values; (v) repeat steps 3 and 4 iteratively until converge or hit a user-defined value, and use the updated PSSM as the new query for each new search.

11.3.1.2 SCORPION Yaseen and Li [319] reported an approach, called SCORPION, using neural networks with the statistical context-based scores as the encoded features to achieve the improvement on the secondary structure prediction accuracy. A sliding window of 15 residues is selected. To predict the SS state of that residue in the middle of the window, SCORPION utilizes 20 PSSM values, 1 extra value, and the context-based scores of the predefined SS to train a neutral model. Some studies

have shown that the types of nearby residues play a predominant role to the SS that a residue adopts [320, 321]. The context-based scores are defined as the summation of all mean-force potential for the singlet R_i, doublet R_iR_{i+1}, and triplet $R_iR_{i+1}R_{i+2}$, where i indexes the position of the residue in a protein sequence. At first, SCORPION needs to obtain the observed probabilities of a particular secondary structure C_i given the singlet, doublet, and triplet, respectively. The types of C_i can be the three states or the eight states.

$$P_{obs}(C_i|\bullet) = \frac{N_{obs}(C_i, \bullet)}{N_{obs}(\bullet)}, \tag{11.9}$$

where $\bullet$ represents the singlet, doublet, and triplet, and $N_{obs}(C_i, \bullet)$ is the observed number given the SS type and reside(s). Using the PSSM, these observed numbers are calculated as the summation of all protein sequences generated by the PSI-BLAST at the corresponding position in the PSSM. For example, $N_{obs}(C_i, R_i) = \sum^{|MSA|} \sum_{C_i} PSSM_j(R_i)$, where j is the position in the PSSM. The mean-force potential for doublet is defined as follows:

$$U(C_i, R_iR_{i+1}) = -RT \ln \left(\frac{P_{obs}\left(C_i|R_iR_{i+1}\right) P_{ref}(C_i|R_i)}{P_{ref}(C_i|R_iR_{i+1}) P_{obs}(C_i|R_i)} \right), \tag{11.10}$$

where the estimation of the referenced probability is formulated as

$$P_{ref}(C_i|R_i) = \frac{\sum_{C_j=C_i} N_{obs}(C_j, R_j)}{\sum_j N_{obs}(R_j)}. \tag{11.11}$$

The mean-force potential for the single and triplet can be calculated in a similar way. Finally, the context-based scores, $U(C_i, \cdots, R_{i-1}R_iR_{i+1}, \cdots)$, can be written as follows:

$$U(C_i, \cdots, R_{i-1}R_iR_{i+1}, \cdots) = U_{singlet} + U_{doublet} + U_{triplet}. \tag{11.12}$$

The context-based scores are encoded in the training and predicting of a neural network. SCORPION adopts a feed-forward neural network with three phases training. The first phase is the sequence-to-structure training using 20 PSSM values, 1 value for C- and N-terminals and the context-based scores. The second phase is the structure-to-structure training. The third phase is to refine the prediction results. Overall, there are 360 and 435 input values for three- and eight-state prediction, respectively. From the sevenfold cross-validation on several benchmarks, SCORPION achieved higher Q_3 prediction accuracy than some recently developed prediction tools [319].

11.3.1.3 Bamboo Li et al. [322] proposed a Bayesian model, named Bamboo, for the protein SS prediction. Bamboo is based on the knob-socket model, where MSA is used for inferring the prior information. The packing unit of protein residues in an area is called the knob-socket. The packing unit lays the ground rules about how

residues pack to form higher-order protein structures, and it also constructs the relations at both the secondary and tertiary structure levels [323, 324]. A pattern or motif of protein residues is called knob-socket when a single-residue "knob" packs into a three-reside "socket." A basic type of knob-socket involves residues i, $i+1$, $i+3$, and $i+4$ where i is the position on the protein sequence. According to this knob-socket model, Bamboo samples the probability of an SS for a given residue dependent on the ± 4 residues around it. Bamboo considers four types of blocks: (i) Helix (H), including 3_{10} helices, α-helices, or π-helices; (ii) Strand (E), including extended strands in parallel or antiparallel β-sheets; (iii) Turn (T), including hydrogen-bonded turns of length 3 or more amino acids; (iv) Coil (C), including β-bridge residues, bends, or random coils. The Bayesian model for these four SSs is described as follows:

$$p(\eta, \lambda|a) \propto p(a|\eta, \lambda)p(\eta, \lambda), \tag{11.13}$$

where η denotes a vector of the types of blocks, λ denotes a vector of the length values of blocks, and a denotes the protein residue string. For $p(a|\eta, \lambda)$, we take the coil as an example to show the probability mass functions for a coil block:

$$\begin{aligned} p_{Coil}(a_i, \cdots, a_j) &= p_{C_1}(a_i) \times p_{C_2}(a_{i+1}|a_i) \\ &\quad \times p_{C_3}(a_{i+2}|a_i, a_{i+1}) \times \cdots \times p_{C_3}(a_j|a_{j-2}, a_{j-1}), \end{aligned}$$

where p_{C_1} is a multinomial distribution with a category for each of the 20 amino acids, p_{C_2} is a 20-dimensional multinomial distribution conditioned on the antecedent amino acid, and p_{C_3} is a multinomial distribution conditioned on the two previous amino acids. The evaluation of these probability mass functions is based on the training data. $p_{C_\bullet}$ are calculated as the posterior means from the Bayesian model, which assumes a noninformative Dirichlet prior with all hyperparameters equal to 1 and a multinomial sampling model.

For the prior distribution $p(\eta, \lambda)$, Bamboo incorporates the information from MSA. To get the MSA given a set of proteins, whose SSs are already known, similar to the query sequence a, Bamboo uses the PSI-BLAST to search on the nrPDB database with E-values criterion set to 0.001. With the count vector X for the number of times of four secondary structures, and the assumption that the posterior distribution, $\phi|X$, follows a Dirichlet distribution, the prior distribution can be estimated as

$$p(\eta, \lambda) = \prod^{M} \prod^{l_m} \phi_{\eta, l}, \tag{11.14}$$

where M is the total blocks and l indexes the positions.

To infer the SS (η, λ), Markov chain Monte Carlo (MCMC) method is employed to sample from Formula 11.13 . The update scheme of the SSs includes switching types between two random blocks, block boundary changing, splitting a block, and merging two adjacent blocks. From the results on two classic protein structure data sets, the incorporation of MSA data provides a good balance and improves the prediction accuracy [322].

11.3.1.4 S2D Recently, Sormanni et al. [325] introduced a computational method, named S2D, to predict secondary structure populations from amino acid sequences and intrinsic disorder of proteins simultaneously. Many proteins or protein regions have been recognized as intrinsically disordered proteins (IDPs) and intrinsically disordered protein regions (IDPRs), which means that they are highly dynamical and are not able to maintain stable 3D structures [326, 327]. IDPs and IDRs prove to be involving in many biological processes, such as regulation and signaling [328]. As many structured proteins contain some disordered regions, and disordered proteins contain some structured parts at the same time, the boundary between "ordered" and "intrinsically disordered" proteins are not well defined. It would be desirable to have a predictor that provides structure and disorder simultaneously to reveal their intertwined properties of protein molecules.

The S2D predictor uses the artificial neural networks with extreme learning machine (ELM). Compared to many other machine learning methods, an important aspect of ELM is that the universal approximation capability has been proven. For each protein target, a sequence profile is generated using the PSSM by the PSI-BLAST program. Each residue in a protein sequence is described by 22 values with 21 from PSSM and additional one parameter to represent the identity of the amino acid found at that position. The training procedure of S2D uses is in three steps as follows:

1. S2D uses two single-hidden layer feed-forward neural networks (SLFNs), where the first SLFN employs a sliding window of 11 amino acids with 242 input neurons and the second SLFN employs a sliding window of 15 amino acids with 330 input neurons. Both networks have 4000 hidden nodes and 3 output neurons corresponding to α-helix, β-strand, and coil of the middle residue in the sliding window, respectively. Based on ELM, the first layer of weights is randomly assigned.
2. The second step of S2D is to predict the mean secondary structure populations of the entire protein by an N-to-1 network taking 22 values from PSSM and the 6 additional values corresponding to the outputs of the two SLFNs in the first step. This N-to-1 network has 150 hidden nodes and predicts three output values as well.
3. The third step of S2D employs another SLFN with a sliding window of 5 amino acids and 3600 hidden nodes. The network uses the same input values per residue as the N-to-1 network in the second step. The N-to-1 network provides this SLFN with valuable information about the global properties of the protein sequence, which can be utilized to interpret the stabilizing effect of the tertiary structure.

The experimental results showed that S2D predicted the SS populations assigned to the X-ray structures with a Q_3 score above 79% and identified disordered regions with an accuracy about 85–88% [325].

11.3.2 Measures of Quality of Protein SS Prediction

11.3.2.1 Performance Metrics We assess the quality of protein SS prediction at two levels, that is, the residue level and the segment level. To calculate any metrics at these two levels, we need the numbers of residues in a native SS state i with the prediction as in state j. If we aim at three-state prediction, i and j are in a set S of {helix H, strand E, coil C}. Let N_{ij} represent these numbers. Six residue-level measures commonly used are defined as follows:

$$\begin{aligned}
Q_{\mathrm{H}pre} &= \frac{N_{\mathrm{HH}}}{\sum_{i\in S} N_{i\mathrm{H}}},\\
Q_{\mathrm{E}pre} &= \frac{N_{\mathrm{EE}}}{\sum_{i\in S} N_{i\mathrm{E}}},\\
Q_{\mathrm{C}pre} &= \frac{N_{\mathrm{CC}}}{\sum_{i\in S} N_{i\mathrm{C}}},\\
Q_{\mathrm{H}obs} &= \frac{N_{\mathrm{HH}}}{\sum_{j\in S} N_{\mathrm{H}j}},\\
Q_{\mathrm{E}obs} &= \frac{N_{\mathrm{EE}}}{\sum_{j\in S} N_{\mathrm{E}j}},\\
Q_{\mathrm{C}obs} &= \frac{N_{\mathrm{CC}}}{\sum_{j\in S} N_{\mathrm{C}j}}.
\end{aligned}$$

The $Q_{\mathrm{H}pre}$, $Q_{\mathrm{E}pre}$ $Q_{\mathrm{C}pre}$ measure the percentages of correctly predicted helix, strand and coil residues, respectively. Similarly, $Q_{\mathrm{H}obs}$, $Q_{\mathrm{E}obs}$, and $Q_{\mathrm{C}obs}$ measure the percentages of correct helix, strand, and coil predictions among all native helix, strand, and coil residues, respectively. Another commonly used parameter measuring the accuracy is the parameter Q_3, which quantifies the overall rate of correct predictions for three SS states. To quantify the helix prediction errors, the $Q_{\mathrm{HE}error}$ is applied when a native helix residue is predicted as a strand and vice versa.

$$\begin{aligned}
Q_3 &= \frac{\sum_{i\in S} N_{ij}}{N},\\
Q_{\mathrm{HE}error} &= \frac{N_{HE}+N_{EH}}{N},
\end{aligned}$$

where N is the total number of residues in the protein sequence.

For the segment level, we use the segment overlap coefficient (SOV), which is an average overlap between the observed and the predicted segments. SOV contains the predicted helix (SOV_{H}), strand (SOV_{E}), coil (SOV_{C}) segments, and SOV_3, which is similar to the Q_3, is the sensitivity measurement.

$$SOV_{\mathrm{H}} = \frac{1}{N_{\mathrm{H}}} \sum_{S_{\mathrm{H}}} \frac{minOV(s_1, s_2) + \delta(s_1, s_2)}{maxOV(s_1, s_2)},$$

$$SOV_3 = \frac{1}{N} \left(\sum_{i\in S} \sum_{S_i} \left(\frac{minOV\left(s_1, s_2\right) + \delta(s_1, s_2)}{maxOV\left(s_1, s_2\right)} \times len(s_2) \right) \right),$$

$$\delta(s_1, s_2) = min \begin{cases} maxOV\left(s_1, s_2\right) - minOV(s_1, s_2) \\ minOV(s_1, s_2) \\ int\left(\frac{1}{2} len\left(s_1\right)\right) \\ int\left(\frac{1}{2} len\left(s_2\right)\right) \end{cases},$$

where s_1 and s_2 are the observed and predicted secondary structure segments in the H state; S_H is the set of all segment pairs (s_1, s_2) with at least one residue in H state in common; $minOV(s_1, s_2)$ is the overlap length between s_1 and s_2; $maxOV(s1_1, s_2)$ is the length of the segments covered by either s_1 or s_2 in H state; N_H is the total number of amino acid residues natively in the H state; $len(\bullet)$ is the number of amino acid residues in the segment. Usually, the measurements mentioned earlier are scaled in the percentage represent.

11.3.2.2 Benchmark Database Resources The performance of various prediction programs depends on the test databases and on the protein sequence data sets for which the predictions are made. There are some traditional public benchmarks for protein secondary structure prediction, such as CB513, Manesh215, Carugo338, and CASP9 targets. We briefly introduce these public benchmarks as follows:

- CB513. This data set includes 513 nonredundant domains, which was collected by Cuff and Barton in 1999. It has been frequently used to evaluate protein SS prediction. It can be freely accessed at http://comp.chem.nottingham.ac.uk/disspred/data-sets/CB513.
- Manesh data set, also known as Manesh215, consists of 215 nonhomologous protein chains with less than 25% pairwise sequence similarity. The data set is available at http://www.rtc.riken.go.jp/~/netasa/rvp-net/manesh-215/.
- CASP, that is, Critical Assessment of protein Structure Prediction, starting from 1994, is a community-wide and worldwide experiment for protein structure prediction performed every 2 years. Its data is freely available at http://predictioncenter.org/casp9/targetlist.cgi.
- SCOP, that is, Structural Classification of Proteins database, is a manually curated ordering of protein structural domains. Update to November 2015, the latest version of SCOP is 1.75 with 38221 PDB entries. The database is publicly accessible at http://scop.mrc-lmb.cam.ac.uk/scop/.
- ASTRAL. The ASTRAL compendium provides databases and tools useful for analyzing protein structures and protein sequences. Its current version is 2.05, released on February 5, 2015. It is freely available at http://astral.berkeley.edu/.

11.4 CONCLUSION AND FUTURE WORK

In this chapter, we briefly introduce two MSA-based applications, RNA and protein SS predictions. The key idea behind MSA-based RNA SS prediction is the conserved

patterns of base pairs inferred from a collection of homologous sequences, that is, some base pairs tend to vary at the same time during evolution for maintaining structural integrity. In general, MSA provides three types of information, that is, mutual information, covariation information, and phylogenetic information. Regarding the order of aligning and folding processes in RNA SS prediction, the computational approaches can be classified into three categories: Prior Aligning Ulterior Folding Method, Simultaneously Folding and Aligning Method, and Prior Folding Ulterior Aligning Method. We use three distinct approaches falling into these three groups to demonstrate the fundamental strategies of aligning and folding based on MSA. Although protein does not have exactly the same variation pattern of base pairs as in RNA SS prediction, some studies showed that the SS state of the amino acid residue is significantly affected by the types of adjacent residues. Due to this fact, many modern methods for protein SS prediction have been extensively studied using machine learning approaches, such as support vector machines, hidden Markov model, Bayesian networks, and artificial neural networks, which often utilize the PSSM generated from MSA as a significant indicator. We introduced the PSI-BLAST and how to use PSI-BLAST to create PSSM, and then employed three recently proposed machine learning approaches to illustrate how to incorporate the information from PSSM when predicting protein SSs. For the quality assessing of these two MSA applications, we provide a list of evaluation metrics and available benchmarks freely available on the Internet.

For the RNA SS prediction, the computational cost for the second-type methods, that is, Simultaneously Folding and Aligning Method, is still a big computational challenge, although various techniques for reducing the computation time have been developed. More efficient algorithms that incorporate mature computing techniques, such as cloud computing, are needed. Despite the difference of protein SS prediction methods, the cross-validated 80% accuracy barrier is still present and has not yet been overcome. Kihara [329] pointed out the importance of long-range interactions on the formation of protein SS. He argued that as long as SS predictions are based on a sliding window, the long-range effects not only on β-sheets but even on helices will be treated in a limited manner. So nonsliding-window-based methods could make progress in protein SS prediction with innovative machine learning strategies such as hierarchical temporal memory, deep neural networks, and computational biology thinking. Therefore, the future direction of protein secondary structure prediction could be an ensemble approach to the utilization of a group of techniques to include most if not all merit aspects of machine learning. Lastly, novel applications of other machine learning techniques, such as graph theory, data visualization, and biomedical framework, along with in silico simulation and even DNA computing, promise to give rise to the eventual vision of routine tertiary and even quaternary protein structure prediction. Perhaps these techniques will eventually lead us to the nascent field of "biomolecular intelligence," where molecular biology and machine learning converge and become one discipline.

REFERENCES

1. Maxam, A. and Gilbert, W. (1977) A new method for sequencing DNA. *Proc. Natl. Acad. Sci. U.S.A.*, **74**, 560–564.
2. Sanger, F. and Coulson, A. (1975) A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J. Mol. Biol.*, **3**, 441–448.
3. Olsvik, O., Wahlberg, J., Petterson, B., et al. (1993) 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.
4. Bentley, D.R., Balasubramanian, S., Swerdlow, H.P., Smith, G.P., Milton, J., Brown, C.G., Hall, K.P., Evers, D.J., Barnes, C.L., Bignell, H.R. et al. (2008) Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, **456** (7218), 53–59.
5. Ratia, K., Pegan, S., Takayama, J., Sleeman, K., Coughlin, M., Baliji, S., Chaudhuri, R., Fu, W., Prabhakar, B.S., Johnson, M.E., Baker, S.C., Ghosh, A.K., and Mesecar, A.D. (2008) A noncovalent class of papain-like protease/deubiquitinase inhibitors blocks SARS virus replication. *Proc. Natl. Acad. Sci. U.S.A.*, **42**, 16 119–16 124.
6. Akiyama, Y., Hosoya, T., Poole, A., and Hotta, Y. (1996) The gcm-motif: a novel DNA-binding motif conserved in Drosophila and mammals. *Proc. Natl. Acad. Sci. U.S.A.*, **25**, 14 912–14 916.
7. Chothia, C. and Lesk, A. (1986) The relation between the divergence of sequence and structure in proteins. *EMBO J.*, **5**, 823–826.
8. Hubbard, T. and Blundell, T. (1987) Comparison of solvent-inaccessible cores of homologous proteins: definitions useful for protein modelling. *Protein Eng.*, **1**, 159–171.

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

9. Greer, J. (1990) Comparative modeling methods: application to the family of the mammalian serine proteases. *Proteins*, **7** (4), 317–334.
10. Boeckmann, B., Bairoch, A., Apweiler, R., et al. (2003) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, **31**, 365–370.
11. Bairoch, A., Apweiler, R., et al. (2005) The Universal Protein Resource (UniProt). *Nucleic Acids Res.*, **33**, 154–159.
12. Wu, C., Yeh, L., Huang, H., Arminski, L., et al. (2003) The protein information resource. *Nucleic Acids Res.*, **31**, 345–347.
13. Berman, H.M., et al. (2000) The protein data bank - (PDP). *Nucleic Acids Res.*, **28**, 235–242.
14. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., and Wheeler, D.L. (2006) GenBank. *Nucleic Acids Res.*, **34** (Suppl. 1), D16–D20.
15. Mizuguchi, K., Deane, C., Blundell, T., and Overington, J. (1998) HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci.*, **7**, 2469–2471.
16. Pruitt, K., Tatusova, T., and Maglott, D. (2005) NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, **33**, 501–504.
17. Gibbs, A.J. and McIntyre, G.A. (1970) The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *Eur. J. Biochem.*, **16** (1), 1–11.
18. Smith, T.F. and Waterman, M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.
19. Needleman, S.B. and Wunsch, C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48** (3), 443–453.
20. Mount, D.W. (2004) Mount, Bioinformatics Hardcover, 2nd edition. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, p. 692, ISBN-10: 0879696877; ISBN-13: 978-0879696870.
21. Ollivier, E., Soldano, H., and Viari, A. (1991) 'Multifrequency' Location and Clustering of Sequence Patterns from Proteins. Oxford University Press, Oxford, **7**, pp. 31–38, doi: 10.1093/bioinformatics/7.1.31.
22. Gotoh, O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162** (3), 705–708, doi: 10.1016/0022-2836(82)90398-9.
23. Lipman, D. and Pearson, W. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435–1441.
24. Altschul1, S.F., Gish1, W., et al. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
25. Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C. (1978) A model of evolutionary change in proteins. Matrices for detecting distant relationships. *Atlas Protein Seq. Struct.*, **5** (Suppl. 3), 345–358.
26. Jukes, T. and Cantor, C. (1969) Evolution of Protein Molecules. Academic Press, New York, pp. 21–123.
27. Henikoff, S. and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, **89** (22), 10 915–10 919.
28. Edwards, A. (1963) The measure of association in a 2x2 table. *J. R. Stat. Soc.*, **1**, 109–114.
29. Mosteller, F. (1968) Association and estimation in contingency tables. *J. Am. Stat. Assoc.*, **321**, 1–28.

30. Cornfield, J. (1951) A applications to cancer of the lung, breast, and cervix. *J. Natl. Cancer Inst.*, **11**, 1269–1275.
31. Altschul, S. (1991) Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Bol.*, **219**, 555–565.
32. Eddy, S. (2004) Where did the BLOSUM62 alignment score matrix come from? *Nat. Biotechnol.*, **22** (8), 1035–1036.
33. Gonnet, G.H., Cohen, M.A., and Benner, S.A. (1992) Exhaustive matching of the entire protein sequence database. *J. Sci.*, **256**, 1443–1445.
34. Lesk, A. and Chothia, C. (1980) How different amino acid sequences determine similar protein structures: the structure and evolutionary dynamics of the globins. *J. Mol. Biol.*, **136**, 225–270.
35. Carillo, H. and Lipman, D. (1988) The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, **48** (5), 1073–1082.
36. Karlin, S. and Brocchieri, L. (1996) Evolutionary conservation of RecA genes in relation to protein structure and function. *J. Bacteriol.*, **178** (7), 1881–1894.
37. Armon, A., Graur, D., and Ben-Tal, N. (2001) 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.
38. Gonnet, G.H., Korostensky, C., and Benner, S. (2000) Evaluation measures of multiple sequence alignment. *J. Comput. Biol.*, **7** (1), 261–276.
39. Wu, T. and Kabat, E. (1970) An analysis of the sequences of the variable regions of Bence Jones proteins and myeloma light chains and their implications for antibody complementarity. *J. Exp. Med.*, **132** (2), 211–250.
40. Jores, R., Alzari, P., and Meo, T. (1990) Resolution of hypervariable regions in T-Cell receptor beta chains by a modified Wu-Kabat index of amino acid diversity. *Proc. Natl. Acad. Sci. U.S.A.*, **87** (23), 9138–9142.
41. Lockless, S. and Ranganathan, R. (1999) Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, **286** (5438), 295–299.
42. Shannon, C.E. (1948) A mathematical theory of communication. *Bell Syst. Tech. J.*, **27**, 379–423.
43. Sander, C. and Schneider, R. (1991) Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, **9** (1), 56–68.
44. Shenkin, P., Erman, B., and Mastrandrea, L. (1991) Information-theoretical entropy as a measure of sequence variability. *Proteins*, **11**, 297–313.
45. Gerstein, M. and Altman, R. (1995) Average core structures and variability measures for protein families: application to the immunoglobulins. *J. Mol. Biol.*, **251**, 161–175.
46. Taylor, W.R. (1986) The classification of amino acid conservation. *J. Theor. Biol.*, **119** (2), 205–218.
47. Zvelibil, M.J., Barton, G.J., Taylor, W.R., and Sternberg, M.J.E. (1987) Prediction of protein secondary structure and active sites using the alignment of homologous sequences. *J. Mol. Biol.*, **195**, 957–961.
48. Valdar, W. and Thornton, J.M. (2001) *Residue conservation in the prediction of protein-protein interfaces*, Ph.D. thesis, University College London.
49. Nguyen, K.D. and Pan, Y. (2007) A reliable metric for quantifying multiple sequence alignment, in *BIBE*, pp. 788–795.

50. Smith, R.F. and Smith, T.F. (1992) Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling. *Protein Eng.*, **5**, 35–41.
51. Hudak, J. and McClure, M.A. (1999) A comparative analysis of computational motif-detection methods. *Pac. Symp. Biocomput.*, **4**, 138–149.
52. Thompson, J.D., Koehl, P., Riip, R., and Poch, O. (2005) BAliBASE 3.0: latest development of multiple alignment benchmark. *Protein*, **61**, 127–136.
53. Edgar, R. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32** (5), 1792–1797.
54. Stoye, J. (1998) Multiple sequence alignment with the divide-and-conquer method. *Gene*, **211** (2), 56.
55. Thompson, J., Higgins, D., Gibson, T., et al. (1994) 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.
56. Morgenstern, B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.
57. Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30** (14), 3059–3066.
58. Notredame, C., Higgins, D., and Heringa, J. (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302** (1), 205–217.
59. Thompson, J., Gibson, T., Plewniak, F., Jeanmougin, F., and Higgins, D. (1997) The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.*, **25** (24), 4876–4882.
60. Sneath, P.H.A. and Sokal, R.R. (1973) Numerical Taxonomy. The Principles and Practice of Numerical Classification. Freeman, San Francisco, CA, p. 573.
61. Saitou, N. and Nei, M. (1987) The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. Oxford University Press, New York, **4**, pp. 406–425.
62. McQueen, J.B. (1967) Some methods for classification and analysis of multivariate observations, in *Proceeding of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.
63. Dunn, J.B. (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. Cybern.*, **32** (3), 32–57.
64. Dunn, J.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc.*, **39**, 1–38.
65. Muthukrishnan, S. and Sahinalp, S.L.C. (2000) Approximate nearest neighbors and sequence comparison with block operations, in *procedding of 32nd ACM on Theory Computing*, pp. 416–424.
66. Nguyen, K. and Pan, Y. (2010) KB-MSA: knowledgebase multiple sequence alignment, in *Preceeding of ISBRA 2010*, **6**, pp. 68–71.
67. Graham, R.L. and Foulds, L.R. (1982) Unlikelyhood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Math. Biosci.*, **60**, 133–142.
68. Wang, L. and Jiang, T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol.*, **1** (4), 337–348.

69. Thompson, J.D., Plewniak, F., and Poch, O. (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, **27**, 2682–2690.

70. Lipman, D., Altschul, S., and Kececioglu, J. (1989) A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. U.S.A.*, **86** (12), 4412–4415.

71. Perrey, S.W. and Stoye, J. (1997) FDCA: Fast and Accurate Approximation to Sum-of-Pairs Score Optimal Multiple Sequence Alignment. Universität Bielefeld, **114**.

72. Feng, D. and Doolittle, R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **60**, 351–360.

73. Corpet, F. (1988) Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.*, **16** (22), 10881–10890.

74. Lassmann, T. and Sonnhammer, E.L. (2005) Kalign - an accurate and fast multiple sequence alignment algorithm. *Commun. ACM*, **6**, 298.

75. Wu, S. and Manber, U. (1992) Fast text searching allowing errors. *Commun. ACM*, **35**, 83–91.

76. Smith, R.F. and Smith, T.F. (1990) Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Natl. Acad. Sci. U.S.A.*, **87** (1), 118–122.

77. Yamada, S., Gotoh, O., and Yamana, H. (2006) Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinf.*, **7**, 524.

78. Gotoh, O. (1996) 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.

79. Golver, F. and Laguna, M. (1997) Tabu Search. Kluwer Academic Publishers, Boston, MA.

80. Glover, F., Taillard, E., and de Werra, D. (1993) A user's guide to tabu search. *Ann. Oper. Res.*, **41**, 3–28.

81. Riaz, T., Wang, Y., and Li, K.B. (2004) Multiple sequence alignment using tabu search, in *Conferences in Research and Practice in Information Technology*, **29**, pp. 223–232.

82. Baldi, P. and Chauvin, Y. (1994) Smooth on-line learning algorithms for hidden Markov models. *Neural Comput.*, **6** (2), 307–318, doi: 10.1162/neco.1994.6.2.307.

83. Krogh, A., Mian, I.S., and Haussler, D. (1994) A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Res.*, **22** (2), 307–318.

84. Grasso, C. and Lee, C. (2004) 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.

85. Sze, S.H., Lu, Y., and Yang, Q. (2006) A polynomial time solvable formulation of multiple sequence alignment. *J. Comput. Biol.*, **13**, 309–319.

86. Do, C., Mahabhashyam, M., Brudno, M., and Batzoglou, S. (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.

87. Huang, X. and Miller, W. (1991) A time-efficient, linear space local similarity algorithm. *Adv. Appl. Math.*, **12**, 337–357.

88. O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D.G., and Notredame, C. (2004) 3DCoffee: combining protein sequences and structures within multiple sequence alignments. *J. Mol. Biol.*, **340**, 385–395.

89. Bray, N., Dubchak, I., and Pachter, L. (2003) AVID: A Global Alignment Program. Cambridge University Press, Cambridge, **13**, pp. 97–102.

90. Gusfield, D. (1997) Algorithms on Strings, Trees, and Sequences: Science and Computational Biology. Cambridge University Press, Cambridge.

91. Zhang, Y. and Waterman, M.S. (2005) An eulerian path approach to local multiple alignment for dna sequences. *Proc. Natl. Acad. Sci. U.S.A.*, **102**, 1285–1290.

92. Pevzner, P.A., Tang, H., and Waterman, M.S. (2001) An eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A.*, **98**, 9748–9753.

93. Holland, J. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

94. Golberg, D. (1989) Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Reading, MA.

95. Chellapilla, K. and Fogel, G.B. (1999) Multiple sequence alignment using evolutionary programming, in *Congress on Evolutionary Computation*, pp. 445–452.

96. Zhang, C. and Wong, A. (1997) Toward efficient multiple molecular sequence alignment: a system of genetic algorithm and dynamic programming. *IEEE Trans. Syst. Man Cybern.*, **27**, 918–932.

97. Notredame, C., Higgins, D., and Journals, O. (1996) SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Res.*, **24** (8), 1515–1524.

98. Taheri, J. and Zomaya, A.Y. (2009) RBT-GA: a novel metaheuristic for solving the multiple sequence alignment problem. *BMC Genomics*, **10** (Suppl. 1), S10.

99. Lee, Z.J., Su, S.F., Chuang, C.C., and Liu, K.H. (2008) Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. *Appl. Soft Comput.*, **8**, 55–78.

100. Cai, L., Juedes, D., and Liakhovitch, E. (2000) Evolutionary computation techniques for multiple sequence alignment, in *Proceedings of the Congress on Evolutionary Computation*, **2**, pp. 829–835.

101. Nguyen, H., Yamamori, K., Yoshihara, I., and Yasunaga, M. (2003) Improved ga-based method for multiple protein sequence alignment, in *The 2003 Congress on Evolutionary Computation (CEC '03)*, **3**, pp. 1826–1832.

102. Liu, L.f., Huo, H.w., and Wang, B.s. (2004) Aligning multiple sequences by genetic algorithm, in *International Conference on Communications, Circuits and Systems (ICCCAS 2004)*, pp. 994–998.

103. Dorigo, M. (1992) *Optimization, learning and natural algorithms*, Ph.D. thesis, Politecnico di Milano, Italie.

104. Liu, D., Xiong, X., Hou, Z., and DasGupta, B. (2005) Identification of motifs with insertions and deletions in protein sequences using self-organizing neural networks. *Neural Netw.*, **18**, 835–842.

105. Chakrabarti, S., Bhardwaj N., Anand, P.A., and Sowdhamini, R. (2004) Improvement of alignment accuracy utilyzing sequentially conserved motifs (FMALIGN). *BMC Bioinf.*, **5**, 167.

106. Walle, I.V., Lasters, I., and Wyns, L. (2005) SABmark-a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, **21**, 1267–1268.

107. Nguyen, K. and Pan, Y. (2011) Multiple sequence alignment based on dynamic weighted guidance tree. *Int. J. Bioinf. Res. Appl.*, **7** (2), 168–182.

108. Carl von, L. and Lars, S. (1758) *Caroli Linnaei Systema naturae per regna tria naturae :secundum classes, ordines, genera, species, cum characteribus, differentiis, synonymis, locis*, **1**, Holmiae: Impensis Direct. Laurentii Salvii, p. 881, http://www.biodiversitylibrary.org/item/10277, http://www.biodiversitylibrary.org/bibliography/542.

109. Mayr, G., Pohl, B., and Peters, D.S. (2005) A well-preserved archaeopteryx specimen with theropod features. *Science*, **310** (5753), 1483–1486, doi: 10.1126/science.1120331.

110. Huang, C.H. and Biswas, R. (2002) Parallel pattern identification in biological sequences on clusters, in *IEEE International Conference on Cluster Computing*, p. 127, doi: 10.1109/CLUSTR.2002.1137737.

111. Lee, H.C. and Ercal, F. (1997) RMESH algorithms for parallel string matching, in *3rd International Symposium on Parallel Architectures, Algorithms, and Networks, I-SPAN '97 Proceedings*, pp. 223–226.

112. Lima, C.R.E., Lopes, H.S., Moroz, M.R., and Menezes, R.M. (2007) Multiple sequence alignment using reconfigurable computing, in *ARC'07: Proceedings of the 3rd International Conference on Reconfigurable Computing*, Springer-Verlag, Berlin, Heidelberg, pp. 379–384.

113. Liu, Y., Schmidt, B., and Maskell, D.L. (2009) MSA-CUDA: multiple sequence alignment on graphics processing units with CUDA, in *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 121–128, doi: 10.1109/ASAP.2009.14.

114. Sarkar, S., Kulkarni, G.R., Pande, P.P., and Kalyanaraman, A. (2010) Network-on-chip hardware accelerators for biological sequence alignment. *IEEE Trans. Comput.*, **59**, 29–41, doi: 10.1109/TC.2009.133.

115. Raju, V.S. and Vinayababu, A. (2006) Optimal parallel algorithm for string matching on mesh network structure. *Int. J. Appl. Math. Sci.*, **3**, 167–175, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.9932&rep=rep1&type=pdf.

116. Raju, V.S. and Vinayababu, A. (2007) Parallel algorithms for string matching problem on single and two dimensional reconfigurable pipelined bus systems. *J. Comput. Sci.*, **3**, 754–759, http://thescipub.com/PDF/jcssp.2007.754.759.pdf.

117. Takefuji, Y., Tanaka, T., and Lee, K. (1992) A parallel string search algorithm. *IEEE Trans. Syst. Man Cybern.*, **22** (2), 332–336.

118. Oliver, T., Schmidt, B., Nathan, D., Clemens, R., and Maskell, D. (2005) Using reconfigurable hardware to accelerate multiple sequence alignment with clustalW. *Bioinformatics*, **21** (16), 3431–3432, doi: 10.1093/bioinformatics/bti508.

119. Oliver, T., Schmidt, B., Maskell, D., Nathan, D., and Clemens, R. (2006) High-speed multiple sequence alignment on a reconfigurable platform. *Int. J. Bioinf. Res. Appl.*, **2**, 394–406, doi: 10.1504/IJBRA.2006.011038.

120. Huang, X. (1990) A space-efficient parallel sequence comparison algorithm for a message-passing multiprocessor. *Int. J. Parallel Program.*, **18** (3), 223–239, doi: 10.1007/BF01407900.

121. Aluru, S., Futamura, N., and Mehrotra, K. (2003) Parallel biological sequence comparison using prefix computations. *J. Parallel Distrib. Comput.*, **63** (3), 264–272, doi: 10.1016/S0743-7315(03)00010-8.

122. Dally, W.J. and Towles, B. (2001) Route packets, not wires: on-chip interconnection networks, in *Proceedings of the 38th Design Automation Conference*, pp. 684–689, doi: 10.1109/DAC.2001.156225.

123. Tan, G., Feng, S., and Sun, N. (2005) Parallel multiple sequences alignment in SMP cluster, in *HPCASIA '05: Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, IEEE Computer Society, Beijing, China, p. 426, doi: 10.1109/HPCASIA.2005.70.

124. Luo, J., Ahmad, I., Ahmed, M., and Paul, R. (2005) Parallel multiple sequence alignment with dynamic scheduling, in *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, **I**, IEEE Computer Society, Washington, DC, pp. 8–13, doi: 10.1109/ITCC.2005.223.

125. Shi, H., Ritter, G.X., and Wilson, J.N. (1995) Simulations between two reconfigurable mesh models. *Inf. Process. Lett.*, **55** (3), 137–142, doi: 10.1016/0020-0190(95)00082-N.

126. Pan, Y., Li, K., and Hamdi, M. (1999) An improved constant-time algorithm for computing the radon and hough transforms on a reconfigurable mesh. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, **29** (4), 417–421, doi: 10.1109/3468.769762.

127. Bourgeois, A.G. and Trahan, J.L. (2000) Relating two-dimensional reconfigurable meshes with optically pipelined buses. *Parallel and Distributed Processing Symposium, International*, p. 747, doi: 10.1109/IPDPS.2000.846060.

128. Trahan, J.L., Bourgeois, A.G., Pan, Y., and Vaidyanathan, R. (2000) Optimally scaling permutation routing on reconfigurable linear arrays with optical buses. *J. Parallel Distrib. Comput.*, **60** (9), 1125–1136, doi: 10.1006/jpdc.2000.1643.

129. Nguyen, K.D. and Bourgeois, A.G. (2006) Ant colony optimal algorithm: fast ants on the optical pipelined R-mesh, in *International Conference on Parallel Processing (ICPP'06)*, pp. 347–354.

130. Cordova-Flores, C.A., Fernandez-Zepeda, J.A., and Bourgeois, A.G. (2007) Constant time simulation of an R-Mesh on an LR-Mesh, in *Parallel and Distributed Processing Symposium, International*, p. 269, doi: 10.1109/IPDPS.2007.370459.

131. Vaidyanathan, R. and Trahan, J.L. (2004) Dynamic Reconfiguration: Architectures and Algorithms. Kluwer Academic/Plenum Publishers, New York.

132. Bertossi, A.A. and Mei, A. (2000) Constant time dynamic programming on directed reconfigurable networks. *IEEE Trans. Parallel Distrib. Syst.*, **11**, 529–536, doi: 10.1109/71.862204.

133. Chen, L., Juan, C., and Pan, Y. (2005) Fast scalable algorithm on LARPBS for sequence alignment, in *Proceedings of the 2005 International Conference on Parallel and Distributed Processing and Applications (ISPA'05)*, Springer-Verlag, Berlin, Heidelberg, pp. 176–185, doi: 10.1007/11576259_20.

134. EMBL-EBI (2012) The European Bioinformatics Institute, http://www.ebi.ac.uk (accessed 4 February 2016).

135. Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soding, J., Thompson, J.D., and Higgins, D.G. (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol. Syst. Biol.*, **7**, 539.

136. Thompson, J.D., Plewniak, F., Thierry, J.C., and Poch, O. (2000) DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res.*, **28** (15), 2919–2926, doi: 10.1093/nar/28.15.2919.

137. Katoh, K. and Toh, H. (2007) PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics*, **23** (3), 372–374, doi: 10.1093/bioinformatics/btl592.

138. Blackshields, G., Sievers, F., Shi, W., Wilm, A., and Higgins, D. (2010) Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms Mol. Biol.*, **5** (1), 21, doi: 10.1186/1748-7188-5-21.
139. MacQueen, J.B. (1967) Some methods for classification and analysis of multivariate observations, in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, **1** (eds L.M.L. Cam and J. Neyman), University of California Press, Berkeley, CA, pp. 281–297.
140. Wilm, A., Higgins, D.G., and Notredame, C. (2008) R-Coffee: a method for multiple alignment of non-coding RNA. *Nucleic Acids Res.*, **36** (9), e52, doi: 10.1093/nar/gkn174.
141. Loytynoja, A. and Goldman, N. (2010) webPRANK: a phylogeny-aware multiple sequence aligner with interactive alignment browser. *BMC Bioinf.*, **11** (1), 579, doi: 10.1186/1471-2105-11-579.
142. Löytynoja, A. and Goldman, N. (2005) An algorithm for progressive multiple alignment of sequences with insertions. *Proc. Natl. Acad. Sci. U.S.A.*, **102**, 10 557–10 562.
143. National center for biotechnology information (2012) http://www.ncbi.nlm.nih.gov/ (accessed 4 February 2016).
144. Papadopoulos, J.S. and Agarwala, R. (2007) COBALT: constraint-based alignment tool for multiple protein sequences. *Bioinformatics*, **23** (9), 1073–1079, doi: 10.1093/bioinformatics/btm076.
145. DNA Data Bank of Japan (2012) http://www.ddbj.nig.ac.jp/index-e.html (accessed 4 February 2016).
146. GenomeNet (2012) http://www.genome.jp/ (accessed 4 February 2016).
147. Institute Pasteur (2012) http://bioweb2.pasteur.fr/ (accessed 4 Febraury 2016).
148. Computtational Biology Research Center (2012) http://www.cbrc.jp/index.eng.html.
149. Welcome trust Sanger institute (2012) http://www.sanger.ac.uk/resources/software/ (accessed 4 February 2016).
150. Ning, Z., Cox, A., and Mullikin, J. (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11** (10), 1725–1729.
151. Center for biological sequence analysis-Technical University of Denmark (2012) http://www.cbs.dtu.dk/biotools/ (accessed 4 Febrauary 2016).
152. Swiss Institute of Bioinformatics (2012) http://expasy.org/ (accessed 4 Febrauary 2016).
153. Bielefeld University Bioinformatics Server (2012) http://bibiserv.techfak.uni-bielefeld.de/ (accessed 4 February 2016).
154. Institut deg-n-tique et microbiologie (2012) http://www-archbac.u-psud.fr/ (accessed 4 February 2016).
155. Pole bioinformatique lyonnais (2012) http://prabi.ibcp.fr.
156. Corpet, F. (1988) Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.*, **16** (22), 10 881–10 890, doi: 10.1093/nar/16.22.10881.
157. HPC @ NIH (2012) Helix Systems High-Performance Computing at the NIH. http://helixweb.nih.gov/multi-align/ (accessed 4 February 2016).
158. UTSouthwestern Medical Center (2012) Grishin lab at Southwestern Medical Center - University of Texas, USA, http://prodata.swmed.edu/Lab/HomeLAB.htm (accessed 4 February 2016).
159. CSC - It Center for Science Ltd (2012) Kajaani Data Center, Finland, http://www.csc.fi/english/research/software (accessed 4 Febrauary 2016).
160. Center for BioInformatic at Peking University, China, 1996, http://www.cbi.pku.edu.cn/ (accessed 15 March 2016).

161. Lassmann, T. and Sonnhammer, E.L.L. (2005) Automatic assessment of alignment quality. *Nucleic Acids Res.*, **33** (22), 7120–7128.

162. Nguyen, K. and Pan, Y. (2011) An improved scoring method for protein residue conservation and multiple sequence alignment. *IEEE Trans. NanoBiosc.*, **10**, 275–285.

163. Schwartz, A.S. and Pachter, L. (2007) Multiple alignment by sequence annealing. *Bioinformatics*, **23** (2), e24–e29, doi: 10.1093/bioinformatics/btl311.

164. Gilbert, D. (2002) Seqread, http://iubio.bio.indiana.edu/.

165. Raghava, G.P., Searle, S.M., Audley, P.C., Barber, J.D., and Barton, G.J. (2003) OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinf.*, **4**, 47.

166. Van Walle, I., Lasters, I., and Wyns, L. (2005) SABmark – a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, **21** (7), 1267–1268, doi: 10.1093/bioinformatics/bth493.

167. Subramanian, A., Weyer-Menkhoff, J., Kaufmann, M., and Morgenstern, B. (2005) DIALIGN-T: an improved algorithm for segment-based multiple sequence alignment. *BMC Bioinf.*, **6** (1), 66, doi: 10.1186/1471-2105-6-66.

168. Blackshields, G., Wallace, I.M., Larkin, M., and Higgins, D.G. (2006) Analysis and comparison of benchmarks for multiple sequence alignment. *In Silico Biol.*, **6**, 321–339.

169. Van Walle, I., Lasters, I., and Wyns, L. (2004) Align m: a new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics*, **20** (9), 1428–1435, doi: 10.1093/bioinformatics/bth116.

170. Pei, J., Sadreyev, R., and Grishin, N.V. (2003) PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics*, **19** (3), 427–428, doi: 10.1093/bioinformatics/btg008.

171. Notredame, C. and Higgins, D. (2000) T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

172. Metzker, M.L. (2010) Sequencing technologies [mdash] the next generation. *Genome Res.*, **11**, 31–46.

173. Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I., Belmonte, M., Lander, E., Nusbaum, C., and Jaffe, D. (2008) ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Res.*, **8**, 810–820.

174. Jiang, H. and Wong, W. (2008) SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, **24**, 2395–2396.

175. Li, H., Ruan, J., and Durbin, R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18**, 1851–1858.

176. Smith, A., Xuan, Z., and MQ, Z. (2008) Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC Bioinf.*, **9**, 128.

177. Lin, H., Zhang, Z., Zhang, M.Q., Ma, B., and Li, M. (2008) Zoom! Zillions of oligos mapped. *Bioinformatics*, **24** (21), 2431–2437, doi: 10.1093/bioinformatics/btn416.

178. Eaves, H.L. and Gao, Y. (2009) MOM: maximum oligonucleotide mapping. *Bioinformatics*, **25** (7), 969–970, doi: 10.1093/bioinformatics/btp092.

179. Homer, N., Merriman, B., and Nelson, S.F. (2009) BFAST: an alignment tool for large scale genome resequencing. *PLoS ONE*, **4** (11), e7767, doi: 10.1371/journal.pone.0007767.

180. Campagna, D., Albiero, A., Bilardi, A., Caniato, E., Forcato, C., Manavski, S., Vitulo, N., and Valle, G. (2009) PASS: a program to align short sequences. *Bioinformatics*, **25** (7), 967–968, doi: 10.1093/bioinformatics/btp087.

181. Chen, Y., Souaiaia, T., and Chen, T. (2009) PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*, **25** (19), 2514–2521, doi: 10.1093/bioinformatics/btp486.

182. Schatz, M.C. (2009) CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, **25** (11), 1363–1369, doi: 10.1093/bioinformatics/btp236.

183. Rumble, S.M., Lacroute, P., Dalca, A.V., Fiume, M., Sidow, A., and Brudno, M. (2009) SHRiMP: accurate mapping of short color-space reads. *PLoS Comput. Biol.*, **5** (5), e1000 386, doi: 10.1371/journal.pcbi.1000386.

184. Weese, D., Holtgrewe, M., and Reinert, K. (2012) RazerS 3: faster, fully sensitive read mapping. *Bioinformatics*, **28** (20), 2592–2599, doi: 10.1093/bioinformatics/bts505.

185. Clement, N.L., Snell, Q., Clement, M.J., Hollenhorst, P.C., Purwar, J., Graves, B.J., Cairns, B.R., and Johnson, W.E. (2010) The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics*, **26** (1), 38–45, doi: 10.1093/bioinformatics/btp614.

186. Zhang, G., Fedyunin, I., Kirchner, S., Xiao, C., Valleriani, A., and Ignatova, Z. (2012) FANSe: an accurate algorithm for quantitative mapping of large scale sequencing reads. *Nucleic Acids Res.*, **40** (11), e83, doi: 10.1093/nar/gks196.

187. Yang, X., Charlebois, P., Gnerre, S., Coole, M., Lennon, N., Levin, J., Qu, J., Ryan, E., Zody, M., and Henn, M. (2012) De novo assembly of highly diverse viral populations. *BMC Genomics*, **13** (1), 475, doi: 10.1186/1471-2164-13-475.

188. Broder, A.Z., Glassman, S.C., Manasse, M.S., and Zweig, G. (1997) Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, **29** (8-13), 1157–1166, doi: 10.1016/S0169-7552(97)00031-7.

189. Weiner, P. (1973) Linear pattern matching algorithms, in *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (swat 1973)*, IEEE Computer Society, Washington, DC, pp. 1–11, doi: 10.1109/SWAT.1973.13.

190. Kurtz, S., Phillippy, A., Delcher, A., Smoot, M., Shumway, M., Antonescu, C., and Salzberg, S. (2004) Versatile and open software for comparing large genomes. *Genome Biol.*, **5** (2), R12, doi: 10.1186/gb-2004-5-2-r12.

191. Meek, C., Patel, J.M., and Kasetty, S. (2003) OASIS: an online and accurate technique for local-alignment searches on biological sequences, in *In VLDB*, pp. 910–921.

192. Abouelhoda, M.I., Kurtz, S., and Ohlebusch, E. (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, **2** (1), 53–86, doi: 10.1016/S1570-8667(03)00065-0.

193. Hoffmann, S., Otto, C., Kurtz, S., Sharma, C.M., Khaitovich, P., Vogel, J., Stadler, P.F., and Hackermüller, J. (2009) Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput. Biol.*, **5** (9), e1000 502, doi: 10.1371/journal.pcbi.1000502.

194. Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10** (3), R25, doi: 10.1186/gb-2009-10-3-r25.

195. Li, H. and Durbin, R. (2009) Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25** (14), 1754–1760, doi: 10.1093/bioinformatics/btp324.

196. Li, R., Yu, C., Li, Y., Lam, T.W., Yiu, S.M., Kristiansen, K., and Wang, J. (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25** (15), 1966–1967, doi: 10.1093/bioinformatics/btp336.

197. Burrows, M. and Wheeler, D. (1994) *A Block Sorting Lossless Data Compression Algorithm*. Technical Report 124. Digital Equipment Corporation, Systems Research Center 130 Lytton Avenue Palo Alto, CA 94301.

198. SHRiMP - Short Read Mapping Package, http://compbio.cs.toronto.edu/shrimp/ (accessed 4 February 2016).

199. Li, R., Li, Y., Kristiansen, K., and Wang, J. (2008) SOAP: short oligonucleotide alignment program. *Bioinformatics*, **24**, 713–714.

200. Li, H. and Homer, N. (2010) A survey of sequence alignment algorithms for next-generation sequencing. *Briefings Bioinf.*, **11** (5), 473–483, doi: 10.1093/bib/bbq015.

201. Ruffalo, M., LaFramboise, T., and Koyut-rk, M. (2011) Comparative analysis of algorithms for next-generation sequencing read alignment. *Bioinformatics*, **27** (20), 2790–2796, doi: 10.1093/bioinformatics/btr477.

202. Ku, C.S., Loy, E.Y., Salim, A., Pawitan, Y., and Chia, K.S. (2010) The discovery of human genetic variations and their use as disease markers: past, present and future. *J. Hum. Genet.*, **55** (7), 403–415.

203. Sharp, A.J., Mefford, H.C., Li, K., Baker, C., Skinner, C., Stevenson, R.E., Schroer, R.J., Novara, F., De Gregori, M., Ciccone, R. et al. (2008) A recurrent 15q13. 3 microdeletion syndrome associated with mental retardation and seizures. *Nat. Genet.*, **40** (3), 322–328.

204. McLendon, R., Friedman, A., Bigner, D., Van Meir, E.G., Brat, D.J., Mastrogianakis, G.M., Olson, J.J., Mikkelsen, T., Lehman, N., Aldape, K. et al. (2008) Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, **455** (7216), 1061–1068.

205. Mitelman, F., Johansson, B., and Mertens, F. (2007) The impact of translocations and gene fusions on cancer causation. *Nat. Rev. Cancer*, **7** (4), 233–245.

206. Guo, X., Zhang, J., Cai, Z., Du, D.Z., and Pan, Y. (2015) DAM: a bayesian method for detecting genome-wide associations on multiple diseases, in *Bioinformatics Research and Applications*, Springer-Verlag, pp. 96–107.

207. Guo, X., Yu, N., Gu, F., Ding, X., Wang, J., and Pan, Y. (2014) Genome-wide interaction-based association of human diseases-a survey. *Tsinghua Sci. Technol.*, **19** (6), 596–616.

208. Ding, X., Wang, J., Zelikovsky, A., Guo, X., Xie, M., Pan, Y. (2015) Searching high-order SNP combinations for complex diseases based on energy distribution difference. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **12** (3), 695–704.

209. Tamaki, K. and Jeffreys, A.J. (2005) Human tandem repeat sequences in forensic DNA typing. *Legal Med.*, **7** (4), 244–250.

210. Karas-Kuzelicki, N. and Mlinaric-Rascan, I. (2009) Individualization of thiopurine therapy: thiopurine S-methyltransferase and beyond. *Pharmacogenomics*, **10** (8), 1309–1322.

211. Katsanis, S.H. and Katsanis, N. (2013) Molecular genetic testing and the future of clinical genomics. *Nat. Rev. Genet.*, **14** (6), 415–426.

212. Pulley, J.M., Denny, J.C., Peterson, J.F., Bernard, G.R., Vnencak-Jones, C.L., Ramirez, A.H., Delaney, J.T., Bowton, E., Brothers, K., Johnson, K. et al. (2012) Operational implementation of prospective genotyping for personalized medicine: the design of the vanderbilt predict project. *Clin. Pharmacol. Ther.*, **92** (1), 87–95.

213. Keinan, A. and Clark, A.G. (2012) Recent explosive human population growth has resulted in an excess of rare genetic variants. *science*, **336** (6082), 740–743.

214. Curtis, K., Talwalkar, A., Zaharia, M., Fox, A., and Patterson, D.A. (2015) SiRen: leveraging similar regions for efficient & accurate variant calling.

215. Metzker, M.L. (2010) Sequencing technologies-the next generation. *Nat. Rev. Genet.*, **11** (1), 31–46.

216. Ahmed, M., Ahmad, I., and Ahmad, M.S. (2015) A survey of genome sequence assembly techniques and algorithms using high-performance computing. *J. Supercomput.*, **71** (1), 293–339.

217. Consortium, G.P. et al. (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491** (7422), 56–65.

218. Weinstein, J.N., Collisson, E.A., Mills, G.B., Shaw, K.R.M., Ozenberger, B.A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J.M., Network, C.G.A.R. et al. (2013) The cancer genome atlas pan-cancer analysis project. *Nat. Genet.*, **45** (10), 1113–1120.

219. Ng, S.B., Buckingham, K.J., Lee, C., Bigham, A.W., Tabor, H.K., Dent, K.M., Huff, C.D., Shannon, P.T., Jabs, E.W., Nickerson, D.A. et al. (2010) Exome sequencing identifies the cause of a mendelian disorder. *Nat. Genet.*, **42** (1), 30–35.

220. Guo, X., Meng, Y., Yu, N., and Pan, Y. (2014) Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering. *BMC Bioinf.*, **15** (1), 102.

221. Standish, K.A., Carland, T.M., Lockwood, G.K., Pfeiffer, W., Tatineni, M., Huang, C., Lamberth, S., Cherkas, Y., Brodmerkel, C., Jaeger, E. et al. (2015) Group-based variant calling leveraging next-generation supercomputing for large-scale whole-genome sequencing studies. *BMC Bioinf.*, **16** (1), 304.

222. Auwera, G.A., Carneiro, M.O., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J. et al. (2013) From fastq data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Curr. Protoc. Bioinf.*, **43**, 11.10.1–11.10.33.

223. Guo, X., Yu, N., Ding, X., Wang, J., and Pan, Y. (2015) DIME: a novel framework for de novo metagenomic sequence assembly. *J. Comput. Biol.*, **22** (2), 159–177.

224. Redon, R., Ishikawa, S., Fitch, K.R., Feuk, L., Perry, G.H., Andrews, T.D., Fiegler, H., Shapero, M.H., Carson, A.R., Chen, W. et al. (2006) Global variation in copy number in the human genome. *Nature*, **444** (7118), 444–454.

225. MacDonald, J.R., Ziman, R., Yuen, R.K., Feuk, L., and Scherer, S.W. (2014) The database of genomic variants: a curated collection of structural variation in the human genome. *Nucleic Acids Res.*, **42** (D1), D986–D992.

226. Wang, J., Wang, W., Li, R., Li, Y., Tian, G., Goodman, L., Fan, W., Zhang, J., Li, J., Zhang, J. et al. (2008) The diploid genome sequence of an asian individual. *Nature*, **456** (7218), 60–65.

227. McCarroll, S.A., Kuruvilla, F.G., Korn, J.M., Cawley, S., Nemesh, J., Wysoker, A., Shapero, M.H., de Bakker, P.I., Maller, J.B., Kirby, A. et al. (2008) Integrated detection and population-genetic analysis of SNPS and copy number variation. *Nat. Genet.*, **40** (10), 1166–1174.

228. Freeman, J.L., Perry, G.H., Feuk, L., Redon, R., McCarroll, S.A., Altshuler, D.M., Aburatani, H., Jones, K.W., Tyler-Smith, C., Hurles, M.E. et al. (2006) Copy number variation: new insights in genome diversity. *Genome Res.*, **16** (8), 949–961.

229. Patro, R., Mount, S.M., and Kingsford, C. (2014) Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.*, **32** (5), 462–464.

230. Gardner, S.N. and Hall, B.G. (2013) When whole-genome alignments just won't work: KSNP v2 software for alignment-free SNP discovery and phylogenetics of hundreds of microbial genomes.

231. Hauser, M., Mayer, C.E., and Söding, J. (2013) kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinf.*, **14** (1), 248.

232. Ghodsi, M., Liu, B., and Pop, M. (2011) DNACLUST: accurate and efficient clustering of phylogenetic marker genes. *BMC Bioinf.*, **12** (1), 271.

233. Chor, B., Horn, D., Goldman, N., Levy, Y., Massingham, T. et al. (2009) Genomic DNA k-mer spectra: models and modalities. *Genome Biol.*, **10** (10), R108.

234. Zhou, W., Zhao, H., Chong, Z., Mark, R.J., Eterovic, A.K., Meric-Bernstam, F., and Chen, K. (2015) ClinSeK: a targeted variant characterization framework for clinical sequencing. *Genome Med.*, **7** (1), 1–9.

235. Lam, H.Y., Mu, X.J., Stütz, A.M., Tanzer, A., Cayting, P.D., Snyder, M., Kim, P.M., Korbel, J.O., and Gerstein, M.B. (2010) Nucleotide-resolution analysis of structural variants using breakseq and a breakpoint library. *Nat. Biotechnol.*, **28** (1), 47–55.

236. Kidd, J.M., Sampas, N., Antonacci, F., Graves, T., Fulton, R., Hayden, H.S., Alkan, C., Malig, M., Ventura, M., Giannuzzi, G. et al. (2010) Characterization of missing human genome sequences and copy-number polymorphic insertions. *Nat. Methods*, **7** (5), 365–371.

237. Höhl, M. and Ragan, M.A. (2007) Is multiple-sequence alignment required for accurate inference of phylogeny? *Syst. Biol.*, **56** (2), 206–221.

238. Langmead, B., Trapnell, C., Pop, M., Salzberg, S.L. et al. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10** (3), R25.

239. Langmead, B. and Salzberg, S.L. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9** (4), 357–359.

240. Li, H. and Durbin, R. (2009) Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25** (14), 1754–1760.

241. Li, H., Ruan, J., and Durbin, R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18** (11), 1851–1858.

242. Li, R., Yu, C., Li, Y., Lam, T.W., Yiu, S.M., Kristiansen, K., and Wang, J. (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25** (15), 1966–1967.

243. Hasan, M.S., Wu, X., and Zhang, L. (2015) Performance evaluation of indel calling tools using real short-read data. *Hum. Genomics*, **9** (1), 1–14.

244. Harris, S.R., Török, M., Cartwright, E.J., Quail, M.A., Peacock, S.J., and Parkhill, J. (2013) Read and assembly metrics inconsequential for clinical utility of whole-genome sequencing in mapping outbreaks. *Nat. Biotechnol.*, **31** (7), 592–594.

245. Perna, N.T., Plunkett, G., Burland, V., Mau, B., Glasner, J.D., Rose, D.J., Mayhew, G.F., Evans, P.S., Gregor, J., Kirkpatrick, H.A. et al. (2001) Genome sequence of enterohaemorrhagic escherichia coli o157: H7. *Nature*, **409** (6819), 529–533.

246. Loman, N.J., Misra, R.V., Dallman, T.J., Constantinidou, C., Gharbia, S.E., Wain, J., and Pallen, M.J. (2012) Performance comparison of benchtop high-throughput sequencing platforms. *Nat. Biotechnol.*, **30** (5), 434–439.

247. Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O., and Salzberg, S.L. (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27** (11), 2369–2376.

248. Angiuoli, S.V. and Salzberg, S.L. (2011) Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics*, **27** (3), 334–342.

249. Treangen, T.J., Ondov, B.D., Koren, S., and Phillippy, A.M. (2014) The harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes. *Genome Biol.*, **15** (524), 2.

250. Trappe, K., Emde, A.K., Ehrlich, H.C., and Reinert, K. (2014) Gustaf: detecting and correctly classifying SVs in the NGS twilight zone. *Bioinformatics*, **30** (24), 3484–3490.

251. Bartenhagen, C. and Dugas, M. (2015) Robust and exact structural variation detection with paired-end and soft-clipped alignments: SoftSV compared with eight algorithms. *Briefings Bioinf.*, **17** (1), 51–62.

252. Weiner, P. (1973) Linear pattern matching algorithms, in *Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium on*, IEEE, pp. 1–11.

253. Gusfield, D. (1997) Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology, 1st edition. Cambridge University Press, New York, p. 556, ISBN-10: 0521585198; ISBN-13: 978-0521585194.

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

255. Delcher, A.L., Phillippy, A., Carlton, J., and Salzberg, S.L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30** (11), 2478–2483.

256. Döring, A., Weese, D., Rausch, T., and Reinert, K. (2008) SeqAn an efficient, generic c++ library for sequence analysis. *BMC Bioinf.*, **9** (1), 11.

257. Rausch, T., Emde, A.K., Weese, D., Döring, A., Notredame, C., and Reinert, K. (2008) Segment-based multiple sequence alignment. *Bioinformatics*, **24** (16), i187–i192.

258. Castillo-Ramírez, S., Corander, J., Marttinen, P., Aldeljawi, M., Hanage, W.P., Westh, H., Boye, K., Gulay, Z., Bentley, S.D., Parkhill, J. et al. (2012) Phylogeographic variation in recombination rates within a global clone of methicillin-resistant staphylococcus aureus. *Genome Biol.*, **13** (12), R126.

259. Deloger, M., El Karoui, M., and Petit, M.A. (2009) A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. *J. Bacteriol.*, **191** (1), 91–99.

260. Garrison, E. and Marth, G. (2012) Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*.

261. Ma, B., Tromp, J., and Li, M. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18** (3), 440–445.

262. Ewing, B., Hillier, L., Wendl, M.C., and Green, P. (1998) Base-calling of automated sequencer traces usingphred. I. Accuracy assessment. *Genome Res.*, **8** (3), 175–185.

263. Kehr, B., Weese, D., and Reinert, K. (2011) STELLAR: fast and exact local alignments. *BMC Bioinf.*, **12** (Suppl. 9), S15.

264. Abyzov, A. and Gerstein, M. (2011) AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision. *Bioinformatics*, **27** (5), 595–603.

265. Huang, W., Li, L., Myers, J.R., and Marth, G.T. (2012) ART: a next-generation sequencing read simulator. *Bioinformatics*, **28** (4), 593–594.

266. Balzer, S., Malde, K., Lanzén, A., Sharma, A., and Jonassen, I. (2010) Characteristics of 454 pyrosequencing data-enabling realistic simulation with flowsim. *Bioinformatics*, **26** (18), i420–i425.

267. Shcherbina, A. (2014) FASTQSim: platform-independent data characterization and in silico read generation for NGS datasets. *BMC Res. Notes*, **7** (1), 533.

268. McElroy, K.E., Luciani, F., and Thomas, T. (2012) GemSIM: general, error-model based simulator of next-generation sequencing data. *BMC Genomics*, **13** (1), 74.

269. Angly, F.E., Willner, D., Rohwer, F., Hugenholtz, P., and Tyson, G.W. (2012) Grinder: a versatile amplicon and shotgun sequence simulator. *Nucleic Acids Res.*, **40** (12), e94.

270. Hu, X., Yuan, J., Shi, Y., Lu, J., Liu, B., Li, Z., Chen, Y., Mu, D., Zhang, H., Li, N. et al. (2012) piRS: profile-based illumina pair-end reads simulator. *Bioinformatics*, **28** (11), 1533–1535.

271. Bartenhagen, C. and Dugas, M. (2013) RSVSim: an R/Bioconductor package for the simulation of structural variations. *Bioinformatics*, **29** (13), 1679–1681.

272. Pattnaik, S., Gupta, S., Rao, A.A., and Panda, B. (2014) SInC: an accurate and fast error-model based simulator for SNPs, Indels and CNVs coupled with a read generator for short-read sequence data. *BMC Bioinf.*, **15** (1), 40.

273. Pratas, D., Pinho, A.J., and Rodrigues, J.M. (2014) XS: a FASTQ read simulator. *BMC Res. Notes*, **7** (1), 40.

274. Tian, R., Basu, M.K., and Capriotti, E. (2015) Computational methods and resources for the interpretation of genomic variants in cancer. *BMC Genomics*, **16** (Suppl. 8), S7.

275. Liu, B., Morrison, C.D., Johnson, C.S., Trump, D.L., Qin, M., Conroy, J.C., Wang, J., and Liu, S. (2013) Computational methods for detecting copy number variations in cancer genome using next generation sequencing: principles and challenges. *Oncotarget*, **4** (11), 1868.

276. Gilbert, L.A., Larson, M.H., Morsut, L., Liu, Z., Brar, G.A., Torres, S.E., Stern-Ginossar, N., Brandman, O., Whitehead, E.H., Doudna, J.A. et al. (2013) CRISPR-mediated modular RNA-guided regulation of transcription in eukaryotes. *Cell*, **154** (2), 442–451.

277. Huntzinger, E. and Izaurralde, E. (2011) Gene silencing by microRNAs: contributions of translational repression and mRNA decay. *Nat. Rev. Genet.*, **12** (2), 99–110.

278. Castel, S.E. and Martienssen, R.A. (2013) RNA interference in the nucleus: roles for small RNAs in transcription, epigenetics and beyond. *Nat. Rev. Genet.*, **14** (2), 100–112.

279. Ray, S.S. and Pal, S.K. (2013) RNA secondary structure prediction using soft computing. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **10** (1), 2–17.

280. Gutell, R.R., Lee, J.C., and Cannone, J.J. (2002) The accuracy of ribosomal RNA comparative structure models. *Curr. Opin. Struct. Biol.*, **12** (3), 301–310.

281. Neerincx, P.B. and Leunissen, J.A. (2005) Evolution of web services in bioinformatics. *Briefings Bioinf.*, **6** (2), 178–188.

282. Cheng, J., Tegge, A.N., and Baldi, P. (2008) Machine learning methods for protein structure prediction. *IEEE Rev. Biomed. Eng.*, **1**, 41–49.

283. Koculi, E., Cho, S.S., Desai, R., Thirumalai, D., and Woodson, S.A. (2012) Folding path of P5abc RNA involves direct coupling of secondary and tertiary structures. *Nucleic Acids Res.*, **40** (16), 8011–8020.

284. Tinoco, I. and Bustamante, C. (1999) How RNA folds. *J. Mol. Biol.*, **293** (2), 271–281.

285. Mathews, D.H. (2006) Predicting RNA secondary structure by free energy minimization. *Theor. Chem. Acc.*, **116** (1-3), 160–168.

286. Grüner, W., Giegerich, R., Strothmann, D., Reidys, C., Weber, J., Hofacker, I.L., Stadler, P.F., and Schuster, P. (1996) Analysis of RNA sequence structure maps by exhaustive enumeration II. Structures of neutral networks and shape space covering. *Monatsh. Chemie/Chemical Mon.*, **127** (4), 375–389.

287. Freyhult, E., Moulton, V., and Gardner, P. (2005) Predicting RNA structure using mutual information. *Appl. Bioinf.*, **4** (1), 53–59.

288. Gutell, R., Power, A., Hertz, G., Putz, E., and Stormo, G. (1992) Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *Nucleic Acids Res.*, **20** (21), 5785–5795.

289. Nussinov, R. and Jacobson, A.B. (1980) Fast algorithm for predicting the secondary structure of single-stranded rna. *Proc. Natl. Acad. Sci. U.S.A.*, **77** (11), 6309–6313.

290. Hofacker, I.L., Fekete, M., and Stadler, P.F. (2002) Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.*, **319** (5), 1059–1066.

291. Knudsen, B. and Hein, J. (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res.*, **31** (13), 3423–3428.

292. Mak, D.Y. and Benson, G. (2009) Consensus RNA secondary structure prediction by ranking K-length stems, in *BIOCOMP*, pp. 521–527.

293. Marti-Renom, M.A. and Capriotti, E. (2008) Computational RNA structure prediction. *Curr. Bioinf.*, **3** (1), 32–45.

294. Sato, K., Kato, Y., Akutsu, T., Asai, K., and Sakakibara, Y. (2012) DAFS: simultaneous aligning and folding of RNA sequences via dual decomposition. *Bioinformatics*, **28** (24), 3218–3224.

295. Lin, C., Tang, Z., Jiang, Y., Yang, C., and Zou, Q. (2011) GA combined with structural comparison to improve the prediction of RNA secondary structure from comparative sequence alignment. *J. Convergence Inf. Technol.*, **6** (11), 400–408.

296. Ruan, J., Stormo, G.D., and Zhang, W. (2004) An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, **20** (1), 58–66.

297. Bernhart, S.H., Hofacker, I.L., Will, S., Gruber, A.R., and Stadler, P.F. (2008) RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinf.*, **9** (1), 474.

298. Sankoff, D. (1985) Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, **45** (5), 810–825.

299. Meyer, I.M. and Miklós, I. (2007) SimulFold: simultaneously inferring RNA structures including pseudoknots, alignments, and trees using a Bayesian MCMC framework. *PLoS Comput. Biol.*, **3** (8), e149.

300. Xu, X., Ji, Y., and Stormo, G.D. (2007) RNA Sampler: a new sampling based algorithm for common RNA secondary structure prediction and structural alignment. *Bioinformatics*, **23** (15), 1883–1891.

301. Do, C.B., Foo, C.S., and Batzoglou, S. (2008) A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics*, **24** (13), i68–i76.

302. Will, S., Reiche, K., Hofacker, I.L., Stadler, P.F., and Backofen, R. (2007) Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput. Biol.*, **3**, (4), e65.

303. McCaskill, J.S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

304. Nussinov, R., Pieczenik, G., Griggs, J.R., and Kleitman, D.J.(1978) Algorithms for loop matchings. *SIAM J. Appl. Math.*, **35** (1), 68–82.

305. Shapiro, B.A. and Zhang, K. (1990) Comparing multiple RNA secondary structures using tree comparisons. *Comput. Appl. Biosci. CABIOS*, **6** (4), 309–318.

306. Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S. (2003) Local similarity in RNA secondary structures, in *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE, IEEE*, pp. 159–168.

307. Siebert, S. and Backofen, R. (2005) MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, **21** (16), 3352–3359.

308. Jiang, T., Wang, L., and Zhang, K. (1995) Alignment of trees-an alternative to tree edit. *Theor. Comput. Sci.*, **143** (1), 137–148.

309. Gorodkin, J., Stricklin, S.L., and Stormo, G.D. (2001) Discovering common stem–loop motifs in unaligned RNA sequences. *Nucleic Acids Res.*, **29** (10), 2135–2144.

310. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000) The protein data bank. *Nucleic Acids Res.*, **28** (1), 235–242.

311. Andronescu, M., Bereg, V., Hoos, H.H., and Condon, A. (2008) RNA STRAND: the RNA secondary structure and statistical analysis database. *BMC Bioinf.*, **9** (1), 340.

312. Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S.R., and Bateman, A. (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, **33** (Suppl. 1), D121–D124.

313. Gardner, P.P. and Giegerich, R. (2004) A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinf.*, **5** (1), 140.

314. Kabsch, W. and Sander, C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22** (12), 2577–2637.

315. Chou, P.Y. and Fasman, G.D. (1974) Prediction of protein conformation. *Biochemistry*, **13** (2), 222–245.

316. Kloczkowski, A., Ting, K.L., Jernigan, R., and Garnier, J. (2002) Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence. *Proteins Struct. Funct. Bioinf.*, **49** (2), 154–166.

317. Pavlopoulou, A. and Michalopoulos, I. (2011) State-of-the-art bioinformatics protein structure prediction tools (review). *Int. J. Mol. Med.*, **28** (3), 295–310.

318. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J.(1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25** (17), 3389–3402, doi: 10.1093/nar/25.17.3389.

319. Yaseen, A. and Li, Y. (2014) Context-based features enhance protein secondary structure prediction accuracy. *J. Chem. Inf. Model.*, **54** (3), 992–1002.

320. Rata, I.A., Li, Y., and Jakobsson, E. (2010) Backbone statistical potential from local sequence-structure interactions in protein loops. *J. Phys. Chem. B*, **114** (5), 1859–1869.

321. Garnier, J., Gibrat, J.F., and Robson, B. (1996) GOR method for predicting protein secondary structure from amino acid sequence. *Methods Enzymol.*, **266**, 540–553. PubMed PMID: 8743705.

322. Li, Q., Dahl, D.B., Vannucci, M., Joo, H., and Tsai, J.W. (2014) Bayesian model of protein primary sequence for secondary structure prediction. *PLoS ONE*, **9** (10), e109832.

323. Joo, H., Chavan, A.G., Phan, J., Day, R., and Tsai, J. (2012) An amino acid packing code for α-helical structure and protein design. *J. Mol. Biol.*, **419** (3), 234–254.

324. Joo, H. and Tsai, J. (2014) An amino acid code for β-sheet packing structure. *Proteins Struct. Funct. Bioinf.*, **82** (9), 2128–2140.

325. Sormanni, P., Camilloni, C., Fariselli, P., and Vendruscolo, M. (2015) The s2D method: simultaneous sequence-based prediction of the statistical populations of ordered and disordered regions in proteins. *J. Mol. Biol.*, **427** (4), 982–996.

326. Habchi, J., Tompa, P., Longhi, S., and Uversky, V.N. (2014) Introducing protein intrinsic disorder. *Chem. Rev.*, **114** (13), 6561–6588.

327. Babu, M.M., van der Lee, R., de Groot, N.S., and Gsponer, J. (2011) Intrinsically disordered proteins: regulation and disease. *Curr. Opin. Struct. Biol.*, **21** (3), 432–440.

328. Uversky, V.N., Oldfield, C.J., and Dunker, A.K. (2008) Intrinsically disordered proteins in human diseases: introducing the D2 concept. *Annu. Rev. Biophys.*, **37**, 215–246.

329. Kihara, D. (2005) The effect of long-range interactions on the secondary structure formation of proteins. *Protein Sci.*, **14** (8), 1955–1963.

INDEX

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications, First Edition. Ken Nguyen, Xuan Guo, and Yi Pan.

Wiley Series on

Bioinformatics: Computational Techniques and Engineering

Bioinformatics and computational biology involve the comprehensive application of mathematics, statistics, science, and computer science to the understanding of living systems. Research and development in these areas require cooperation among specialists from the fields of biology, computer science, mathematics, statistics, physics, and related sciences. The objective of this book series is to provide timely treatments of the different aspects of bioinformatics spanning theory, new and established techniques, technologies and tools, and application domains. This series emphasizes algorithmic, mathematical, statistical, and computational methods that are central in bioinformatics and computational biology.

Series Editors: **Professor Yi Pan and Professor Albert Y. Zomaya**
pan@cs.gsu.edu zomaya@it.usyd.edu.au

Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications
Xiaohua Hu and Yi Pan

Grid Computing for Bioinformatics and Computational Biology
Edited by El-Ghazali Talbi and Albert Y. Zomaya

Bioinformatics Algorithms: Techniques and Applications
Ion Mandiou and Alexander Zelikovsky

Machine Learning in Bioinformatics
Yanqing Zhang and Jagath C. Rajapakse

Biomolecular Networks
Luonan Chen, Rui-Sheng Wang, and Xiang-Sun Zhang

Computational Systems Biology
Huma Lodhi

Analysis of Biological Networks
Edited by Björn H. Junker and Falk Schreiber

Computational Intelligence and Pattern Analysis in Biological Informatics
Edited by Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Jason T. L. Wang

Mathematics of Bioinformatics: Theory, Practice, and Applications
Matthew He and Sergey Petoukhov

Introduction to Protein Structure Prediction: Methods and Algorithms
Huzefa Rangwala and George Karypis

Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications
Edited by Mourad Elloumi and Albert Y. Zomaya

Mathematical and Computational Methods in Biomechanics of Human Skeletal Systems: An Introduction
Jiří Nedoma, Jiří Stehlik

Rough-Fuzzy Pattern Recognition: Applications in Bioinformatics and Medical Imaging
Pradipta Maji and Sankar K. Pal

Data Management of Protein Interaction Networks
Mario Cannataro and Pietro Hiram Guzzi

Algorithmic and Artificial Intelligence Methods for Protein Bioinformatics
Yi Pan, Jianxin Wang, and Min Li

Classification Analysis of DNA Microarrays
Leif E. Petersen

Biological Knowledge Discovery Handbook: Processing, Mining, and Postprocessing of Biological Data
Edited by Mourad Elloumi and Albert Y. Zomaya

Computational Methods for Next Generation Sequencing Data Analysis
Ion Mandoiu and Alexander Zelikovsky

Pattern Recognition in Computational Molecular Biology: Techniques and Approaches
Mourad Elloumi, Costas S Iliopoulos, Jason T.L Wang, and Albert Y. Zomaya

Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications
Ken Nguyen, Xuan Guo, and Yi Pan

Evolutionary Computation in Gene Regulatory Network Research
Hitoshi Iba and Nasimul Noman

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.