SELECTING THE "CLOSEST TO OPTIMAL" MULTIPLE SEQUENCE ALIGNMENT USING MULTI-LAYER PERCEPTRON

by

Catherine Anderson

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professors Stephen Scott and Etsuko Moriyama

Lincoln, Nebraska

May, 2017

SELECTING THE "CLOSEST TO OPTIMAL" MULTIPLE SEQUENCE ALIGNMENT USING MULTI-LAYER PERCEPTRON

Catherine Anderson, Ph.D.

University of Nebraska, 2017

Adviser: Professors Stephen Scott and Etsuko Moriyama

Many bioinformatics analyses use multiple sequence alignments (MSAs) as their input data. Therefore, the quality of an MSA is critical. When selecting an MSA, users often rely on the overall accuracy reported in published studies where various MSA programs are evaluated using only a small number of benchmark datasets. For protein sequences, such benchmark alignments are often generated based on protein 3D-structure information, limiting the numbers and types of alignments that can be tested. The main objective of this study is to develop a method that can improve the quality of MSAs. Toward this goal, we first developed SuiteMSA, a graphical MSA viewing and assessment software package. It helps users to visually and quantitatively assess MSAs produced by any automated programs. A learning problem of this nature requires a large number of reference protein alignments and currently available benchmark databases are not sufficiently large nor diverse. Therefore, we constructed a new simulated alignment benchmark database, SimDom. It includes a large number of protein sets with a wide range of properties and levels of divergence as well as multi-domain architectures. Using this benchmark, we evaluated the performance of five MSA programs and developed a system of measures that quantify the shift in performance between the programs. We determined which aspects of the sequence sets and resulting alignments influenced the performance shift. Based on this knowledge, we developed a multi-class classifier based on a multi-layer perceptron

to select the alignment closest to the optimal. Using this "**S**e**L**ecting an **A**lignment **P**rogram" (**SLAP**) classifier, we successfully increased the average quality score of the selected alignments by as much as 0.052 for the simulated datasets and by as much as 0.259 for the non-simulated datasets over the best effort of a single program. Successful selection of the alignment closest to the optimum will allow for better results from downstream analyses and thus contribute to the improvement of various bioinformatics and molecular evolutionary analyses.

DEDICATION

This is dedicated to my best friend and my companion, Ronald Lee Heyen, without whose loving support I would have had a much harder row to hoe.

ACKNOWLEDGMENTS

I would like to acknowledge and thank the following people for their time and assistance:

Etsuko Moriyama, who has supported me both financially and academically through the whole PhD program.

Stephen Scott, for our many academic and professional discussions.

Leen-kiat Soh, for stepping in at the last minute to be first reader in spite of being on sabbatical.

Steven Dunbar, for stepping in at the last minute to be second reader.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Table 1: List of Acronyms and Abbreviations

Guide tree topologies

FU	**F**ull **U**ltrametric
FN	**F**ull **N**on-ultrametric
CU	**C**lustered **U**ltrametric
CN	**C**lustered **N**on-ultrametric
RU	**R**andom **U**ltrametric
RN	**R**andom **N**on-ultrametric

Alignment programs

MUSCLE	Muscle
LINSI	MAFFT L-INS-i algorithm
PROB	Probalign
CLTW2	ClustalW version 2
OMEGA	Clustal-Omega

Benchmark databases

BB3	BAliBASE 3
OX3	OXBench
HOM	HOMSTRAD

Metrics

CSS	Cline shift score
SPS	Sum of pairs score (developer score)
CS	Column score
INFO	Information score

Table 2: List of Alignment Quality Measures

Quality measure for each sequence set

CSS(id, p)	CSS for alignment from sequence set with id and program p
maxCSS(id)	maximum CSS between the ve alignments for a specic sequence set with id
$\Delta 12$	difference between maxCSS and the second highest CSS between the five alignments
$\Delta 23$	difference between the second highest CSS and the third highest CSS between the five alignments
$\Delta 34$	difference between the third highest CSS and the fourth highest CSS between the five alignments
$\Delta 45$	difference between the fourth highest CSS and the lowest CSS between the five alignments
$\Delta 15$	difference between maxCSS and the lowest CSS between the five alignments

Quality measure for a dataset

BEST	average maxCSS for a dataset
CSS_{SLAP}	average CSS of the **SLAP** predicted alignment for a specific dataset
CSS_{CLTW2}	average CSS of the CLTW2 alignment on a specific dataset
CSS_{LINSI}	average CSS of the LINSI alignment on a specific dataset
CSS_{MUSCLE}	average CSS of the MUSCLE alignment on a specific dataset
CSS_{PROB}	average CSS of the PROB alignment on a specific dataset
CSS_{OMEGA}	average CSS of the OMEGA alignment on a specific dataset
$\Delta\text{BP}(p)$	BEST for a specific dataset - CSS_p where p is the program used to generate the alignment
$\Delta\text{SP}(p)$	$\text{CSS}_{SLAP} -$ CSS_p for a specific dataset where p is the program used to generate the alignment

Ratio variables

SLAP_RAT	ratio between the **SLAP** predicted CSS_{SLAP} and the maxCSS
RAT_CLTW2	ratio between the CLTW2 CSS and the maxCSS
RAT_LINSI	ratio between the LINSI CSS and the maxCSS
RAT_MUSCLE	ratio between the MUSCLE CSS and the maxCSS
RAT_PROB	ratio between the PROB CSS and the maxCSS
RAT_OMEGA	ratio between the OMEGA CSS and the maxCSS

Chapter 1

Introduction

1.1 Motivation

The multiple sequence alignment (MSA) plays a central role in nearly all bioinformatics and molecular evolutionary analyses. It is involved in, e.g., similarity search, phylogenetic analysis, identification of conserved motifs and domains, and prediction of protein structures. In all of these tasks, the first step is to compare the sequences by building MSAs. The process of building an MSA is to infer homologous positions between the sequences and place gaps in the sequence in order to align these homologous positions. These gaps represent evolutionary events of their own. Gaps are caused by either insertions or deletions of nucleotides or amino acids (therefore also called indels) on a particular lineage of sequences during the evolution. In this sense, building an MSA is to reconstruct the evolutionary history of the sequences involved.

With the advent of new technology making the sequencing of genomes cheaper and faster, there is an abundance of sequence data to be analyzed. With this increasing volume of data comes an increasing dependence on automated programs to generate MSAs quickly and accurately. Alignment quality reflects how accurately the alignment depicts the evolutionary relation between a set of sequences. Alignment quality is critical to the accuracy of the subsequent studies. For example:

- A higher false positive rate in the screening for positive selection events has been linked to alignment quality [1]. By positive selection, we mean the process by which new advantageous genetic variants enter the population. Decreasing accuracy in the MSA causes non-homologous positions to be erroneously aligned,leading to more sites incorrectly identified to be under positive selection.

- Differences in alignments can cause a disagreement in the outcomes of phylogenetic analysis. For example, Morisson and Ellis [2] found that variations in the MSA produced greater differences in the resulting trees than were produced by using different phylogeny reconstruction methods on the same alignment.

- The prediction of protein 3D-structure is also very sensitive to alignment quality. Raghava et al.[3] stated in reference to protein 3D-structure, "the utility of any prediction is completely dependent on the accuracy of the alignment." This opinion is shared and nicely summed up by D. T. Jones, a professor of Bioinformatics at the University College London and the author of many protein structure prediction methods: "As the familiar joke goes, there are really only three things that govern the overall accuracy of comparative modeling: alignment quality, alignment quality, and ... alignment quality." [4]

Due to its significant impact on many bioinformatics and molecular evolutionary analyses, MSA reconstruction is one of the most heavily scrutinized bioinformatics fields. In the quest for better automated MSAs, numerous MSA reconstruction methods have been developed. Assessment of MSAs, however, is usually reserved for power users. Often regular users simply run one MSA method on default parameters and proceed directly to the next analysis without examining the alignment output. Considering the importance of MSAs, it is desirable if quality assessment of MSA can be

performed more easily and more intuitively by all researchers who are interested in sequence analysis. As Morrison [5] also pointed out, visual inspection of multiple MSAs would greatly help improve the quality of MSAs and consequently the reconstruction of phylogenies and other downstream analyses.

When assessing alignment methods, reports are usually made on the average accuracy of an alignment program using a small number of benchmark datasets generated usually based on structural alignments of proteins. Not only are these benchmark datasets relatively low in the variety of reference alignments they offer, but also they represent only a best educated approximation of a correct alignment. Compounding this with the fact that these studies do not examine the case by case difference in alignment, either qualitatively or quantitatively, then the results can at best be viewed as only vague recommendations. Even if an alignment program had the highest average accuracy across all alignment problems tested, it is not known if all sequence sets of specific characteristics can be aligned with the highest accuracy by this single alignment program, even if this alignment program had the highest average accuracy across all alignment problems tested. In other words, an alignment program that would not be recommended for use due to its lower average accuracy score could actually produce the alignment closest to the optimal on sequence set of different characteristics.

In this study, our main objective was to develop a method that can improve the quality of multiple sequence alignments. To help the user select the alignment that was closer to the optimal alignment, we first developed SuiteMSA, a graphical MSA viewing and assessment program. A project of this nature requires a large number of protein sequence sets along with reference alignments to model as training and testing data. Currently available benchmark databases do not have a sufficient amount of data to successfully train a classifier of sufficient capability to learn this difficult

problem. Therefore, we constructed a large simulated alignment benchmark database, SimDom. The SimDom benchmark MSA database along with utilities provided by SuiteMSA were used to compare the performance of five currently available MSA programs. We determined the characteristics of sequence sets and alignments that indicated where one program would outperform another. Using this knowledge, we developed a classifier that makes the choice of the "closest to optimal" alignment from those generated by a group of alignment programs. The novelty of our approach is that it uses the strengths of each of the five alignment program, each shown to produce high quality alignment on specific sequences set, to help improve the accuracy of alignment over all sequence sets.

1.2 Thesis

No single alignment program consistently produces an alignment closer to the optimal alignment than any other. With a benchmark database consisting of a large number of true alignments of realistic protein sequences, the characterizing attributes of the alignments resulting from different alignment programs can be used to train a multi-class classifier to select an alignment closer to the optimal than if the choice of alignment is made simply by the average performance of the individual alignment program alone.

1.3 Outline of dissertation and contributions

The rest of the dissertation is organized as follows:

- Chapters 2 gives general background information on the concepts of protein sequences, domain structures, protein evolution, multiple sequence alignments (MSAs), and representative MSA methods.

- Chapter 3 discusses the representative metrics used to assess MSA quality. It also discusses the currently available non-simulated benchmark databases and two previous studies that attempted to use multiple programs to improve MSA quality.

- Chapter 4 presents the SuiteMSA package we developed for both visual and computational assessment of MSA. Three novel viewers that assist the user in evaluating protein sequence alignments are highlighted. This work has been published in [6] and [7]

- Chapter 5 discusses the development of a new simulated alignment benchmark database, SimDom, which contains 144,000 sets of multi-domain protein sequences and their true alignments. Our choice of the sequence simulation method, the simulation design, and the novel approach of how the protein sequences and their domain architectures are modeled are described.

- Chapter 6 starts with a survey of the alignment evaluation studies that have been conducted over the last several years. We then evaluate the relative performances of the five alignment programs most frequently used using the SimDom database. We highlight our method of determining the extent of the improvement possible by employing the strength of the five alignment programs on different sequence sets. We then analyze the characteristic of the alignment to determine which can be used as indicators of the shift in relative performance between the individual alignment programs and used as attributes in the data model of our multi-class classifier described in Chapter 7.

- Chapter 7 discusses the design of the data model of the input vector that represent each sequence set used to train the classifier, **S**e**L**ecting an **A**lignment

Program, for the "closest to optimal" problem. This classifier will select an alignment closer to the optimal than if the choice of alignment is made simply by the average performance of the individual alignment program alone. We assembled a set of attributes and performed attribute selection trials. The critical difference between our approach and what has been done in other studies is that we use attributes that describe each alignment from the group of alignments to be chosen from for a specific sequence set. Previous studies confined their attributes to sequence set based characteristics.

- Chapter 8 discusses the training and evaluation of the results for the classifier **SLAP** using the sequence sets from SimDom. We discuss the development of a metric, SLAP_RAT, which shows the alignment quality achieved by the classifier **SLAP** relative to the maximum quality possible from the set of alignment programs. Using this metric, we demonstrated that in spite of the rather low level of classifier accuracy (¡ 66%, measured using the Cline shift score), significant increase in the average alignment quality was observed (as much as 5% depending on the datasets and alignment programs compared against). We further employed the classifier **SLAP** on non-simulated datasets. Again significant increase in the average alignment quality as much as 26% was observed.

- Chapter 9 concludes by summarizing the work performed in the dissertation, the contributions of this work and the work for the future.

1.4 Contributions

The overall objective of this study was to improve the quality of multiple sequence alignments. Our contributions towards this end are as follows:

- **Development of the SeLecting an Alignment Program(SLAP) classifier**. We provided a classifier to help identify alignment closest to the optimal. It achieved a substantial improvement in average alignment accuracy for both simulated and non-simulated protein sequences.

- **Development of SuiteMSA**. We provided a user-friendly program that contains three novel alignment viewers to assist in the visual assessment of MSA quality. These three viewers are as follows:

    1. MSAviewer. It allows graphical assessments of MSAs using functional information such as secondary structures and transmembrane prediction
    2. MSAcomparator. It allows the detailed comparison between two alignments with column-wise alignment quality and conservation scores.
    3. PixelPlot. It allows a large-scale visual comparison among multiple MSAs. It can incorporate functional information such as secondary structures and transmembrane prediction as color schemes.

- **Development of SimDom database**. We created a large-scale simulated protein alignment benchmark database, SimDom. It specifically designed for the evaluation of alignment programs for maximum quality. SimDom has the following advantages:

    1. Availability of true (error free) reference alignments for alignment quality evaluation
    2. Inclusion of a large (1000) protein types incorporating 1750 different domain models
    3. Inclusion of 144 individual evolutionary scenarios per protein type

4. Inclusion of a total of 144,000 sequence sets with true alignments
5. Possibility to be easily supplemented by additional simulations

- **Development and performance of an alignment program evaluation protocol**. We developed an alignment program evaluation protocol that included a system of measures to document relative performance differences in quality between the alignment programs evaluated. Using this protocol we showed that the shift in relative performance was much larger than indicated in previous evaluation studies where only the average quality values had been used. We also revealed trends that have not been seen previously (i.e., CLTW2 can generate much better MSAs from highly divergent sequence sets compared to other MSA methods.

- **Positive impacts for downstream analyses**. Successful selection of the alignment closest to the optimum will allow for better results from downstream analyses. Therefore, the results described in this dissertation contributes to the improvement of various bioinformatics and molecular evolutionary analyses.

Chapter 2

Background on Proteins and Multiple Sequence Alignments

2.1 Amino acids and proteins

Proteins are the products of the genes located in the DNA of an organism and are one of the main the materials with which an organism is built. Each individual protein molecule is made up of a series of amino acids. Amino acids are organic compounds containing an amine (-NH2) group and a carboxyl (-COOH) group (Figure 2.1). Its this side chain gives each amino acid its distinctive characteristic. There are twenty types of amino acids that can be found in proteins. Table 2.1 shows a listing of each amino acid along with its single letter code and their physico-chemical characteristics. Amino acids can be grouped by common physico-chemical characteristics, such as the presence of sulfur in the side chain, the size of the molecule, its polarity or how it reacts with water. Figure 2.2 shows one example of how these twenty amino acids can be grouped.

A protein, also known as a poly-peptide, is a series of amino acids joined together through the peptide bond (Figure 2.1). The peptide bond occurs when the nitrogen in the amine group of one amino acid forms a double bond with the carbon in the carboxyl group of the next amino acid, liberating the oxygen of the carboxyl group and the two hydrogens of the amine group as a molecule of water. The order of amino acids within a protein is encoded by the DNA sequence of a gene. This code is

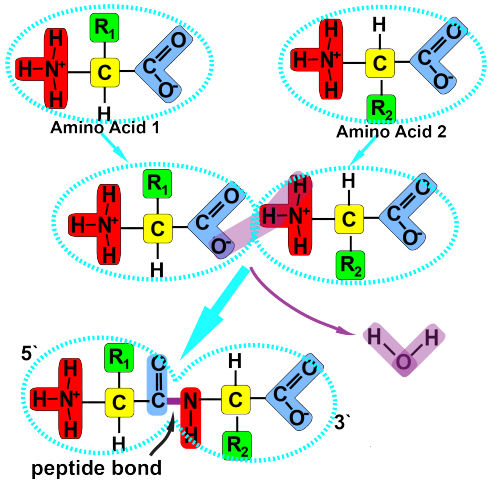


Figure 2.1: The polypeptide bond. A schematic representation of forming a single polypeptide bond is shown. The red background represents the amine group, while the blue background represents the carboxyl group. R_1 and R_2 represent the side chains of the individual amino acid. The purple back ground in the middle and bottom of the figure represents where a peptide bond is formed producing a side-product of one water molecule.

a series of a three-nucleotide set (codon), which is transcribed to the messenger RNA (mRNA), then translated into the amino acid sequence. The amino acid sequence is referred to as the primary structure of the protein.

The secondary structure of the protein occurs when the side chains of the amino acid next to each other in the sequence react in such a way as to cause the amino acids to form compact regular arrangements. Protein secondary structures can be categorized into three general types: 1) α-helix, 2) β-strand or sheet, or 3) loop or coil (Figure 2.3). The α-helix is a compact and rigid helical arrangement of adjacent amino acids. There is one complete turn of the helix for approximately 3.6 amino acids. α-helices can contain anywhere from four to over forty consecutive residues.

A β-strand is a segment of the polypeptide that assumes a flat arrangement. When two or more of these strands are loosely connected by hydrogen bonds, they

Table 2.1: Twenty amino acids. Amino acids are listed with their single letter code and side chain characteristics.

		Side Chain				molecular	
name	code	class	polarity	charge	hydropathy	weight	frequency
Aspartic acid	D	acid/amide	acidic polar	negative	-3.5	133.1	5.49
Glutamic acid	E	acid/amide	acidic polar	negative	-3.5	147.13	6.32
Alanine	A	aliphatic	nonpolar	neutral	1.8	89.09	8.76
Asparagine	N	acid/amide	polar	neutral	-3.5	132.12	3.93
Cysteine	C	sulfur	nonpolar	neutral	2.5	121.15	1.38
Glutamine	Q	acid/amide	polar	neutral	-3.5	146.15	3.9
Glycine	G	aliphatic	nonpolar	neutral	-0.4	75.07	7.03
Isoleucine	I	aliphatic	nonpolar	neutral	4.5	131.18	5.49
Leucine	L	aliphatic	nonpolar	neutral	3.8	131.18	9.68
Methionine	M	sulfur	nonpolar	neutral	1.9	149.21	2.32
Phenylalanine	F	aromatic	nonpolar	neutral	2.8	165.19	3.87
Proline	P	cyclic	nonpolar	neutral	-1.6	115.13	5.02
Serine	S	hydroxyl	polar	neutral	-0.8	105.09	7.14
Threonine	T	hydroxyl	polar	neutral	-0.7	119.12	5.53
Tryptophan	W	aromatic	nonpolar	neutral	-0.9	204.23	1.25
Tyrosine	Y	aromatic	polar	neutral	-1.3	181.19	2.91
Valine	V	aliphatic	nonpolar	neutral	4.2	117.15	6.73
Arginine	R	basic	basic polar	positive	-4.5	174.2	5.78
Lysine	K	basic	basic polar	positive	-3.9	146.19	5.19
Histidine	H	basic	basic polar	positive(10%) neutral(90%)	-3.2	155.16	2.26

form β-sheets, which tend to take a pleated sheet shape. The third category of the secondary structure, a loop or coil (Figure 2.3), is more the absence of either an α-helix or β-strand or sheet. The coil sections give the protein the flexibility it needs to assume its ultimate structure and function. Amino acids have different propensities for forming these secondary structure arrangements. The prediction of their presence from an amino acid sequence is mainly based on this propensity.

The tertiary structure of a protein, referred to as a "fold", is the 3-dimensional structure that a protein assumes when its synthesis is complete. A protein transitions to this structure by what is historically known as the polypeptide "folding" on itself, which is the source of the 3D structure of a protein being referred to as a "fold". Proteins can exist in stable form after assuming their 3D shape. However, there are quite a few that form larger complexes with other proteins, where each individual protein is considered a subunit of the larger structure. When this occurs where proteins are actual subunits of a larger structure, the final complex is referred to as

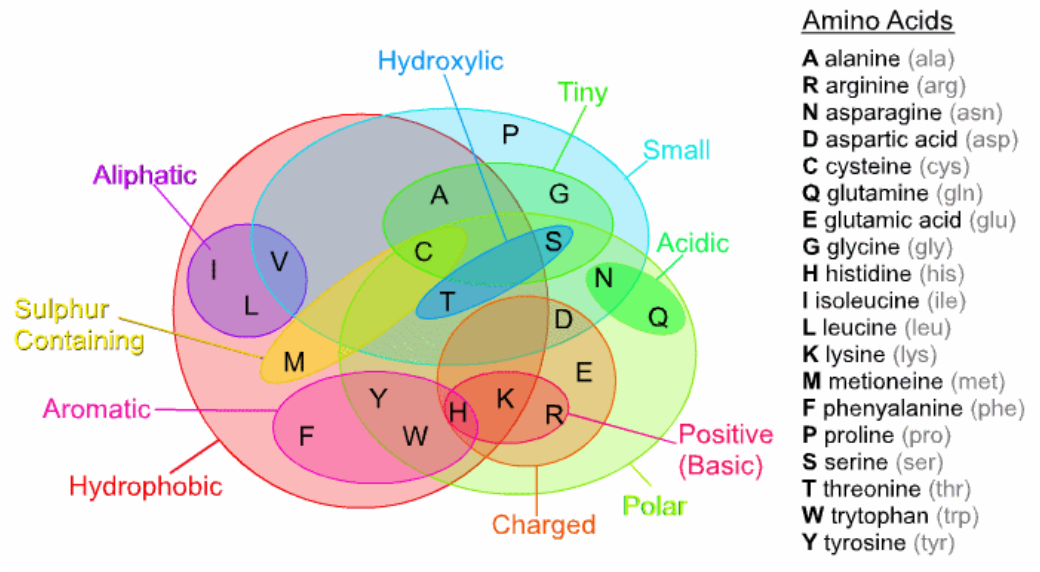


Figure 2.2: Groupings of amino acids. A Venn diagram demonstrates the shared characteristics between amino acids [8].

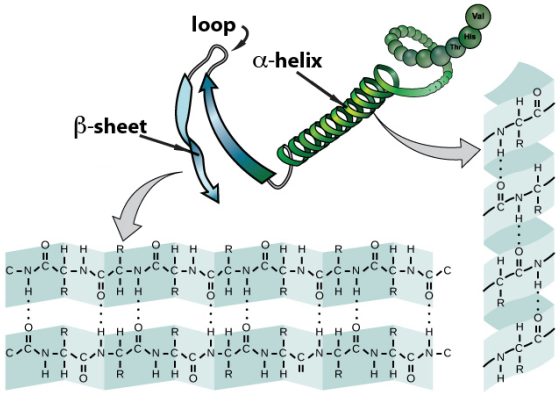


Figure 2.3: Secondary structures of protein. The three predominant secondary structure arrangements for protein, the α-helix and the β-sheet, along with coil or loop section, which is essentially the absence of secondary structure, are illustrated. [9].

the quaternary structure of the protein. Not all proteins have a quaternary structure.

The central dogma of molecular biology is that DNA is transcribed to mRNA, then translated into the amino acid sequence. The amino acid sequence determines the secondary structure, which folds into the tertiary structure and where required,

assembled into the quaternary structure, which provides the function of the protein. The final shape of a protein is critical to its function. Changes to the amino acid sequence can lead to changes in its final structure, which can cause an alteration in its function. By way of evolution, this is how new functions arise. More frequently, however, these changes are deleterious resulting in the weakening and death of the individual organisms, and are eventually eliminated from the population.

2.2 Protein sequence evolution

There are a large number of proteins that occur in natureand they have different structures and functions. The simplest mutation that can occur within a protein is a change in a single amino acid in a specific position of the sequence. If the new amino acid is similar in the specific characteristic critical to the function of the protein as the amino acid replaced, the mutation is more likely to be accepted and passed to the next generation. If the amino acid is too different in this specific characteristic, then the mutation is likely to alter the structure of the protein to the point of rendering the function limited or completely disabled. However, Mother Nature is not so much an inventor as a tinkerer. Therefore proteins that perform the same function usually are more similar in sequence and structure than different. For instance, the human genome shares up to 80% of its genes with cows and up to 60% with chickens [10]. According to the Tree of Life Web Project [10], "Evidence from morphological, biochemical, and gene sequence data suggests that all organisms on Earth are genetically related, and the genealogical relationships of living things can be represented by a vast evolutionary tree, the Tree of Life." Therefore, we share not only genes with other red blooded animals, but also with sharks, shiitake mushrooms, corn, seaweed and bacteria.

Inferring the relationship between different organisms starts by comparing the the sequences of "homologous" genes or proteins that are presumed to have been derived from the common ancestor. Homologous proteins usually maintain similar amino acid sequences and perform the same function. Therefore, these proteins are grouped into the same family of proteins. Protein families can be also grouped into a protein superfamily based on their sequences and functions.

Similarity of two proteins can be quantified based on the similarity of their amino acid sequences. They will most likely not be exactly the same, since with time, DNA changes will have been introduced to to either or both of the two gene sequences, causing changes to the amino acid sequences of the coded proteins.

The evolutionary more diverged the two species are, the less similar their DNA and hence their protein sequences will be. The most frequent change is a substitution mutation in the DNA some of which can introduce a change in the amino acid in a specific site in the sequence. The next type of change is when genetic material has been inserted into the DNA, resulting in the addition of amino acids within the sequence, or, conversely, when genetic material has been deleted from the DNA causing the removal of amino acids from the sequence. The former is referred to as an **insertion event** and the latter as a **deletion** event. Together these are referred to as indel events or simply indels.

2.3 Pairwise protein sequence alignment

In order to compare two sequences, the sequences must be aligned. This means they must be placed, one over the other, with the order in each sequence preserved, so that all amino acids (for protein sequences) that are believed to come from the same position in the ancestral sequence are aligned one over the other.

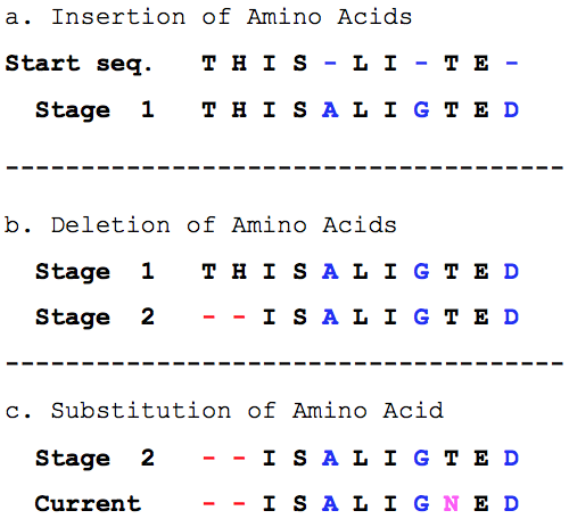


Figure 2.4: An example of a sequence evolution. The series of evolutionary events from the starting sequence (Start seq.) to the current sequence (Current), demonstrating the insertion (a), deletions (b), and substitution event (c) is illustrated. The gap only columns in c are merely place holders for the deletions that occurred in Stage 2.

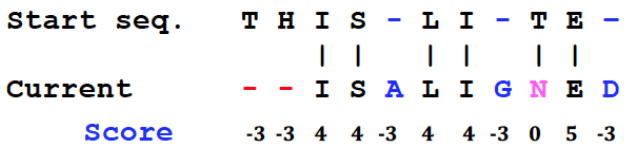


Figure 2.5: Alignment of homologous positions. The homologous positions inherited from the starting sequence (Start seq.) to the current sequence (Current) are aligned and marked with the vertical lines at those positions. The scores listed at the bottom are based on a linear gap penalty, -3 for the each insertion or deletion site, and the BLOSUM 62 scoring matrix for each aligned amino acid pair.

As an example, we will examine the toy problem of two short evolutionary related sequences, the ancestral sequence "T H I S L I T E " and the descendant sequence "I S A L I G N E D". Figure 2.4 shows one possible evolutionary scenario from the starting (ancestral) sequence to the current (descendant) sequence. Three insertion events occurred between the Start sequence and Stage 1 (Figure 2.4.a). Next are the three deletion events between Stages 1 and 2 (Figure 2.4.b). Finally between

B 62	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 2.6: The BLOSUM62 scoring matrix [11]. This matrix is symmetric about the upper left to lower right diagonal. This indicates the value for X_{ij} is equal to X_{ji}.

Stage 2 and Current (Figure 2.4.c), a substitution occurred in the 7^{th} residue of Stage 2, resulting in the current sequence. This progression is only one of many possible evolutionary scenarios.

In essence, to create an alignment is to make a choice between two alternatives for each amino acid position: 1) decide the two amino acids come from the same ancestral position and assign them to the same column or 2) decide that the two amino acids are not from the same ancestral sites and introduce a gap. If an insertion event has occurred in one sequence, dashes are placed in the corresponding columns of the other sequence. Likewise, when a deletion event has occurred in a sequence, dashes are placed in that sequence at the position where the deletion occurred. In this manner,as shown in Figure 2.5, when the process is complete, the two rows of this pairwise alignment are the same length (including gaps).

Making the decision between these two alternatives is assisted by the use of scoring matrices and gap penalties. Scoring matrices express by a numeric value how "likely" the alignment between two specific amino acids is. This value will be at a maximum

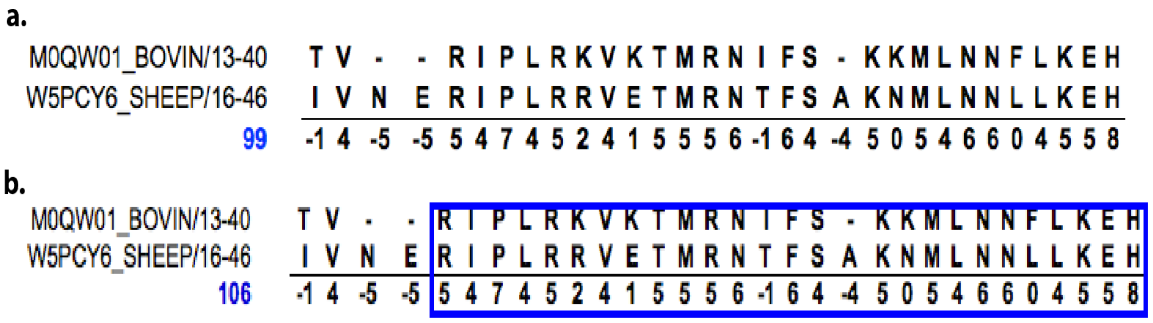


Figure 2.7: Comparison of global (a) and local (b) pairwise alignment. The value for each column is given using the BLOSUM62 scoring matrix and -5 for the gap penalty. A local alignment ends when the running total of the score drops below 0. The blue outline indicates the positions included in the local alignment. The total alignment scores (99 for the global alignment and 106 for the local alignment) are shown in blue numbers.

for an individual amino acid when it is paired with another amino acid of the same kind and will decrease from there for different amino acids. This number therefore reflects the likelihood that the two amino acids appear at homologous positions. The actual calculation for these values vary from matrix to matrix, reflecting the change in likelihood with different evolutionary distances [12][11]. Figure 2.6 gives an example of a scoring matrix, the BLOSUM62 [11]. Notice that the maximum value between like amino acids is not the same. The higher the frequency of the two amino acids occurring as a homologous pair, the higher this maximum value is.

When creating an alignment, the objective is to maximize the alignment score, which is the summation of all column-wise scores. When two residues are aligned in a column, the score for that column is taken from the scoring matrix. When a residue is aligned with a gap, a gap penalty is assigned as the column score. In the toy alignment given in Figure 2.7, where -3 is assessed for a gap and the BLOSUM62 scoring matrix is used for aligned positions, the final score for the alignment is 6.

The gap penalty scheme used in this sample alignment problem is a **linear gap penalty**, where all gaps are treated equally and incur the same penalty. An al-

ternative gap penalty scheme is the **affine gap penalty**. This scheme breaks the penalty into two parts: the gap opening penalty, gp_{open} and the gap extension penalty, gp_{extend}. The total gap penalty, gp for an uninterrupted gap section in an alignment is given as $gp = gp_{open} + gp_{extend}L$ where L is the length of the uninterrupted gap section. This scheme is most often used in many programs both for pairwise and multiple alignments.

The first efficient algorithms that would both align two sequences and guarantee the best alignment based on a specific scoring scheme are named after their developers:

1. The Needleman and Wunsch [13] algorithm creates a global alignment of two sequences where all amino acids in both sequences are aligned. Figure 2.7.a shows the global alignment between two sequences, where a gap penalty of -5 and the BLOSUM62 scoring matrix is used. The total score can be calculated as 99 in this example.

2. The Smith and Waterman [14] algorithm creates a local alignment of two sequences, where only segments of the sequences whose consecutive alignment score is positive are considered (Figure 2.7.b). The segment with the highest score is the final local alignment.

Both of these algorithms are based on dynamic programming and take $\Theta(nm)$ for both time and memory, where n and m are the lengths of the two sequences to be aligned. When the two sequences are of comparable lengths, both algorithms take roughly $\Theta(n^2)$ for both time and memory. Pairwise alignment is used, for example, when calculating the evolutionary distance or identity between two sequences and sequence similarity search.

2.4 Multiple sequence alignment

Bioinformatics studies more often than not involve many more than two sequences. When more than two sequences are involved in an alignment, it is referred to as a **multiple sequence alignment** or **MSA**. An MSA is a representation of the evolutionary relation of the sequences to each other. Any sequence not related to the other, if included in the alignment process, can introduce error to the final alignment.

To extend the dynamic programming solutions described in Section 2.3, to include more than two sequences would increase the complexity to $\Theta(n^k)$, where k is the number of sequences. Any growth rate above $\Theta(n^2)$ quickly becomes intractable in time. As such heuristics had to be devised to generate multiple sequence alignments more quickly than $\Theta(n^3)$ while still maintaining confidence in the solution.

The most used heuristic multiple alignment algorithm is a progressive alignment [15]. Progressive alignment methods require several steps, and how each step is accomplished gives rise to the variety of programs. The basic steps of the progressive alignment algorithm are as follows:

1. Form a pairwise distance matrix which gives a measure of how divergent each pair of sequences is from the other.

2. Create a phylogeny from this distance matrix and use this as a guide tree.

3. Form the alignment using a heuristic, following the guide tree by starting with the most closely related sequences and adding the others in order of relation.

Progressive alignment algorithms makes intuitive sense since more similar sequences are more likely related to one another from a common ancestral sequences and are aligned first. More closely related sequences are aligned easily and these alignments are more likely to be done correctly. Therefore, these sequences are aligned

first. However, the disadvantage of any progressive alignment method is that after a sequence has been added to the growing alignment, it is not possible to go back and correct mistakes from either the alignment procedure itself or for an error in the estimated guide tree. The introduction of error early on in the alignment process can cause larger mistakes toward the end. To correct this potential source of error programs have added a refinement step after the initial progressive alignment process has been completed [16] [17]. Some of these programs are described in the next section.

2.5 Discussion of specific alignment programs

In this section, we will discuss the five alignment programs, all variations of the progressive alignment strategy, that are used in this work. All five are readily available for free download and lend themselves to batch file usage for aligning large numbers of sequence sets which is required for the purpose of this study. The five alignment programs (and their versions) used in this study are:

- ClustalW version 2.1 [18]
- MAFFT version 7.157 [19]
- MUSCLE version 3.8.31 [16]
- Probalign version 1.4 [20]
- Clustal-Omega version 1.2.1.2 [21]

2.5.1 ClustalW2

ClustalW [18] was one of the first progressive alignment programs developed that also came with a user-friendly interface [22], both of which made it one of the most

widely used alignment programs for constructing MSAs [8][17]. ClustalW estimates a guide tree by performing pairwise alignments of all sequences and computing a distance matrix (no multiple hit correction is used for the guide tree construction). This distance matrix is then used to construct the guide tree using the Neighbor-Joining (NJ) method [23]. This guide tree is used for determining the order that the sequences are aligned and the branch length are used for determining the weighting factors for the score during alignment.

The program uses the affine gap penalty scheme. Both the gp_{open} and gp_{extend}, are adjusted as the alignment process proceeds based on several factors: the alignment length, the difference in length of either sequence, amino acid composition and the location of adjacent gaps in the alignment. For example, if a gap has occurred in the same position in the growing alignment, the penalty is lowered. However, if a gap has occurred within eighth positions of adjacent gaps, the penalty is increased. An example of the composition adjustment is the increased penalty for opening a gap for runs of hydrophobic amino acids. This is because indels tend to occur in loop areas, which tend to be hydrophilic, but not in structured areas such as the transmembrane, which are hydrophobic. Sequences are also assigned a weight based on the guide tree which is used when tallying alignment scores, so that the alignment is not overly biased due to a large number of very similar sequences in the group. To accomplish this, the weight factor of the non-similar sequences will be higher than those of the similar. ClustalW does not performs any refinement step after the alignment procedure is complete. In the newer version of ClustalW (version 2 or ClustalW2), an option for the iteration has been included although this option is not the default.

2.5.2 MAFFT (the L-INS-i) algorithm

MAFFT stands for **M**ultiple sequence **A**lignment based on **F**ast **F**ourier **T**ransform [24][17]. It is a program that offers various strategies to optimize the alignment process for a sequence set of different characteristics. We will focus our discussion on only the L-INS-i strategy, which assumes that there is only one conserved area within the sequences. This strategy uses the Smith-Waterman algorithm for the local alignment of two sequences (described in Section 2.3) to establish the distance matrix from which to generate a guide tree using the UPGMA method[25].

Before the alignment process, each of these initial pairwise alignments is broken into n gap-free segments. An "importance value" for each position that occurs in these gap-free segments is calculated based on the frequency that each residue in a gap-free segment actually appears in a gap-free segment. This importance value is used during the progressive alignment step to calculate to assist in aligning two groups of sequences.

After the progressive alignment step, the alignment is then subjected to iterative refinement steps which optimize the objective scores involving the weighted sum-of-pair score and the importance value.

2.5.3 MUSCLE

The major advantage introduced by MUSCLE (**MU**ltiple **S**equence **C**omparison by **L**og- **E**xpectation) [26] is its faster method of estimating the distance matrix of the first iteration, using the k-mer distance. The three stages this program goes through are as follows:

1. A pairwise distance matrix is configured using k-mer counting with a compressed alphabet. A compressed alphabet of size N is a partition of the standard 20-

letter amino acid alphabet into N disjoint subsets (classes) containing similar amino acids [27]. Using the distance matrix configured from a compressed alphabet, a phylogeny is generated using the UPGMA method. Then, as with many of the other alignment programs, a progressive alignment is carried out using this phylogeny as a guide tree, aligning sequences in a prefix order (children of a common node before the children of the siblings of the common node). At each internal node, the profile for each subtree is created and aligned.

2. Using the alignment created above, a new distance matrix is generated using the Kimura's method [28]. A new phylogeny is created from this using the UPGMA method. A progressive alignment is performed on sections of the alignment where the branching orders changed relative to the first tree produced.

3. A refinement step is carried out by systematically dividing the tree into two subtrees, forming profiles for each tree, and then realigning these profiles. If the Sum of Pairs Score (SPS score, the alignment score totaled from o the value taken form the scoring matrix for each aligned pair of amino acids) is improved, the new alignment is kept; otherwise it is thrown out. This refinement step is repeated until convergence or an iteration limit has been reached.

2.5.4 Probalign

Probalign [20] estimates an amino acid posterior probability matrix for each pair of sequences in the set to be aligned using the Gonnet 160 scoring matrix [29] along with gap open and extension penalties set at -20 and -1, respectively. A probabilistic consistency transformation is applied to improve these estimated probabilities.

A guide tree is computed from this matrix using the UPGMA method with the pairwise expected accuracy alignment scores as distance. Sequence profiles are then

aligned in a post-order walk along this tree. Finally, iterative refinement is performed to improve the alignment for up to 100 rounds.

2.5.5 Clustal-Omega

Clustal-Omega [21] is a combination of several existing packages to allow for the alignment of very large sequence sets. It also has the feature that allows new sequences to be added to the alignment without the full alignment procedure being carried out, which is beneficial when aligning with sets in excess of 1000 sequences. The first innovation employed in this program is in the production of the initial guide tree. To start this, mBed [30] is used which transforms each sequence to an n dimensional space where n is proportional to $\log N$ (N being the number of sequences). Each sequence is then replaced by an n-element vector, where each element of the vector is the distance to one of n 'reference' sequences. These vectors, each of which now represent each sequence, are then clustered to form the guide tree. The input sequences are now aligned using the HHalign package, which is a process that aligns an HMM (described in the next section) to an HMM [31].

The refinement step recommended when dealing with large sequences sets forms an HMM from this initial alignment and using it as an 'External Profile Alignment' (EPA). The input sequences are then realigned and the guide tree reconfigured in multiple iterations. This refinement step is repeated as needed.

2.6 Modeling the MSA with a profile HMM

An MSA can be modeled with an HMM using the profile-HMM architecture [32] (shown in Figure 2.8). The hidden states allow for the three processes that take place in evolution, substitution, insertion and deletion. By adjusting the emission

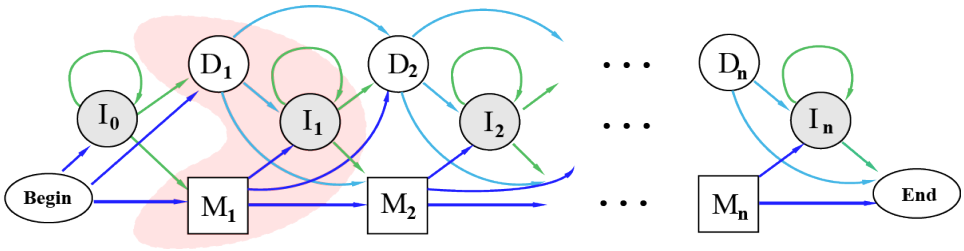


Figure 2.8: Diagram of a profile HMM modeling an MSA. States marked with an 'M' are match states, with an 'I' are insertion states, and 'D' are deletion states. Arrows indicate the directional transitions between states. The pink area represents the repeated pattern of three states.

frequencies per state, position specific evolutionary rates can be achieved. For the profile HMM, the model is configured as a series of three distinct states interconnected with directed edges. These three states are a match state, an insertion state and a deletion state. Figure 2.8 shows how these sets of three states are laid out. The pink background high-lights the single set of three states that is repeated to create the model. Each state has a probability of transition to three other states, one of each type. The insertion state is the only state that includes a loop back onto itself. This allows a single state to model the multiple insertions between two conserved columns in an MSA.

The MSA used to build the profile-HMM can be viewed as the training data. To form the profile HMM from an MSA, the frequency of each residue in each column is calculated. Figure 2.9.a shows the start of a DNA alignment from which the profile HMM will be modeled. A DNA alignment is used for simplicity of the example, having only four residues (A,T,C, and G), versus the 20 residues for an amino acid alignment. The principles for the construction of the profile HMM are the same for both types of sequences (DNA vs. amino acid).

Once the actual counts for those residues present have been calculated (Fig.2.9.b), pseudo-counts (default probability of an element not represented in the column) are

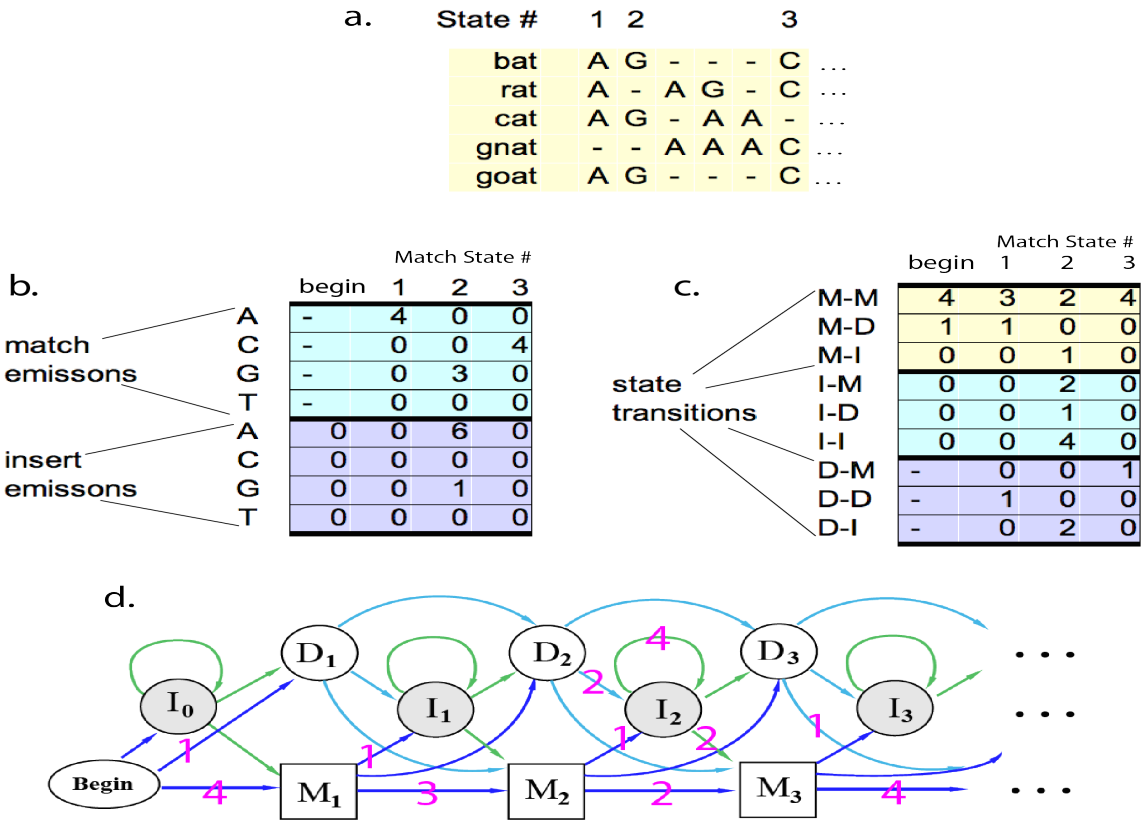


Figure 2.9: Modeling a profile HMM from an MSA. The process of creating a profile HMM from an MSA is illustrated. a) the DNA alignment to be modeled, b) counts for individual residues, c) the transitions represented by the alignment, and d) the transition counts based on the alignment. There is an assumption not visible on the image and that is the next column, not shown on the alignment, would be a full column designated as a match state, $M4$. All unlabeled transitions would be given appropriate default frequencies.

added to account for residues that do not appear in the column. The values of these are dependent on the number of sequences in the alignment. The frequency of the missing residue should be less than any residue that actually appears in the column and can be adjusted to reflect the nature of the sequences being modeled. If the column under consideration is in a highly conserved areas the pseudo-counts or frequencies would be much lower than if the expected variation within the section of the sequences that the column appears was higher.

The strategy for assigning columns to states is straightforward. The non-gapped

columns are assigned to match states. If gaps are present in the column, the number of gaps is compared to the number of sequences in the MSA. If the ratio is below a specific threshold, then the column is labeled a match state and the gaps are accounted for by the transition probability to the deletion state from the preceding match state. If the ratio is above the specific threshold, then the whole column can be treated as an insertion state from the preceding column and the frequency of residues counted as the emission from this insertion state. This threshold can be adjusted to account for varying amounts of divergence in the MSA. As can be seen in Figure 2.9.b, of the six columns of the alignment shown in 2.9.a, three of the residues have been assigned as match states and the other three have been assigned to a single insertion state between M_2 and M_3, labeled in Figure 2.9 as I_2.

Once the states have been assigned, the state transitions for each state can be counted. As an example, from the data given in Figure 2.9 the first two gaps in the **gnat** sequence are counted as a M-D transition (begin state to D_1) and a D-D transition from D_1 to D_2. The first 'A' in the **gnat** sequence then contributes one count to the D_1 to I_2 transition. The next two 'A's contribute two counts to the I_2 to I_2 transitions, followed by a single I_2 to M_3 transition. To complete the model, the appropriate pseudo-counts or frequencies are added to the transition, again dependent on the size of the training data (number of sequences in the MSA the model is based on) and on prior knowledge of the sequences involved.

The profile HMM can be used to give the total probability that any individual sequence "belongs" to the same family that the profile HMM was modeled on. The algorithm used for this is called the *Forward Algorithm* [33], which uses dynamic programming to calculate the full probability of a sequence. The equations involved are given by Equation 2.1. These equations are expressed in log space to avoid the underflow error that occurs when representing small numbers.

$$F_j^M(i) = \log \frac{e_{M_j(x_i)}}{q_{x_i}} + \log[a_{M_{j-1}M_j} \exp(F_{j-1}^M(i-1)) + a_{I_{j-1}M_j} \exp(F_{j-1}^I(i-1)) + a_{D_{j-1}M_j} \exp(F_{j-1}^D(i-1))]; \quad (2.1)$$

$$F_j^I(i) = \log \frac{e_{I_j(x_i)}}{q_{x_i}} + \log[a_{M_jI_j} \exp(F_j^M(i-1)) + a_{I_jI_j} \exp(F_j^I(i-1)) + a_{D_jI_j} \exp(F_j^D(i-1))]; \quad (2.2)$$

$$F_j^D(i) = \log[a_{M_{j-1}D_j} \exp(F_{j-1}^M(i)) + a_{I_{j-1}D_j} \exp(F_{j-1}^I(i)) + a_{D_{j-1}D_j} \exp(F_{j-1}^D(i))]; \quad (2.3)$$

The initialization conditions for this calculation are $F_{begin}^M(0) = 0$ and $F_0^I(0) = 0$.

Profile HMMs are used by Pfam (discussed in the next section) to model protein domains (conserved regions) and families and in the sequence simulation program we will discuss in Section 5.2.2.

2.7 Pfam

Each protein has a specific function that is determined by the structure of the protein. Those parts of the protein sequences that are involved with important function are bound by what is referred to as a functional constraints. The amount of change allowed from generation to generation for this part of the sequence is suppressed and low, resulting in these parts of the sequence being much more conserved than the section of the protein that are not involved in important functions. Note that by conserved, we mean that the sequences remain the same or nearly the same as the genetic material is passed from generation to generation.

The conserved regions in protein sequences are referred to as domains. A protein generally has one or more of these regions or domains. Domains are separated by sections of amino acid sequences called linkers, taken from the idea that they link

the domains together to form the sequence. Different combinations of domains in different proteins give rise to a large number of types of proteins (protein families) along with their associated functions. Determining the functions of a domains in one sequence allows that knowledge to be applied to proteins that have not been studied. When the same series of domains occurs in two different proteins, the probability of the two proteins having the same function is very high. As such when the function of a protein has not been determined by experiment, computationall identifying the domains within a protein allows the function of the protein to be conjectured.

The Pfam database [34][35][36] is a reference database containing domain families and their profile HMMs. A Pfam entry is generally classified as either a domain, family or clan. However, there are less frequent instances where an entry can be classified as a disorder region, motif, a repeat, a coiled coil, areas of low complexity, or transmembrane. The current version of Pfam(version 30) has about 16,306 domain entries.

Each Pfam entry has a seed alignment and a profile HMM that is based on that seed alignment. Usually there is no profile HMM for Pfam entries such as disordered areas or low complexity areas. To demonstrate the information available through Pfam, we will discuss an example domain, EFG_IV (PF03764). Figure 2.10 shows an HMM logo [37] which visualizes the profile HMM for this domain. Figure 2.11 shows the seed alignment that was used to generate the profile HMM logo. It can be seen that the four areas in the alignment with high proportion of gaps in each column, corresponds to the four red vertical lines in the logo, which indicate areas of high probability for insertions. In Pfam, this domain is given the following description, "This domain is found in elongation factor G, elongation factor 2 and some tetracycline resistance proteins and adopts a ribosomal protein S5 domain 2-like fold". It has the specific function within the protein elongation factor (EF2) of extending from the

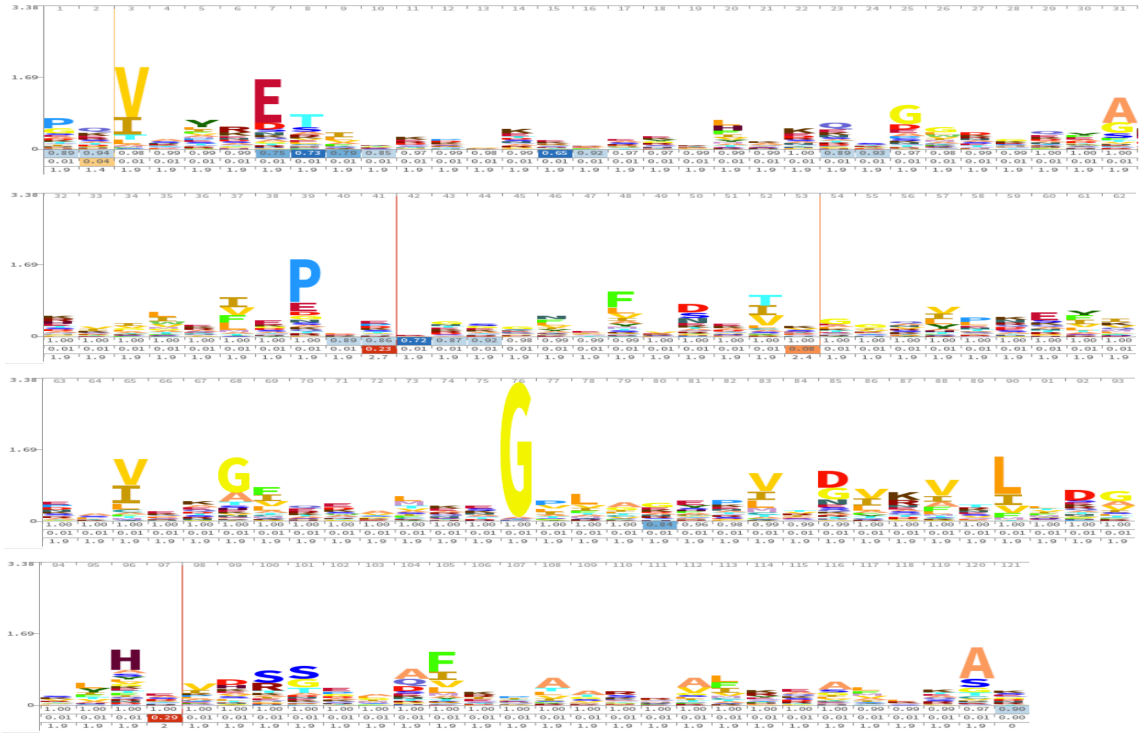


Figure 2.10: The profile-HMM logo for the Pfam domain EFG_IV (PF03764). The height of the letters is proportional to the frequency that the symbol appears in the column. The three rows of values under the logo represent the probability of occupancy, the probability of insertion and the probable insertion length, respectively. The areas of blue tinting in the probability of occupancy represent the areas where the probability is below 0.99. The red vertical line mark the areas where the probability of insertion is higher than the background probability of 0.01.

'body' much like a lever arm, and is essential for the structural transition needed for the translocation of peptidyl-tRNA and mRNA to take place (described in InterPro entry IPR005517). Any protein in which this domain is found is likely to employ it in a similar function.

Pfam is organized into two divisions: Pfam-A, which contains the alignments of highly curated families and Pfam-B, which contains the the alignments of automatically generated families. The work in this study deals only with data found in Pfam-A. Pfam-A families are assigned using the following four-step process.

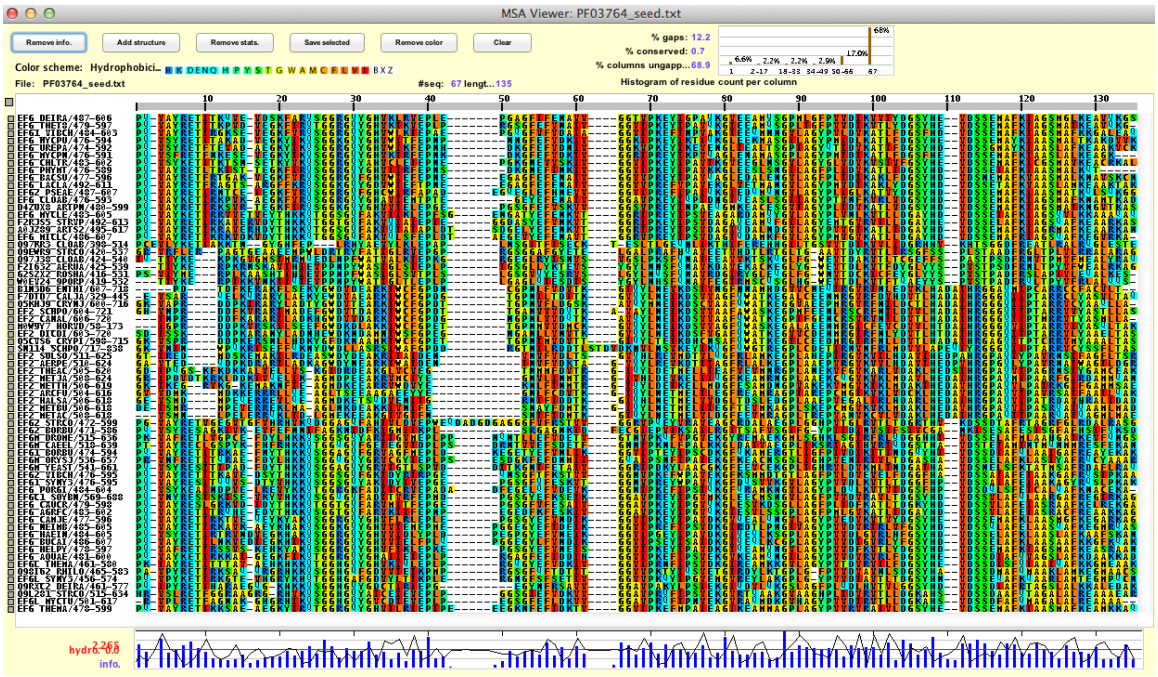


Figure 2.11: The seed alignment for the Pfam domain EFG_IV (PF03764). This alignment image was generated by the SuiteMSA application [6] using the alignment viewer tool employing the hydrophobicity color scheme. The column-wise information and hydrophobicity graph is shown at the bottom of the alignment.Additional information is available from the statistics histogram featured on this alignment, specifically that 68% of the columns are gap free and the average pairwise protein identity is 32%. .

1. A high-quality MSA (the seed alignment) is constructed and adjusted manually.

2. A profile HMM is constructed from the seed alignment (using HMMER3 [38])

3. The profile HMM is searched against the UniProtKB [39] sequence database for sequences that contain the domain represented by the profile HMM.

4. Both a sequence and a domain gathering threshold are selected and all sequence regions that score above the domain gathering threshold are included in the full alignment (as opposed to the seed alignment) for the family.

In addition to the seed alignment and profile HMM, Pfam also provides a set of domain architectures for each of its entries. A domain architecture is a specific

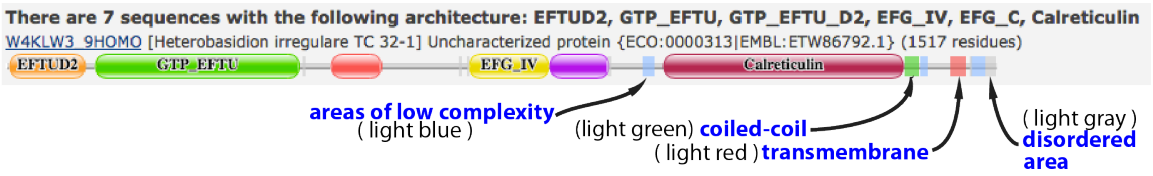


Figure 2.12: One of the domain architectures where the domain EFG˙IV (PF03764) is found. The colored rounded corner shapes represent the domains. The non-domain features, such transmembrane, disorder, and low complexity areas, are represented by square cornered light colored shapes.

arrangement of protein domains and other Pfam entries that combine to create a sequence. Sequences that form proteins using this arrangement are associated with the architecture. The version 30 of Pfam has more than 184,000 architectures. Continuing with the example of domain EFG_IV, there are 76 domain architectures for this domain, with a total of 7,048 protein sequences that contain EFG_IV. Figure 2.12 shows an example of one of the 76 architectures where the domain EFG_IV has been identified. In this diagram, there are six individual domains. From the annotation of the diagram found above the architecture, it can be seen that there are seven sequences that use the architecture shown.

Pfam also provides an unrooted phylogeny for each seed alignment. This phylogeny was constructed using FastTree 2.1.9 [40], an application that can infer approximately-maximum-likelihood phylogenetic trees from an MSA for very large numbers of sequences where the traditional methods of inferring a phylogeny would require substantially more time. It has been shown that FastTree is more accurate than PhyML 3 [41] with default settings, and much more accurate than the distance-matrix methods (NJ and UPGMA) that are traditionally used for large alignments [42]. We use these trees, as downloaded from Pfam, to calculate the average pairwise divergence of each model domain used in our simulation (discussed in Chapter 3).

Pfam also provides links to other databases and established annotation such as the Gene Ontology [43], the Protein Data Bank [44], SCOP [45] classification, and InterPro informatio [46]. These additional features were not used in this study and as such are not discussed here.

Chapter 3

Assessment of MSAs

In this chapter we discuss the various metrics that are available to assess the quality of an alignment. With the increasing awareness of the need for producing quality MSAs, quite a few metrics have been developed to capture different aspects of an MSA and to assist the user in assessing and comparing MSAs. We group these metrics into two groups: 1) characteristic metrics that do not require a reference alignment for comparison and 2) comparison metrics that do require a reference alignment. The former is discussed as possible values to be used as attributes in the data model of a specific sequence set while the latter as a metric to be used to determine the most accurate, the closest to the optimal, alignment. We analyze the comparison metrics and explain why we chose the Cline shift score (CSS) over the more frequently used developer score for this study. We also discuss in detail the two previous studies that bring together several alignment programs in an effort to improve the overall accuracy

3.1 Characteristic metrics

We define "characteristic metrics" as those that convey information on a specific characteristic of an alignment but do not require a reference or true alignment. These metrics include the following statistics for an alignment:

- The percent of gaps in the alignment compared to the minimum number needed

- The percent of un-gapped columns
- The percent of completely conserved columns (those columns containing only one type of amino acid)
- Average pairwise protein identity (discussed later in this section)

Characteristic metrics also include formal scoring schemes such as:

- Information score (used in this study; discussed in detail later in this section)
- NorMD [47]: A scheme that uses a scoring matrix, such as BLOSUM62 or PAM100 to calculate a score for the full alignment. This metric is used in AQUA (described in Section 3.5.1). The final score is strongly influenced by the composition of the sequences as much as their similarities. As such it cannot be used as a comparative indicator of alignment quality between different sequence sets. Consequently, this metric was not used in this work.
- Guidance [48]: A computationally expensive scoring system that involves creating a large number of bootstrapped alignments, distance calculations, tree construction and re-alignments using the resulting trees as the guide trees. It requires that the alignment program being evaluated be able to take a guide tree as input to guide the order of alignment. Not all alignment programs allow the input of a guide tree, i.e.Probalign [20], which is one of the programs used in this study. Consequently, this metric was not used in this work. Recently an updated version [48] was released, which featured options to locate unreliable sections of an alignment, which could be incorporated into our future work.

3.1.1 Average pairwise protein identity

average pairwise protein identity (or protein identity in short) is the average number of identical amino acids aligned together between each pair of protein sequences within an MSA, divided by the total number of alignment positions within the pairwise alignment. The algorithm for this calculation is shown in Algorithm 1.

Data: MSA

Result: Calculated Average Pairwise Identity for MSA

Initialize total Pair Identity = 0;

Initialize pair Count = 0;

forall *pairs of sequences in the MSA* **do**

  initialize matches = 0;

  initialize aligned Pair Count = 0;

  for *each aligned pair of residues between the sequences of a pair* **do**

    if *both residues are the same* **then**

      increment matches;

    end

    increment aligned Pair Count;

  end

  pair Identity = matches / aligned Pair Count;

  add pair Identity to total Pair Identity ;

  increment pair Count;

end

average Pairwise Identity = total Pair Identity/pair Count ;

Algorithm 1: Calculation of the average pairwise protein identity.

As can be seen in this algorithm, residues aligned to the gap symbol do not contribute to the score. In the extreme case, by making an alignment extremely gappy, even if a very low number of amino acids are involved in the alignment, misleadingly

high protein identity can be achieved. This score has been used as an indicator of the level of divergence among sequences. For example, the range of protein identities between 10% and 30%, so called the twilight zone of protein similarity, is known to be indicative of sequence divergence large enough to cause false positives when identifying homologous protein sequences [49].

3.1.2 Information score

The information score is the average column-wise information score [50]. It is based on Shannons entropy [51], which is given in Equation 3.1. The higher the entropy of the column, the less information is present and the lower the score should be for the column. The calculation of the information score for a protein sequence is shown in Equation 3.2. A fully conserved column, i.e. one that has no gaps and occupied by the same amino acid, represents the situation of minimum entropy (0) and consequently maximum information, which for protein sequences with 20 different amino acids, would be $\log_2(20) = 4.31$. For DNA, with only four nucleotides, the maximum information would be $\log_2(4) = 2.0$. The average information score for an alignment is the summation of the information score from all columns of the alignment divided by the number of columns (shown in Equation 3.3). Because of the relationship between the entropy and information, the average column-wise information score can be viewed as a measure of alignment conservation.

$$\textbf{Shannon's entropy} = \Sigma_k \; p_k \cdot \log_2(p_k) \tag{3.1}$$

$$\textbf{Information} : I = \log_2(20) - \Sigma_k \; p_k \cdot \log_2(p_k) \tag{3.2}$$

$$\textbf{average information score} = (\Sigma_{i=1}^{L} I(i))/L \tag{3.3}$$

where L is the number of columns in the alignment.

Gaps are not counted as a residue in the entropy calculation. Their presence is accounted for as a discount of the resulting score [50]. For example, when calculating the column-wise entropy for an MSA consisting of ten sequences, for a column with one gap and all other residues the same, the entropy of such a column would be 0 resulting in the maximum score for the column. Therefore, it is necessary to discount the score for such a column. This can be done by dividing the number of residues in the column by the number of sequences. In the case above, 9/10 = 0.9. Since for twenty amino acids, the maximum score is 4.31, for the example above (one gap out of 10 residues) the final column-wise information score would be 4.31 x 0.90 = 3.888. Likewise, the score for a column occupied by only a single residue and nine gaps would result in a column-wise information score of 4.31 x 0.10 = 0.431.

3.2 Comparison metrics

For the purposes of this study, a single numeric metric that would best describe the quality or optimality of a reconstructed alignment when compared to the reference alignment is needed. Frequently, two or more of these metrics must be compared to obtain the full picture of the alignment quality. Several metrics have been defined to compare two alignments of the same sequences [52][53][54]. For this work, we want to be able to declare that one alignment from a set of alignments generated from the same sequence set is closer to the optimal alignment than any of the others in the set. The optimal alignment would be identical to the reference alignment and as such must be indicated by the maximum value of the metric used. We describe the several

of these metrics and discuss which we decided to use in this study.

3.2.1 Column score

The column score (CS) [55] is based on the number of full columns in the reference alignment that were completely reproduced in the candidate alignment. Traditionally only full columns (columns with no gaps) are considered in the calculation of CS. In the case where there are x columns in the alignment with only y of these columns with no gaps, if z of these columns were reproduced identically from the reference, then the CS is yz . If all columns were considered, the CS would be xz .In the case of very closely related sequences where the proportion of no-gap columns is much higher than those with gaps, the two CS values can be close. However, as the divergence between the sequences in the alignment increases, the number of gaps will also increase, causing y to decrease, which in turn causes the difference between the two CS values to widen. Therefore, CS could be representative of only a very small portion of an alignment. The percent of columns involved in the column score would be needed to better interpret the quality conveyed by the score.

In SuiteMSA, we have implemented a more liberal version of this traditional CS, where the user may select a threshold of gaps, under which a column is eligible to participate in the final score. This modified version is called the "CS with gaps" and is explained in Section 4.3.

3.2.2 Developer and modeler score

The developer score, which is also referred to as the sum-of-pairs score (SPS), is a directional metric; it is seen from the point of view from a specific alignment [56]. If the point of view is changed, the value of the metric would most likely not be the same, except if the two alignments were identical. When comparing a candidate

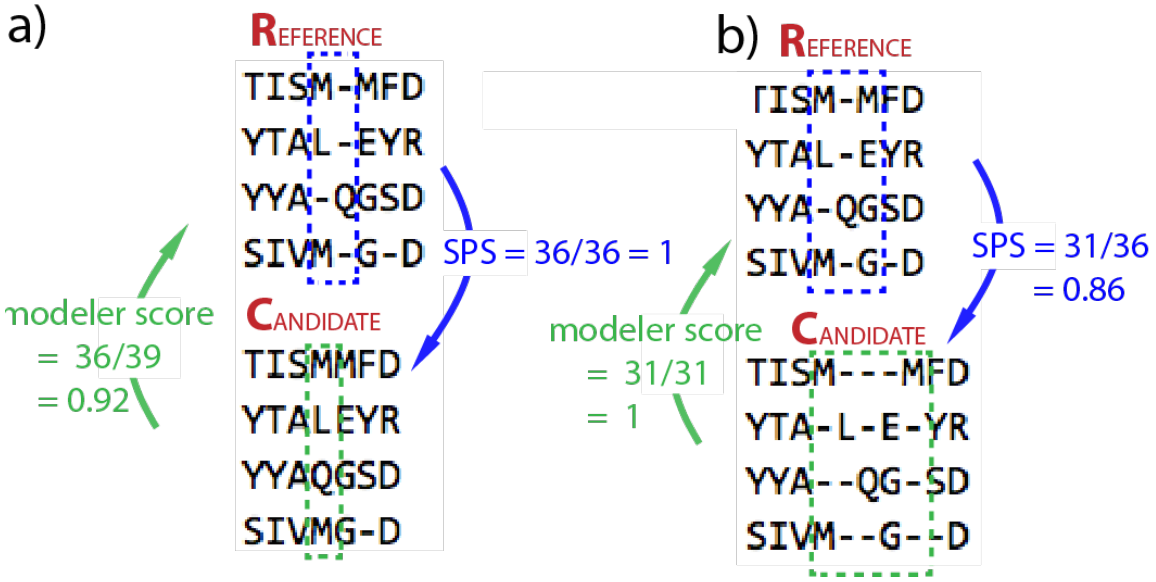


Figure 3.1: Comparison of developer and modeler scores. a) Illustration of the insensitivity of the developer score (SPS) to the error of over-alignment. b) Illustration of the insensitivity of the modeler score to the errors of under-alignment.

alignment C with a reference alignment R, C can be viewed as a test set and each aligned residue pair within C can be viewed as a single prediction. As such, the developer score can be seen as a measure of the recall and is expressed as

$$S_{developer} = \frac{ap_{rc}}{ap_r}, \tag{3.4}$$

where ap_{rc} is the number of aligned pairs in both the reference alignment and the candidate alignment (true positives) and ap_r is the number of aligned pairs in the reference (true positives plus false negatives). This measure is invariant under a change in the number of false positives, which is the number of aligned pairs in C that are not in R. This would occur when C is over-aligned (has too few gaps) which generates a more compact alignment. Figure 3.1a gives an illustration of this. The first three and last two columns of this sample alignment are identically aligned in

both R and C. The differences between the two alignments occur within the sections indicated by the dotted outline. The two alignments are not identical. However, since all pairs within the orange dotted square of R are also aligned in C, even though the alignments are not identical, the developer score (SPS) is $36/36 = 1$, which is the maximum value possible.

The modeler score [56] switches the perspective of the developer score. If C is viewed as the test set, the modeler score can be viewed as the precision and is given as

$$S_{modeler} = \frac{ap_{rc}}{ap_c}, \tag{3.5}$$

where ap_{rc} is the number of aligned pairs in both the reference alignment R and the candidate alignment C (true positives) and ap_c is the number of aligned pairs in C (true positives and false positives). This measure is invariant under a change in false negatives, or the number of aligned pairs in R that are not aligned in C. This occurs when the alignment is under-aligned (has too many gaps). Figure 3.1b gives an illustration of this under-alignment. Once again, the first three and last two columns of this sample alignment are identically aligned in both R and C. The differences between the two alignments are indicated by the dotted outline. All aligned pairs that occur in the orange dotted outline in C also occur in R generating a modeler score of $31/31 = 1$. Once again, even though the two alignments are not identical, the developer score is at the maximum value of 1.0.

Both of these scores, therefore, suffer from insensitivity to a specific type of errors which should not be present in a quality/optimality score. When used together, the amount of under-alignment and over-alignment can be noticed by the range in the difference between the two numbers. Both of these scores are thus needed to fully interpret the alignment quality.

3.2.3 Cline shift score

The Cline shift score (CSS) represents a more robust metric that rewards correct alignments while penalizing both over- and under-alignment [57]. This is accomplished by quantifying the shift information. "Shift" is a pairwise measurement of misalignment. Given two alignments of the same sequence set, R and C, the CSS is calculated as follows:

- Let $|R|$ and $|C|$ be the number of aligned pairs in R and C, respectively.
- Let S_A and S_B be two sequences present in these two alignments.
- Let a_i be residue i of S_A and let it be aligned to b_j of sequence S_B in C but to b_k in R. Then $shift(a_i, b_k)$ is calculated as the number of residues from b_j to b_k. Let b_j of sequence S_B in C be aligned to a_g in R
- If a_i is not aligned to any residue of S_B, then $shift(a_i)$ is undefined.
- The score for aligned-pair i, C_i, involving a_j and b_k is given as

$$cs_R(C_i) = \begin{cases} s(a_j, b_k) + s(b_k, a_j) & \text{if } a_j \wedge b_k \neq \text{gaps} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

  where the subscore $s(a_j, b_k)$ is given by

$$s(a_j, b_k) = \begin{cases} \frac{1+\epsilon}{1+|shift(a_j,b_k)|} - \epsilon & \text{if } shift(a_j, b_k) \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

  where ϵ is a scoring parameter set to 0.2.

- CSS for the alignment C with respect to R is given as

$$CSS_R(C) = \frac{\sum_{i=1}^{|C|} cs_R(C_i)}{|C|+|R|}. \tag{3.8}$$

- CSS for the alignment R with respect to C will be the same value.

The range in $cs_R(C_i)$ runs between -0.4 and 2 (-2ϵ). When an aligned pair appears in both C and R, i.e., they were correctly aligned, both $s(a_j, b_k)$ and $s(b_k, a_j)$ equal 1, and $cs_R(C_i) = 2$, effectively increasing the CSS, thereby demonstrating that this metric rewards properly aligned pairs. For the situation where the aligned pair does not exist in R, as long as $shift(a_i, b_k)$ is less than 5, the alignment will contribute to the CSS but at a discounted rate (the purpose of ϵ). Furthermore, if $shift(a_i, b_k)$ is larger than 5, indicating a largely misaligned section, the contribution of $cs_R(C_i)$ is negative, deducting from the score achieved by better aligned areas, demonstrating that this metric penalizes improperly aligned pairs.

Now we will examine how this metric treats over- and under-alignment. First, we will consider the case where C is identical to R. In the calculation for $CSS_R(C)$ (Equation 3.8), each aligned pair will contribute a count of 2 to the numerator totaling $2|R|$ and the denominator will be effectively be $|R|+|C|= |R|+|R|$, so that the $CSS_R(C) = \frac{(2|R|)}{(2|R|)} = 1$, which is the maximum score.

In the case of the over-aligned alignment, as shown in Figure 3.1a, all aligned pairs in R are also aligned in C. However, due to over-aligning, C has 3 more aligned pairs than R. The denominator of the $CSS_R(C)$ (Equation 3.8) is $|R|+|C|= 2|R|+3$. However, $shift$ between this over-aligned pairs does not exist, resulting in $cs_R(C_i) = 0$ for these over-aligned pairs and as such, there is no contribution to the numerator for them. Hence the only contribution to the numerator is for the $|R|$ aligned pairs that exist in R. Since these are all properly aligned, their contribution to the numerator

is $2|R|$ resulting in $CSS_R(C) = \frac{2|R|}{(2|R|+3)}$. It results in a lower than maximum score, demonstrating that this metric penalizes over-alignment.

The same argument can be made for under-alignment. If five aligned pairs in R are not present in C due to under-alignment as shown in Figure 3.1b, then the denominator of $CSS_R(C)$ can be expressed as $|R|+|R|-5$. However, $shift$ for the aligned pairs in R where one of the amino acids is aligned to a gap is not defined, 2 counts for each missing aligned pair will not be contributed to the nominator. For this example the numerator can be expressed as $2|R|-2(5)$, resulting in an expression for the $CSS_R(C) = \frac{2|R|-2(5)}{(2|R|-5)}$ which is less than 1. This demonstrates that the metric penalizes under-alignment.

Since CSS rewards proper alignment and penalizes misalignment as well as over- and under-alignment, CSS is a more robust metric than either the developer or modeler scores.

3.2.4 Other assessment tools

In this section we will discuss other assessment tools for comparing two alignments. These other tools, which were not used in this study, include the following:

- Mumsa [58]: This assessment tool determines the quality of an alignment by comparing it to multiple alignments of the same sequence set. The assessment is based on the concept that the areas in which the different alignments agree can be considered reliably aligned. Hence, the score of each of the different alignments is based on the same sections, and thus generating the same score. While this assessment does determine candidates for reliable sections for a group of alignments, it does not give a good indication of how the individual alignments compare with each other. Hence, this metric was not used in this work.

- AlignStat [58]: This metric starts by finding for each column in the reference alignment, the best match column in the candidate alignment. This choice is based on maximizing a similarity objective which considers only exact matches of amino acid. Once the matching column has been identified, each amino acid in a column is compared with its matching column and sorted into five scenarios: two amino acids correctly aligned, a correct match of a conserved gap region, a match of an amino acid in the reference with a gap in the candidate, a gap in the reference matched with an amino acid in the candidate alignment, or finally, if an amino acid in the reference is mismatched with an amino acid in the candidate.

  The resulting column-wise score for each category is used for different purposes. To calculate the comparison score, the average column-wise correct match count is divided by 1 minus the average column-wise correct match of conserved gap. The other three categories are used to highlight the alignment in a visual graphic to indicate the misaligned areas. This scoring system suffers from the same insensitivity to over- and under-alignment as the developer and modeler scores.

After considering various metrics, we decided the CSS to be the more sensitive metric for our purposes.

3.3 Benchmark datasets

Assessment of the various MSA programs requires benchmark MSA datasets where alignments have at least three or more sequences. Having only two sequences is simply a pairwise alignment and consequently, does not present the challenges that are inherent when attempting to align more than two sequences. The three benchmark datasets that are discussed in this section and used in this work, are those that are

most frequently used in assessment studies for protein sequence alignments. Table 3.1 shows the number of alignments in each dataset along with the average alignment length and number of taxa. Table 3.2 shows the average protein identity and information scores for these benchmark databases.

Table 3.1: Three benchmark databases currently available. Average alignment length and average number of taxa are obtained from the sequence sets that have three or more sequences are used in the analysis and are counted here. SD: standard deviation.

name	total number of alignments	Number of of alignments > 2 seqs	Alignment of length (SD)	Number of taxa (SD)
HOMSTRAD	605	402	243.3 (173.5)	5.5 (4.5)
BAliBASE	608	608	864.8 (812.5)	27.9 (29.4)
OXBench	672	399	155.0 (88.1)	8.3 (14.0)

Table 3.2: The protein sequence divergence for the three benchmark databases. These average values are based on the alignments that contain three or more sequences.

name	Average protein identity (SD)	Information score (SD)
HOMSTRAD	0.389 (0.147)	3.34 (0.347)
BAliBASE	0.477 (0.141)	2.86 (0.430)
OXBench	0.534 (0.214)	3.52 (0.429)

3.3.1 HOMSTRAD

HOMSTRAD (**HOM**ologous **STR**ucture **A**lignment **D**atabase) was formed as a resource for identifying functions in new protein sequences [59][60][61]. It provides alignments consisting of protein sequences from the same family as determined by the similarity of the structure of the domain areas present in the sequences. The 3D-structure information is taken from the Protein Data Bank (PDB)[44]. The alignments are heavily annotated with connections to Pfam [34], SCOP [45][62], and Swis-

sProt [63]. HOMSTAD has 402 alignments that contain three of more sequences, with an average of 5.46 sequences and an average alignment length of 243 amino acids.

3.3.2 BAliBASE

BAliBASE was designed specifically for the evaluation of MSA programs when aligning complete sequences [64][65]. It contains manually refined reference alignments based on 3D-structure superpositions in the areas of each sequence where the solved 3D-structures are available. Because of this, the areas of the alignments based on homologous structures are assumed to be more accurate than those areas free of 3D-structure. BAliBASE 3 includes 6255 full protein sequences aligned in 608 curated alignments. These are broken into the following six different groups each of which represents a different alignment problem [65]:

1. 164 alignments with equidistant sequences of variable sequence lengths and different degrees of variability.
2. 88 alignments that contain one or more highly divergent orphan sequence.
3. 82 alignments that contain divergent subfamilies.
4. 60 alignments that contain long end extension.
5. 49 alignments that contain large internal insertions.
6. 224 alignments containing protein families with linear motifs.

BAliBASE also contains several reference sets of alignments based on the structural portion of full sequences. These were constructed by fragmenting the full sequences into shorter sections containing only the portions of the sequence that can

be verified by 3D-structure. As these are not full sequences, they are not used in this work.

3.3.3 OXBench

OXBench was also formed for the purpose of providing quality reference alignments for the assessments of MSA programs [3]. It is based on both 3-D structure and sequence similarity and is heavily dependent on the STAMP structural alignment program [66]. It contains three broad grouping of alignments:

- "Master set" contains the sections of the sequences that contain the structural domain. The author recommends this set for general use when not attempting to optimize parameters of the program used. This set contains 672 alignments, 399 of which contain more than two sequences.
- "Full set" contains the full protein sequences (domains along with linker sections). Additional reference alignments are not provided for this set.
- "Extended set" contains proteins of unknown structure combined with the master set sequences. These are provided to be used to show the effect of having more sequences, especially those sequences not determined to be homologous. Reference alignments are not provided for this set.

We used 399 alignments from the master set in this study. These had an average of 155.0 amino acids, with an average of 8.3 sequences per alignment.

3.3.4 Pairwise reference alignment databases

For two other alignment databases, SABmark and PREFAB, the reference alignments are provided only in pairs [67] [68]. In the previous studies using these databases, the

accuracy of multiple sequence alignments was compared on a pairwise scoring system with these pairwise reference alignments. This scoring scheme is not compatible with the full alignment metrics used in this study. Therefore, the alignments in these two databases were not used in this study.

3.4 Strategies for improving alignment quality

In the pursuit of higher quality multiple sequence alignments, the following three main strategies have emerged:

- to develop a new alignment program that takes a set of sequences and outputs an alignment that is better than those produced by currently available methods
- to develop an alignment refinement technique that takes existing alignments and adjusts them to create higher quality alignments or
- use a suite of existing alignment programs to create a set of MSAs from which to select the highest quality alignment.

In the following sections, each of these strategies will be discussed and explain why we invested our effort in the third strategy.

3.4.1 Developing a new alignment program

As discussed in Section 2.3, there exist dynamic programming algorithms that can guarantee the top scoring alignment between any two sequences for a specific scoring matrix. However, these algorithms have $\Theta(n^2)$ growth where n is the length of the sequences. To extend this algorithm would increase the exponent of n in Θ by one for each sequence added, which is intractable in calculation. To circumvent this growth rate, heuristics are employed to align more than two sequences which introduce variety

in both the type of heuristic and in the implementation of the heuristic. This variety results in variation in the resulting alignments. In all evaluation studies performed, not only have differences in relative performance been noted, but also that no single alignment program consistently achieves the highest quality for all sequence sets. In other words, there were always sequence sets where an alignment program that achieved the lower average quality score produced a higher quality alignment than the program that had achieved the highest average quality. Therefore, we concluded that while a new alignment program could achieve more accurate alignments for a specific type of sequences, it is not likely that a new alignment technique will produce better alignments for all sequence sets. Rather it is more reasonable and practically more useful if we can identify a specific method that is more likely to perform better than other methods for a given alignment problem.

3.4.2 Refining an existing alignment

The process of refining an alignment starts with an existing alignment or group of alignments created from either a manual alignment process or as the result of an automated alignment process. The refined alignment is then configured by different techniques. Some of the refinement techniques are:

- RASCAL first clusters MSAs into subfamilies, identifies "core blocks", uses NorMD objective function [47] and finally realigns by a progressive method [69].
- REFINE uses a predetermined block mode as a constraint, and iteratively re-align individual sequences with block shifting and block editing [70].
- ComAlign is a variant of the dynamic programming technique and combines sub-alignments iteratively at combination points [71].

- M-Coffee is an extension of T-Coffee [72] and combines multiple MSAs using consistency-based objective function[73].
- MUMSA groups pairs-of-aligned residues into occurrence patterns among alignments, extracts identically aligned regions, and disentangles unreliably aligned residues [58].
- MergeAlign constructs consensus MSAs using a weighted directed acyclic graph and a dynamic programming approach that incorporates multiple substitution matrices [74].

These techniques can improve the accuracy and hence the quality of an alignment but this is not always the case. For example, if you use MUMSA, the sections of the refined alignment where the base alignments disagree are disentangled, which means each amino acid is placed in a column by itself. If any aligned parts that appear in the reference alignment are in a section that is disentangled then there will be a decrease in both CSS and SPS scores. If any disagreements leading to disentanglement involved full columns in the reference alignment then the CS can also decrease. The main drawback to these techniques, therefore, is that while they can increase the quality, there is always a chance that the resulting alignment can be overall less accurate and hence lower quality than the starting alignment(s).

3.4.3 Selecting the best alignment from a group of alignments

The third strategy involves forming multiple MSAs using a group of alignment programs and selecting the best one from the group. Two previously developed methods are:

- AQUA selects the best MSA from those built by MUSCLE, MAFFT, and those refined by RASCAL based on NorMD [47]. scores [75]
- AlexSys is a decision-tree-based learning algorithm that selects an aligner from six methods depending on the sequence properties [76]

Although both AQUA and AlexSys had limited success (both are discussed in detail in Section 3.5), this is the strategy we chose in this study. It allows the strengths of each individual alignment programs to be used based on different types of sequence sets. After examining the shortcomings of AQUA and AlexSys, we concluded that with more thorough analysis of alignment problems and choosing or developing better metrics that can efficiently identify quality alignments, this strategy has the best potential in improving alignment quality. Another advantage of this approach is that it can be used as either a final selection process as in the work described in this dissertation or as a subroutine in a refinement program that will decide if the original or the refined alignment is better. This selection strategy also makes provisions for the first strategy in that if a better alignment program is developed, it can be incorporated into the suite of programs that created the base alignments from which the classifier will choose the best.

3.5 Previous studies on improving the overall alignment quality

There have been two published studies on approaches to improving alignment quality based on using several alignment programs. Each one is discussed here with a description of the approach, a summation of the results of the approach and a discussion of factors leading to their very limited success.

3.5.1 AQUA

AQUA is a method developed to improve the quality of an alignment [75]. It uses two existing alignment programs, MUSCLE [16] and MAFFT [77] in conjunction with RASCAL [69] (a refinement algorithm), and the characterizing metric, NorMD (described in Section 3.1). The results of the methods were judged on the ranked values of the developer score (SPS, discussed in Section 3.1). The following is a review of each component of the method:

- MUSCLE - This is one of the alignment programs used in the work of this study. As discussed in Section 2.5.3, MUSCLE is a progressive alignment program with refinement steps to improve alignment.

- MAFFT - As described in Section 2.5), this is also a progressive alignment program with options that allow you to select the mode that best suits your application. These options deal with the number of refinement step, the method of calculating the initial guide tree and how gaps are handled. The sub-programs that use the options that have the more accurate results, one of which is L-INS-i (the version of MAFFT used in their study), take longer to run. Although the article on AQUA does not indicate which version of MAFFT was employed on their study, the default option (–auto mode) chooses the method automatically based on, e.g., the data size.

- RASCAL - This is refinement program that uses a knowledge-based approach that detects most common errors in aligning real families of proteins [69]. It divides an alignment into vertical and horizontal zones and evaluates which zones are reliably aligned and which contain error. The program evaluates and corrects the following types of errors: poorly aligned core blocks of a member of

a subfamily, inappropriately aligned orphaned sequences (a sequence much more distantly related than the sequences it is being aligned to), or more generally unreliable zone. This program has been shown to increase the quality of some alignments as measured by SPS and CS (discussed in Section 3.2.2 and 3.2.1, respectively), and has been to not affect the alignment in the areas it determines are reliably aligned. Thus, it can improve an alignment but will not deteriorate the quality. It was also shown that the ending alignment is dependent on the input. In other words, two alignments starting at different levels of accuracy will not be corrected to the same level. Results are difficult to interpret as there was little explanation to how the graph was generated. There does seem to be "improvement" in subfamilies of proteins, where the sequences within a subfamily are more closely related than they are to the other subfamilies in the alignment. The improved performance for the reference 3 dataset is an expected result as RASCAL uses a knowledge-based approach that is developed for detecting errors often found in alignments involving protein families.

- NorMD This characteristic metric was described in Section 3.1 [47]. It evaluates an alignment based on a scoring matrix, most commonly the BLOSUM62 [11]. This metric is heavily influenced by the composition of the sequence set and gives no indication of over-alignment or under-alignment.

An outline of the method is as follows:

- Align the sequence set with both MAFFT and MUSCLE , producing two base alignments
- Refine each of the base alignment with RASCAL, producing two refined alignments.

- Evaluate each of the four alignments using NorMD.
- Output the alignment that has the highest NorMD score.

The authors tested AQUA on several datasets including the benchmark database BAliBASE 3.0. The performance of AQUA was compared with the two original alignments by MUSCLE and MAFFT as well as those refined by RASCAL based on the CS value. Although AQUA seems to be able to choose better alignments, it is not necessarily the best alignment and especially when divergent sequences were aligned, it showed very low performance (CS<0.48). The minimum amount of the results reported in their paper, unfortunately made quantitative and fair performance comparison difficult. This method was successful in demonstrating that improvement can be made in a final alignment if a selection can be made from a group of alignments. However, the success was limited by several factors.

- The selection of alignment was limited to only two base alignment programs. As shown in Chapter 6, the difference between various programs can change depending on sequence set. Although two other alignments were included by generating refinement alignments by RASCAL, as the results shown in [69] indicate, except for when working with sequence sets that represent multiple clusters of protein subfamilies, the expected improvement is 1% or less, which is well below the average difference between programs[64]. RASCAL can also sometimes lower the alignment quality as can be seen in the data presented in [75] where the curves representing the refined alignments are consistently below those of the base alignment. All of these factors must have contributed to the limited success of AQUA.

- As discussed in Section 3.2.1, using CS or SPS by itself cannot provide the full picture of the significance of any change in alignments. For example, if the true reference of an alignment had only ten conserved columns, and a candidate alignment had only five of these columns correctly align, the CS would be 0.5. But it would take only one more correct column to see an increase of CS in 0.1. Therefore either CS or SPS by itself does not truly reflect the accuracy in an alignment.

- The decision process in based on only one aspect of the sequence set and that is the NorMD score. No account is given to such as the number of taxa, sequence divergence, or other characterizing metrics. As described in Chapter 6, various sequence and alignment properties are shown to be indicators of shifting relative performance between alignment programs.

3.5.2 AlexSys

AlexSys uses machine learning to select an aligner a priori from a group of six alignment programs depending only on the nature of the input sequences [76]. Decision trees representing each alignment program are polled for a prediction of the alignment program being either a strong or weak aligner for a specific sequence set. The attributes are gathered from the sequence set, which include a number of sequences, average protein identity, and a large number of annotations from existing databases such as PDB and Pfam. The use of decision trees is based on the explicit requirement of keeping the chain of decisions human readable. This means that the decisions can be traced from the root of the tree to the prediction. The objective of AlexSys is also to keep the amount of work needed to produce a high quality alignment as low as possible, although it was accepted that this might be at a cost of accuracy.

The quality of the final alignment chosen by AlexSys was tested using SPS. The dataset used were 214 sequence sets from BAliBASE and 672 of the extended alignment dataset from OXBench. The authors performed a 10-fold cross-validation trial using AlexSys and each alignment. The performance by AlexSys was not statistically greater than that by MAFFT and took an average more than 100 times the cpu time as MAFFT. The single program Probcons [78] outperformed AlexSys by 1.2% (non-parametric Wilcoxon signed rank test, $p = 0.15 \times 10^{-6}$), although it did take between 4 and 5 time the cpu time.

AlexSys is the first attempt to evaluate the MSA problem with machine learning. Its limited success can be accounted for by the following:

- The decision trees are used to determine if the alignment program is 'weak' or 'strong' (although no clear denition is given for these classes). Then if more than one aligner received a classification of strong, AlexSys selected the alignment program based on speed. While a 10-fold cross-validation showed that the decision tree had 95% or higher accuracy, the criteria on which the label was based do not seem to be a good discerner for quality, as indicated by the very low overall results of the AlexSys.

- The training data is limited for the decision trees; 218 and 672 sequence sets from BAliBASE and OXBench, respectively, were used to train and evaluate the binary decision trees. 80% of the data was used to train and 20% was used to evaluate using a 10-fold cross-validation (two folds for each test set).

- The insistence that the decision be human understandable (expressed as a series of distinct decision points by the nodes of a decision tree) is to say that the surfaces that separate the classes are well defined. The evaluation we performed

in Chapter 6 indicates that there are no clear-cut indicators for the relative behavior of the alignment programs.

- The objective that a selection of a high quality alignments be made as efficiently as possible indicates that the approach is not optimized to seek the best, only a "high quality" alignment.

- This objective also disallowed the formation of the alignments from all the candidate alignment programs which in turn made it not possible to draw inferences from the aspects of all resulting alignment to better decide on the best alignment

In AlexSys, the authors attempted to use multiple existing alignment programs to create the best alignment for each sequence set. As discussed previously in Sections 3.4.3, we consider this as an efficient approach. The existing programs can be used on sequence sets where they perform the best and newer programs should be used where they have a niche

Our approach to using multiple alignment programs is different in several points, all of which are discussed in more detail in Chapter 7. The major points of the difference are as follows:

- We used a multi-class classifier trained using the multi-layer perceptron as opposed to binary classifiers.

- Our data model for the training and testing data used include both sequences-based, as AlexSys did, but also alignment-based attributes.

- We developed and fined-tuned our data model using a simulated database that had both true reference alignments, and had sufficient numbers and types of alignments to cover more of the instance space of the classifier during training.

- We developed a systematic method for quantifying the performance difference amoung alignment programs and tracking the improvement in alignment quality against the maximum improvement possible for a specific group of alignment programs and reference alignments.

Chapter 4

SuiteMSA

4.1 Motivation

Due to the significant impact of the MSA on many bioinformatics and molecular evolutionary studies, multiple sequence alignment is one of the most scrutinized bioinformatics fields [54][79]. While a number of statistics has been developed to assess an MSA, there is no definite answer on how to measure the 'biological correctness' of MSAs. It remains for the end user to incorporate the available statistics into their evaluation of this 'biological correctness'. Often the users simply run one MSA method and procede to the next analysis without examining their alignment output [5]. Considering how MSA quality affects the outcomes of further analysis, assessment of MSAs should be included as regular part of sequence analysis.

There are a number of programs available that generate, display, and/or let users analyze MSAs such as SeaView [80], ClustalX2 [81], Se-Al [82], Jalview [83], AliView [84], AlignStat [85], webPRANK [86], as well as MEGA [87]. However, none of the MSA viewing/editing programs currently available allows the user to display functional information of sequences (e.g., secondary structures, transmembrane predictions) along with the alignment or make direct comparisons between two or more MSAs to determine the best one to procede with. Considering the importance of MSA quality in a wide range of research, it is desirable for MSA assessment to be

performed routinely and . To this end, we have developed SuiteMSA, a Java-based application that provides unique MSA viewers to assist in this assessment [6][7]. To aid MSA quality assessment, with SuiteMSA, functional information of sequences can be overlaid onto alignment. Multiple MSAs can be directly compared and where the MSAs agree (are consistent) or disagree (are inconsistent) can be visually assessed. Several alignment statistics and metrics are provided to assist such comparisons. In addition to MSA comparison tools, SuiteMSA provides a GUI for a feature-rich biological sequence simulator, indel-Seq-Gen v2.1 [88].

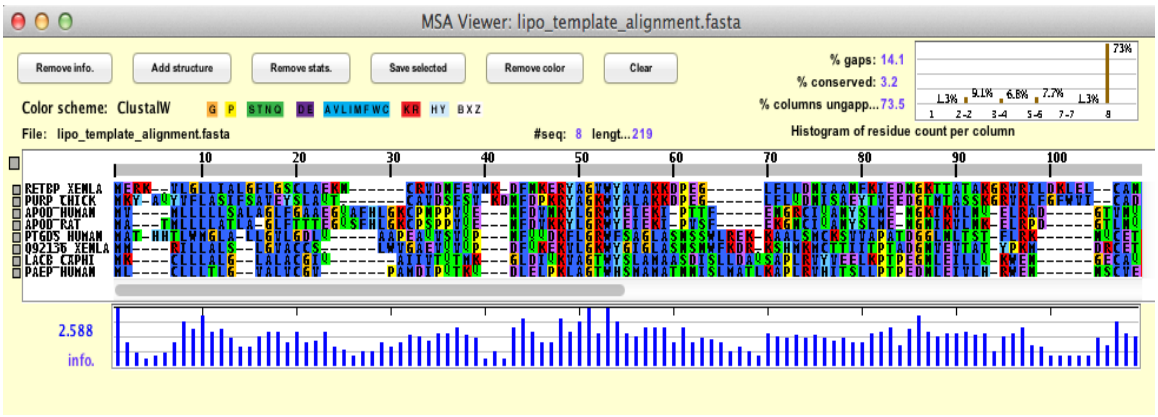
In this chapter, we describe the following three MSA assessment tools available with SuiteMSA, emphasizing the unique feature of each tool:

- The MSAviewer: single alignment viewer and editor.
- The MSAcomparator: pairwise alignment viewer for comparison of two alignments.
- The Pixel plot: multiple alignment viewer for visual comparison of large alignments.

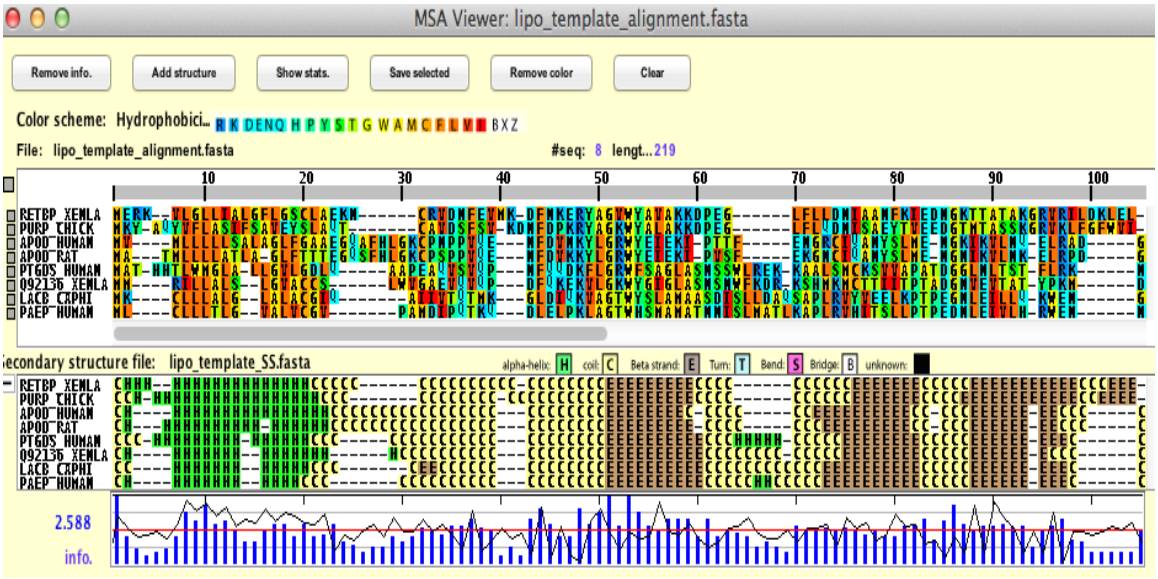
4.2 The MSAviewer

The MSAviewer provides utilities for displaying and assessing a single MSA. It provides many of the features that other alignment viewers have, such as displaying of the MSA using various color schemes, editing of the MSA by way of inserting, deleting or moving gaps within the display, and utilities to produce publication-ready images. It also provides several unique utilities to assist in the process of assessment.

The MSAviewer generates several metrics for the alignment. Two of these are frequency distribution of gaps and the column-wise information score [51]. (Figure



(a) A screen-shot of the MSAviewer showing the frequency distribution of gaps within the alignment and the column-wise information score.



(b) A screen-shot of the MSAviewer showing each amino acid using the hydrophobicity color scheme with the second display showing the secondary structure prediction for the alignment above it. The average hydrophobicity for each column is also plotted (bottom display).

Figure 4.1: MSAviewer from SuiteMSA. The assessment options available with the MSAviewer. The file *lipo_template_alignment.fasta* is used to demonstarte each function.

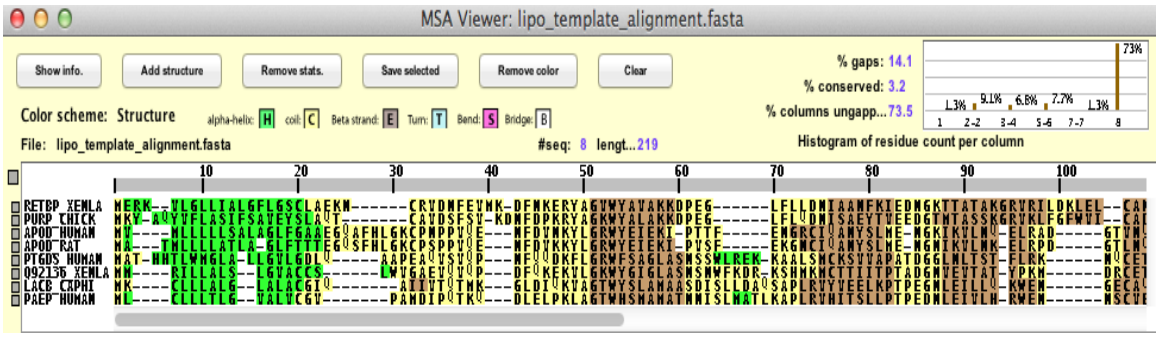


Figure 4.2: MSAviewer with the secondary structure Color scheme. The secondary structure prediction was provided in an additional le containing the prediction information for each amino acid in each sequence.

4.1a). The column-wise information score is graphically displayed in the bar chart. As mentioned in Section 3.1, the column-wise information score can be used to determine the level of conservation of the individual columns. To give an example of this, in Figure 4.1a, columns 1, 51 and 53 all have only one type of amino acid residue and no gaps, indicating complete conversation of the column. This is represented on the information graphics as a full height blue bar under the column. When more than one type of residue is present in a column, the information score is reduced corresponding to the reduced height of the blue bar in that column. An example of this is shown in column 52 which contains three different types of amino acids with no gaps.

Another feature of this viewer is that when the hydrophobicity color scheme is selected, an average column-wise hydrophobicity plot is displayed (overlaid on the information graphic). The alignment in Figure 4.1b is shown with the hydrophobicity color scheme. In this color scheme, each amino acid is assigned a color between blue and red, with the color getting progressively less blue and going toward red as the hydrophobicity scale of the amino acid increases. The average hydrophobicity scale for each column is plotted across the information score bar chart at the bottom. The red line bisecting the blue bars represents the 0.0 hydrophobicity scale. Positive values (above the red line) indicate more hydrophobic while negative values (below

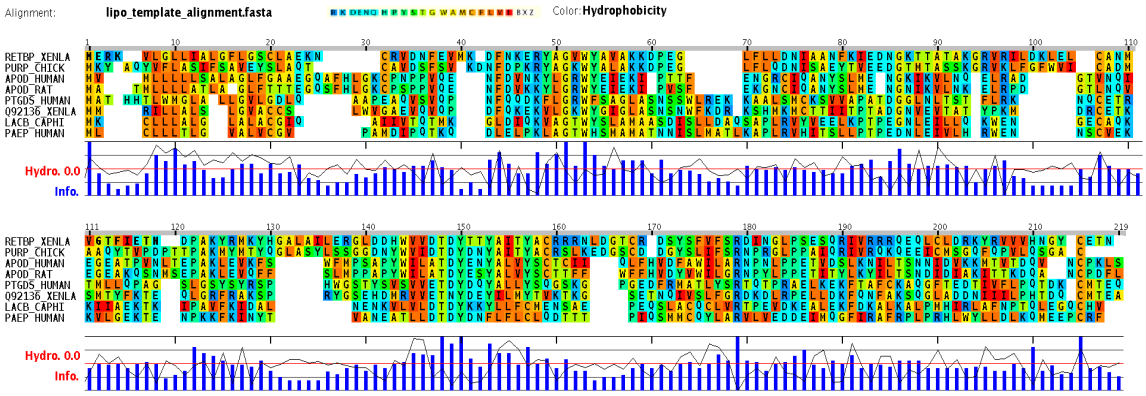
the red line) represent more hydrophilic.

The MSAviewer has two modes for the display of functional information associated with the sequences involved in the alignment: as either a separate graphic under the alignment or as the color scheme of the alignment display. Color schemes are available for such as transmembrane and secondary structure prediction. There are also two gradient scales (blue to green and red to yellow) for any data ranging between 0.0 and 1.0. Examples of these gradient schemes would be for solvent accessibility or disorder scores for each amino acid.

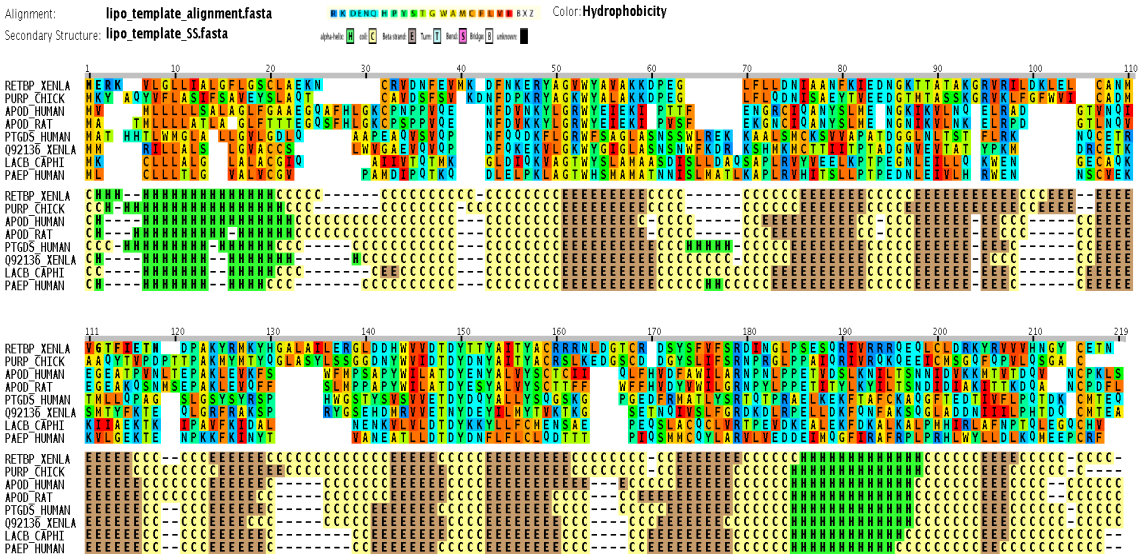
The separate graphic mode can be used with any of the various types of functional information. For example, in Figure 4.1b, the secondary structure for the sequences of the alignment in lipo˙template˙alignment.fasta is displayed underneath the MSA. The secondary structure information can be obtained from either prediction or data confirmed from 3D-structure databases. As described in Section 2.1, protein secondary structure includes -helix, strand, coil or turn. In the second display of the alignment, each amino acid in the alignment is replaced with a symbol (H, E, C, or L) representing the predicted secondary structures.

The second mode for displaying the functional information is to apply the color scheme directly to the alignment. Figure 4.2 shows the secondary structure color scheme applied. These two modes of display allow multiple types of functional information to be displayed simultaneously, assisting in the assessment of the MSA. All functional information is communicated to the MSAviewer by way of a text file in fasta format.

The MSAviewer also provides a utility that will generate publication-ready images of the various assessment tools used. While other assessment packages also provide for the production of publication-ready graphics, the MSAviewer allows publication ready images of the information graphics and functional information graphics to be

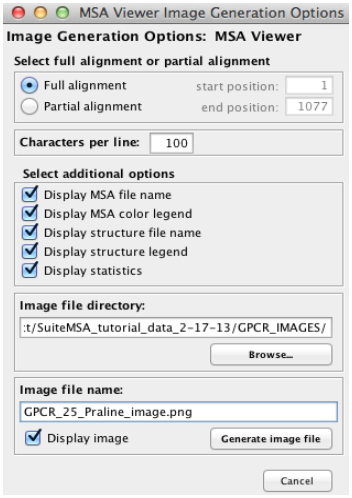


(a) Hydrophobicity color scheme with the information score display.

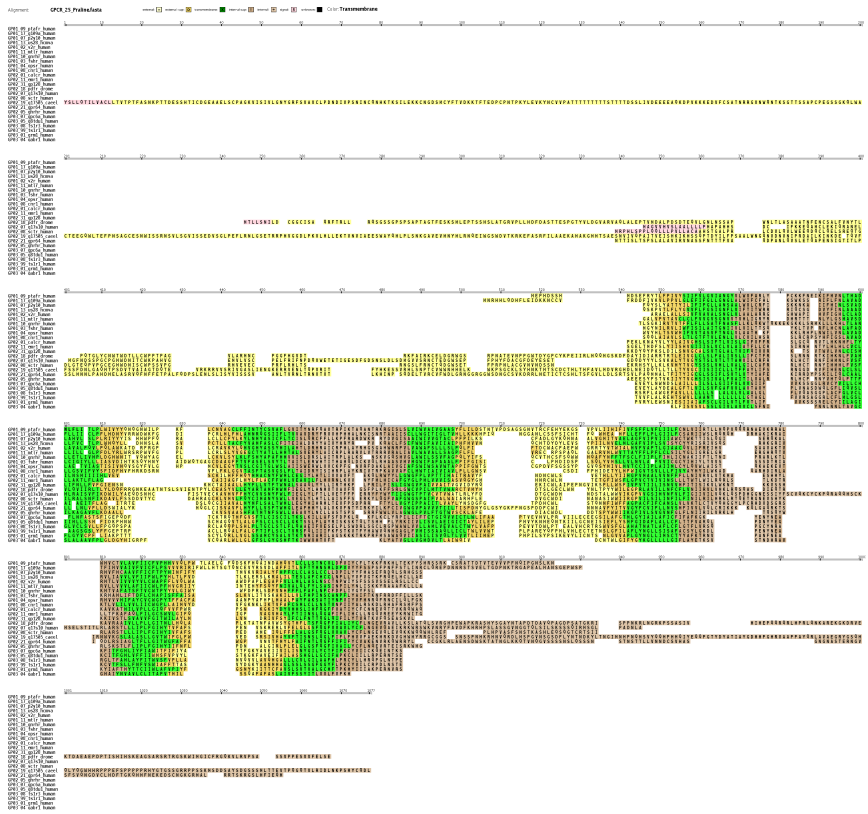


(b) Hydrophobicity color scheme with secondary structure display.

Figure 4.3: Publication ready images generated by the MSAviewer. This image was created using the data in the file *lipo_template_alignment.fasta*.

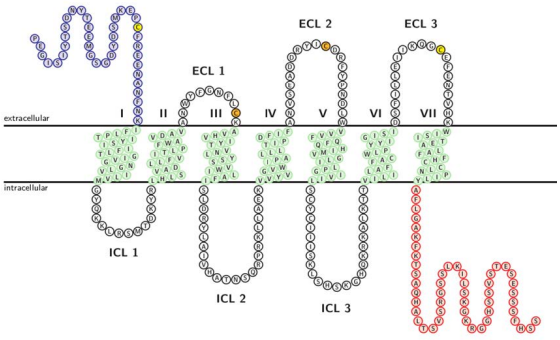


(a) The menu to configure the image file of the MSAviewer.

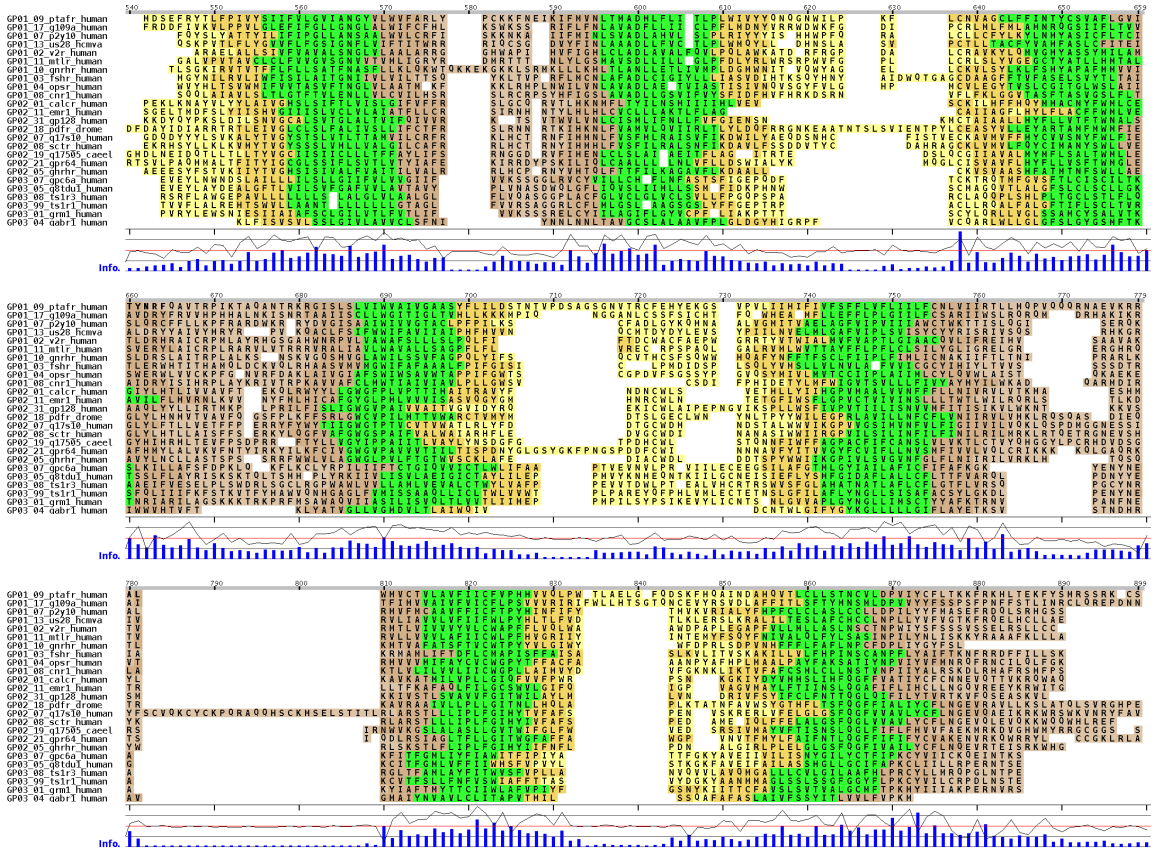


(b) The full alignment for 7 transmembrane protein sequences that contains an extremely long N-terminal region.

Figure 4.4: Publication ready image utility.



(a) The topology of a typical 7-transmembrane protein. There are seven transmembrane regions (labeled I-VII) [89].

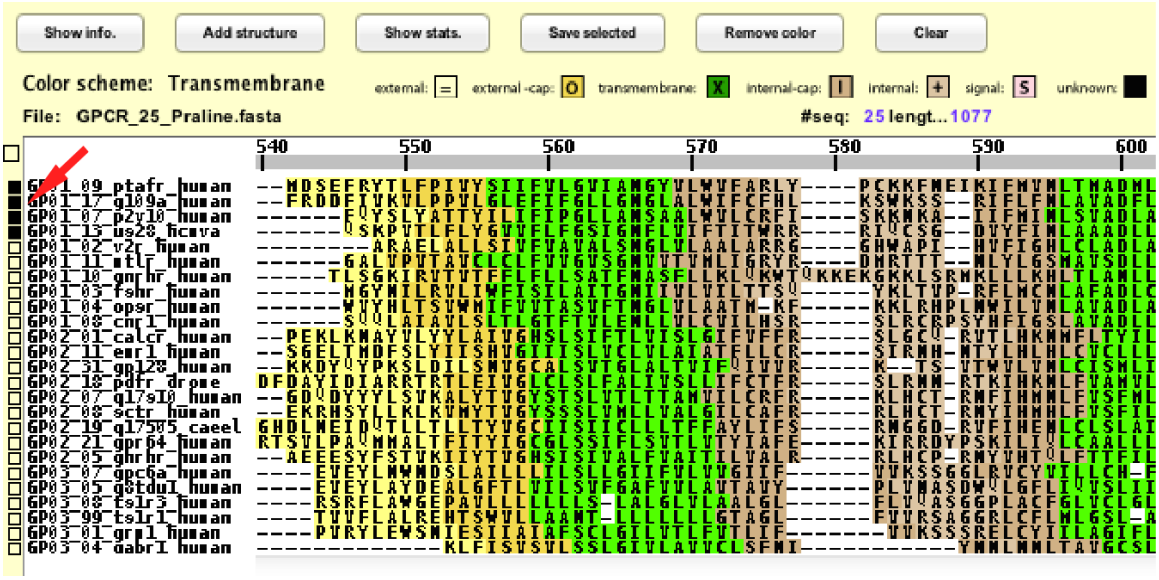


(b) Use of the image utility to generate a publication ready image including specic section of the alignment. In this case, only the region covering the seven transmembrane regions (from positions 540 to 899) was output. Amino acids predicted to be in transmembrane regions are shown in green.

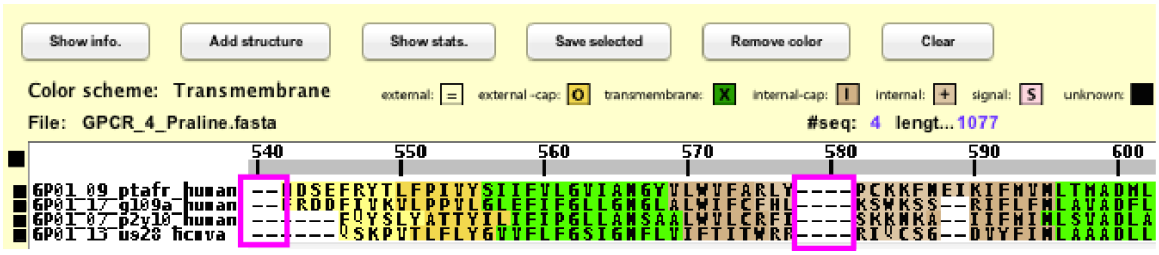
Figure 4.5: Display of Transmembrane MSA using MSAviewer.

produced. Figures 4.3a and 4.3b show how this utility can generate an image of the alignment and . Figure 4.3 shows how this utility can generate an image of the alignment and other information. These images can be custom configured to allow the user to select the specific graphic or sections of the alignment to be included in the image. Figure 4.4a shows the options available for this utility. Creating images is especially useful when an alignment is too long to be shown across the breadth of a computer monitor. An image will allow full alignment to be studied. The partial alignment feature of this utility allows for a specific section of an alignment to be extracted and displayed. For example, Figure 4.4b shows the full alignment of 7-transmembrane proteins. Some of these sequences contain extremely long N- and C-terminal regions. If the user wishes to extract the portion of the alignment that contains only the transmembrane areas, the partial alignment option on the image generation menu can be used, resulting in a image such as shown in Figure 4.5b. The hydrophobicity plot shows that the predicted transmembrane regions (color coded in green) have high hydrophobicity (high positive values).

Another feature of the MSAviewer is the utility provided for extracting subsets of sequences from the alignment displayed. In Figure 4.6a, a series of squares are visible to the left of the sequence names (indicated by the red arrow). When the square is dark, the sequence is selected. The default state is selected. If you click on the square, it will turn white, indicating that the sequence is no longer selected. After selecting sequences, the sub-alignment (Figure 4.6b) can be saved. If the resulting sub-alignment contains all gap columns, another utility can be used to remove all gap columns (Figure 4.6c). All gaps in the alignment can be also removed leaving just amino acids in preparation for realigning the subset of sequences.



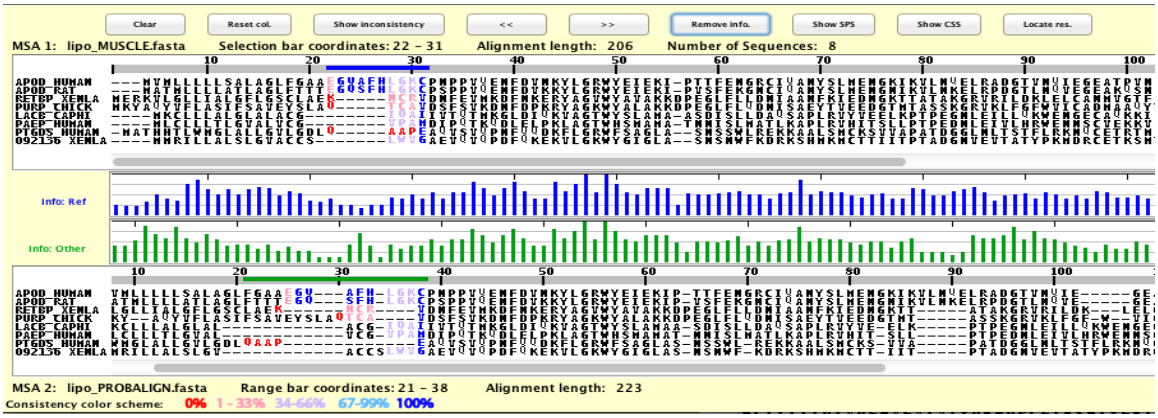
(a) A screenshot of the MSAviewer illustrating the use of the subset selection utility. The black squares (indicated by the red arrow) show that these sequences are selected to be included in the subset.



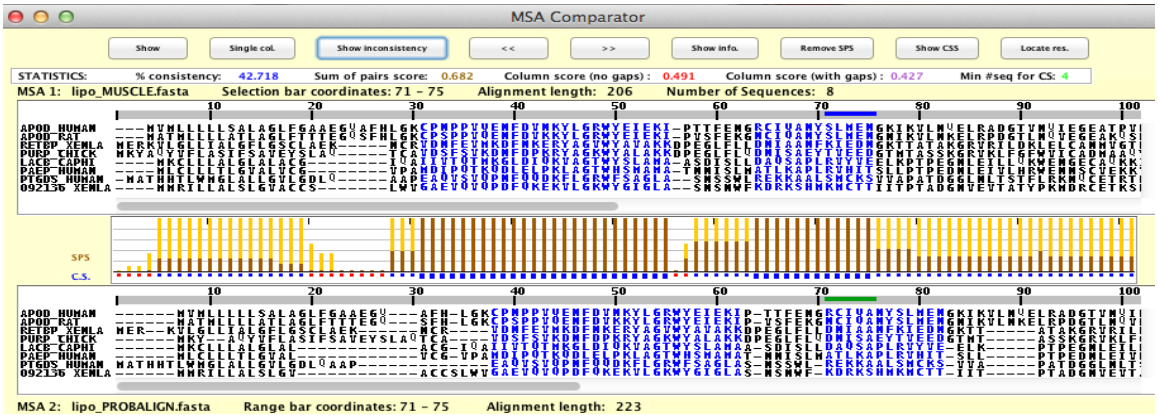
(b) A screenshot of the MSAviewer displaying the subset of sequences. All gap columns are outlined in magenta

(c) A screenshot of the MSAviewer displaying the subset of sequences after all gap columns are removed.

Figure 4.6: Use of subset selection utility.



(a) A screenshot of the MSAcomparator with column-wise information scores shown in bar charts. The residues under the blue selection bar of the reference alignment are highlighted in the bottom alignment. The consistency color scheme is used (indicated in the image legend).



(b) MSAcomparator screenshot with column-wise SPS and CS. The color scheme to highlight the columns that are 100alignments in blue. The graphic display between the two alignments show the column-wise SPS and CS.

Figure 4.7: MSAcomparator from SuiteMSA. The alignment generated using Probalign (the bottom) is compared against the one using MUSCLE.

4.3 The MSAcomparator

The MSAcomparator allows for the detailed viewing and comparison of two alignments of the same set of sequences. The top alignment is considered to be the "reference" alignment, and the comparisons are with respect to this alignment.

The MSAcomparator display has a position scale above each alignment. It con-

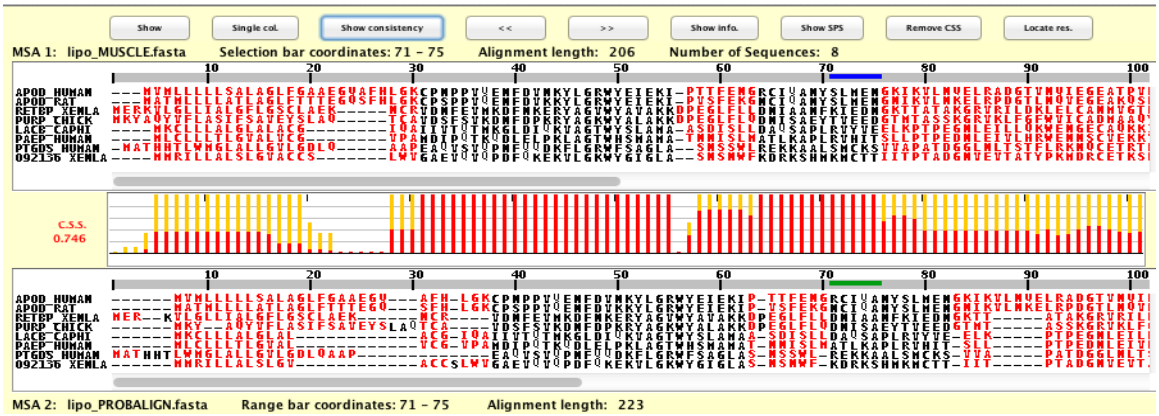


Figure 4.8: MSAcomparator screenshot with column wise CSS. The MSAcomparator is shown with the color scheme set to high lighting in red any column not 100The graphic display between the two alignments shows the column wise Cline Shift Score (CSS).

tains a colored bar, blue for the reference or top alignment and green for the bottom or candidate alignment. The blue bar is referred to as the selection bar, because the bar marks the column positions that are compared and highlighted in both alignments. For example, in Figure 4.7a, the selection bar is 10-positions wide and is shown positioned over columns 22-31. The width of this bar can be changed to facilitate finer or larger comparisons. The residues in the columns under the selection bar are highlighted in both alignments. The colors indicate the amount of consistency the individual residue has between the alignment. The color indicates the percent of residues that any specific residue is aligned to in reference alignment that it is also aligned to in the candidate alignment. For instance, in column 22, the amino acid E in the first sequence is shaded in pink (1-33% consistency). This is due to the fact that in the reference alignment, this E is aligned with four other residues: E, K, Q and Q. In the candidate alignment, the same amino acid E is aligned with only one of these four residues. Therefore, the top E is aligned with one out of four, or 25was aligned with in the reference alignment. In the same column, the K or either of the two Qs are shaded in red (0% consistency). This is because in the candidate alignment these

amino acids are aligned with none of those of their column in the reference alignment. In column 23, in the reference alignment, there are only two residues and since both are aligned with the other in the candidate alignment, regardless of the additional residues that appear in the column in the candidate alignment, they are aligned with 100% of the residues they had in the reference alignment. They are thus shaded in dark blue.

The consistency scheme provided under the selection bar is from the viewpoint of each individual residue in the reference alignment and is referred to as residue consistency. This is different from column consistency, which is defined as occurring when a specific column in the reference alignment is aligned identically as a column in the candidate alignment.

The MSAcomparator has an additional graphic tool, the SPS graphic (Figure 4.7b). It illustrates two metrics: 1) the column-wise contribution to SPS, and 2) the columns involved in the CS calculation. The vertical bars in this graphic represent the column-wise SPS (see Section 3.1 for a full description). The gold vertical bars of the SPS graphic represent the maximum SPS for a specific column. The cap on the maximum SPS score for any column is determined by the number of sequences in the alignment and is given by $N \times (N-1)/2$, where N is the number of sequences in the alignment. However, since a gap is not counted as a sequence, their presence will reduce the maximum SPS possible for a specific column to $n \times (n-1)/2$, where n is the number of amino acids in the column. The red bars represent the actual SPS obtained by the specific column. If the column was identically reproduced in the candidate alignment, then the achieved SPS will equal the maximum and the red bar will hide the gold bar. As such the gold bar shows only in those columns where the reference column was not identically reproduced.

The second metric in the SPS graphic, CS, is illustrated by the colored square

underneath the vertical SPS bar for each position. A column that has a dark blue square under this vertical bar has no gaps in it and will be used to calculate the column score. The size of the square indicates the consistency of the column between the two alignments. A large blue square indicates that the column is identical in both alignments. Conversely, a small square indicates that there are inconsistencies for that specific column between the two alignments. The average CS for the full alignment is shown above the alignment.

In addition to the traditional CS, we have also provided for a more liberal definition of the CS, where the user can determine the threshold of gaps that the column may contain before it is not included in the calculation. We refer to this as the "CS with Gaps". If the column contains a lower number of gaps than the set threshold, a red square is placed under the column. If the column is identically reproduced in the candidate alignment, this square is larger than if it is not. Again, the un-gapped CS for the whole alignment is shown above the alignments. It is calculated by dividing the total number of columns included in the calculation (illustrated by those having either a red or blue squares beneath them) by the number identically reproduced in the candidate alignment (illustrated by those columns having large squares, either red or blue, underneath them). The threshold number (maximum number of gaps allowed) is also shown above the alignments.

A numeric value is also given for the full consistency, which is the total number of columns identically reproduced in the candidate alignment regardless of number of gaps in the column. This would be equivalent to the gap threshold set to the number of sequences in the alignment to include all columns.

The final display available with the MSAcomparator is the CSS display (shown in Figure 4.8; CSS is described in Section 3.2.3). Just as with the SPS display, the gold bars represent the maximum column-wise CSS. The red bar represents the

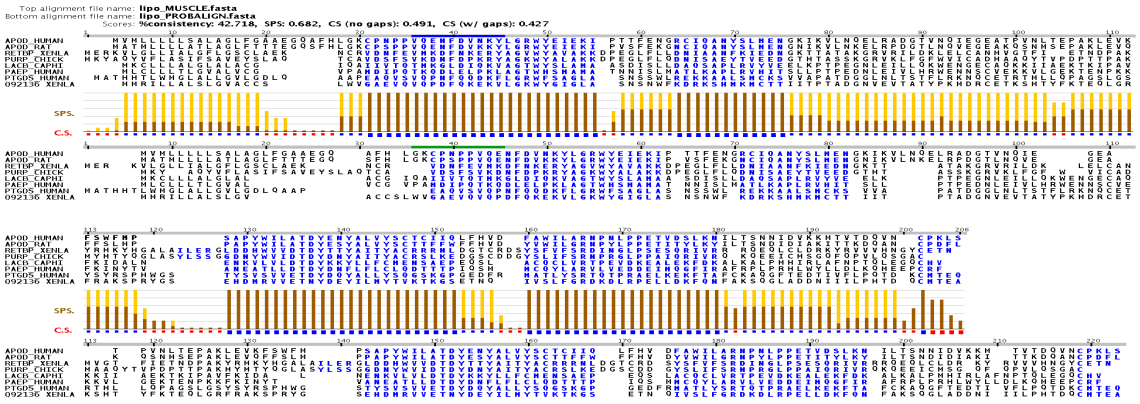


Figure 4.9: Example of a publication ready MSAcomparator image.

achieved column-wise score and when the actual value equals the maximum, that gold bar is hidden. Figure 4.8 also shows an alternative highlighting scheme where any column not identically reproduced in the candidate alignment are highlighted with red. This feature is independent of the CSS display and is the complement for the blue consistency highlighting, where each column identically reproduced in the candidate column is highlighted dark blue.

A utility to configure publication-ready images is also available with the MSAcomparator. Figure 4.9 shows the full image of the alignment shown in Figure 4.7b. As with the MSAviewer, all additional graphic can also be reproduced as publication-ready images, with the same degree of freedom in choosing which graphics and which part of the alignment to include in the image.

4.4 Pixel plot

The pixel plot is a novel display unique to SuiteMSA. It allows for the viewing, comparing and assessing of multiple MSAs of the same set of sequences in a novel graphic format developed for larger datasets. The basic format of the pixel plot consists of a black pixel representing each residue and a white pixel representing a

gap. The resulting reduced scale allows for the gap pattern of the alignment to be assessed visually over a smaller space. Questions such as "do the gappy areas of the alignment cluster in specific segments of the alignment or are them scattered throughout?" can be answered using this tool. Any number of alignments can be compared, limited only by the resources (monitor size, resolution and RAM) of the system that SuiteMSA is being run on.

Figure 4.10 shows the pixel plot being used to compare five alignments created by 5 different alignment programs using a set of transmembrane protein sequences. This figure is a basic black and white pixel plot. It can be noted that the shape of each of the alignments being compared is significantly different from the others. However, the long insertion (outlined in green in Figure 4.10.a) could indicate a gap section that was consistently captured in each alignment. There is no known or agreed upon reference alignment for these sequences, so it is not possible to compare these alignments with a reference. However, the areas where the alignments agree, can be explored using the selection bar.

In Figure 4.10.b, the selection bar has a width of 10 alignment positions and is located starting at position 646 of the top alignment. Just as with the MSAcomparator, the residues that fall under the selection bar of the top alignment are highlighted in magenta in all of the other compared alignments. The fact that for the most part the magenta highlighting in the lower four alignments has smooth edges, with the exception of the ClustalW2 alignment (the bottom alignment), indicates that in this section, the alignments have a high degree of consistency.

In Figure 4.10.c, the selection bar is located starting at position 718, which is to the immediate right of a rather large gapped area. The very large amount of disagreement between the different alignments is made apparent by the magenta highlighting. For example in the second and fifth alignments, there is a good amount

of magenta to the left of the gapped areas. This is contrasted by all of the magenta being to the right of the gapped area in the top alignment. Also, the jagged edges of the magenta highlighting in all of the bottom four alignments indicates the large amount of inconsistency within this section of the alignments.

In Figure 4.11a, the selection bar is located to the immediate left of a large gapped area where disagreement is found mainly in the placement of the large insertion in the 15^{th} sequence. In Figure 4.11b, the width of the selection bar has been increased to 28 positions and moved to be just above the position of the insertion. Very few of the sequences have residues highlighted in magenta in the top alignment. If you examine where the highlighting is found in the other alignments, it shows that in the different alignments, there is disagreement on which residues were involved in the insertion. In fact, with MSAs 2 and 4, the insertion shifted by up to 15 residues from its relative position in the top alignment, indicating different residues selected for the insertion event.

In addition to the basic black and white pixel plots, it can be colored using a given color scheme and data. For example, Figure 4.11c has the color scheme for the transmembrane protein data applied to it. With this color scheme in place, we can see that the large gapped areas occur in the regions of the inner (beige) or the outer (yellow) loops while the transmembrane regions (green) are relatively free of gaps in most of the alignments. The pixel plot also comes with a utility for producing the same level of publication images as the MSAviewer and the MSAcomparator.

4.5 Batch utilities

In addition to the three novel alignment viewers, Suite MSA has a variety of utilities that can be launched from a script and run on large numbers of sequences sets. These

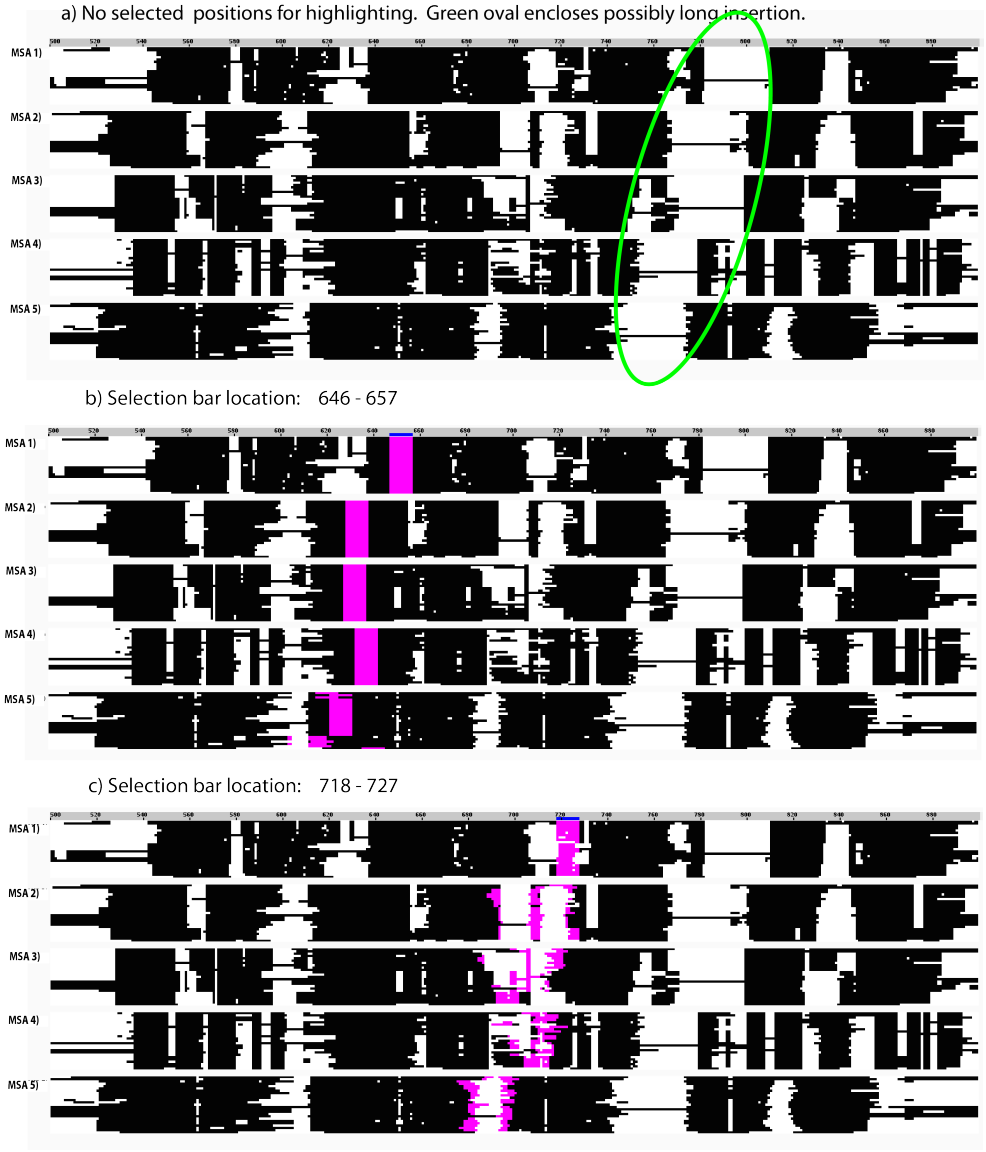


Figure 4.10: Use of the pixel plot. a) The basic pixel plot format displaying the gap pattern. The green outline shows an area where the gaps appear to be largely consistently placed. b) Illustration of the use of the selection bar. It works similar to the selection bar in the MSAviewer, over an area of the alignments with a high degree of consistency between the alignments. c) Illustration of the selection bar over an area of the alignments that show a high level of inconsistency. Starting from the top of the pixel plot image, the alignments were created by: Praline [90], Promals [91], MAFFT [24], MUSCLE [26], and ClustalW2 [22].

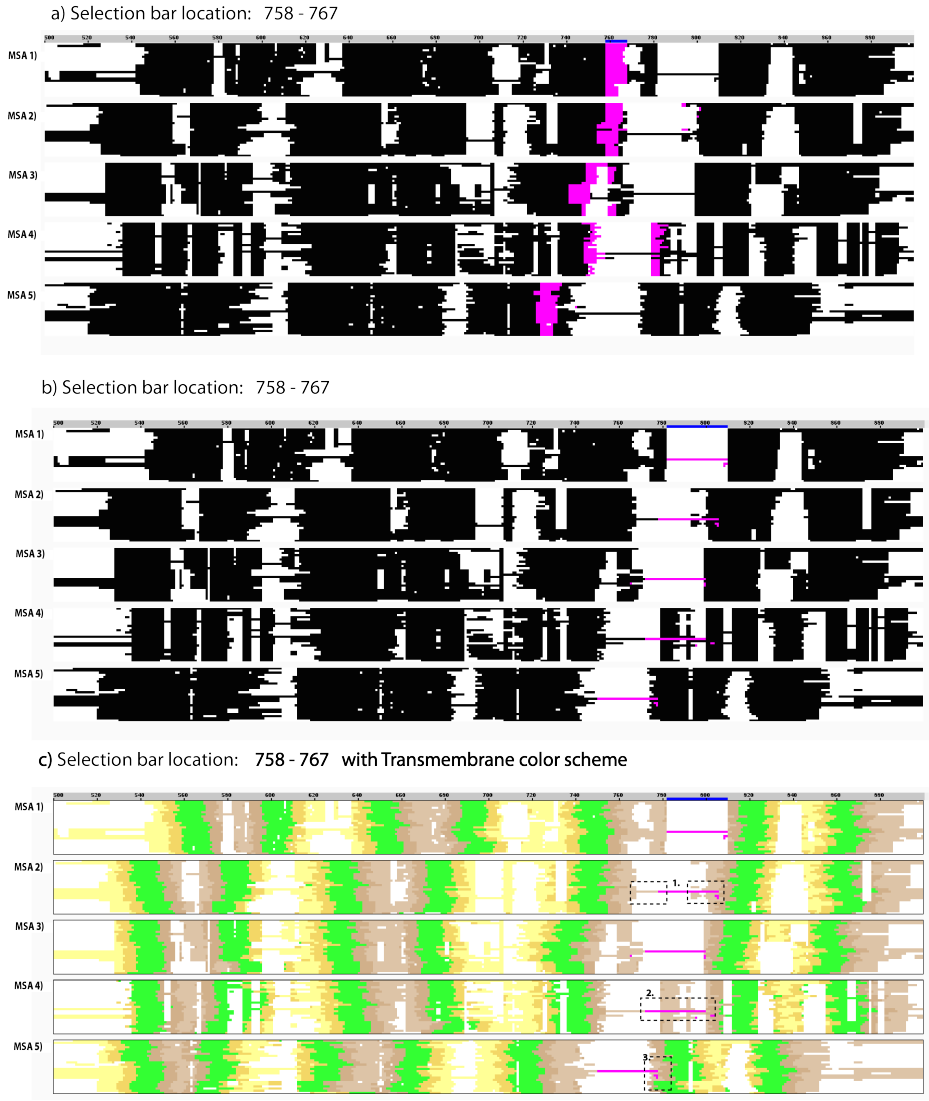


Figure 4.11: : The Use of the pixel plot selection bar. a) The selection bar is located on the left side of the large gapped section. b) The selection bar width adjusted to 28 positions wide and located on the large gapped section. From where the magenta marks fall in the other four alignments, it is obvious that this gapped section was not well captured in each of the alignments, specically with the MUSCLE alignment, MSA 4. c) The transmembrane prediction color-scheme can assist in the assessment of the inconsistency of the long gapped regions. The selection bar is located over a coil section on the inside of the cell, which has lower functional constraint than the actual transmembrane sections and it accounts for the large length variation the sequences in this area, and hence the high level of inconsistencies in the various alignments made from the different alignment programs. The black dashed outlines indicate areas of high inconsistency for this section of the alignment.

utilities were used to perform the evaluation study conducted in Chapter 6 and to form the alignment attributes for the input vectors discussed in Chapter 7. The capability of these utilities include the following:

- calculation of column-wise information score with output file of column-wise score
- calculation of column-wise protein identity with
- calculation of column-wise Kumar protein distance
- calculation of SPS (developers score) with output file of column-wise SPS
- calculation of CSS (Cline shift score) with output file of column-wise CSS
- calculation of CS (column core)

Chapter 5

Development of a simulated alignment benchmark database, SimDom

In this study, we develop an algorithm that selects the MSA that is closest to the optimal from a group of alignments produced by five chosen alignment programs. We use simulated benchmark datasets in the development and test of this algorithm. In this chapter, we discuss the motivation for using simulated data in the evaluation of alignment programs and the importance of using simulated data for training our algorithm. We describe and explain our decision process of the simulation program. We review the guidelines for developing a benchmark database for alignment program evaluation and discuss how the design of our database SimDom follows this standard. We then layout the specific procedure we followed in the creation of SimDom to ensure that the resulting database was sufficiently large in the number of sequences sets and in the variety of protein types and evolutionary histories, so that it can be used both to evaluate alignment programs and to train our algorithm to select the alignment closest to the optimal.

5.1 Motivation

For effective evaluation of MSA programs, benchmark or reference alignments are required. These are the MSAs that are considered reliable enough to represent the

evolutionary history of the sequences involved in the alignment. However, with non-simulated sequences, their true evolutionary history cannot be known. As such, for non-simulated protein sequences, for example, reference alignments are most often inferred based on the structural alignment information. This is usually available for only a limited number of protein sequences or sections of the protein sequences. Currently available non-simulated benchmark MSA databases for protein sequences include PREFAB [68], OXBench [3], HOMSTRAD [59], BAliBASE [65], and SABmark [67] (as described in Section 3.3). Because solving the 3D-structure of proteins is time-consuming and expensive, such information is usually available for only a limited number of protein sequences or sections of the protein sequences. Therefore, the sample size of alignments that contain three or more sequences in these databases is very low compared to the number of known protein families.

There have been questions regarding the heavy reliance on structural alignments to validate sequence alignments [53]. For example, different methods of structural alignment can create alignments that disagree with each other, not only in the structural elements but also in the sequential elements. As such, it is believed that structure can be relied on as accurate only in the same situations where the majority of sequence alignment programs agree, i.e., with closely related sequences [53].

Researchers who are very familiar with the sequences they are studying will also often adjust MSAs manually to create a "reference" alignment. However, there is no standard way to adjust or improve an alignment and the procedure depends heavily on the researcher's knowledge of the function of specific proteins. Manual adjustment is also very time consuming and just as with structural alignment, a manually adjusted alignment cannot always be fully resolved; meaning only those areas in the sequence that are highly conserved can be aligned with confidence. Therefore, while a small number of alignments can be manually adjusted for a specific study, it is not a reliable

and viable method with which to create a database sufficient in size and accuracy to evaluate the automated process of sequence alignment.

A solution to many of the issues of establishing a reference alignment is offered by Hillis [92]. He advocates sequence evolution simulation as an alternative method for obtaining reference MSAs to evaluate MSA algorithms. Sequence simulation methods can generate a set of related sequences with a known evolutionary history, i.e., providing a fully-resolved reference MSA. The datasets generated by simulation, with various evolutionary parameter settings, are also useful for evaluating the robustness, consistency, and efficiency of phylogenetic reconstruction methods based on different MSA methods. In addition to the wide variety of simulation conditions that can be used, this strategy can generate sequence sets that represent a much larger number of protein families than those currently available in non-simulated databases (see Section 3.3).

For these reasons, we created SimDom, a database of biologically realistic simulated protein sequences, complete with true reference alignments and multi-domain protein architectures. We based the configuration of domains and domain architectures on the library of domain prole HMMs provided by Pfam [34]. By doing this, we created a database much larger than any existing non-simulated database. Furthermore, SimDom can always be augmented by additional simulations to allow for growth as demanded by the discovery of new types of proteins, including also proteins without domains. The design of this database allows us to perform a quantitative analysis of alignment programs based on characteristics such as sequence divergence, number of taxa, and number of domains. This will assist us in determining the relative performance of alignment programs and to identify the factors that can be used as indicators of a relative performance shift between the programs. These indicators will be incorporated into the data model for the multi-class classifier that will be

trained to select the alignment that is "closest to the optimal".

5.2 Simulation program

In order to develop an MSA machine learning algorithm that can be successfully applied to any protein sequence alignment problems both presently available as well as potentially new, biologically realistic protein sequences need to be simulated with a sufficient size and scope. To create biologically realistic sequence sets with true reference alignments, a program must be able not only to simulate the evolutionary events including amino acid substitutions, insertions and deletions but also to simulate sequence evolution maintaining the protein domain functions. In this section, we discuss the available protein sequence evolution simulation programs and why we chose REvolver. How REvolver models functional constraints using profile HMMs is also briefly described.

5.2.1 Selection of simulation program

The database we needed to create must contain a set of sequences that are sufficiently similar to real protein sequences containing one or more functional domains. Therefore, we decided to simulate protein sequences including one or more domains. Since not all sequence evolution simulation programs have the ability to generate realistic protein sequences, the choice of the program is critical. Because the type of events that can be included during the simulated evolution is limited to the capacity of the simulator, we decided to choose a simulator that had not only the capability to model amino acid substitutions as well as insertion/deletion events, but also the capability to evolve segments that have domains based on given profile HMMs. Followings are the capabilities we required for the simulation program to create our benchmark dataset:

1. able to provide inputs for guide trees of varying topologies, number of taxa and a multiplicative factor to change branch lengths (evolutionary distances between nodes of the guide tree).

2. able to provide an input to allow for change in evolutionary models based on standard scoring matrices, i.e., PAM120 [12], BLOSUM62 [11], or JTT [93].

3. able to provide an input to set insertion and deletion rates and to limit the areas on the sequences where indels can occur.

4. able to provide for evolving a functional domain in designated areas on the simulated sequences that maintains the constraint of the domain over time.

Table 5.1: A comparison of the capabilities of simulation programs for protein sequence evolution.

	year	guide tree	adjustable model	adjustable indel model	domain modeling	reference
ROSE	1998	✓	✓	✓		[94]
SIMPROT	2005	✓	✓	✓		[95]
INDELible	2009	✓	✓	✓		[96]
iSGv2	2009	✓	✓	✓		[88]
REvolver	2012	✓	✓	✓	✓	[97]

We examined five protein sequence simulators for these required capabilities. The results are summarized in Table 5.1. All five of these programs provide for substitution events. ROSE was the first program to provide for the simulation of insertion and deletion events [94]. SIMPROT was one of the first program to incorporate the application of different rates over different sections of a sequence, which was the first step in providing functional constraint [95]. Nuin et al. [98] performed an evaluation of nine alignment programs using the sequence sets and alignments created by SIMPROT. At the time, SIMPROT was the state of the art protein simulation program with its ability to include and vary indel rates along different sections of

the sequences. However, the mechanism provided by SIMPROT was not sufficient to allow for the preservation of motifs or domains The next large step in the simulation of biologically realistic protein sequences came with the release of two successive versions (1 and 2) of indel-Seq-Gen [99][88]. indel-Seq-Gen version 2 (iSGv2) provided for a mechanism to control both substitution events and indels events on a position by position bases, which, however, required detailed knowledge of each domain to be simulated. While this was a step forward, it still lacked the facility to make use of the domain models as rendered by, e.g., profile HMMs, available from databases such as Pfam [35]. In 2012, REvolver [97] was released. This was the first program to fully provide for the simulation of functionally constrained areas of a sequence by using profile HMMs. REvolver has been shown to maintain 95% of the domain identity over time [97]. Along with the ability to simulate domains, REvolver provides all other evolutionary events: substitutions, insertions and deletions, and allows for multiple areas of differing evolutionary condition on the same sequence. As the program that provides the most biologically realistic sequences, we decided to use REvolver to create a database of protein sequences and reference alignments for our development effort of the MSA machine learning algorithm. We call this database, SimDom.

5.2.2 Simulation models

In this section, we describe the process we used with REvolver to simulate protein sequences that include both functionally non-constrained and constrained sections and how profile HMMs were used with REvolver to simulate domain sections.

We included the models of one to five domains in a sequence set in the design of SimDom. The overview of the simulation model for a full protein sequence containing multiple functional domains is represented in Figure 5.1. The proteins illustrated in this figure, contain one to five domains. The proportion of domain with linker

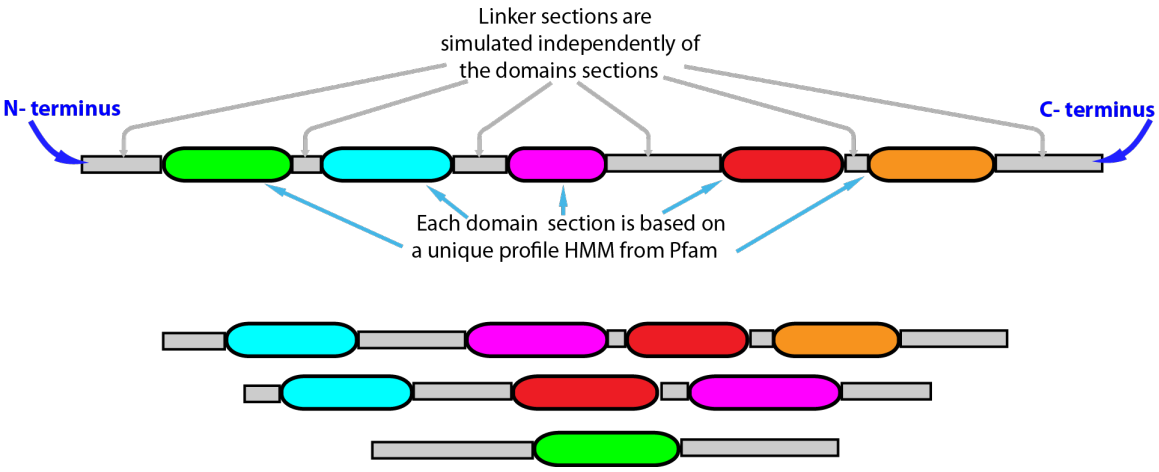


Figure 5.1: Full protein simulation scheme. This illustration shows how the functional domains (colored oblong shapes) are connected with unconstrained linker sections (gray rectangles). Note that the N- and C- terminal regions of proteins are treated just like as "linker" sections for this simulation model.

section also varies. It can be seen in Figure 5.1 that the single domain protein has a lower proportion of amino acids involved in domains than the protein sequence containing four domains. Each domain and linker section is simulated independently and as such can be subjected to varying evolutionary conditions. Using REvolver, this can be accomplished in two ways: 1) by giving REvolver the full layout of the desired protein containing the parameters that would describe the rates and models for each section to be simulated, or 2) by giving REvolver each individual section to be simulated separately but using the same guide tree, the results of which are joined after simulation. For this work, we chose the latter method for the ability to evaluate domain and linker section separately.

REvolver uses the Gillespies algorithm [100] (shown in Algorithm 2) to simulate the sequences. The total event rate, Λ is given as $\Lambda_S + \Lambda_I + \Lambda_D$, where Λ_S, Λ_I and Λ_D are the substitution, insertion and deletion rates, respectively. The following subsections discuss how substitution, insertion and deletion events for both functionally

un-constrained (linker) and constrained (domain) areas are simulated.

Data: Gillespies simulation algorithm

Result: Simulated set of sequences with reference alignment

$\Lambda \leftarrow \Lambda_S + \Lambda_I + \Lambda_D$;

t_{rem} $t_w \sim \text{EXP}(\Lambda)$;

while $t_w \leq t_{rem}$ **do**

randomVariable $\sim$ Uniform();

if *randomVariable* $\leq \Lambda_I/\Lambda$ **then**

doInsertion();

else if *randomVariable* $\leq (\Lambda_I + \Lambda_D)/\Lambda$ **then**

doDeletion();

else

doSubstitution();

Λ = updateEventRate();

$t_{rem} \leftarrow t_{tem} - t_w$;

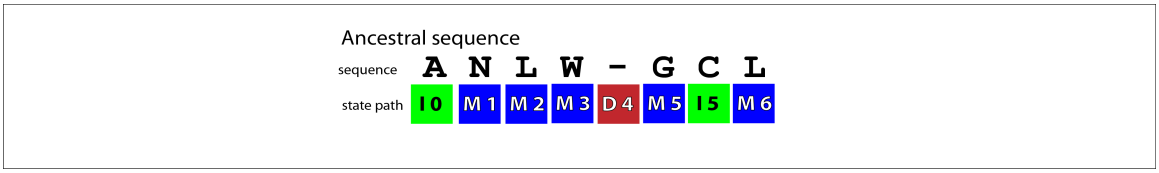
$t_w \sim \text{Exp}(\Lambda)$;

end

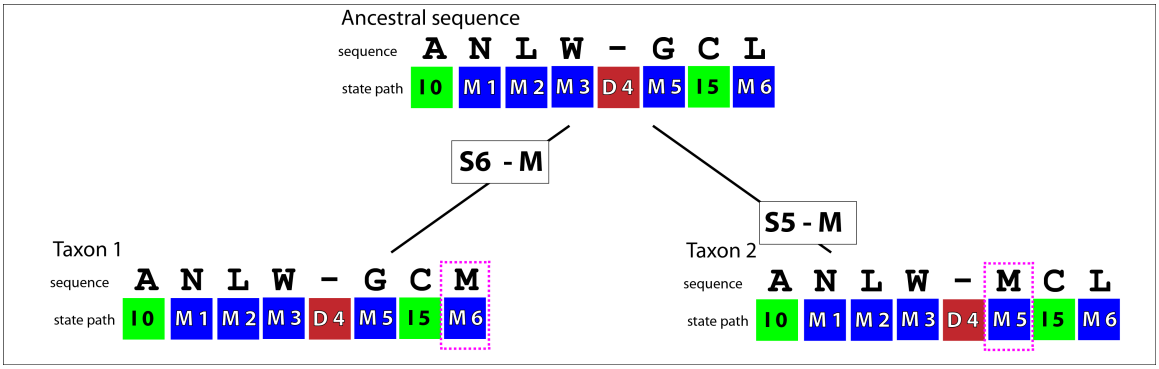
Algorithm 2: The Gillespies algorithm used for protein sequence simulation in REvolver [97]. Λ is the total event rate and $\Lambda = \Lambda_S + \Lambda_I + \Lambda_D$ where Λ_S, $\Lambda_I = (L+1)\delta_l$ and $\Lambda_D = L\lambda_D$ are the total substitution, insertion, and deletion rates, respectively.

5.2.2.1 Substitution models

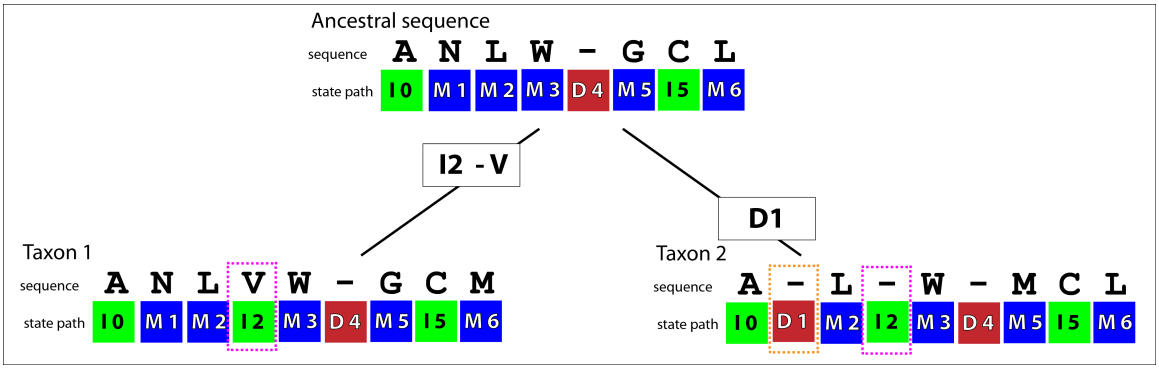
REvolver uses different algorithms to simulate substitution events, depending on the presence or absence of a domain model (profile HMM). Non-constrained segments, those that contain no domains, include the N- and C- terminals of a sequence as well as the linker regions between domains. These segments will, for simplicity, be referred



(a) State path of the ancestral sequence. The color of the units indicates the type of state responsible for the emission of the amino acid indicated above it. The dark blue units indicate a match state, the green units indicate an insertion state and the red units indicate a deletions state. The specific state is shown in the unit label. This illustration emphasizes that deletions are possible in the ancestral sequence.



(b) Tracking substitution events. The two substitution events, L to M at position 6 (S6-M) and from G to M at position 5(S5-M), are illustrated. These events cause a change in the sequence but no change in the state path.



(c) Tracking indel events. The magenta outline indicates the position of the insertion event that occurred in Taxon 1 (I2-V) along with the corresponding gap insertion in Taxon 2. The orange outline indicates the deletion event that occurred in Taxon 2 (D1) with no corresponding change in the state path of Taxon 1.

Figure 5.2: Tracking evolutionary events during the simulation of a domain sequence. How different evolutionary events are tracked during the simulation is illustrated.

to as linkers. The ancestral sequence for a linker segment is modeled by a Markov chain characterized by Q, a matrix containing the instantaneous rates q_{ij}, which is the product of the relative substitution rate from amino acid i to amino acid j, ρ_{ij}, and the amino acid frequency π_j. In REvolver, there are 14 substitution matrices for the user to select from. In this study, we used the JTT substitution model. The substitution rates for any amino acid is given as $q_i = \Sigma_{j \neq i} q_{ij}$ and the total substitution rate is given as $\Lambda_S = \Sigma_{l=1}^{L} q_{i_l}$ where l is the position on the ancestral sequence and L in the sequence Length..

A substitution rate at site l of the ancestral sequence is calculated as $q_i r_l$, where r_l is the scaling factor and i is the current amino acid at site l, where the substitution occurs. l is chosen proportional to $q_i r_l$ and the probability that amino acid i is substituted with amino acid j is proportional to q_{ij}/q_i for $i \neq j$.

When a domain model is used, the simulation is started by generating the ancestral sequence using the profile HMM, yielding both a sequence of amino acids and gaps, and a state path. The state path is the series of the states from the profile HMM that generated the symbols of the sequence. An example of an ancestral sequence with its state path is shown in Figure 5.2a. As the ancestral sequence evolves along the guide tree, the evolutionary events are recorded in the sequence as well as in the state path of the sequence.

Substitutions are determined by Q_l, the customized model for each site l. The rates in the model for each site are based on the substitution model selected by the user (in this study, JTT) as well as the probabilities of the profile HMM for the site. For each site l, the components of Q_l are given as $q_{ij} = \Sigma_{j \neq i} \rho_{ij} e_{j_{M_x}}$ where ρ_{ij} is the relative rate of amino acid substitution taken from the substitution model (i.e., JTT) and $e_{j_{M_x}}$ is the specific emission probability for the amino acid as given in the profile HMM. The probability that amino acid i is substituted with amino acid j is

proportional to q_{ij}/q_i where for $q_i = \Sigma_{j \neq i} q_{ij}$. A substitution event does not affect the state path. As an example, Figure 5.2b shows substitutions along two branches of the guide tree, resulting in changes of the amino acid in different positions in the two taxa while no change occurred in their state paths.

5.2.2.2 Indel models

In an unconstrained linker section, the rates for insertions and deletions, are λ_I and λ_D, respectively. The total deletion rate is given as $\Lambda_D = L\lambda_D$, where L is the length of the sequence. Since insertions can occur before the first amino acid and after each amino acid of the current sequence, the total insertion rate is given as $\Lambda_I = (L+1)\lambda_I$. The positions of indel events are uniformly distributed, with the length of the individual event determined by the Zipfian length probability distribution [101]. After the length of an insertion is determined, the amino acids are selected by the equilibrium frequency, π. We used an indel rate of 0.0025 and the geometric distribution for length of indel event in the simulation of this benchmark dataset.

In the domain section, indel events cause change to both the sequence and the state path. The probability of placing an insertion after position l is $P(M_x, I_x)$ if associated with a match state, or $P(I_x, I_x)$ if l is associated with an insertion state. The length of the insertion is drawn from the Zipfian length probability distribution [101]. The amino acid is selected by the emission probability of I_l and placed immediately to the right of I_l. For an insertion resulting from the transition between M_i to I_i, the new state I_i is inserted between the M_i and M_{i+1} of the state path of all the taxa. If the insertion resulted from the transition between I_i to I_i, the new state I_i is inserted between the last I_i and M_{i+1} of the state path. An example shown in Figure 5.2c illustrates an indel occurring in position 2 resulting in a unit labeled I2 placed between M2 and M3 in both taxa. The symbol for the inserted amino acid V is shown above

this unit in Taxon 1 while a gap is placed above the unit in Taxon 2.

For a deletion, if the site is associated with the match state M_{x-1} then that match state M_x is replaced by the corresponding delete state D_x in the state path and a gap symbol is placed in the sequence. The deletion probability is $P(M_{x-1}, D_x)$. If l is associated with the deletion state D_{x-1}, then again M_x is replaced by deletion state D_x and the deletion probability is $P(D_{x-1}, D_x)$. In this case, no change is needed in the state paths of the other taxa.

5.3 Guidelines for the benchmark dataset used for alignment program evaluation

When evaluating the performance of alignment programs, in addition to the "true" reference alignments, the type of sequences in the database must be appropriate to the purpose of the evaluation. As such, the developers of the benchmarks must have clear objectives when using the specific benchmark and select sequence sets whose characteristics are suitable for the purpose of the analysis. To assist in this endeavor, guidelines for establishing good benchmark datasets have been proposed [102][103]. We have incorporated these guidelines into designing our benchmark dataset. There are six criteria. Each of these criteria and how our design conforms to it is discussed below:

- **Relevance**: Benchmarks should be adapted to the application. Our objective is to evaluate the relative performance of various alignment techniques on full protein sequences containing one or more domains. We need to establish a set of sequence characteristics indicative of where the different programs outperform other programs (as measured by alignment metrics). It should also allow users to select the individual subgroups of this database that correspond to the char-

acteristics of the sequences they are studying, such as the number of sequences, sequence similarity, amino acid composition, and number of domains.

- **Solvability**: The tasks specified by the benchmark should not be either too easily solved nor too difficult. Making the task too easy prevents the results from being applied to the more difficult problems present in ongoing research. If the tasks are too difficult, the results are poor across the board and yield little data for comparisons. The difficulty of the alignment problem as represented by the sequence sets of SimDom represents the full range of alignment problems as seen in various studies: from very closely related sequences, as when studying individuals of the same population, to very distantly related as when comparing paralog proteins. The SimDom dataset includes sequences that are:

  - equally distant sequences from the root sequence,
  - uniformly varying in distance from the root sequence and
  - randomly varyingly distant from the root sequence.

  Additionally, the evolutionary rates were varied among the sequences as well as between the domains and linkers in a single sequence.

- **Scalability**: The task specified by the benchmark should be sufficiently large so that mature algorithms can be fully tested, but not too large that newly developed techniques cannot be properly tested. In SimDom, the sequence sets along with their evolutionary history (as represented by their true alignment) vary from single domain proteins with sequences of 500 amino acids (aa) to those with five domains that are more than 3000 aa long. There is also variation in the number of sequences (taxa) per alignment (8, 16 and 32 taxa). This will allow the user to select the subgroup of alignments from our datasets that best

complement those that are being studied to optimize the alignment programs with.

- **Accessibility**: The datasets along with test results must be publicly and easily available along with test results. We have all of our data available on a website that contains the datasets broken down into various libraries that will allow the user to pick and choose according to their needs (see Section 5.6). Along with this, our own test results will be available.

- **Independence**: The method or approach of evaluation should be independent of any of the alignment techniques evaluated. This is to avoid bias toward any particular algorithm. The reference or true alignment in SimDom were generated during the simulation process. Therefore, they represent the exact evolutionary relationship between the sequences. No other alignment program or adjustment technique was used to create the reference alignment.

- **Evolution**: The database needs to be updated to present the current challenges in sequence alignment. This is to prevent stagnation of "evolution" of the alignment techniques, meaning that researchers inadvertently fine-tune algorithms to optimize performance on specific types of alignments. New sequence sets should be added as new problems arise to avoid stagnation. We provide the full plan for the creation of SimDom along with scripts and parameters used. Therefore, if a challenge arises that does not have sufficient representation within the benchmark we have developed, it would be possible to simulate new sequence sets that more closely match the sequences involved in the challenge. With REvolver, it is also possible for the user to select any evolutionary model including a custom model, any distribution of indels, both for occurrence and for length of events, and any profile HMMs.

5.4 Design of the SimDom database

We designed SimDom to be used as both a benchmark database for the evaluation of alignment programs and the source of alignments to form the training data for the development and proof of concept for of the machine learning algorithm to select the alignment "closest to the optimal". In addressing both purposes, we needed to remedy the two most prevalent issues faced when dealing with non-simulated databases: the number of alignments available and the scope of alignments available. By scope we mean the number of different domains present in alignments of the dataset, the variation in the amount of evolutionary distance present within the sequence set, and the variation between the proportion of the segments in the alignment that contain domains and those that do not. To create a database with a large number of sequence sets and a variation in its scope, we based our simulation on a pattern of 1000 combinations of domains with 144 variations of evolutionary parameters.

A total of 1750 domain models were obtained from Pfam (version 29, 2016) [36]. The combination of one to five domain models was determined using the information provided in the sequence architecture database obtained from Pfam, where an architecture is a list of the Pfam domains that appear together in a protein sequence. We set two conditions for a domain to be selected as a model in SimDom: 1) each domain must have a seed alignment. If a domain does not have a seed alignment, it does not have a profile HMM and could not be modeled by REvolver; and 2) all domains put together in a combination to simulate a multi-domain sequence must have occurred together in at least one architecture as reported by Pfam.

The procedure for forming domain combinations is given in Algorithm 3. Note that this domain selection process does not allow for duplicated domains within any single combination or in more than one combination. This was to have the largest possible

number of different domains present in the limited number of domain architectures our database can contain. It should also be noted that the Pfam database is large including over 138000 domains in more than 187,000 architectures, many more than we plan to use for our simulatation. It was thus not possible to include all domains in all architectures in our simulation.

For each domain combination, a set of lengths of linkers (a linker length is the number of amino acids in the ancestral root sequence) is determined by a random number assignment ranging from 25 to 150 amino acids (aa). As such, each domain architecture is defined by a given combination of domains and a set of linker lengths and 1,000 of domain architectures were produced as summarized in Table 5.2.

For each of the 1000 domain architectures, simulation of protein sequences was done using 144 different sets of evolutionary parameters with 6 types of guide tree topologies, 8 variations of evolutionary rates and 3 variations on taxon numbers as shown in Table 5.3. The first type of variation is the difference in the guide tree topologies. For each tree type, the taxon number varies (8, 16 or 32 taxa). The type of guide trees are as follows (Figures 5.4-5.9 are provided at the end of this section):

- Full Ultrametric (FU) tree (Figure 5.4). This is a full binary tree. The 32-taxa tree has the same distance of 1.0 from the root to each tip. The 16-taxa tree is derived from the 32-taxa tree, such that the distances between taxa of the same name are the same in both trees. Likewise, the 8-taxa tree is derived from the 16-taxa tree.

- Full Non-ultrametric (FN) tree (Figure 5.5). This tree allows for different distances from the root to the tips with the maximum distance being 1.0.

- Clustered Ultrametric (CU) tree (Figure 5.6). In this tree, all taxa have the distance of 1.0 from the root to the tips. The taxa are grouped in clusters of

```
Data: List of architectures, List of seed Alignments
Result: List of domain combinations
2, 1)
array of combo_counts ← (25, 50, 125, 250, 550)
allowed_combinations ← empty list
forall Architectures do
    qualified_domains ← empty list
    foreach domain in current Architecture do
        if domain is in List of seed Alignments and not in qualified_domains
         then
            add to qualified_domains
        end
    end
    if qualified_domains is not empty then
        create combo with all domains in qualified_domains add combo
         allowed_combinations list.
    end
end
sort allowed_combinations by number of Ids in each combination
domain_combos ← empty list
used_domains ← empty list
for  n = each domain_number do
    number_combos = 0;
    while number_combos < combo_counts for n do
        current_combo ← ""
        domainsAdded ← 0
        extract top allowed_combinations from list
        while domainsAdded < n] do
            take next domain from current combination if  add domain to
             current_combo add domain to used_domains increment
             domainsAdded then  domains not in used_domain list
            end
        end
        add combo to domain_combos
        increment number_combos
    end
end
return the domain_combos
```

Algorithm 3: Creates 1000 combinations of domains to be used as the base pattern for SimDom.

four which creates a different number of nodes between each taxon and the root. As before, the 16- and 8-taxa trees are derived from the 32- and 16-taxa trees, respectively. The green outline in Figure 5.6 indicates the cluster made up of taxa A1-A4, which is present in all three trees

- Clustered Non-ultrametric (CN) tree (Figure 5.7). This tree allows for different distances from the root to the tips, the maximum being 1.0. The taxa are grouped in clusters of four, which are much more obvious than on the ultrametric trees shown in Figures 5.4and 5.6. As before, the 16- and 8-taxa trees are derived from the 32- and 16-taxa trees, respectively.

- Random Ultrametric (RU) Tree (RU) (Figure 5.8). This tree allows for different numbers of nodes between the taxon and the root while the distances from the root to the tips are 1.0 for all taxa. The simulation for each of the 1000 protein architecture uses its own unique tree. However, the same trees are used for each of the 144 different simulations for a given protein architecture. For this topology group, 32-, 16- and 8-taxa trees are generated independently.

- Random Non-ultrametric (RN) Tree (Figure 5.9). This tree allows both different distances from the root to the tips, the maximum being 1.0, and different numbers of nodes between the taxon and the root. Each protein architecture has its own unique tree. However, these trees are shared across the 144 difference simulation conditions for a given protein architecture.

The next types of variation in evolutionary parameters specify the amount of divergence in sequences. The base-level divergence of each sequence region was determined based on the "inherent divergence" associated with a specific domain profile HMM. The inherent divergence for each domain was calculated from the phylogeny

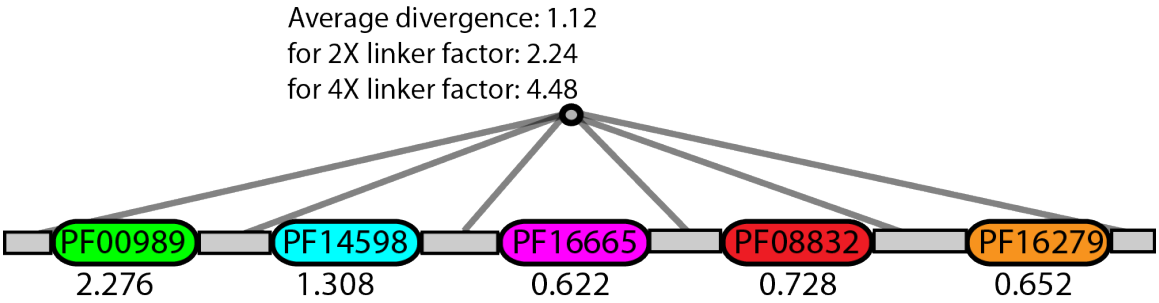


Figure 5.3: The example of the linker divergence calculation. The protein sequence in the example has five domain regions, each given with the Pfam ID and its inherent divergence (shown below in the number of amino acid substitutions per site). The average inherent divergence for this protein is calculated as 1.12. With the tree branch factor of 1.0 and linker factor of 2 and 4, the linker divergence can be calculated as 2.24 and 4.48, respectively.

provided by Pfam for each of the seed alignments of the domain. For each domain, the average pairwise distance between taxa was calculated from the phylogeny. This average distance, divided by 2, was used as the inherent divergence of the domain.

A branch factor, or more specifically "tree branch factor", is used to scale the evolutionary rate for a simulation. We used four branch factors, referred to as the tree branch factors values: 0.5, 1.0, 1.5, and 2.0. For segments of the sequences that contain domains, the total divergence applied during a simulation is given by the inherent divergence times the tree branch factor.

The linker sections are free of functional constraints and they tend to be much more divergent than the domain regions. Thus, we used an additional branch factor during the simulation of the linker sections, which is referred to as "linker branch factor", in conjunction with the tree branch factor. An example below illustrates how these branch factors are applied to the linker sections shown in Figure 5.3:

- The inherent divergence of each domain to be used in the protein sequence is averaged. For the five domains shown in Figure 5.3 the average inherent divergences is 1.12.

- The average inherent divergence is multiplied by the tree branch factor: either 0.5, 1.0, 1.5 or 2.0, depending on the simulation. In the example, 1.0 is used.
- This product is further multiplied by the linker branch factor (either 2 or 4). The resulting linker divergence is applied to all of the linker sections in the protein sequence during the simulation.

One novel aspect of how we created the SimDom database was that the simulation of protein sequences was done only for the 32-taxa guide trees for FU, FN, CU and CN. As described before and shown in Figures 5.4-5.7, the sequence sets for the lower number of taxa were taken from the 32-taxa simulations by extracting specific sequences according to the guide trees of the lower taxa tree. This strategy allows a quantitative measure of how the increased number of sequences affects the relative performance of the programs.

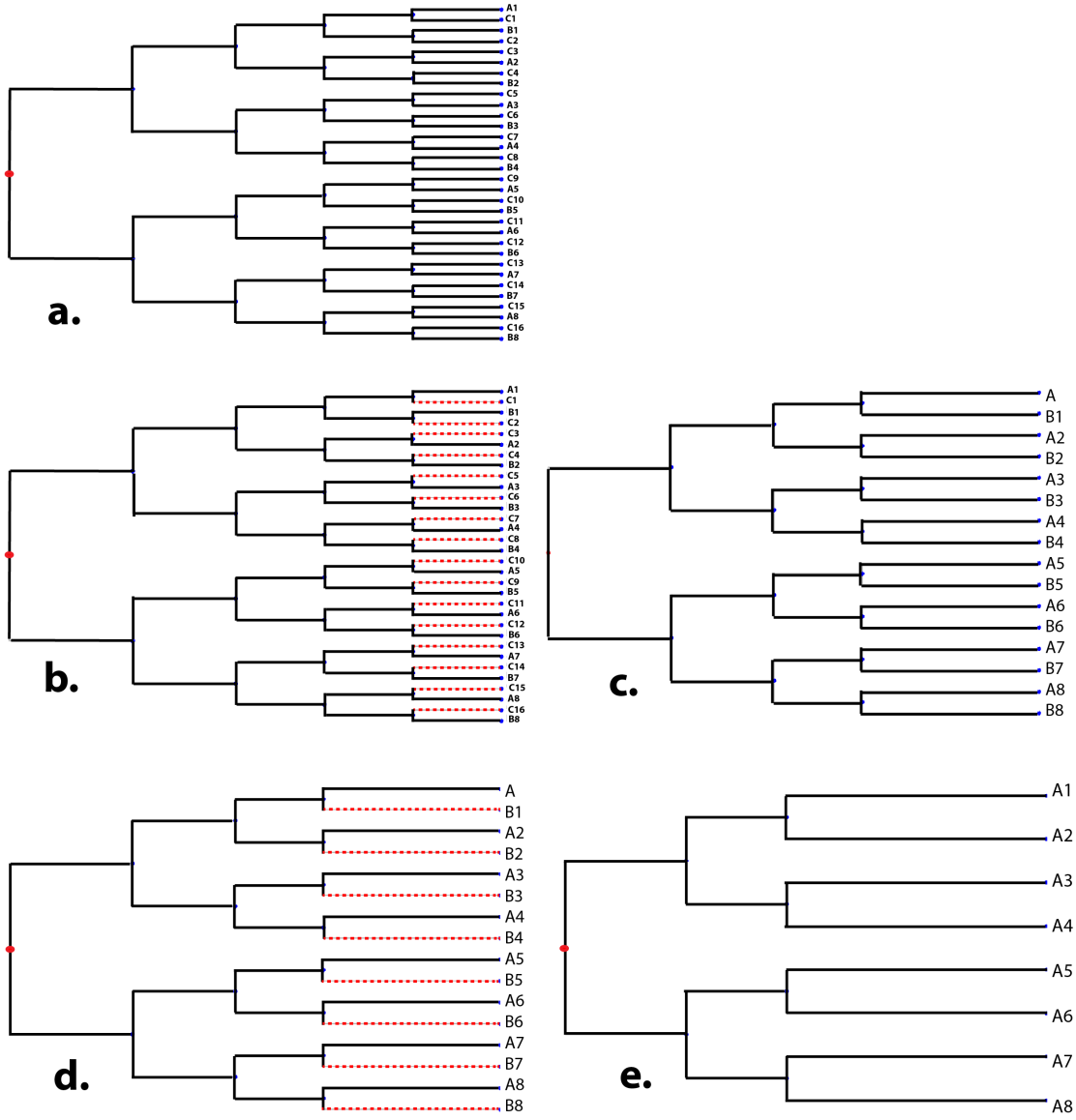


Figure 5.4: Guide trees used in the subset FU (Full Ultrametric). a: A complete binary tree with 32 taxa (FU32). b: Dotted red lines indicate the taxa removed from the 32-taxa tree to create the 16-taxa tree shown in c. c: A complete binary tree with 16 taxa (FU16). d: Dotted red lines indicate the taxa removed from the 16-taxa tree to create the 8-taxa tree shown in e. e: A complete binary tree with 8 taxa (FU8). All trees have equal total branch lengths from the root to taxa.

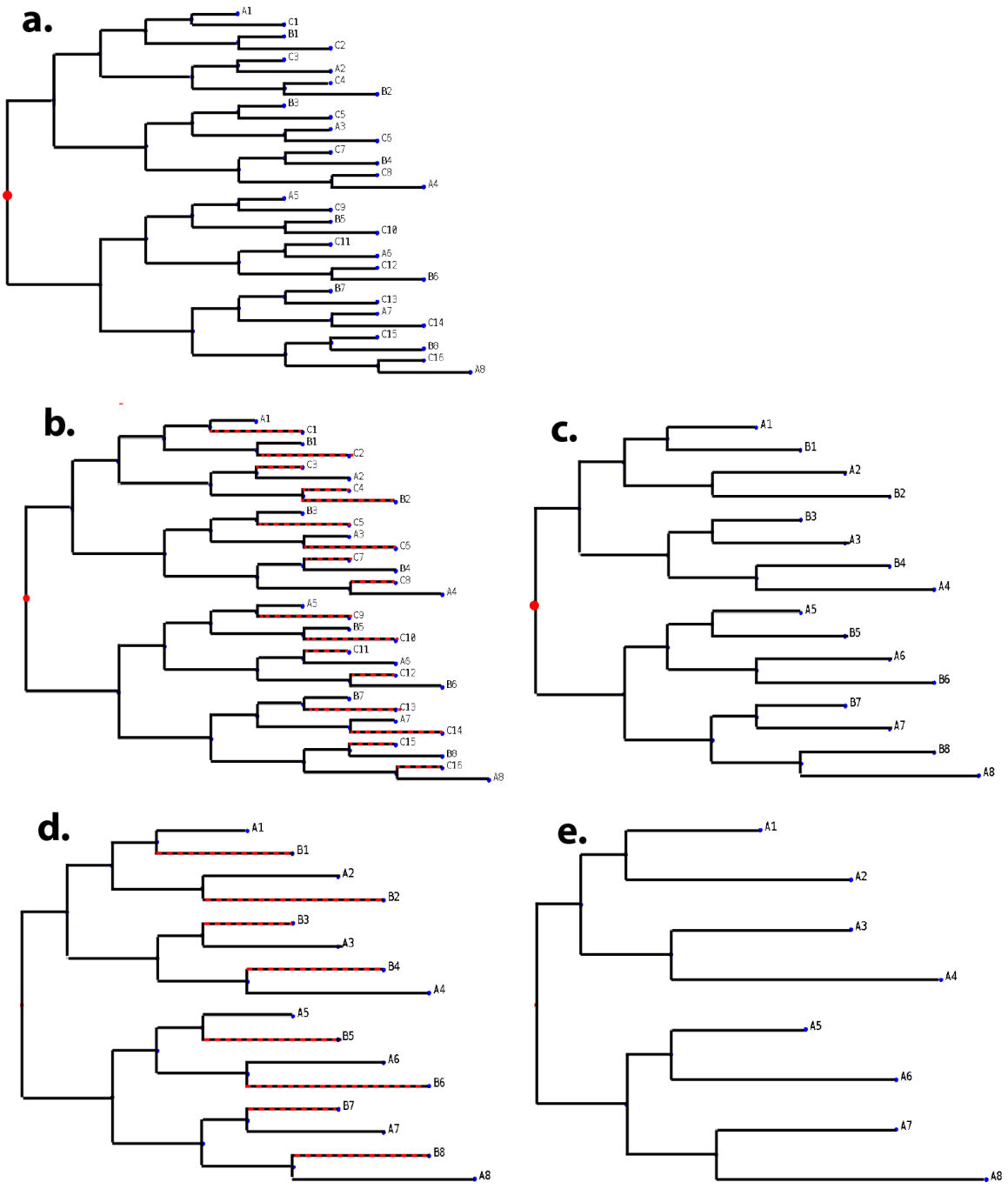


Figure 5.5: Guide tree used in subset type **FN** (Full Non-ultrametric). a: A complete binary tree with 32 taxa (FN32). b: Dotted red lines indicate the taxa removed from the 32-taxa tree to create the 16-taxa tree shown in c. c: A complete binary tree with 16-taxa (FN16). d: Dotted red lines indicate the taxa removed from the 16-taxa tree to create the 8-taxa tree shown in e. e: A complete binary tree with 8 taxa (FN8). Taxa have different total branch lengths from the root.

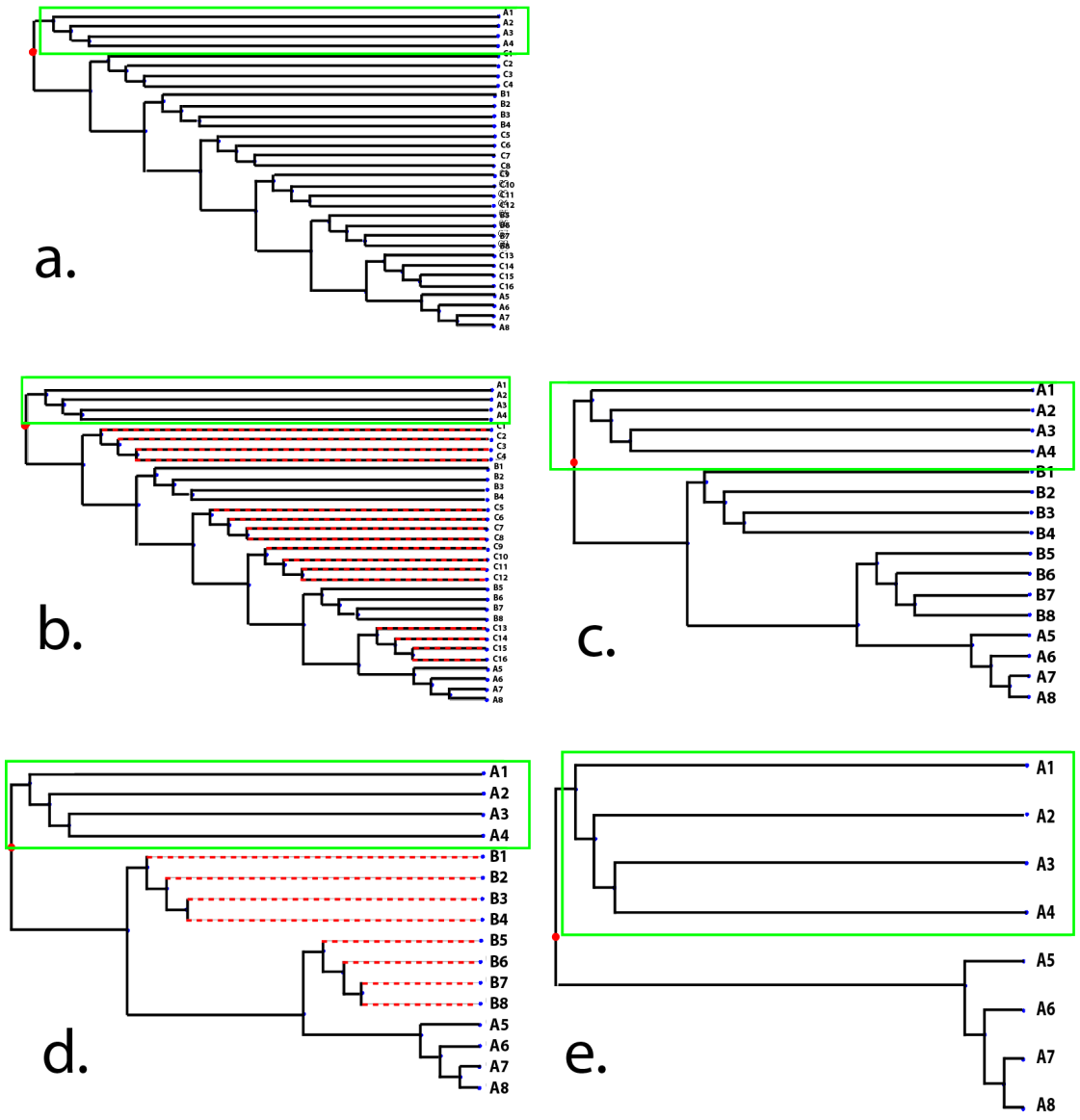


Figure 5.6: Guide tree used in subset type **CU** (Clustered Ultrametric). a: A binary tree with 32 taxa clustered in 8 groups (CU32). b: Dotted red lines indicate the taxa removed from the 32-taxa tree to create the 16-taxa tree shown in c. c: A binary tree with 16 taxa clustered in 4 groups (CU16). d: Dotted red lines indicate the taxa removed from the 16-taxa tree to create the 8-taxa tree shown in e. e: A binary tree with 8 taxa clustered in 2 groups (CU8). All trees have equal branch lengths from the root to taxa. Green boxes indicate the example of the same set of four taxa (A1-A4) in 32-, 16-, and 8-taxa trees.

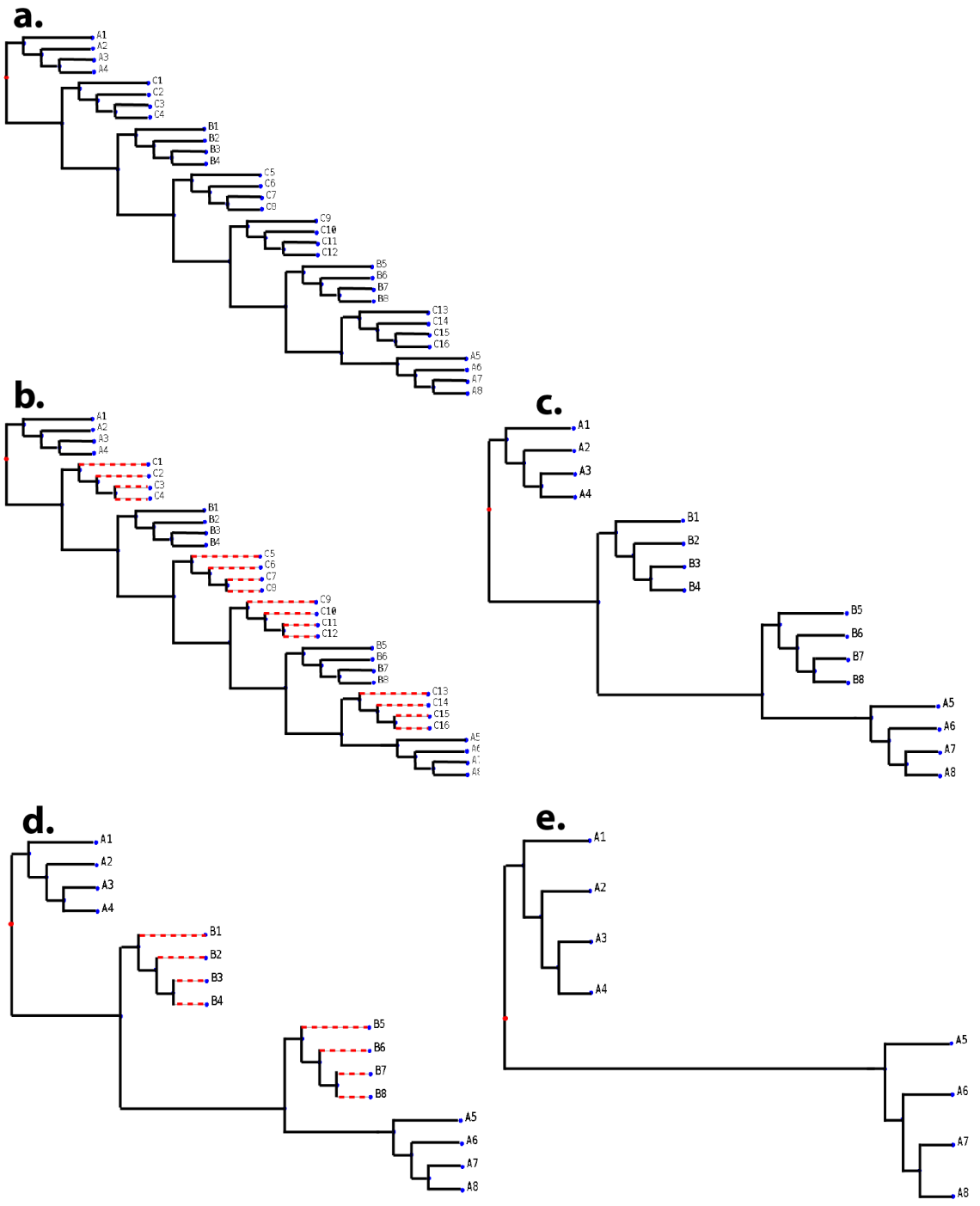


Figure 5.7: Guide tree used in subset type **CN** (Clustered Non-ultrametric).). a: A complete binary tree with 32 taxa (CN32). b: Dotted red lines indicate the taxa removed from the 32-taxa tree to create the 16-taxa tree shown in c. c: A complete binary tree with 16 taxa (CN16). d: Dotted red lines indicate the taxa removed from the 16-taxa tree to create the 8-taxa tree shown in e. e: A complete binary tree with 8 taxa (CN8). Groups of taxa have different total branch lengths from the root.

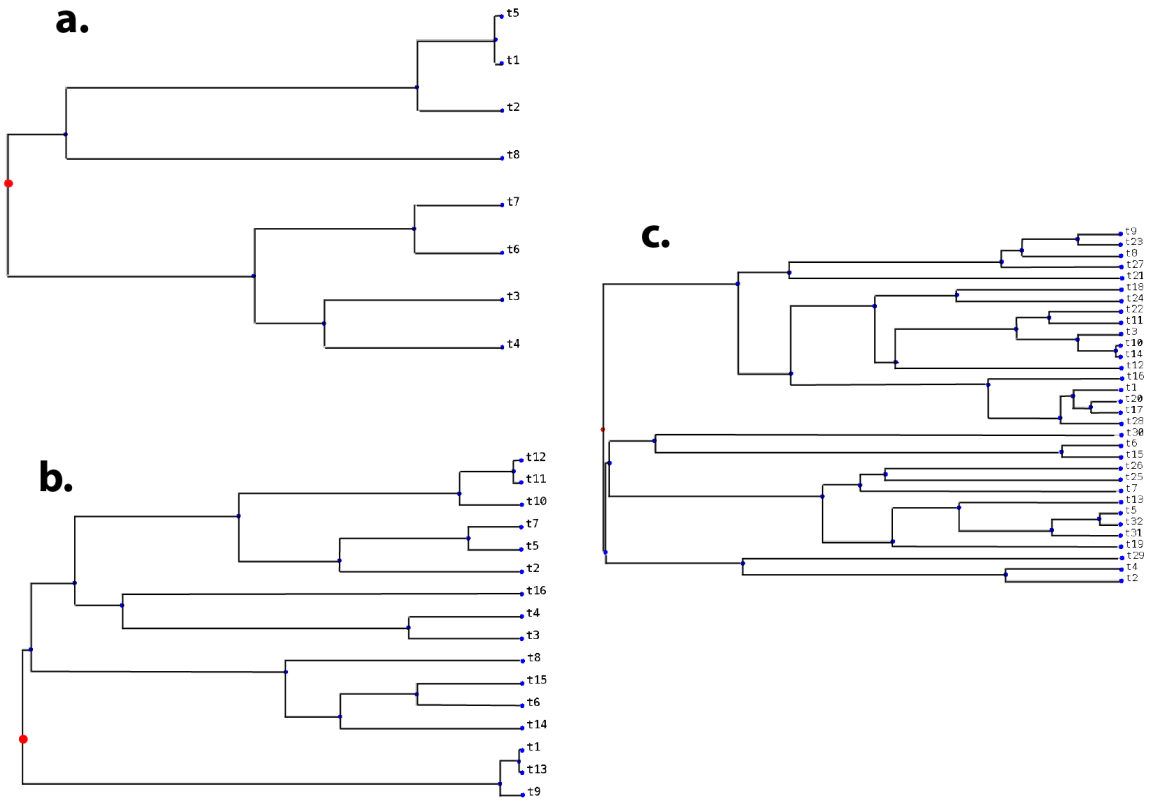


Figure 5.8: Guide tree used in subset type **RU** (Random Ultrametric).). a: A random binary tree with 8 taxa (RU8). b: A random binary tree with 16 taxa (RU16). c: a random binary tree with 32 taxa (RU32). All trees have equal total branch length from the root to taxa. For all trees, each path from the root to a leaf has a random number of nodes. The tree used for each of the 1000 domain combination is unique but the same tree is used with the same domain combination through each of the 144 simulation scenarios.

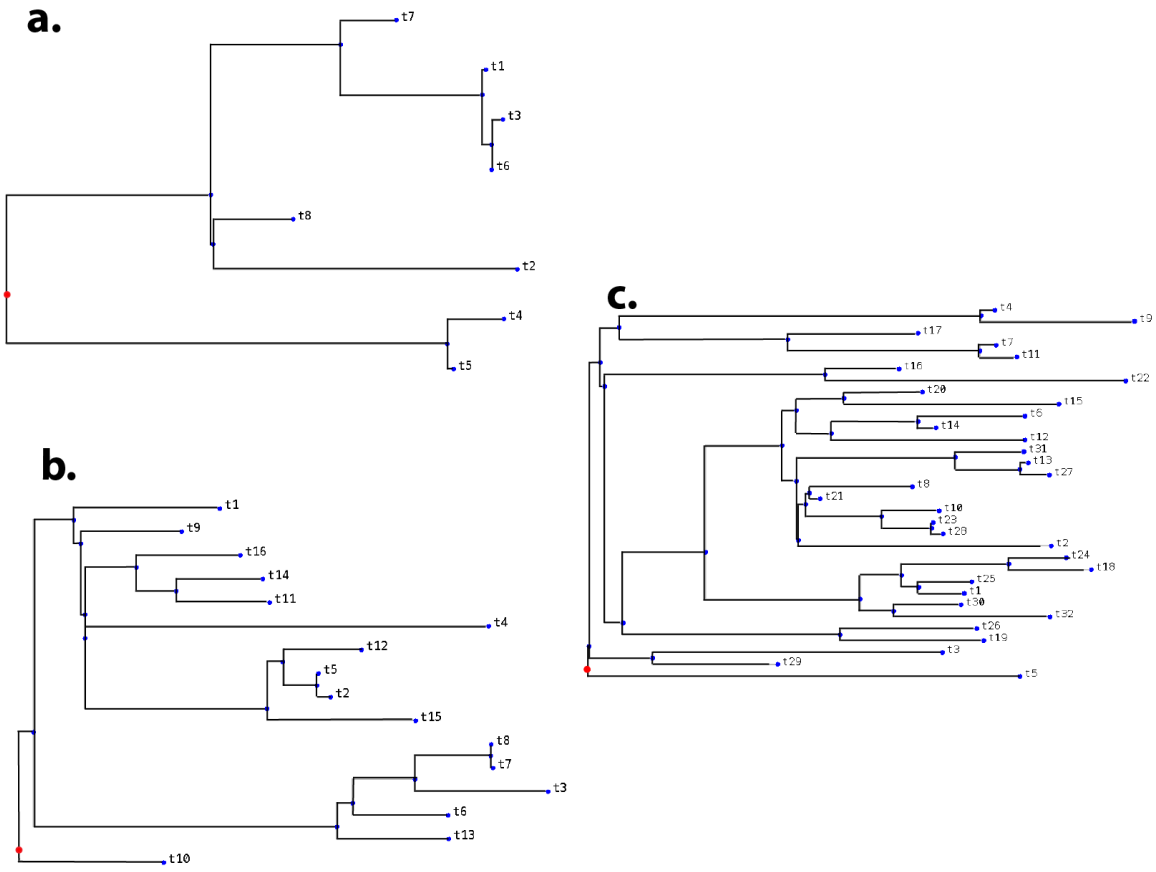


Figure 5.9: Guide tree used in the subset **RN** (Random Non-ultrametric). a: A random binary tree with 8 taxa (RN8). b: A random binary tree with 16 taxa (RN16). c: A random binary tree with 32 taxa (RN32). All trees have randomly determined total branch lengths from the root to taxa as well as randomly determined number of nodes. The tree used for each of the 1000 domain combination is unique but the same tree is used with the same domain combination through each of the 144 simulation scenarios.

Table 5.2: Number of simulated protein sequence sets and domain numbers.

number of domains	number of sets per 1000	ID range per 1000	Total number of sequences
5	25	1-25	3,600
4	50	26-75	7,200
3	125	76-200	18,000
2	250	201-450	36,000
1	450	451-1000	79,200

Table 5.3: Number of variables.

Area	number values	values
Guide tree topology	6	FU, CN, CU, CN, RU, RN
Number of taxa	3	8, 16, 32
Branch factor	4	0.5, 1.0, 1.5, 2.0
Linker factor	2	2X, 4X

5.5 Sequence properties of the SimDom benchmark database

The SimDom database is composed of 144 subsets of alignments, each subset containing 1000 alignments and created by varying the simulation conditions. These datasets are grouped into four categories based on the type of guide trees (ultrametric or nonultrametric) and the linker branch factor (2X or 4X).

In Tables 5.4 - 5.7, we summarized several sequence properties including alignment lengths, average pairwise identity, and average information score.

The ultrametric tree subgroups have higher average sequence divergence than the non-ultrametric ones. This is because while the sequences generated by ultrametric tree simulation have the same distance from the root, some of the sequences generated by non-ultrametric simulation have shorter distances from the root than others. The longest distance from the root in a non-ultrametric tree (e.g., FN) is equal to the distance of all sequences in the ultrametric tree counterpart (e.g., FU).

For all subgroups increasing evolutionary distance (branch length) increases the alignment length while both the protein identity and the information score are low-

ered. As described in Section 3.1, the information score is used as an indication of the conservation. Thus, the positive correlation between the protein identity and the information score as well as the negative correlation between the evolutionary distance and these two quantities is expected. It was also expected that the ultrametric topology groups would have lower protein identities resulted from having more evolutionary distance during simulation than their non-ultrametric counterparts.

5.6 The advantage of the SimDom database

The advantages of the SimDom design can be summarized as follows:

- SimDom contains 144,000 sequence sets each with the true reference alignment involving the full length of the sequences. In contrast, as described in Section 5.1, non-simulated protein benchmark databases (e.g., BAliBASE) do not have true alignments of their sequences. The alignments included in these databases are educated approximations for only those areas of the sequences whose 3D-structures are known. For the regions where there is no 3D information, confidence for the alignments are even lower. It has also been noted that structural alignments are prone to mistakes when sequence divergence is high, similar to the problem sequence alignment suffers.

- SimDom contains biologically realistic simulated sequences. The simulation program we used, REvolver, uses profile HMMs to capture evolutionary constraints in the domain sequences. By preserving the domains during protein sequence evolution, our simulation more likely preserves the functions of simulated proteins.

- SimDom uses combinations of domains as they appear in real protein sequence architectures obtained from Pfam. This again ensures our protein sequence simulations to be biologically realistic.

- Each combination of domains is subjected to 144 different simulation conditions, giving rising to a large variation in alignment features on which to evaluated the performance of alignment programs. This large number of datasets is grouped by specific characteristics such as sequence number, number of domains, guide tree topology and sequence divergence level. This large variety in protein sequences and their evolutionary patterns allows the evaluation of alignment programs on many types of proteins. It also avoids over-training of programs by optimizing alignment functions on a limited set of data.

5.7 Accessibility to the SimDom database

The SimDom database is packaged in the various bundles based on a specific sequence set characteristic. Table 5.8 summarizes these packages. The groups are based on the simulation parameters as well as characteristics of the reference MSAs.

For each sequence set, the linker sections and the domain sections are clearly specified in each LorD (Linger or Domain) file. In this FASTA format text file, each amino acid position for each sequence is represented with a character, either 'D' or 'L', indicating the type of section each amino acid originated from: 'D' for part of a domain simulation or 'L' as part of the linker simulation. Also available is the complete set of ancestral sequences generated from each simulation, along with the reference (true) alignments.

The database will be available from: http://bioinfolab.unl.edu/canderson/SimDom/

Table 5.4: The statistics on the ultrametric tree subset of the SimDom database with the linker factor of 2. Each subgroup has 1000 sequence sets. "Branch": tree branch factor; "Algn. len": alignment length; "Ident": the average pairwise protein identity; "Info" the average column-wise information score; and "SD": standard deviation.

	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	538.5 (276.8)	0.38 (0.06)	2.872 (0.18)
	1	557.7 (287.1)	0.23 (0.04)	2.503 (0.13)
	1.5	576.8 (296.1)	0.18 (0.03)	2.378 (0.11)
	2	591.6 (304.7)	0.15 (0.02)	2.332 (0.11)
16 Taxa	0.5	548.2 (281.7)	0.39 (0.06)	2.692 (0.21)
	1	577.9 (299.0)	0.25 (0.05)	2.249 (0.17)
	1.5	604.7 (310.1)	0.19 (0.03)	2.091 (0.14)
	2	631.1 (327.7)	0.16 (0.03)	2.039 (0.15)
32 Taxa	0.5	560.5 (287.4)	0.40 (0.06)	2.629 (0.22)
	1	600.8 (310.8)	0.26 (0.05)	2.156 (0.19)
	1.5	635.0 (325.6)	0.20 (0.03)	1.974 (0.15)
	2	671.8 (349.8)	0.17 (0.03)	1.912 (0.16)

Clustered Ultrametric, 2X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	537.2 (275.7)	0.45 (0.06)	3.091 (0.14)
	1	552.9 (286.1)	0.32 (0.04)	2.782 (0.11)
	1.5	569.3 (297.6)	0.27 (0.03)	2.661 (0.10)
	2	583.2 (299.5)	0.23 (0.03)	2.605 (0.09)
16 Taxa	0.5	551.0 (283.0)	0.43 (0.06)	2.819 (0.20)
	1	578.5 (300.3)	0.29 (0.05)	2.407 (0.15)
	1.5	608.3 (317.1)	0.22 (0.04)	2.266 (0.14)
	2	637.4 (331.7)	0.19 (0.03)	2.220 (0.14)
32 Taxa	0.5	579.8 (295.9)	0.43 (0.06)	2.713 (0.20)
	1	632.8 (327.3)	0.28 (0.05)	2.321 (0.16)
	1.5	689.3 (362.1)	0.22 (0.04)	2.228 (0.16)
	2	748.2 (402.3)	0.18 (0.03)	2.215 (0.17)

Random Ultrametric, 2X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	541.9 (281.0)	0.40 (0.10)	2.961 (0.26)
	1	558.8 (287.1)	0.26 (0.08)	2.621 (0.23)
	1.5	576.9 (298.6)	0.21 (0.07)	2.500 (0.18)
	2	594.8 (315.5)	0.18 (0.06)	2.451 (0.16)
16 Taxa	0.5	553.9 (286.1)	0.38 (0.10)	2.702 (0.30)
	1	583.1 (299.7)	0.25 (0.08)	2.307 (0.25)
	1.5	614.9 (320.2)	0.20 (0.06)	2.185 (0.20)
	2	643.2 (337.3)	0.17 (0.05)	2.147 (0.18)
32 Taxa	0.5	578.0 (300.0)	0.36 (0.10)	2.519 (0.31)
	1	631.9 (332.6)	0.23 (0.08)	2.149 (0.25)
	1.5	686.2 (365.8)	0.18 (0.06)	2.071 (0.20)
	2	744.7 (410.7)	0.16 (0.05)	2.086 (0.21)

Table 5.5: The statistics on the non-ultrametric tree subset of the SimDom database with the linker factor of 2. Each subgroup has 1000 sequence sets. "Branch": tree branch factor; "Algn. len": alignment length; "Ident": the average pairwise protein identity; "Info" the average column-wise information score; and "SD": standard deviation.

	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	534.5 (274.6)	0.44 (0.07)	3.038 (0.18)
	1	550.0 (284.0)	0.28 (0.05)	2.623 (0.16)
	1.5	563.4 (289.9)	0.21 (0.04)	2.448 (0.13)
	2	577.0 (295.0)	0.17 (0.03)	2.375 (0.11)
16 Taxa	0.5	542.4 (278.6)	0.45 (0.07)	2.877 (0.21)
	1	566.9 (294.5)	0.29 (0.05)	2.389 (0.19)
	1.5	586.9 (302.7)	0.22 (0.04)	2.176 (0.15)
	2	608.7 (310.7)	0.18 (0.03)	2.089 (0.14)
32 Taxa	0.5	552.9 (283.1)	0.47 (0.07)	2.818 (0.22)
	1	585.1 (304.2)	0.31 (0.05)	2.300 (0.21)
	1.5	613.8 (316.8)	0.23 (0.04)	2.071 (0.17)
	2	642.9 (330.7)	0.19 (0.03)	1.969 (0.15)

Clustered Non-ultrametric, 2X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	526.2 (270.6)	0.61 (0.06)	3.539 (0.13)
	1	532.8 (275.1)	0.46 (0.06)	3.177 (0.15)
	1.5	538.4 (277.0)	0.38 (0.05)	2.958 (0.15)
	2	543.5 (278.5)	0.33 (0.05)	2.807 (0.14)
16 Taxa	0.5	529.8 (272.5)	0.61 (0.06)	3.380 (0.18)
	1	541.2 (281.6)	0.44 (0.06)	2.905 (0.21)
	1.5	549.1 (283.2)	0.35 (0.06)	2.616 (0.20)
	2	557.3 (285.5)	0.29 (0.05)	2.425 (0.18)
32 Taxa	0.5	539.4 (276.9)	0.62 (0.06)	2.886 (1.10)
	1	558.2 (289.7)	0.45 (0.06)	2.777 (0.22)
	1.5	572.8 (295.5)	0.35 (0.06)	2.467 (0.22)
	2	586.8 (299.5)	0.29 (0.05)	2.268 (0.19)

Random Non-ultrametric, 2X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	532.4 (273.6)	0.58 (0.11)	3.417 (0.25)
	1	541.0 (278.3)	0.42 (0.11)	3.026 (0.28)
	1.5	548.8 (283.2)	0.34 (0.10)	2.813 (0.27)
	2	558.6 (288.6)	0.29 (0.09)	2.689 (0.24)
16 Taxa	0.5	538.1 (276.3)	0.56 (0.10)	3.230 (0.28)
	1	552.8 (285.4)	0.40 (0.11)	2.762 (0.31)
	1.5	568.8 (296.7)	0.32 (0.10)	2.517 (0.29)
	2	584.0 (302.3)	0.27 (0.08)	2.378 (0.26)
32 Taxa	0.5	550.1 (283.1)	0.55 (0.10)	3.077 (0.30)
	1	576.3 (298.3)	0.38 (0.10)	2.579 (0.31)
	1.5	602.9 (314.1)	0.30 (0.09)	2.337 (0.28)
	2	632.8 (333.4)	0.25 (0.08)	2.219 (0.25)

Table 5.6: The statistics on the ultrametric tree subset of the SimDom database with the linker factor of 4. Each subgroup has 1000 sequence sets. "Branch": tree branch factor; "Algn. len": alignment length; "Ident": the average pairwise protein identity; "Info" the average column-wise information score; and "SD": standard deviation.

	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	552.8 (283.4)	0.32 (0.06)	2.711 (0.17)
	1	584.2 (299.5)	0.21 (0.04)	2.491 (0.12)
	1.5	620.4 (317.2)	0.17 (0.03)	2.464 (0.13)
	2	648.4 (329.5)	0.15 (0.02)	2.478 (0.15)
16 Taxa	0.5	569.9 (291.2)	0.33 (0.06)	2.480 (0.22)
	1	619.0 (318.5)	0.22 (0.04)	2.224 (0.15)
	1.5	670.7 (342.8)	0.17 (0.03)	2.195 (0.16)
	2	719.3 (368.6)	0.15 (0.02)	2.232 (0.19)
32 Taxa	0.5	589.4 (299.8)	0.34 (0.06)	2.389 (0.23)
	1	656.8 (338.6)	0.23 (0.04)	2.106 (0.17)
	1.5	724.1 (370.0)	0.19 (0.03)	2.064 (0.17)
	2	789.7 (407.1)	0.16 (0.02)	2.096 (0.20)

Clustered Ultrametric, 4X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	548.9 (281.5)	0.40 (0.06)	2.952 (0.15)
	1	577.0 (296.2)	0.29 (0.04)	2.733 (0.11)
	1.5	606.4 (314.3)	0.24 (0.03)	2.668 (0.10)
	2	630.2 (320.5)	0.21 (0.03)	2.651 (0.10)
16 Taxa	0.5	572.8 (293.9)	0.37 (0.06)	2.623 (0.20)
	1	622.3 (319.6)	0.25 (0.04)	2.381 (0.15)
	1.5	675.3 (349.8)	0.20 (0.03)	2.354 (0.16)
	2	724.4 (371.4)	0.18 (0.03)	2.383 (0.19)
32 Taxa	0.5	620.8 (316.9)	0.37 (0.07)	2.533 (0.20)
	1	718.4 (367.4)	0.25 (0.04)	2.382 (0.17)
	1.5	821.1 (427.8)	0.20 (0.03)	2.353 (0.15)
	2	924.9 (488.5)	0.17 (0.03)	2.534 (0.26)

Random Ultrametric, 4X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	554.4 (286.5)	0.34 (0.09)	2.797 (0.25)
	1	582.6 (297.7)	0.24 (0.07)	2.577 (0.18)
	1.5	614.4 (315.2)	0.19 (0.06)	2.535 (0.14)
	2	644.8 (336.8)	0.17 (0.05)	2.540 (0.13)
16 Taxa	0.5	574.3 (294.3)	0.33 (0.09)	2.510 (0.27)
	1	624.2 (318.4)	0.22 (0.07)	2.280 (0.20)
	1.5	675.7 (348.7)	0.18 (0.05)	2.261 (0.16)
	2	726.5 (379.3)	0.16 (0.04)	2.297 (0.16)
32 Taxa	0.5	614.8 (316.3)	0.31 (0.09)	2.346 (0.27)
	1	708.3 (369.2)	0.21 (0.06)	2.200 (0.19)
	1.5	802.0 (425.8)	0.17 (0.05)	2.244 (0.18)
	2	898.4 (492.3)	0.15 (0.04)	2.328 (0.23)

Table 5.7: The statistics on the non-ultrametric tree subset of the SimDom database with the linker factor of 4. Each subgroup has 1000 sequence sets. "Branch": tree branch factor; "Algn. len": alignment length; "Ident": the average pairwise protein identity; "Info" the average column-wise information score; and "SD": standard deviation.

	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	545.5 (279.5)	0.37 (0.07)	2.831 (0.20)
	1	571.4 (292.8)	0.25 (0.05)	2.557 (0.14)
	1.5	597.1 (304.8)	0.19 (0.03)	2.481 (0.12)
	2	622.7 (316.5)	0.17 (0.03)	2.471 (0.13)
16 Taxa	0.5	560.0 (286.6)	0.38 (0.07)	2.620 (0.23)
	1	599.2 (307.3)	0.26 (0.05)	2.297 (0.17)
	1.5	638.1 (325.8)	0.20 (0.04)	2.212 (0.15)
	2	680.1 (346.4)	0.17 (0.03)	2.216 (0.17)
32 Taxa	0.5	576.2 (294.3)	0.39 (0.07)	2.534 (0.25)
	1	629.9 (323.2)	0.27 (0.05)	2.184 (0.19)
	1.5	683.4 (349.0)	0.21 (0.04)	2.091 (0.16)
	2	739.9 (378.6)	0.18 (0.03)	2.088 (0.18)

Clustered Non-ultrametric, 4X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	530.9 (272.5)	0.53 (0.07)	3.326 (0.17)
	1	541.7 (278.3)	0.40 (0.06)	2.979 (0.18)
	1.5	552.0 (282.5)	0.33 (0.05)	2.791 (0.16)
	2	561.9 (287.8)	0.28 (0.05)	2.673 (0.14)
16 Taxa	0.5	537.0 (275.6)	0.52 (0.08)	3.090 (0.23)
	1	555.4 (287.0)	0.38 (0.07)	2.650 (0.23)
	1.5	571.0 (292.0)	0.31 (0.05)	2.426 (0.20)
	2	587.3 (300.7)	0.26 (0.05)	2.294 (0.17)
32 Taxa	0.5	552.4 (282.6)	0.53 (0.08)	2.974 (0.26)
	1	583.1 (299.7)	0.38 (0.07)	2.509 (0.24)
	1.5	610.7 (312.6)	0.30 (0.06)	2.288 (0.20)
	2	639.1 (326.1)	0.26 (0.05)	2.177 (0.17)

Random Non-ultrametric, 4X linker				
	Branch	Algn. len (SD)	Ident (SD)	Info (SD)
8 Taxa	0.5	538.2 (275.6)	0.50 (0.11)	3.195 (0.27)
	1	553.0 (284.1)	0.36 (0.10)	2.858 (0.26)
	1.5	567.3 (291.3)	0.29 (0.09)	2.707 (0.22)
	2	582.4 (298.2)	0.25 (0.08)	2.632 (0.19)
16 Taxa	0.5	547.7 (280.9)	0.48 (0.11)	2.958 (0.31)
	1	573.1 (294.1)	0.34 (0.10)	2.568 (0.28)
	1.5	599.1 (310.6)	0.28 (0.08)	2.408 (0.24)
	2	624.9 (322.7)	0.24 (0.07)	2.339 (0.20)
32 Taxa	0.5	567.9 (291.7)	0.46 (0.10)	2.787 (0.32)
	1	613.8 (315.3)	0.32 (0.09)	2.407 (0.27)
	1.5	658.3 (340.0)	0.26 (0.08)	2.276 (0.22)
	2	704.9 (367.3)	0.22 (0.07)	2.244 (0.19)

Table 5.8: The configuration of the SimDom database.

code	common characteristic	#seq. sets
FU	Full ultrametric guide tree	24000
FN	Full non-ultrametric guide tree	24000
CU	Clustered ultrametric guide tree	24000
CN	Clustered non-ultrametric guide tree	24000
RU	Random ultrametric guide tree	24000
RN	Random non-ultrametric guide tree	24000
T8	8 taxa	48000
T16	16 taxa	48000
T32	32 taxa	48000
D5	5 domains	3600
D4	4 domains	7200
D3	3 domains	18000
D2	2 domains	36000
D1	1 domains	79200
P5	Avg.Prot.Ident > 0.5	12368
P4	Avg.Prot.Ident $<= 0.5$ but > 0.4	17591
P35	Avg.Prot.Ident $<= 0.4$ but > 0.35	12851
P3	Avg.Prot.Ident $<= 0.35$ but > 0.3	16262
P2	Avg.Prot.Ident $<= 0.3$ but > 0.2	46381
P1	Avg.Prot.Ident $<= 0.2$	38547

Chapter 6

Evaluation of alignment programs

We begin this chapter with a synopsis on the need for high quality multiple sequence alignments. We continue with a brief review of recent studies of alignment program performance. We then discuss the objective of our study and how we conducted this relative performance study, followed by a discussion of the results and their implication on the machine-learning problem of selecting the alignment "closest to the optimal".

6.1 Motivation

As described in Chapter 1, multiple sequence alignments (MSAs) are used as the starting point for many of the bioinformatics studies. With the increasing volume of data comes the increasing dependence on automated alignment programs to generate MSAs quickly and accurately. Alignment quality reflects how accurately the alignment depicts the evolutionary relationship between a set of related sequences. As such, alignment quality is critical to the accuracy of the subsequent studies

With this strong need for accuracy in MSAs, many evaluations have been performed to compare different alignment programs. These evaluations have been performed using different sets of alignment programs and a variety of benchmark datasets and scoring metrics, with the exact combination dependent on the motivation of the

Table 6.1: A summary of the evaluation studies on the quality of MSAs produced by various programs. “year”: publication year; ”ref.”: reference; “metric”: the measure used to determine the alignment quality; and “benchmark”: the reference datasets used in the evaluation. The columns labeled MUSCLE, LINSI, PROB, CLTW2 and OMEGA provide the overall rank of the corresponding program received in the study. A 0 indicates the program was not included in the study.

year	ref.	metric	benchmark	MUSCLE	LINSI	PROB	CLTW2	OMEGA
2004	[16]	SPS*	BB3	1	2	0	3	0
2005	[104]	SPS	prefab,HOM	2	1	0	3	0
2009	[20]	SPS*	BB3,HOM,	2	2	1	1	0
2012	[21]	CS	BB3	3	1	1	5	3
2012	[79]	CS	BB3	2	1	0	3	0
2014	[105]	SPS	sim. data (isg)	2	1	0	0	3
2016	[106]	SPS	sim. data (SimProt)	2	1	0	7	0

* test for statistical significance with some significant results

study, i.e., the release of new alignment program, phylogeny reconstruction, protein structural prediction, etc.

Table 6.1 summarizes seven evaluation studies that included the five programs that we used in this study. It should be noticed that the metric used for accuracy is different from study to study and as such cannot be directly compared across studies. What all of these studies showed was that no single program outperformed any other program on all sequence sets. However, in all these studies, the accuracies for each program were averaged over the datasets and usually reported without statistical analysis. Therefore, it is often difficult to determine to what extent one method outperformed another. Nevertheless, in Table 6.1, the methods are ranked based on the average accuracies. Among these evaluations, only two reported on statistical significance. One indicated that across all programs there was statistical significance between some of the ranks but not all [16]. In this study, MUSCLE was found to be significantly ranked above all programs except T-coffee, while only a few rankings between the other programs showed significance. The second study to report on statistical significance showed significance on only three of the 15 pairwise ranks

[20]. Overall, from all of these studies, it is difficult to understand the difference in performance between the alignment programs to gauge the potential in gain for average accuracy if the alignment with the highest score was selected from each set.

In this chapter, we perform our own evaluation of five alignment programs most frequently used in various bioinformatics research. Our objective in this study is both to determine the relative performance difference between the five alignment programs, and to identify those aspects that can be used as attributes to model training data for a classifier to identify the best alignment. The novelty of our approach for this performance study is that we track the difference in relative performance for each sequence set as well as calculate the average accuracy for each program. Identifying the difference in performance among alignment programs when they are applied to different sequence set is critical to our study, which is to increase the average alignment quality by selecting the program that can generate the highest quality alignment for a given set of sequences. In the following section, we discuss the benchmark database we used for testing, the alignment programs we evaluated and most importantly, we define multiple variables for the purpose of tracking and evaluating the relative performance difference between the alignment programs.

6.2 Evaluation methods

This section describes the setup of the evaluation study on the performance of five alignment programs. We discuss the alignment programs, metrics and benchmark alignment datasets that we used, as well as our objectives and the procedure we followed.

6.2.1 Alignment programs compared

The five alignment programs we compared are described in detail in Section 2.5. They are all variations of the progressive alignment strategy and are reported to perform well in many comparative studies. The executable and/or source code files for these programs are readily available for free download and lend themselves to batch file usage for aligning large numbers of sequence sets without reliance on internet connection. The five programs along with the shortened names and version numbers are as follows:

- ClustalW (**CLTW2**) version 2.1 [18]
- Mafft (**LINSI**) version 7.157 [19]
- Muscle (**MUSCLE**) version 3.8.31 [16]
- Probalign (**PROB**) version 1.4 [20]
- Clustal-Omega (**OMEGA**) version 1.2.1.2 [21]

We used the default parameters for each alignment program.

6.2.2 Sequence datasets used

We used the sequence sets from the SimDom database (described in Chapter 5). It contains simulated protein sequence sets designed specifically for the evaluation of alignment program performance in the presence of domains. As described in Section 2.7, domains are the regions of a protein sequence that are under functional constraints and as such tend to be more conserved. The regions outside of domains (linkers) are not under such constraints and their sequences can be highly divergent. The majority

of the benchmark alignment databases currently available contain non-simulated protein sequences that have reference alignments based on the 3-D structure of only the domain areas. We refer to these databases as non-simulated benchmark databases. The advantages of using the SimDom database instead of these non-simulated benchmark databases for this evaluation are discussed in detail in Chapter 5.1. They are briefly reviewed here:

- As a simulated database, each sequence set in SimDom comes with the true reference alignment involving the full length of the sequences. In contrast, non-simulated protein benchmark alignments are educated approximations for only those areas of the sequences whose 3D-structures are known. The alignments derived from other parts of the sequences have even lower confidence.

- There are 144,000 sequence sets in SimDom, representing 1750 domains in 144 different evolutionary scenarios (conditions for sequence simulation are summarized in Table5.3). The most frequently cited non-simulated benchmark database, BAliBASE3[65], for example, has only 603 alignments that contain more than two sequences. As discussed in Section 3.2, BAliBASE3 has the largest number of reference alignments among non-simulated dataset.

- SimDom datasets are grouped based on specific characteristics such as sequence number, number of domains, guide tree topology and sequence divergence level (summarized in Table 5.8). With these different groups of sequence sets, it is possible to evaluate the performance of alignment programs based on specific characteristics.

6.2.3 Characterization of sequence sets and alignments

We analyze the results of the relative performance of alignment programs based on the following data points taken from each sequence set and reference alignment:

- Number of taxa (number of sequences)
- Number of domains in sequences set
- Alignment length
- Average sequence length with standard deviation
- Average pairwise protein identity. This is a metric used to establish the amount of sequence divergence within a specific sequence set (discussed in Section 3.1.1). Sequence divergence is one factor that has been shown to affect the level of disagreement in the alignments created by various programs, which consequently produces alignments of varying accuracies. Alignment programs are generally in agreement (produce similar alignments) when their average pairwise identity is above 60% [107]. At lower protein identities, alignment programs disagree to a much greater extent with this disagreement increasing as protein identity decreases. At the level of 30% and lower identity, which is considered to be the threshold of the "twilight zone", the largest disagreement occurs. Sequence sets that fall below 10% are considered at high risk of containing unrelated sequences and as such are not generally aligned [49].
- Information score. This is a metric of the average column-wise information, which is an indication of how conserved the columns in a specific alignment (discussed in Section 3.1.2.).

6.2.4 Performance analysis

As previously stated, our aim is to better understand those aspects of sequence sets and alignments that can affect the relative performance of alignment programs. We approach this evaluation with the following objectives:

1. Establish that there is a performance difference between the five alignment programs used in this study. By performance difference we mean that one program produces an alignment that is closer to the optimal than another as measured by an accuracy metric. We use the term outperform to describe the situation where one program generates alignments closer to the reference alignment than all the other programs. This one program is said to outperform the others.

2. Establish that the relative performance between the five alignment programs shifts depending on the sequence sets being used. By shifts we mean that the program that outperforms the others changes on different sequence sets.

3. Establish the extent of the performance difference. As the performance difference increases, the choice of the alignment program becomes more critical.

4. Establish how the amount of performance difference is affected by general sequence set characteristics such as the taxon number, number of domains and topology of the guide tree used in the simulation.

5. Establish the effect of sequence divergence level within a sequence set on the relative performance. We expect that sequence divergence is a strong candidate as an indicator for shifts in performance difference. We use protein identity to measure divergence

6. Establish the effect of column-wise conservation on the relative performance. We expect that column-wise conservation is a candidate as an indicator for shifts in performance difference. We use column-wise information score as a measure of conservation (discussed in Section 3.1.2).

To accomplish these objectives, each of the five programs was used to align the 144,000 protein sequence sets from the SimDom database. We calculated the CSS on each alignment against the reference alignment for the sequence set. As we discussed in Section 3.2.3, CSS is an accuracy metric with the desired property of rewarding the correct alignment while penalizing incorrect alignments including under and over alignments. As such, we used CSS to compare the overall "goodness" of each alignment and interpreted the value as the higher the value, the better the alignment. Using CSS as the base, the following summary variables are calculated and used in our performance analysis:

- CSS(id, p) - the CSS for the alignment created by program p on the sequence set identified by id.

- maxCSS - The largest value of CSS(id, p) obtained between all alignments generated by the five alignment programs from a single sequence set, id. The alignment associated with this score is considered the best alignment of the five and therefore the program that produced the best alignment is said to outperform the other programs. When referring to the maxCSS from a specific sequence set, the notation is augmented by the id of the sequence, maxCSS(id).

- $\text{CSS}_{\text{program}}$ - this is the average CSS(id, p) for a specific program p, across all sequence sets in the dataset. This is an example of the "average accuracy" metric that was often been used to compare alignment programs in the past evaluation

studies. For the alignment programs evaluated in this study, the average CSS's are given as: $\text{CSS}_{\text{MUSCLE}}$, $\text{CSS}_{\text{LINSI}}$, CSS_{PROB}, $\text{CSS}_{\text{CLTW2}}$ and $\text{CSS}_{\text{OMEGA}}$.

- BEST - This is the average of the maxCSS for all sequence sets in a dataset. This measure will illustrate the maximum average of alignment quality if the alignment with the maxCSS could be selected for each sequence set, as opposed to using a single program on all sequence sets.

- $\Delta 15$ - The difference between the maxCSS and the lowest $\text{CSS}(id, p)$ for all programs on a single sequence set. This is in essence the range of alignment quality for a specific sequence set.

- $\Delta 12$ - The difference between maxCSS and the second highest $\text{CSS}(id, p)$ for all programs on a single sequence set. Small values of $\Delta 12$ would indicate that the alignments were comparable in accuracy and therefore, there would be little, if any, concern about which alignment of the two was used. Large values of $\Delta 12$ indicates large differences in accuracy between alignments, which would make the choice of alignment more critical. This value will help establish the extent of the relative performance difference per alignment program.

- $\Delta 23$ - The difference between the second highest $\text{CSS}(id, p)$ and the third highest $\text{CSS}(id, p)$ for all programs on a single sequence set. This quantity is used in Chapter 8.

- $\Delta 34$ - The difference between the third highest $\text{CSS}(id, p)$ and the fourth highest $\text{CSS}(id, p)$ for all programs on a single sequence set. This quantity is used in Chapter 8.

- $\Delta 45$ - The difference between the fourth highest CSS(id, p) and the lowest CSS(id, p) for all programs on a single sequence set. This quantity is used in Chapter 8.

- ΔBP - The difference between BEST and $\text{CSS}_{\text{program}}$. For a specific program, p. $\Delta\text{BP}(p)$ shows how far a given program is behind in average CSS.

6.3 Performance evaluation

The performance of each alignment program was evaluated across 18 groups of the SimDom datasets. These groups contain sequence sets that vary by the number of taxa in each sequence set and the topology of the guide tree used in their simulation (described in detail in Section 5.4). The names of each group are shown in the first column of Table 6.2. The first two letters refer to the guide tree topology (described in Section 5.4) while the number (8, 16, or 32) refers to the number of taxa (sequences) in the sequence sets that make up the dataset.

In Table 6.2, the $\text{CSS}_{\text{program}}$ of each program within each dataset is shown along with BEST. There is clearly a difference in the performance between programs. The program with the highest $\text{CSS}_{\text{program}}$ (shown in blue in Table 6.2) varies beween the groups and the range of $\text{CSS}_{\text{program}}$ for each group varies from 0.024 to 0.047 (both shown in green). BEST for each dataset represents the maximum possible average accuracy if the user was able to select the alignment based on the maxCSS.

The high standard deviation of $\text{CSS}_{\text{program}}$ for each program reflects the large heterogeneity in the sequence sets. This results in CSS(id, p) values that range from 0.09 to 1.00. At this time, the distribution of CSS(id, p) is not known and as such we cannot analyze this difference statistically. In Figure 6.1, we compared CSS(id) between each pair of the five alignment programs. These plots show that no single

Table 6.2: Performance comparison among the five alignment programs. $CSS_{program}$ values are compared among alignment programs for each group. Each group includes 8000 sequences sets and the standard deviations are shown in parentheses. The highest $CSS_{program}$ for each dataset is shown with blue font and the lowest values are shown in red. The range is the difference between the highest and the lowest $CSS_{program}$ for the dataset. The minimum and maximum range values are shown in green font.

dataset	CSS_{MUSCLE}	CSS_{LINSI}	CSS_{PROB}	CSS_{CLTW2}	CSS_{OMEGA}	range	BEST
FU8	0.736	0.731	0.717	0.757	0.718	0.040	0.770
	(0.131)	(0.133)	(0.130)	(0.116)	(0.129)		(0.178)
FU16	0.777	0.786	0.762	0.779	0.762	0.024	0.807
	(0.120)	(0.119)	(0.119)	(0.108)	(0.120)		(0.171)
FU32	0.814	0.829	0.782	0.801	0.799	0.047	0.842
	(0.106)	(0.101)	(0.109)	(0.102)	(0.110)		(0.156)
FN8	0.791	0.785	0.768	0.812	0.767	0.045	0.823
	(0.117)	(0.120)	(0.120)	(0.102)	(0.120)		(0.168)
FN16	0.828	0.834	0.804	0.831	0.806	0.030	0.853
	(0.103)	(0.103)	(0.109)	(0.094)	(0.109)		(0.158)
FN32	0.864	0.876	0.831	0.843	0.843	0.045	0.883
	(0.088)	(0.084)	(0.097)	(0.085)	(0.095)		(0.142)
CU8	0.759	0.757	0.739	0.783	0.741	0.044	0.793
	(0.104)	(0.107)	(0.106)	(0.096)	(0.104)		(0.136)
CU16	0.787	0.789	0.766	0.804	0.763	0.041	0.815
	(0.104)	(0.105)	(0.104)	(0.092)	(0.105)		(0.146)
CU32	0.802	0.808	0.780	0.809	0.776	0.033	0.825
	(0.102)	(0.101)	(0.104)	(0.089)	(0.105)		(0.148)
CN8	0.899	0.890	0.872	0.911	0.876	0.038	0.920
	(0.066)	(0.068)	(0.070)	(0.061)	(0.070)		(0.089)
CN16	0.938	0.939	0.911	0.929	0.916	0.028	0.948
	(0.051)	(0.052)	(0.063)	(0.053)	(0.062)		(0.086)
CN32	0.955	0.960	0.928	0.932	0.941	0.032	0.962
	(0.042)	(0.039)	(0.058)	(0.050)	(0.051)		(0.077)
RU8	0.752	0.744	0.722	0.768	0.724	0.046	0.783
	(0.124)	(0.127)	(0.126)	(0.119)	(0.127)		(0.170)
RU16	0.764	0.769	0.737	0.764	0.732	0.037	0.789
	(0.122)	(0.121)	(0.122)	(0.117)	(0.126)		(0.171)
RU32	0.765	0.781	0.740	0.753	0.734	0.047	0.792
	(0.123)	(0.119)	(0.120)	(0.118)	(0.126)		(0.174)
RN8	0.896	0.891	0.870	0.906	0.875	0.035	0.914
	(0.081)	(0.084)	(0.091)	(0.074)	(0.089)		(0.114)
RN16	0.907	0.908	0.878	0.905	0.882	0.030	0.920
	(0.071)	(0.072)	(0.083)	(0.069)	(0.083)		(0.112)
RN32	0.914	0.918	0.884	0.896	0.886	0.033	0.923
	(0.066)	(0.065)	(0.077)	(0.068)	(0.078)		(0.111)

alignment program achieves the highest CSS on all or even most of alignments. The plots also show the variations in the magnitude (distance of each point from the diagonal) of how one program outperformed the other. While these are not quantitative in nature, it does lend support to the concept of improving the overall quality of the selected alignments by being able to select the most accurate alignment from a group of alignments on a per sequence set basis.

However, to show the variation between CSS(id, p) values for a specific alignment, we have performed a pairwise plotting (CSS(id, p_i) vs.CSS(id, p_j)) among all alignment programs, resulting in ten plots, shown in Figure 6.1. Each plot represents the CSS(id, p) from the alignments of the 144,000 sequence sets in SimDom. The two programs, p_i and p_j, compared in each plot is identified from the row label and the column label of the array of plots. CSS(id, p) of the same value for the two programs on the same sequence set, will fall on the black diagonal line. These plots show that no single alignment program achieves the highest CSS on all or even most of alignments. The plots also show the variations in the magnitude (distance of point form the diagonal) of how one program out performed the other. While these are not quantitative in nature, it does lend support to the concept of improving the over all quality of the selected alignments by being able to select the most accurate alignment from a group of alignments on a per sequence set basis.

To determine the improvement that is possible between being able to select the alignment with maxCSS and the performance of a single program p, we calculated ΔBP(p). The results are shown in Table 6.3. For each row in this table, the program with the mininum ΔBP(p), resulting from its CSS being closest to BEST, is considered to be the top-performing program for the group (shown in blue in Table 6.3). For example, the mininum ΔBP for dataset CN32 was achieved by LINSI (0.003) while the minimum BP in dataset RU8 was achieved by PROB (0.014). LINSI

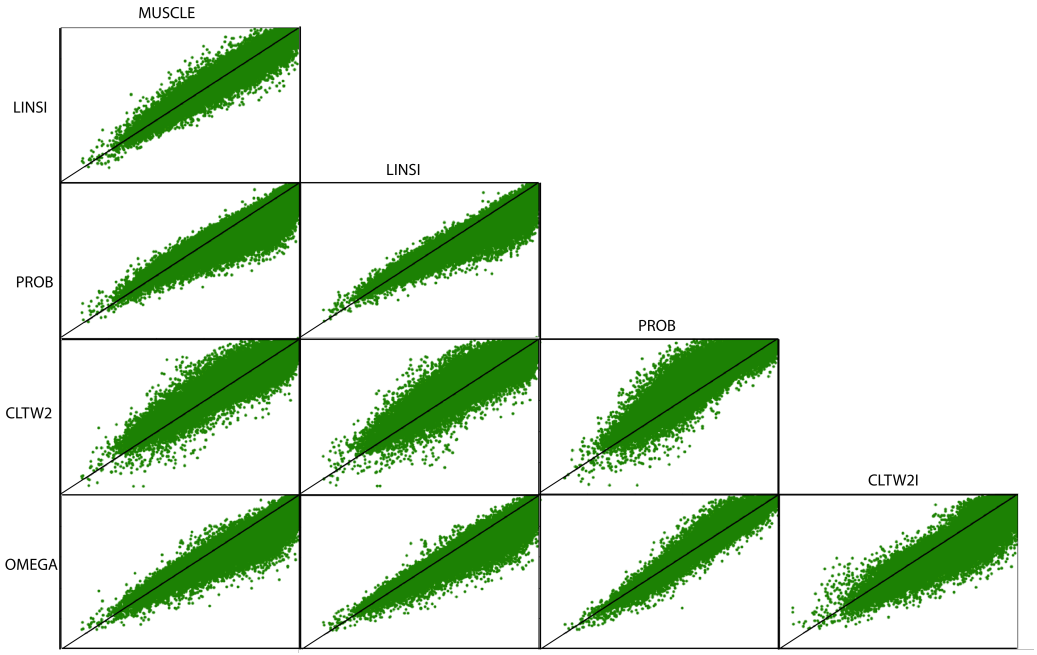


Figure 6.1: Pairwise CSS plots. Each plot shows the CSS(id, p_1) from one program against the CSS(id, p_2) from another program for the entire 144,000 sequence sets in SimDom. The diagonal line represents where the CSS values of the alignments generated by the two programs are equal, the CSS(id, p_1)=CSS(id, p_2). Both x and y axes of each plot range from 0.0 to 1.0.

had the most occurrences of the minimum ΔBP for 10 datasets, while CLTW2 had the second most with 8 occurrences. PROB and OMEGA never had the mininum BP and MUSCLE only twice. These results are unexpected. In previous studies, MUSCLE and OMEGA have always been able to perform better than CLTW2 [26], [21]. PROB has been also shown to outperform LINSI [20]. As we describe later in Section 8.6, both PROB and OMEGA performed better on the non-simulated benchmarks datasets than with SimDom. For example, the minimum ΔBP for dataset CN32 was achieved by LINSI (0.003) while the minimum ΔBP in dataset RU8 was achieved by PROB (0.014). LINSI had the most occurrences of the minimum ΔBP for 10 datasets, while CLTW2 had the second most with 8 occurrences. PROB and OMEGA never had the minimum ΔBP and muscle only twice. These results are

Table 6.3: $\Delta BP(p)$ values for each program for different datasets. $\Delta BP(p)$ values are shown for each dataset with blue and red fonts representing the smallest and largest values, respectively, which correspond to the best and worst average alignment quality, respectively, for each dataset. The "average" shown in the bottom row is the average $\Delta BP(p)$ across all datasets for each alignment program.

dataset	MUSCLE	LINSI	PROB	CLTW2	OMEGA
FU8	0.034	0.040	0.053	0.013	0.052
FU16	0.030	0.021	0.045	0.028	0.045
FU32	0.027	0.013	0.059	0.040	0.043
FN8	0.032	0.037	0.055	0.011	0.056
FN16	0.025	0.019	0.049	0.022	0.047
FN32	0.019	0.007	0.052	0.040	0.040
CU8	0.034	0.036	0.054	0.010	0.052
CU16	0.028	0.026	0.049	0.011	0.052
CU32	0.023	0.018	0.045	0.016	0.049
CN8	0.021	0.029	0.047	0.009	0.043
CN16	0.009	0.009	0.037	0.018	0.032
CN32	0.007	0.003	0.034	0.031	0.022
RU8	0.031	0.039	0.061	0.014	0.059
RU16	0.026	0.021	0.052	0.025	0.057
RU32	0.027	0.011	0.052	0.040	0.058
RN8	0.018	0.024	0.044	0.009	0.039
RN16	0.012	0.012	0.042	0.015	0.038
RN32	0.009	0.006	0.039	0.027	0.037
average	0.018	0.018	0.045	0.020	0.043

unexpected. In previous studies, MUSCLE and Omega have always been able to perform better than CLTW2 [26], [21]. PROB has also been shown to outperform LINSI [20]. As we describe later in Section 8.6, both PROB and OMEGA performed better on the non-simulated benchmarks datasets than with SimDom

We also found a distinct reversal in the relative performance of LINSI and CLTW2 between groups. For example, in dataset FU8, CLTW2 had a ΔBP of 0.013, while LINSI had a much larger ΔBP of 0.040 making CLTW2 the better performer. But in dataset FU32, LINSI had a minimum ΔBP of 0.013 while CLTW2 had a ΔBP of 0.04, indicating that LINSI was the top performer. This reversal of relative performance

also occurred between datasets FN8 and FN32, and between RU8 and RU32. These results show how the relative performance between alignment programs shifts depending on factors concerning the sequence sets, in this case, the guide-tree topology used for simulation and the number of sequences to be aligned. This result establishes that the relative performance does shift.

From this preliminary analysis, two trends are apparent:

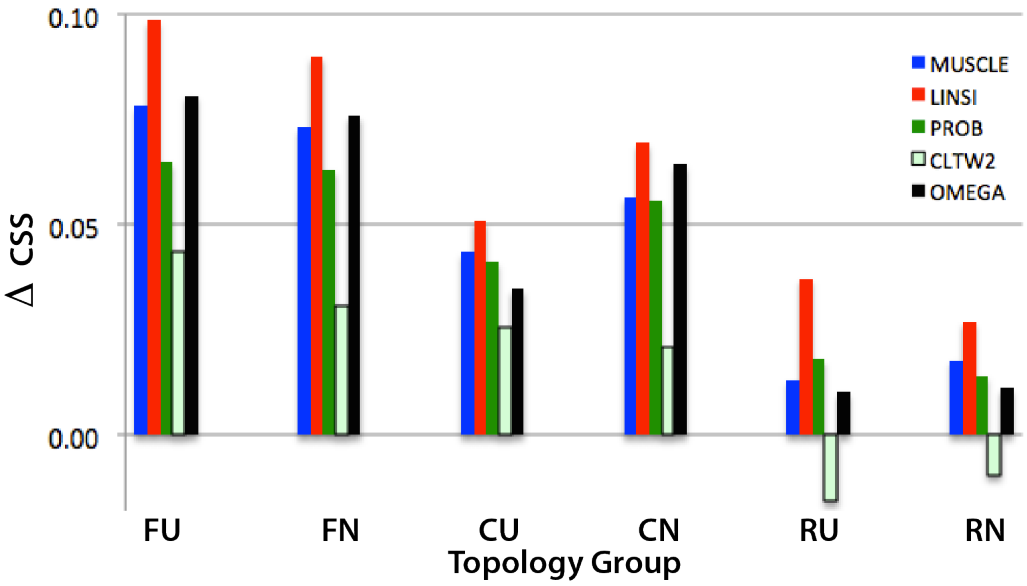


Figure 6.2: Change in $CSS_{program}$ with the increase number of taxa. ΔCSS is obtained for each individual alignment program by subtracting $CSS_{program}$ for the 8-taxon datasets from $CSS_{program}$ for the 32-taxon datasets.

- Within each topology group (FU, FN, CU CN, RU, or RN), the $CSS_{program}$ values for each program tend to increase with the number of taxa, especially from 8 taxa to 32 taxa. This trend is shown in Figure 6.2. The only two exceptions are for CLTW2 in the random tree datasets (both ultrametric RU and non-ultrametric RN datasets). This trend indicates that by increasing the number of sequences within the same divergence range, regardless of the pairwise distance

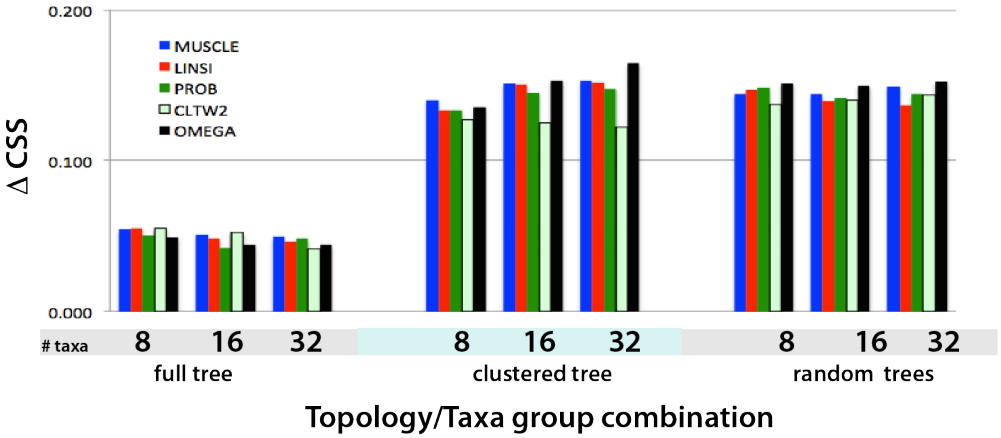


Figure 6.3: Change in $CSS_{program}$ between the ultrametric and non-ultrametric tree groups. ΔCSS is obtained for each individual alignment program by subtracting the $CSS_{program}$ obtained from the datasets generated using non-ultrametric trees from the $CSS_{program}$ obtained from the corresponding ultrametric dataset.

among the sequences, the accuracy of alignments produced by four out of five of the programs, can improve. The decrease in accuracy for CLTW2 with the increase in sequence number has been the noted by those that implemented the method and has been the motivation for various modification to the program [18] [81] [108].

- The $CSS_{program}$ for each program increases from ultrametric topology (FU, CU, or RU) to non-ultrametric topology (FN, CN, or RN). Figure 6.3 shows the difference between the $CSS_{program}$ for each program from the ultrametric datasets and from that of their non-ultrametric counterparts. For instance, the first group of five bars represent the difference (ΔCSS) between the $CSS_{program}$ values for each program obtained in the FU8 dataset and the FN8 dataset (the $CSS_{program}$ for each program obtained from FU8 was subtracted from the $CSS_{program}$ for each program obtained from FN32). This increase is an expected result. As described in Section 5.4, as simulation based on the ultrametric guide trees generates a

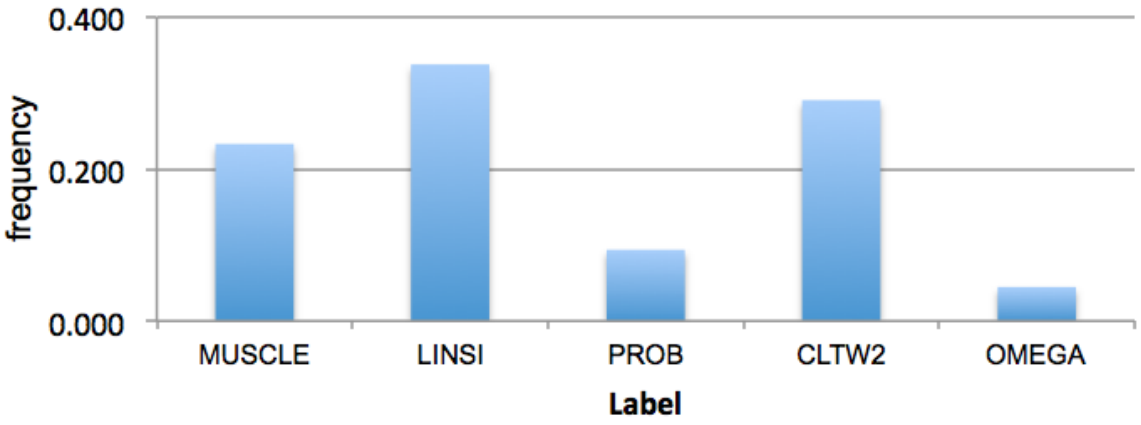


Figure 6.4: Label frequencies. The Label frequencies across the entire 144,000 SimDom datasets. The label is the name of the alignment program with the maxCSS for a specific sequence set. The number of times each alignment program was call as the label is in Table 6.4. the label frequencies were obtained across the entire 144,000 SimDom sequences sets,

group of taxa that have a higher average pairwise distance, representing more divergent sequences and, hence more difficult to correctly align.

While the average accuracy has traditionally been used to judge alignment performance [16] [24] [20] [21], in this study, we concentrate on the relative accuracy of each program for each sequence set. This is to help establish the extent of the performance difference per alignment program. For this purpose we introduce the term "**label**" to be the name of the alignment program that generates the alignment with maxCSS for a specific sequence set and hence, each sequence set has a label. Figure 6.4 shows the label frequencies across the entire SimDom database. Consistent to what we discussed earlier, LINSI and CLTW2 have the two highest label frequencies, followed by MUSCLE, with PROB and OMEGA showing the two lowest label frequencies.

To determine the extent of the differences in the relative performance of alignment programs, we calculated $\Delta 12$ and $\Delta 15$ (see Section 6.1.4 for definitions of these measures). Values for $\Delta 12$ and $\Delta 15$ that are close to 0 indicate that the top two alignments, when considering $\Delta 12$, or all alignments when considering $\Delta 15$, have

Table 6.4: Distribution of label across SimDom. The average (av.), standard deviation (σ), and maximum (max) of Δ12 and Δ15 are also listed.

program	label counts	label frquency	$CSS_{program}$	av.Δ12	$\sigma\Delta$12	maxΔ12	av.Δ15	$\sigma\Delta$15	maxΔ15
MUSCLE	33562	0.233	0.912	0.012	0.018	0.213	0.065	0.060	0.459
LINSI	48642	0.338	0.903	0.012	0.016	0.187	0.057	0.052	0.431
PROB	13496	0.094	0.802	0.008	0.011	0.137	0.032	0.032	0.356
CLTW2	41860	0.291	0.770	0.032	0.033	0.400	0.082	0.056	0.446
OMEGA	6440	0.045	0.821	0.005	0.007	0.070	0.030	0.028	0.353

comparable accuracy. Table 6.4 shows how the values of the average Δ12 and Δ15 changes with label. We note that both the average and standard deviation for Δ12 for CLTW2 are almost double those of the other labels. Similarily, the average and standard deviation for Δ15 are higher for CLTW2 than the other labels. It suggests that when CLTW2 is the label, the alignment performance levels vary widely and CLTW2 performs much better than others. This result is unexpected considering that published studies show CLTW2 generally behind LINSI, MUSCLE and PROB in performance [26][17].

Figure 6.6 shows the frequency distribution of Δ12 for when each program was the label across the entire SimDom datasets. The Δ12 interval from 0.0 to 0.005 contains the sequence sets where maxCSS and the second-place CSS are very close indicating that the alignments are comparable in accuracy. For the sequence sets that fall in this interval, the choice between maxCSS and the second place CSS values is not critical.

The remaining intervals were selected by roughly doubling the length of each successive interval. This generated a distribution of roughly equal proportion in the first three, and taping off in the last three. The height of the bars corresponds to the portion of the 144,000 sequence sets that fall within each labeled bin. The bin label on the x-axis is the upper limit of the Δ12 value that fall within that bin. For example, the far left bin represents those Δ12 values that are between 0.000 and 0.005.

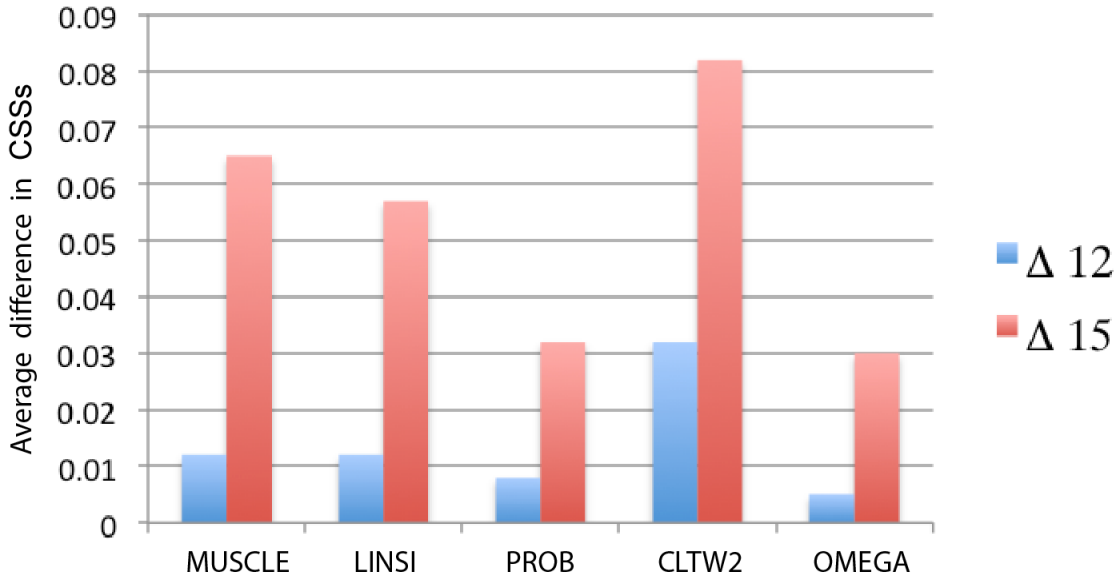


Figure 6.5: Δ12 and Δ15. The average Δ12 and Δ15 when each alignment program was the top performer (the label). These averages were calculated from the entire 144,000 SimDom datasets.

For a large proportion of the alignments, roughly 41%, Δ12 is less than 0.005, which indicates that there is little difference between the label and the program whose alignment achieved the second highest CSS. Each program has a notable proportion in this interval, although LINSI and MUSCLE clearly dominate. However, there is a performance shift as Δ12 increases. When Δ12 is low LINSI has the highest label frequencies. As Δ12 increases,the label frequency for all programs drop with the exception of the label frequency of CLTW2, which increases in the 0.01 - 0.02 interval and becomes the highest when Δ12 is higher than 0.02. This increase in CLW2 as the label with higher Δ12, contributes to the high $\sigma\Delta 12$ for CLTW2 as shown in Table 6.4 .

To produce a finer grained picture of the extent of the performance difference based

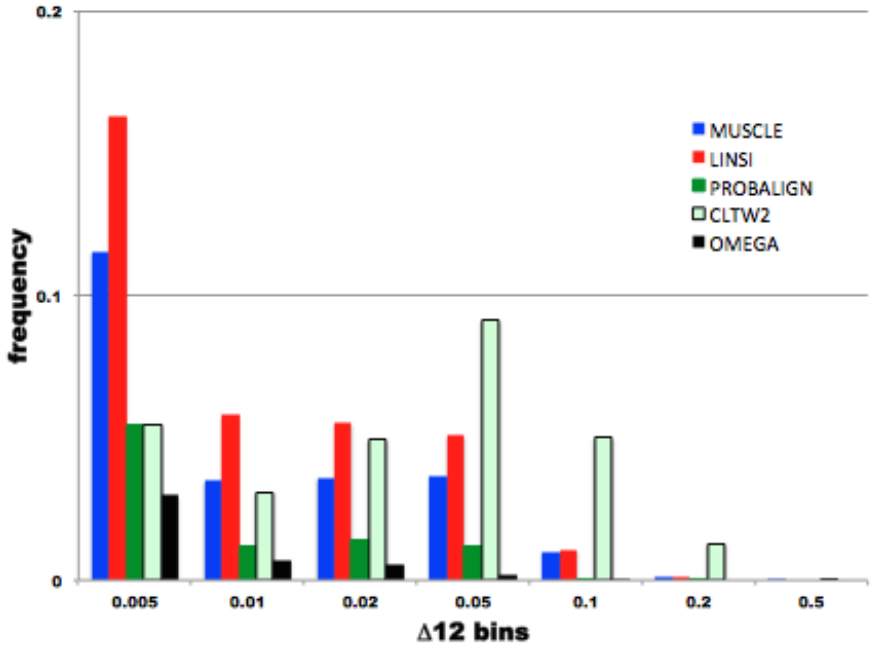


Figure 6.6: Distribution of $\Delta 12$ per alignment program. The frequency is based on the entire 144,000 SimDom sequence sets The numeric value shown for each bin is the upper limit of the bin. The frequency is based on the entire 144,000 SimDom sequence sets. The remaining intervals were selected by roughly doubling the length of each successive interval. This generated a distribution of roughly equal proportion in the first three, and taping off in the last three. The height of the bars corresponds to the portion of the 144,000 sequence sets that fall within each labeled bin. The bin label on the x-axis is the upper limit of the $\Delta 12$ value that fall within that bin. For example, the far left bin represents those $\Delta 12$ values that are between 0.000 and 0.005.

on the characteristics of the dataset of sequences, we examined the label frequency against the value of $\Delta 12$, for each of the 144 subgroups in SimDom (Figures 6.7, 6.8, 6.9). This is to identify any pattern that is associated with large $\Delta 12$ values. A large difference in the accuracy level between the top performing and other alignment programs corresponds to a large $\Delta 12$ value and would signal a sequence set where the selection of alignment program is critical. In Figures 6.7, 6.8, 6.9, the further to the right the maker falls, the larger the $\Delta 12$ value is, indicating sequence sets that cause a large amount of disagreement among alignment programs. The higher on the

plot the marker falls, the more frequently the corresponding program had the best accuracy and consequently, outperformed the others.

Several trends are apparent from these plots:

- Most of the average $\Delta 12$ values for the labels for the 0.5 branch factor datasets (indicated by the circle) in the 2X factor dataset were much less than 0.01, while for the 4X factor datasets, some programs showed the average $\Delta 12$ higher than 0.01. This is an expected result as sequence sets of the 0.5 branch factor datasets have a lower range in divergence due to the low branch factor and as such have lower pairwise distances. For sequence sets of low pairwise distance, alignment programs have higher levels of agreement and as such both $\Delta 12$ values and $\Delta 15$ values can be relatively low, making the choice of alignment programs for these sequence sets less critical.

- For the majority of the 8-taxa datasets, when branch factors were larger than 1.0, CLTW2 showed high label frequency with the average $\Delta 12$ over 0.01. As the taxon number increases, the label frequency for CLTW2 decreased.

- With the exception of the clustered-ultrametric (CU) dataset, LINSI was generally the most frequent label for the 32-taxa datasets, with the average $\Delta 12$ at 0.01 or more. As the number of taxa decreased, the frequency of LINSI to be the label decreased, especially in the 4X linker datasets.

- When MUSCLE was the label, it was associated with an average $\Delta 12$ over 0.1 on the majority of the plots

- There was no dataset where Omega had $\Delta 12$ over 0.01 or a label frequency over 0.10.

- Probalign showed a stronger accuracy in the dataset with 4x linker factor, FU topology, 16 taxa, than any other datasets. It is not immediately obvious why this is the case.

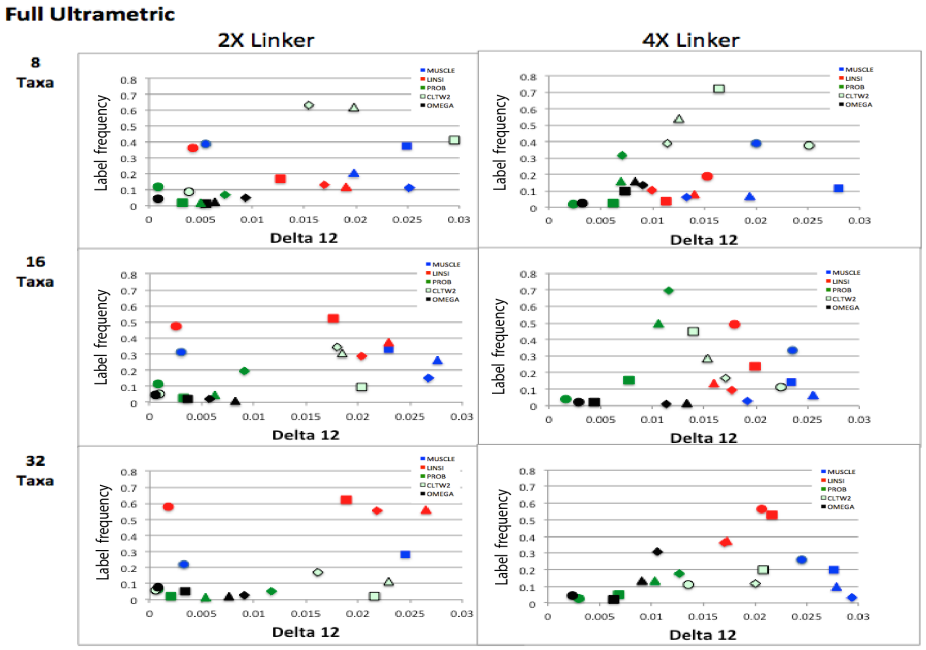
There are broad trends in the accuracy of one alignment program over the other, depending on such characteristics as number of taxa in the sequence set, the guide tree topology used the sequence simulation, and the branch and linker factors. Because these are broad but weak trends, we consider these characteristics to be weak indicators of the shift in relative performance.

In the following sections we will look at various metrics for alignments. We look for trends in the values of specific metrics that will indicate which alignment program produces the most optimal alignment for each sequences set. We will compare the label frequencies and $\Delta 12$ with the values of the following metrics:

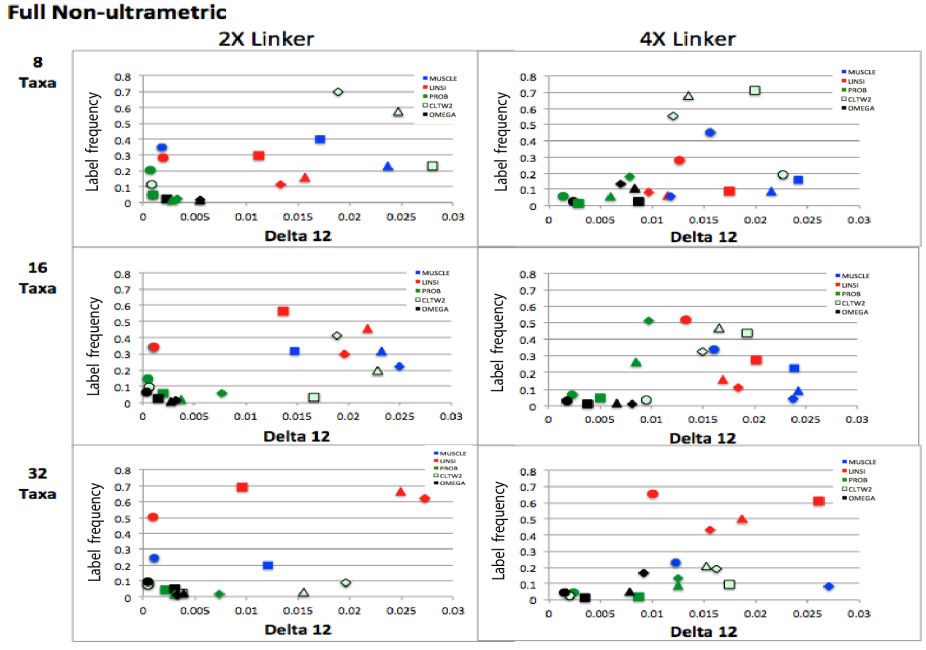
- Average pairwise protein identity
- Information score
- Number of domains

6.4 Protein identity and alignment performance

In the previous section we noted that roughly 41% of the labels have a $\Delta 12$ of less than 0.005. With these sequence sets, the choice of an alignment program is less critical. However, as $\Delta 12$ increases, the importance of selecting the best alignment increases. Therefore, our next aim is to be able to identify the set of sequences that have alignments with high $\Delta 12$, as well as be able to identify from the set of five programs which would create the most accurate alignment (with the highest CSS).

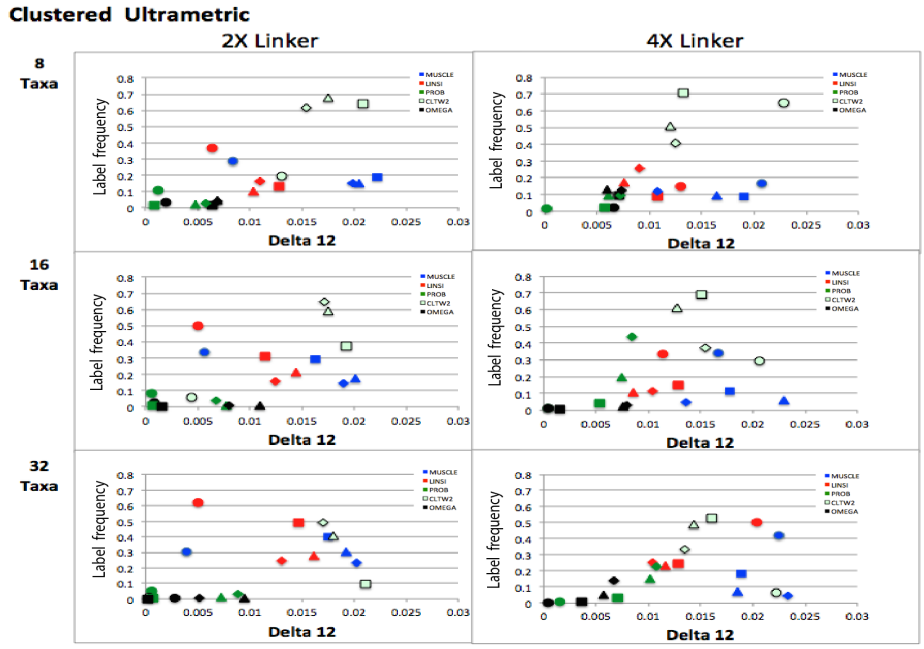


(a) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)

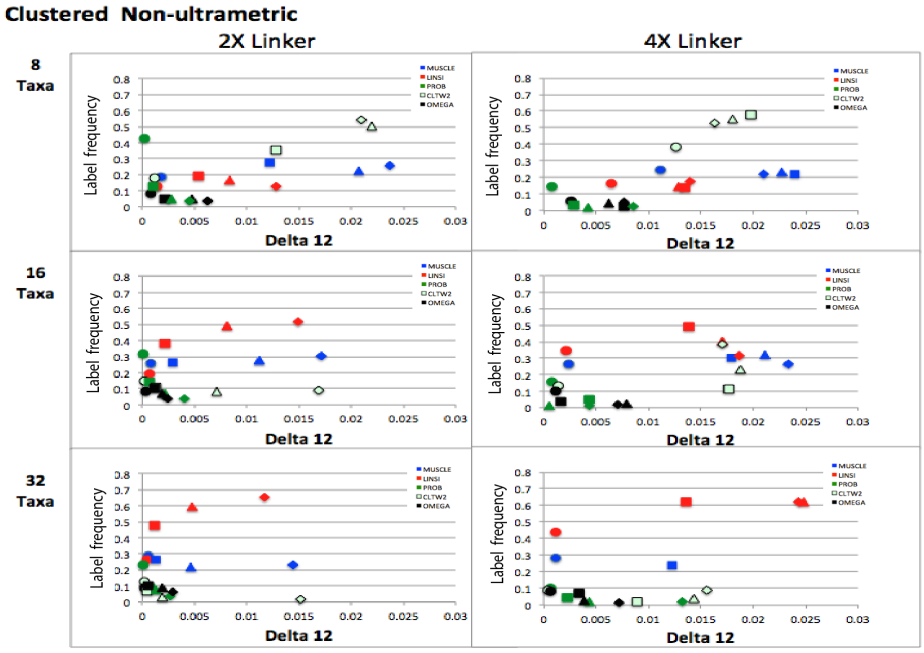


(b) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)

Figure 6.7: Relationship between Δ12 and the label frequency for the "Full" guide-tree datasets. Each dataset (a: FU8, FU16, and FU32; b: FN8, FN16, and FN32) is divided into two groups based on linker factors (2X and 4X). Plots in each panel are shown using different colors for the five alignment programs and different symbols for four branch factors (0.5, 1.0, 1.5, and 2.0).

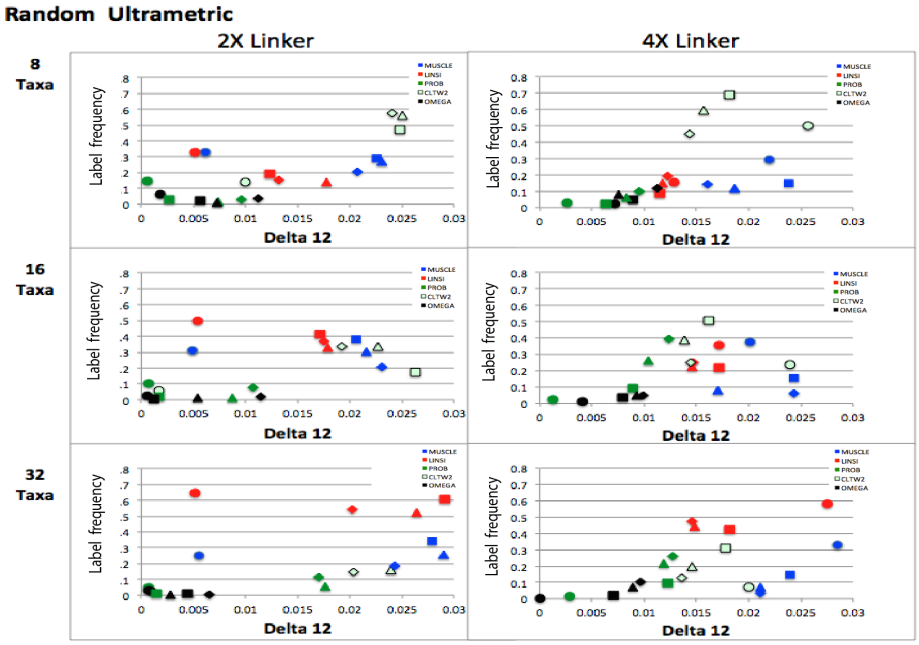


(a) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)

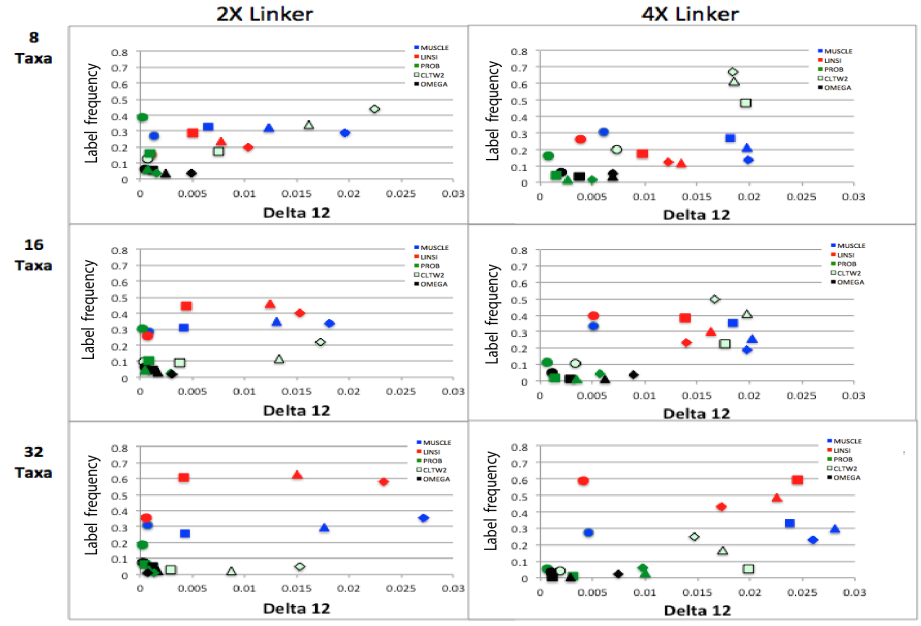


(b) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)

Figure 6.8: Relationship between Δ12 and the label frequency for the "Clustered" guide-tree datasets. Each dataset (a: CU8, CU16, and CU32; b: CN8, CN16, and CN32) is divided into two groups based on linker factors (2X and 4X). Plots in each panel are shown using different colors for the five alignment programs and different symbols for four branch factors (0.5, 1.0, 1.5, and 2.0).



(a) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)



(b) Symbol for branch factors: (◯ - 0.5) (□ - 1.0) (△ - 1.5) (◊ - 2.0)

Figure 6.9: Relationship between Δ12 and the label frequency for the "Random" guide-tree datasets. Each dataset (a: RU8, RU16, and RU32; b: RN8, RN16, and RN32) is divided into two groups based on linker factors (2X and 4X). Plots in each panel are shown using different colors for the five alignment programs and different symbols for four branch factors (0.5, 1.0, 1.5, and 2.0).

In this section we examine the average pairwise protein identity (described in Section 3.1.1) for its potential to be used as an indicator of alignments associated with high $\Delta 12$ values and more accurate alignments.

To perform this analysis, we divided the reference alignment sets into seven bins based on protein identities values, so that there is roughly an equal proportion of alignments in each bin. We then calculated the average $\Delta 12$ for each bin for each program when it was the . In Figure 6.10, the label frequency of each alignment program is plotted against the average $\Delta 12$, with a separate plot for each protein identity bin. For all programs, protein identity is negatively correlated to the average $\Delta 12$ (shown in the summary plot at the bottom right of Figure 6.10). This correlation shows that with decreasing protein identity, the importance of the selection of the most accurate alignment program also increases.

In previous studies [109][105][107], the decrease in average protein identity has been associated with an increase in disagreement between alignment programs indicating higher alignment difficulty. The range of protein identity between 10% and 30% has also been described as the twilight zone where the individual alignment programs are known to have a high level of disagreement [107]. In Figure 6.10, as expected, the average $\Delta 12$ for sequence sets whose average protein identity are within the twilight zone range (between 10% and 30%), are larger than for those with higher identities. This shows that protein identity is a good indicator both for where there is a large $\Delta 12$ and for which of the five programs would be the better one to be used. It is also noteworthy that for highly divergent protein sequences (<25% identity), CLWT2 unexpectedly performed better than other programs with noticeably larger $\Delta 12$ LINSI was the best performer when protein sequences were more conserved (>25% identity). It should also be noted that even when LINSI performed the best more frequently, when CLWT2 generated good alignments,the $\Delta 12$ values were notably higher. Only

when protein sequences were highly conserved as indicated by average protein identities greater than 60%, did PROB produce the most accurate alignments. However, $\Delta 12$ for these alignments were so small (0.0075), at this level of sequence identity that program selection is less critical.

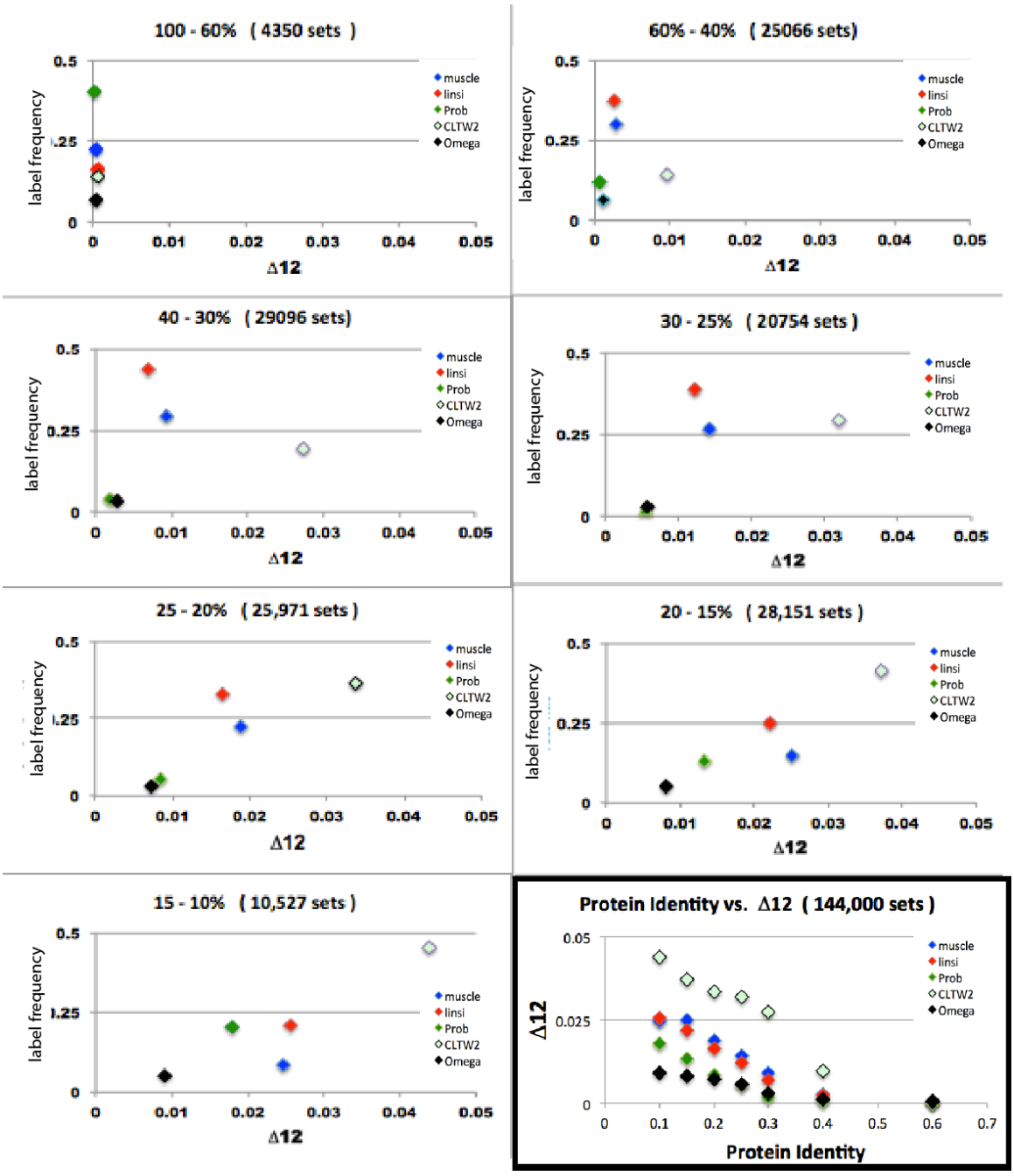


Figure 6.10: Average protein identity distribution. Relationship between the label frequency and Δ12 for different pairwise protein identity. The range of % pairwise protein identity for each bin (and the number of datasets included) is shown at the top of each panel. The lower right plot shows the average Δ12 for each protein identity interval.

6.5 Distribution of the label by information scores

In this section, we examine the relationship between the label frequency of each alignment program and the information score of the reference MSA. This is to determine if the information score can help determine which alignment program will produce the most accurate alignment. As discussed in Section 3.1.2, the information score for an alignment can be used as an indication of the level of conservation in the alignment. The higher the information score, the more conserved the alignment and consequently the more closely the individual alignment program will agree.

We divided the reference alignment sets into seven bins based on the information scores and calculated the average label frequency and $\Delta 12$ for each bin. The results of our calculation are show in Figure 6.11. We again observe negative correlations between the average information scores and $\Delta 12$ (the bottom right panel of Figure 6.11). This is expected because the average information score should be higher when more conserved protein sequences are aligned, and as shown in Figure 6.10, $\Delta 12$ values were negatively correlated with the pairwise sequence identity. However, the correlation was not as strong for information scores as for average protein sequence identity.

The average label frequency for each program remained largely constant regardless of the information scores. LINSI was most frequently the top performing method. However, when the average information scores were between 2.3 and 2.5, CLTW2 performed roughly even with or slightly better than LINSI. Similar to what we observed with pairwise sequence identity, when CLTW2 was the label, the $\Delta 12$ associated with it was substantial, indicating a sequence set where the choice of alignment is critical. However, the observation that all other programs generally maintained the same relative order regardless of the range of the average information score indicates that

this metric is a weak indicator of the label (most optimal alignment).

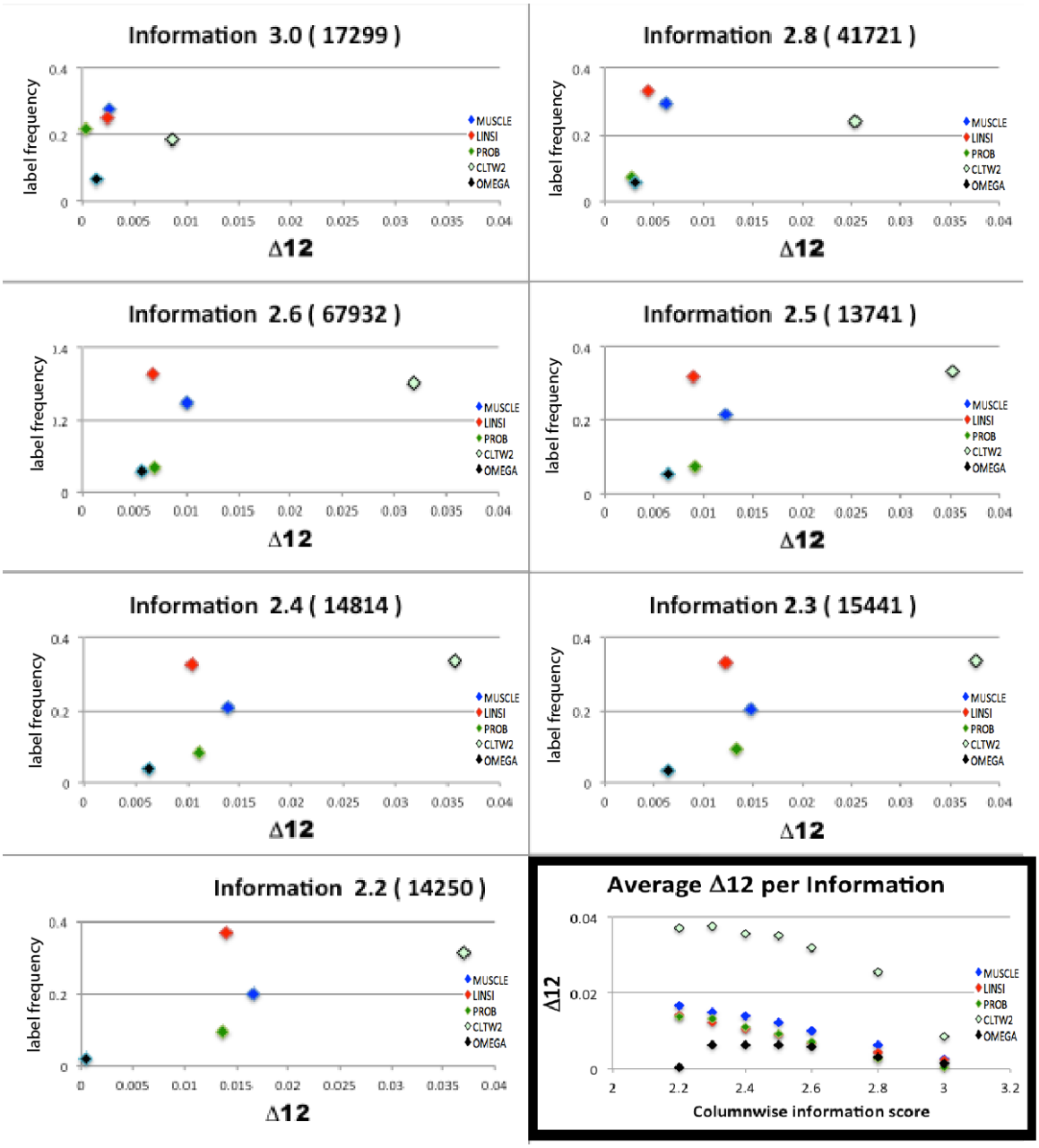


Figure 6.11: Relationship between the label frequency and Δ12 for different information scores. The range of information scores for each bin (and the number of datasets included) is shown at the top of each panel. The lower right plot shows the average Δ12 for each information score interval.

6.6 Domain number number and alignment accuracy

As described in Section 5.5, the SimProt database has 144 simulation groups, each of which has 1000 sequence sets based on a varying number (between 1 and 5) of domains (see Table 5.2). In this section, we examine the relationships between the number of domains in the sequence set and the various metrics (discussed in Section 3.1) to establish if the number of domains can be used as an indicator to determine which program will produce the most accurate alignment (closest to the optimal).

Alignments with the same number of domains are referred to as domain number groups. Figure 6.12 shows the average alignment length for each program as the label within each domain number group, divided for each topology dataset. As expected, the average length of the reference alignments increases as the number of domains increases. There is only a slight variation in alignment lengths between labels within any of the domain number groups, indicating that the accuracy of the alignment programs is not much affected by the length of the reference alignment. This insensitivity of the accuracy of the alignment process to the resulting alignment length has been noted in other studies [58] [106][79].

In Figure 6.13, the average protein identity was compared among domain number groups as well as programs chosen as the label. Within each topology dataset average protein identity varied among the program chosen as the label. However, except for Omega in the single-domain group (especially for ultrametric datasets), the variation pattern among the program was consistent regardless of the number of domains. In Figure 6.14, , the average CSS was compared among domain number groups. The variation pattern among the program was again largely consistent regardless of the domain numbers.

When the average $\Delta 12$ was examined across different domain numbers (Figures

6.15 and 6.16, negative relationships were observed for three programs, CLTW2, LINSI and MUSCLE; $\Delta 12$ decreased with more domain numbers. This corresponds to a decrease in the label frequency for MUSCLE but an increase for LINSI. CLTW2 did not exhibit any noticeable pattern of change in the label frequency associated with domain numbers. PROB had very little uctuation in either the average $\Delta 12$ or label frequency across different numbers of domains. OMEGA showed a slight raise in the 5-domain group but the other groups showed little change. It should be noted that while domain number caused uctuations in label frequency distribution, it did not affect the overall relative performance (ranking). This is to say that the ranking of programs by proportion of the label remains constant with changing domain number.

Over all, choice of the label appeared insensitive to domain number. While there was a small relative change in CSS with change in domain number, the ranking of the individual programs based on the label frequency remained unchanged. There was also small variations in protein identity with a change in domain number. These results suggest that the number of domains is a weak indicator of which alignment program will produce the most accurate alignment.

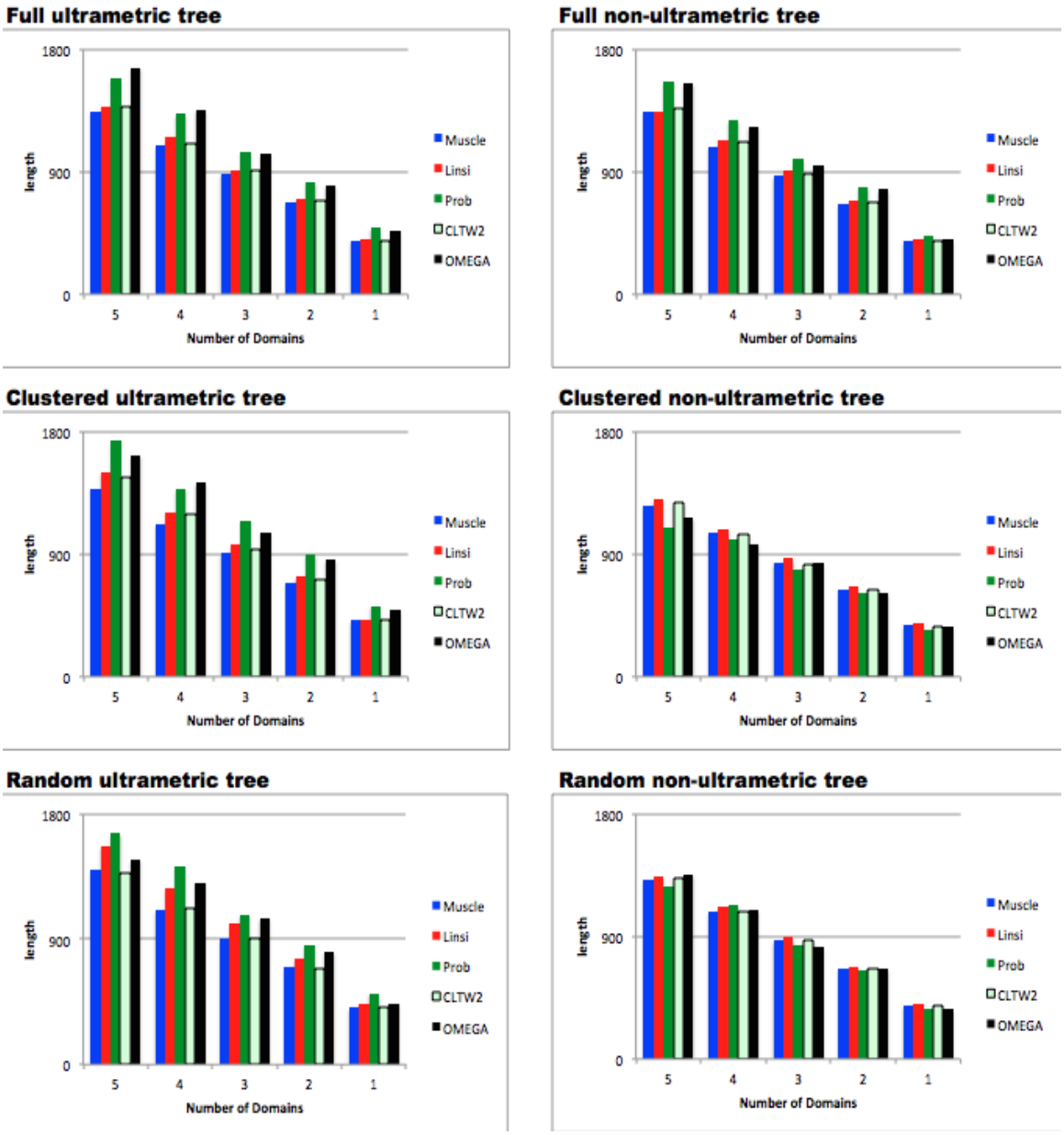


Figure 6.12: Association of alignment length and domain numbers with program performance. The average alignment length (number of amino acids) is based on the reference alignment when the corresponding alignment program was the label (the top performer) for the sequence set. The results are shown for each simulation guide-tree dataset.

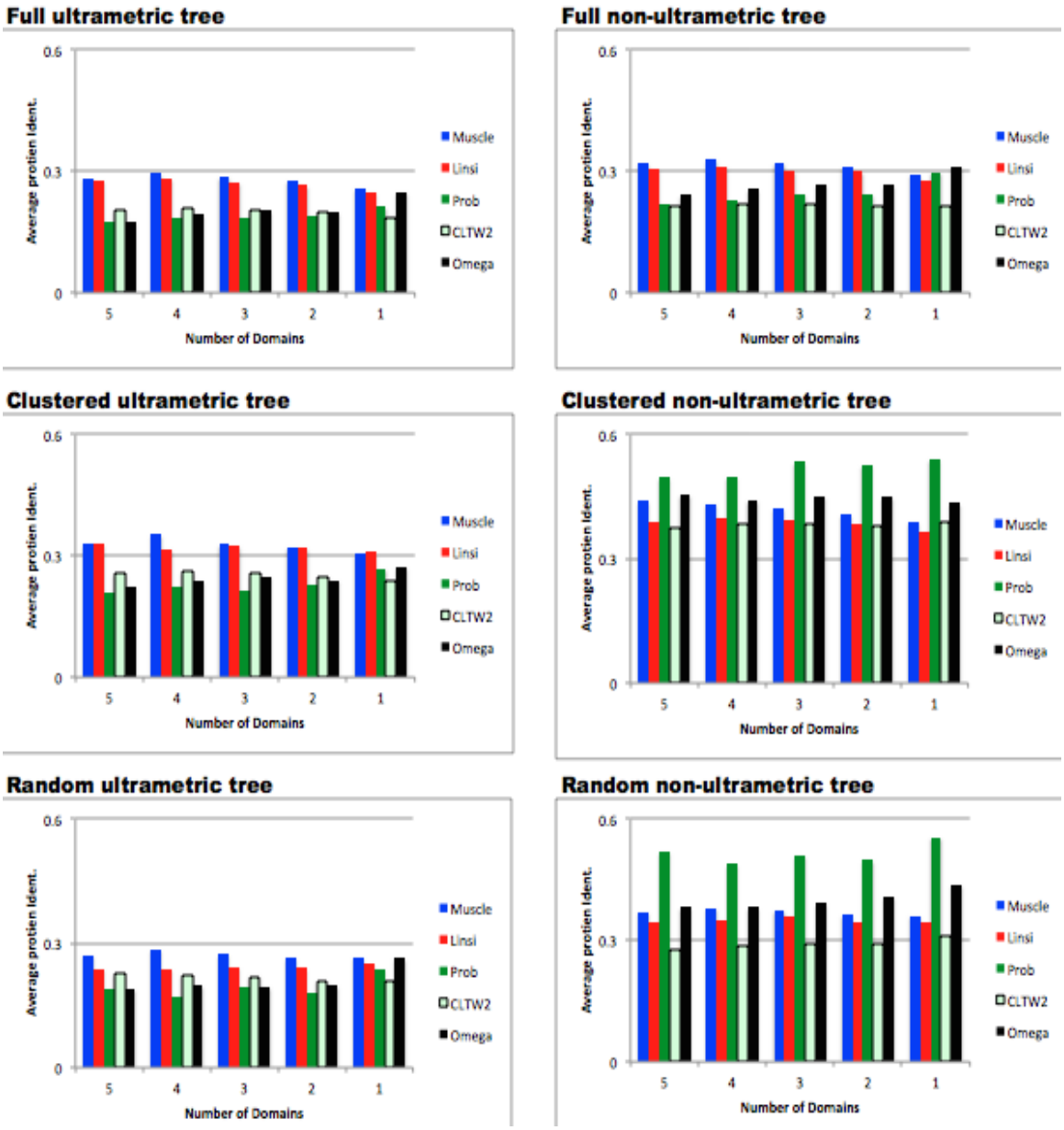


Figure 6.13: Association of protein identity and domain numbers with program performance. The average pairwise protein identity was calculated for the dataset where the corresponding alignment program was the label (the top performer). The results are shown for each simulation guide-tree dataset.

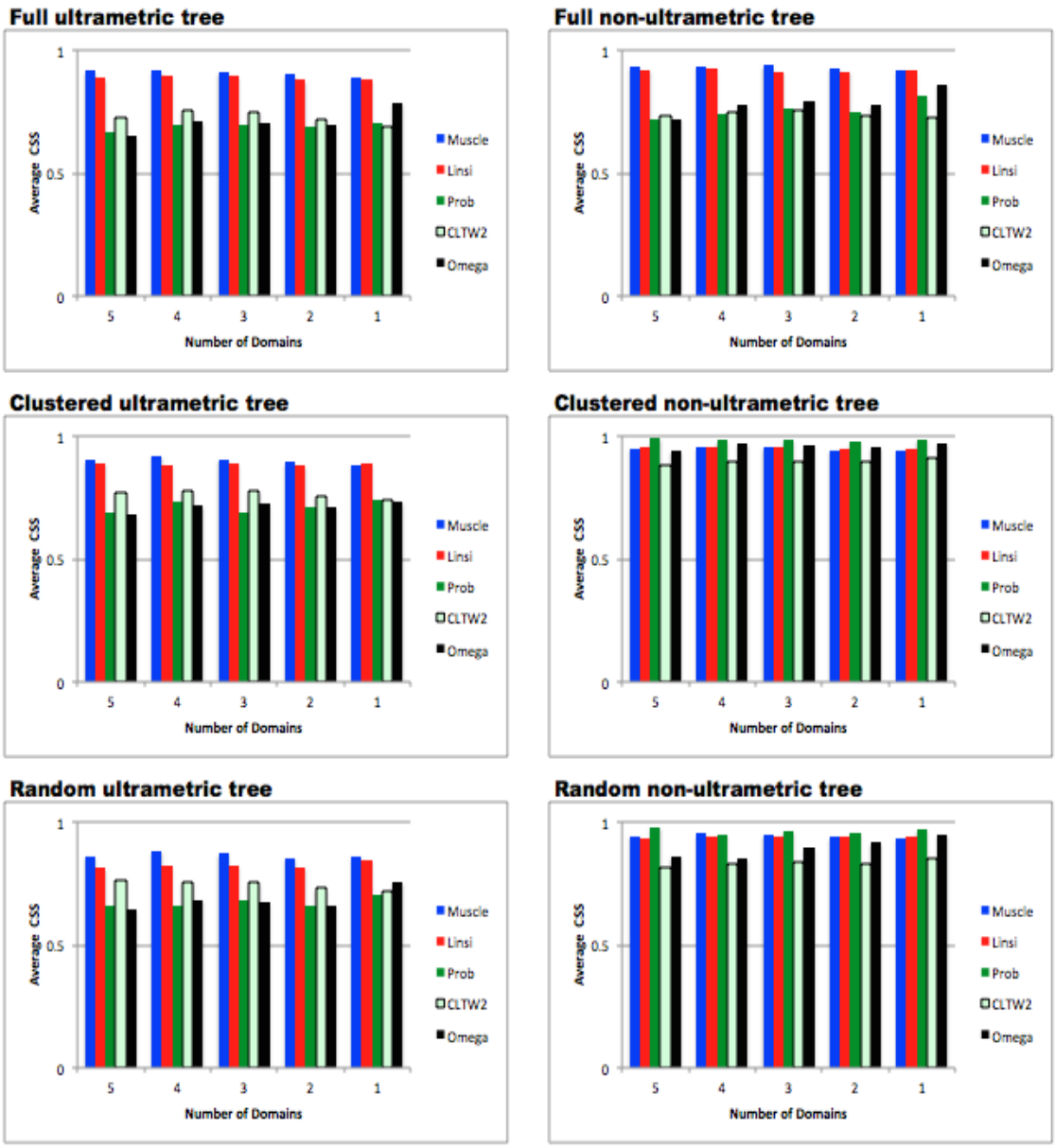


Figure 6.14: : Association of the average CSS and domain numbers with program performance. The average CSS was calculated from the label (top performing) alignment for each alignment program. The results are shown for each simulation guide-tree dataset.

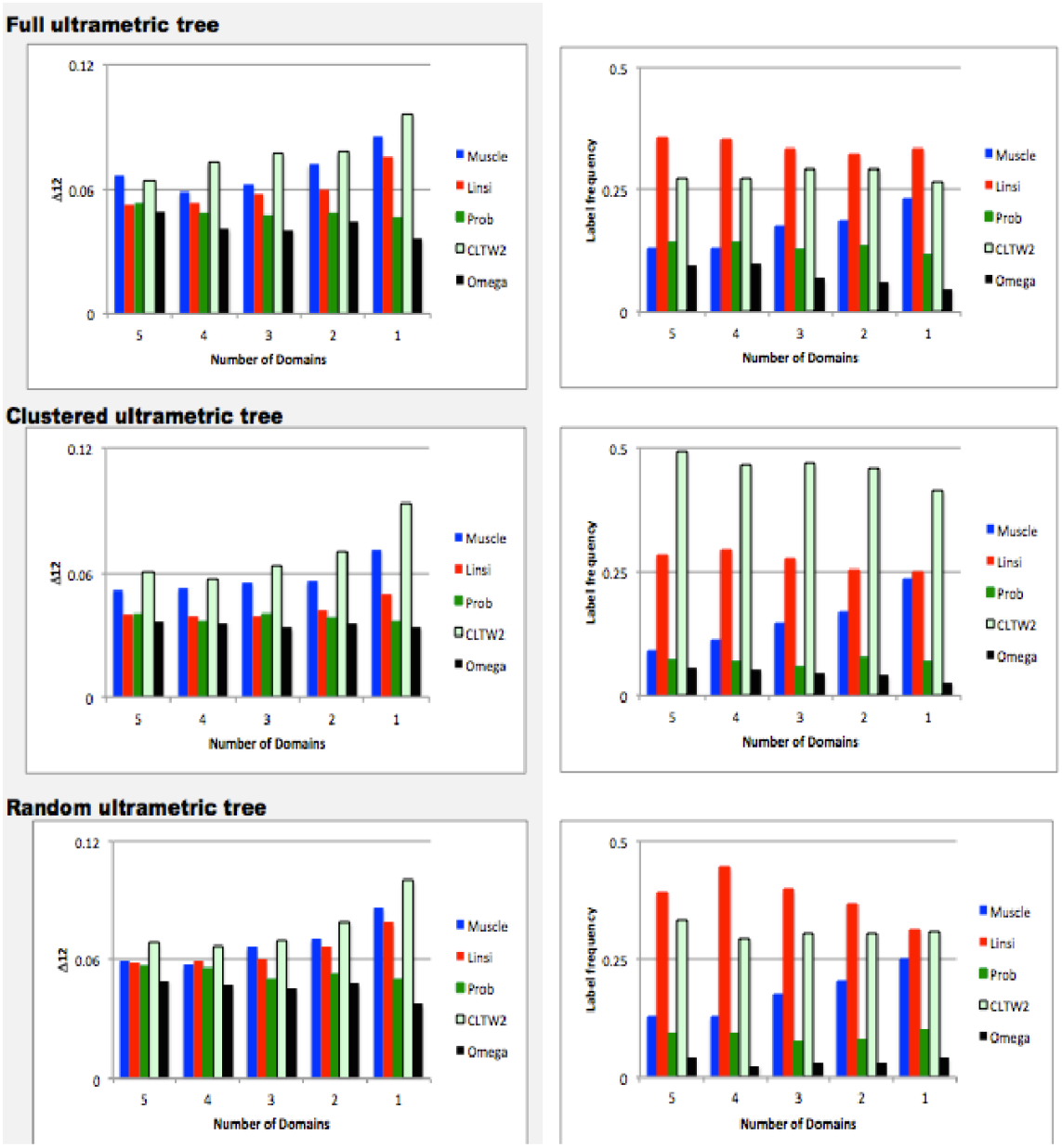


Figure 6.15: Association of $\Delta 12$ and label frequency with domain numbers and alignment performance. The results are shown for each simulation that used the ultrametric guide-tree datasets (FU, CU and RU).

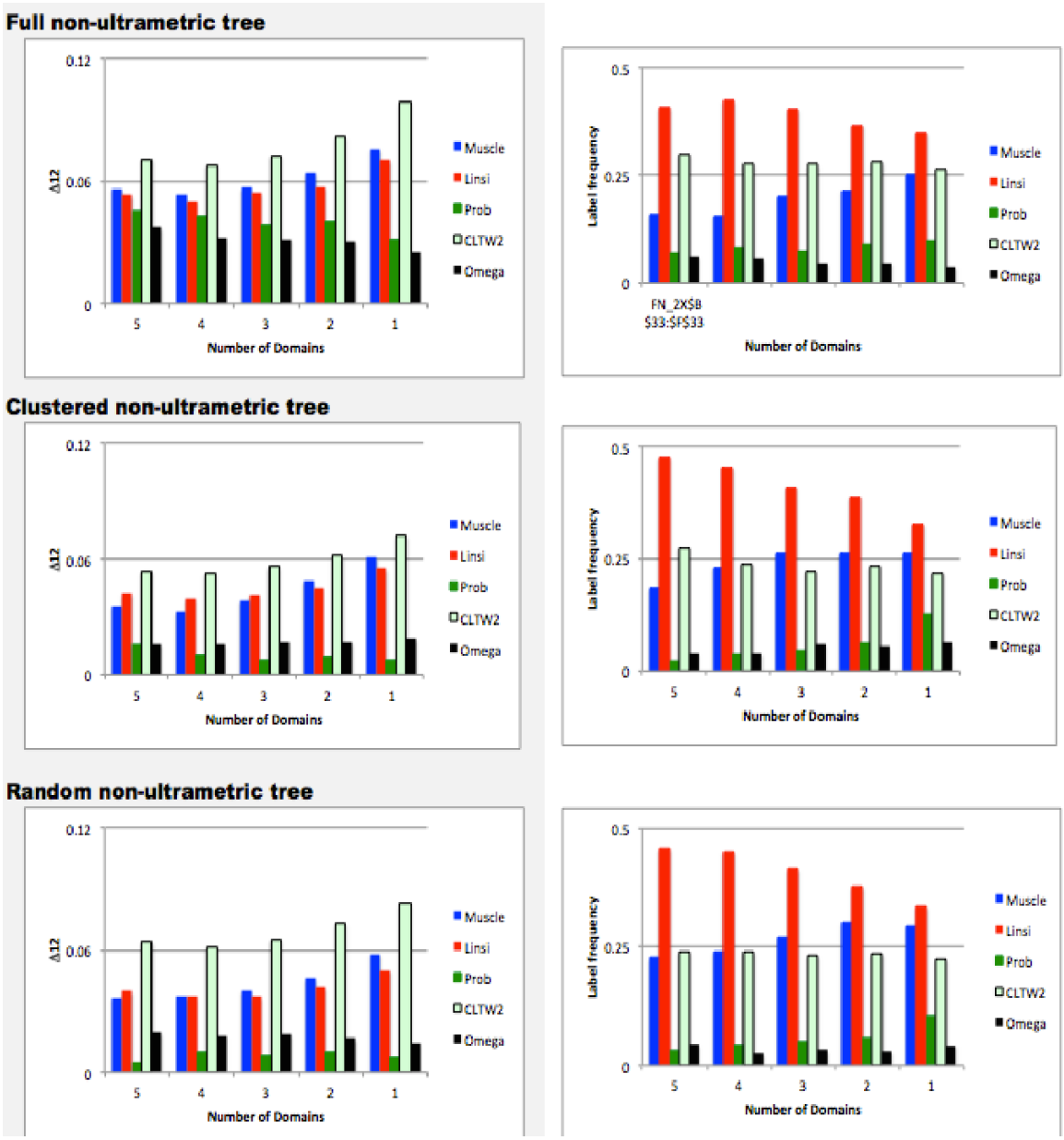


Figure 6.16: Association of $\Delta 12$ and label frequency with domain numbers and alignment performance. The results are shown for each simulation non-ultrametric guide-tree datasets (FN, CN and RN).

6.7 Summary of the evaluation study

In this chapter, we evaluated the relative performance of five alignment programs using CSS as the metric for accuracy. We defined various measures to better quantify any performance difference. We showed that there is a performance difference between the five alignment programs used in this study, and that this difference shifts depends on the sequence sets being used. By shifts we mean that the program that outperforms the others changes depending on different sequence sets. By examining the metric $\Delta 12$ on various groupings of data sets, we were able to establish the extent of this relative performance difference, if any, based on sequence set characteristics such as number of sequences, number of domains and topology of guide tree used in the simulation. We also were able to determine the effect of sequence divergence, as measured by average pairwise protein identity of the reference alignment, and sequence conservation as measured by the column-wise information score, on the relative performance the alignment program. We summarize our findings as follows:

1. We determined that no program consistently out performed another on all sequences sets of SimDom, and that the extent of this difference in performance varies (Figure 6.1).

2. We confirmed that the relative performance (in terms of CSS) of alignment programs shifts when aligning different datasets of sequences (Tables 6.1 and 6.2).

3. Using the metric $\Delta 12$ we found that the extent of the shift in relative performance changes with programs (Table 6.3 and Figure 6.5).

4. We defined BEST (the maximum alignment quality possible), and $\mathrm{CSS}_{\mathrm{program}}$ (the average alignment quality for each specific program), and $\Delta \mathrm{BP}(p)$. We

defined ΔBP(p) (the difference between Best and $CSS_{program}$). This allowed us to determine the range of improvement in quality available (the minimum and maximum values of ΔBP(p)). For the different datasets we evaluated, the maximum ΔBP(p) value ranged from 1.0% to 6.1% (Table 6.2)

5. We determined that the number of taxa, guide tree topology, branch factor and linker factor were, at best, weak indicators of a shift in relative performance between programs.

6. The average protein identity showed a negative correlation with Δ12 and as such is a good indicator of both Δ12 and which program would perform better.

7. We determined that the information score showed a weak negative correlation with Δ12. However,the ranking of the programs based on their label frequencies did not change, making information score a weak indicator of the label (most accurate program). weak (at best) indicator of label (most accurate alignment).

8. The analysis of the average maxCSS values for the sequence sets with the same number of domains showed that domain number is, at best, a weak indicator of relative performance.

With the extent of the performance shift determined and the maximum improvement in alignment quality established, we now have the means of evaluating any selection algorithm we develop. With the analysis of sequence sets and alignment characteristics, we have the information on what would constitute good attributes for the input vector to be used to train a classifier that selects the "closest to optimal" alignment. In the next chapter, development of such a classifier is discussed.

Chapter 7

Development of the machine learning algorithm for selecting the "closest to optimal" alignment

As described in Chapter 1, there are many bioinformatics studies that start with an MSA of the sequences that are under investigation. To achieve better quality in downstream studies, it is therefore critical to start with quality MSAs. However, as described in previous studies [106][105][79][21][20] and also as found in our evaluation discussed in Chapter 6, no single alignment program consistently outperforms all other programs across a wide range of alignment tests. Even in the situation where the sequence sets are grouped by their characteristics, such as sequence divergence (as measured by pairwise protein identity), number of domains, etc., no individual program consistently outperformed the other.

This performance difference indicates that each program has a specific type of sequence set on which they produce better alignments than other programs. Our approach is to harness this difference in performance to use each alignment program where it will produce the best alignment. Our hypothesis is that, by selecting the best alignment from a group produced by a suite of alignment programs, we can increase the average accuracy over the whole set of alignments.

In the following section of this chapter, we discuss the past attempts at using multiple MSA programs to achieve the best alignment for a set of sequences. We then

outline the conceptual points in the development of the "**S**e**L**ecting an **A**lignment **P**rogram" or **SLAP**, the algorithm for the selection of the alignment "Closest to Optimal". We follow this with a discussion of machine learning in general and the implementation of the multilayer perceptron in specific. The chapter concludes with a description of the process we used to configure the data model used with our algorithm. We show the testing procedure and results of our algorithm in the next chapter.

7.1 Discussion of previous studies

In the recent past, two projects have attempted to use more than one alignment program to obtain an alignment with higher quality. These were AQUA [75] and AlexSys [76], both of which are discussed in detail in Sections 3.5.1 and 3.5.2. We believe that the limited success of both of these efforts can be attributed to the following:

- ***Limited selection***: AQUA limited the group of alignments from which to choose to two base alignments created by MAFFT [77] and MUSCLE [16] and two additional alignments created by the refinement program RASCAL [69], operating on each of the two base alignments. This allowed for a selection of four different alignments in total. However, their results showed that the two refined alignments produced by RASCAL scored constantly lower than the two original alignments produced by MAFFT and MUSCLE. The exception was only when the sequences aligned were from closely related protein families. This effectively reduced the choice to the two base alignments. Although their results were not clearly presented, the choice of programs appeared to be too small to

have sufficient differences in relative performance affecting the possibility of improving alignments.

- ***Method of selecting the alignment***: Both projects employed methods that were insufficient to consistently discern the alignment with the best quality. AQUA used the magnitude of the NorMD scoring system to determine the best quality alignment. NorMD is a characterizing metric that uses a scoring matrix, such as BLOSUM62, to score the alignments without consulting a reference alignment. Because the objective function of NorMD is to maximize the alignment score, it tends to score the alignment of non-homologous positions favorably, potentially resulting in higher scores for over-aligned alignment than for a correct alignment. This type of metric can swing the selection of the best alignment to an over-aligned, and thus less optimal alignment. AlexSys used two approaches: a manually constructed decision tree and a polling approach using a series of binary classifiers. In both cases, the input data was based strictly on the characteristics of the sequence set. As we have shown in Chapter 6, there is a large variation in the performance between alignment programs even within sequence sets grouped by characteristics. Because of this phenomenon, the predictive power of a scheme based solely on sequence set characteristics would be weak. The best results reported AlexSys for this project showed only a 0.004 increase in average accuracy score above the best performing single alignment program, MAFFT.

- ***Focus of the goal for the project***. AlexSys, the one project of the two reviewed that used a machine learning approach that had the goal of achieving a "high quality" alignment, but not necessarily the highest. Furthermore, they combined the goal of high quality with that of fast runtime. They stated that

their suite of programs all created high quality alignments, and as such, when lack of discernment in their process of selection ended in ties between multiple alignment programs, they broke the ties by selecting the program with the fastest run time. Therefore, it is not surprising that the improvement achieved by AlexSys was minimum.

7.2 The approach used to develop SLAP

The goal of our algorithm is to select the best alignment available from a group of MSAs produced using a suite of alignment programs. The conceptual points of this development effort are as follows:

- ***Premise***: Our approach is based on the demonstrated phenomena that different programs perform better than others on different sets of sequences (see Figure 6.1). This approach advocates using the best program from a suite of high preforming programs for each set of sequences as opposed to selecting an alignment program based on efficiency, as done in AlexSys or limiting the choice to just a few alignment programs, as was done in AQUA. In this manner we harness the ability of the individual alignment programs to outperform the others on a specific sequence set to achieve an improved average alignment quality over the whole dataset.

- ***Method of selection***: We propose to use an artificial neural network as the learning algorithm to generate a multi-class classifier to select the alignment closest to the optimal. Artificial neural networks have been successfully used in solving machine learning problems such as image recognition [110] and handwriting transcription [111], both problems of which have been studied and analyzed since the early 1990s. Our project is the first attempt in using a multi-class

classifier on the problem of selecting the best multiple sequence alignment. As discussed in the introduction of this chapter, although AlexSys [76] used a series of binary decision trees, its objective was not necessarily to identify the best alignment, but rather, the most efficient program.

- ***Configuration of the data model***: The goal of this algorithm is to select the best alignment from a group of alignments created by specific programs. To achieve this goal it is essential to include characteristics of the individual alignments in the attributes. This is in addition to the sequence set characteristics. As discussed before, we believe the lack of this alignment information with AlexSys was one reason why the system failed to show sufficient improvement in alignment quality.

- ***Alignment quality score***: We based our evaluation of quality on the Cline shift score (CSS). CSS rewards correct alignment and penalizes incorrect alignment including both under-alignment and over-alignment (Section 3.2.3). This is the advantage over SPS and CS, the two most frequently used metrics used for alignment program accuracy. As described in Section 3.2.2, SPS, for example, is completely insensitive to over-alignment and can cause false quality measures with an alignment containing sections of frequent but independent gaps.

- ***Use of a reference dataset of sufficient scope and size***: We use a simulated data set, specifically from the SimDom database, to develop and test our proposed algorithm. SimDom contains sequences modeled after protein sequences that contain one or more domains (see Section 5.4). More importantly, it provides sequence sets that have no ambiguity in their alignments. This ensures that any scoring system applied to the generated alignments will have an absolute standard against which to be compared. In contrast, non-simulated

benchmark datasets (e.g., BaliBASE) have no true reference alignments. Moreover, they often contain limited segments of each sequence. In SimDom, the true alignments are present for the entire length of the sequences. This is especially critical when the sequence sets are highly divergent (protein identity $< 25\%$). Using a simulated reference database such as SimDom, we know all sequences are truely homologous and the reference alignment gives the true evolutionary relationship. Using SimDom will also ensure that there are sufficient training and testing data for our learning algorithm.

- ***Identification of the relative performance shift and its indicators***: Prior to the design of the data model to be used in training the multi-class classifier, as discussed in Chapter 6, we conducted a detailed performance evaluation of all five alignment programs using the SimDom benchmark datasets. We confirmed that there was a shift in performance among the five alignment programs to the extent that no single alignment program consistently outperformed all others. This study allowed us to identify those characteristics of both sequence sets and resulting alignments that can be used as the indicators for which alignment program created the best alignment.

- ***Quantification of the maximum possible improvement***: As described in Chapter 6, we developed a system of measures that allow us to better quantify the relative difference in performance between the alignment programs. Using these measures, we established the maximum possible improvement in alignment quality within any group of sequence sets. These measures identify both the potential of improving the average alignment quality and the goal of our algorithm. Using these measures, we established the existence of large performance variation among MSAs and demonstrated the need for a metric system

that will allow the strength of each alignment program to be used in selecting the best alignment. Unlike AQUA and AlexSys, which simply used the average scores, we leveraged the availability of the maximum quality alignment by tracking the improvement made by our algorithm and showing the increase in average alignment quality as compared to using just a single alignment program. We provide the results from the sign test to demonstrate that the improvement gained when using our algorithm is the closest to the maximum quality than any when using any single alignment program.

- ***Configuration of a library of attributes***: To provide our classifier with a wide choice of attributes, for each sequence set, we compiled a library of attributes representing both sequence set characteristics and alignment characteristics. When optimizing the data model for our algorithm we selected a custom set from this library. This allowed for an efficient reconfiguration of input instances during the design process of the data model. It can also be used to fine-tune the performance of our algorithm on smaller, more isolated regions of the instance space of our problem.

7.3 Overview of machine learning methods

In this section we will discuss some of the principles of machine learning in general and the artificial neural network in particular.

7.3.1 Background of machine learning

A machine learning algorithm is able to learn from experience. The concept of a program learning is succinctly described in the textbook on the subject authored by T. Mitchell, "A computer program is said to learn from experience $\boldsymbol{E}$ with respect to

some class of tasks $\boldsymbol{T}$ and performance measure $\boldsymbol{P}$, if its performance at tasks in $\boldsymbol{T}$, as measured by $\boldsymbol{P}$, improves with experience $\boldsymbol{E}$" [112].

The task $\boldsymbol{T}$ can be any of the many applications that need to process data. A few of the categories of these tasks are as follows:

- Classification, where the program is asked to specify which of k categories an input belongs to. An example would be a program that is asked to distinguish between photos of cats and dogs.

- Transcription, where the program is asked to take as input an unstructured representation of data, such as a scanned handwritten letter, and transcribe it into discrete, textual form.

- Translation, where the program is asked to take as input a sequence of symbols in one language and must convert it to a sequence of symbols in another language. This is most often used with natural languages, such as translating Chinese pictograms to written English.

- voice recognition, where the program is asked to take as input audio data and convert it to a textual form.

The Experience $\boldsymbol{E}$ is the dataset that the program deals with. This dataset is a collection of examples (data points), each of which contains features or attributes of quantitative or qualitative significance. These datapoints are the **instances** the program performs its task on. Usually, an instance is represented as a vector $\mathbf{x} \in \mathbb{R}^n$ where each x_i is a specific attribute describing the particular instance. The performance measure $\boldsymbol{P}$ is specific to $\boldsymbol{T}$. The traditional $\boldsymbol{P}$ for classification tasks is the accuracy of the program, which is determined by the number of correct predictions

the program makes. Thus, machine learning seeks to maximize $\boldsymbol{P}$, which is an optimization problem. However, what makes machine learning more than an optimization problem is that it not only attempts to minimize the error on the training data when fitting the training data to a curve, but it attempts to minimize error for new data that it has not seen yet. In other words, the algorithm "learns" from the old data to better process the new.

In this study, the task $\boldsymbol{T}$ that the program is to learn is one of classification. The classification function can be formalized as $f : \mathbb{R}^n \rightarrow \{$MUSCLE, LINSI, PROB, CLTW2, OMEGA$\}$ where the names represent the various alignment programs. Traditionally, for classification problems, $\boldsymbol{P}$ is the accuracy of the model, defined as the proportion of examples for which the model predicts the correct output. However, due to the nature of our problem the accuracy did not adequately reflect the quality of the outcome of the training. In Section 8.4, we will introduce a novel performance measure that more closely tracks the desired outcome of the combined learning algorithm and problem model of the classifier.

We used a *supervised learning* algorithm in this study, where each instance upon which the algorithm is trained has an associated label indicating the correct classification for the instance. The term supervised learning comes from the analogy that the label serves as an "instructor" who supervises the learning system in what to do.

7.3.2 Multi-layer perceptron

The learning algorithm we used in this study is the multi-layer perceptron (MLP), which is a feed-forward type of artificial neural network (ANN). They are called feed-forward because the input data enters the ANN as x and is processed by the computations that make up the function, f. When all calculations are complete, the ANN outputs a prediction. There is no feedback connection where the output

for each instance is fed-back to the processing function. The design of ANNs was inspired by the densely interconnected network of the brain and are well-suited to problems with specific conditions or characteristics [112]. The following is a list of these characteristics along with an explanation of how the underlying learning problem of our application satisfies them.

- ***Instances are represented by many attribute-value pairs***. The input data for our problem would need to capture the characteristics of both the set of protein sequences and the five alignments generated by the programs in our study in sufficient detail to allow the learning algorithm to correctly predict the most accurate alignment.

- ***Output may be discrete-valued, real-valued or a vector of discrete or real values***. In our problem, the output is one of five discrete labels, which correspond to the names of the individual alignment programs.

- ***Training data may contain errors***. Training data error does not occur when using the SimDom database since we have the true reference alignments on which to base the labels. However, when using a non-simulated sequence database as a source of training data, true alignments are not known and hence there is a greater chance of errors.

- ***Long training times are acceptable***. The anticipated use of this classifier is to train it for a specific dataset and then use it to classify multiple input instances. Frequent retraining during any single study is not anticipated and as such an initial longer training time is not an impediment to the use of the ANN.

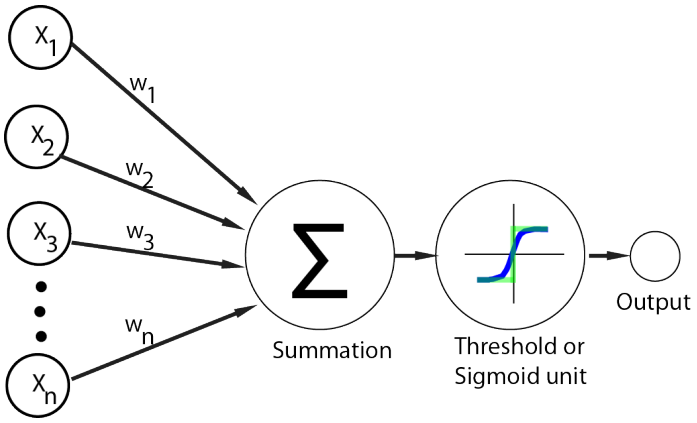


Figure 7.1: Single perceptron unit. An associated threshold (a step function in green) or sigmoid function (blue curve) unit as well as the output unit are also shown.

- ***Fast evaluation of target function***. Once our classifier is trained, evaluation for each input sequence set is rapid for processing large batch jobs which would in turn be used in additional downstream processing.

- ***Target function does not need to be human readable***. As we explain in the next section, the function for an ANN is implemented as multi-connected nodes with weighted edges. In our application of the ANN, it is not necessary for the weights on the node connections to be obviously connected to any specific trait thus rendering it human readable or understandable.

The multi-layer perceptron is based on the unit called a perceptron, which is illustrated in Figure 7.1. The perceptron unit can be viewed as a hyperplane decision surface in the n-dimensional instance space, where n is the number of values in the input vector, and can be used to represent a linear decision surface. Its task is to take the input vector and return either 1 or -1. To obtain a more discerning, nonlinear decision surface, the multiple perceptron units can be connected to form a hierarchical structure. The number of output units is increased to the number of labels and an additional layer of units, referred to as the hidden layer, is inserted between the input

and the output units. Each layer of units is fully connected with the layer to the left of it, resulting in the two layers being fully connected to the hidden layer, and the hidden layer being fully connected with the output layer for this single hidden layer example. An arrangement of this type is referred to as a multiple layer perceptron. A diagram of this hierarchical structure is shown in Figure 7.2. For simplicity, the edge weight labels in the diagram are shown for only the weights on the connections between the input units and the hidden units. The weights on the connections between the hidden layer and the output nodes are present but not shown in this figure. The weights for all connections are determined by applying the gradient descent algorithm, and using a sigmoid unit on each perceptron unit in place of the step function to make it differentiable. This requires multiple iterations through the training data, where the weights on all connections are adjusted each time an incorrect prediction is made for an instance of the training data. Each adjustment incorporates a learning factor which determines the rate at which the weight is adjusted, and a momentum factor, which determines the portion of the previous adjustment that is added to the current adjustment.

For this study we used the source codes from the Weka workbench [113] in conjunction with our custom front-end program to train and test the MLP. All testing in this study was done using a 10-fold cross-validation strategy, which is described in Section 7.4.

7.3.3 Other machine-learning algorithms tested

Prior to the decision to use the MLP for our application, we performed preliminary testing of different learning algorithms using their implementation provided in the Weka workbench [113]. A preliminary dataset used for the training consisted of 300 simulated sequence sets created by indel-seq-gen [88]. 10-fold cross-validation was

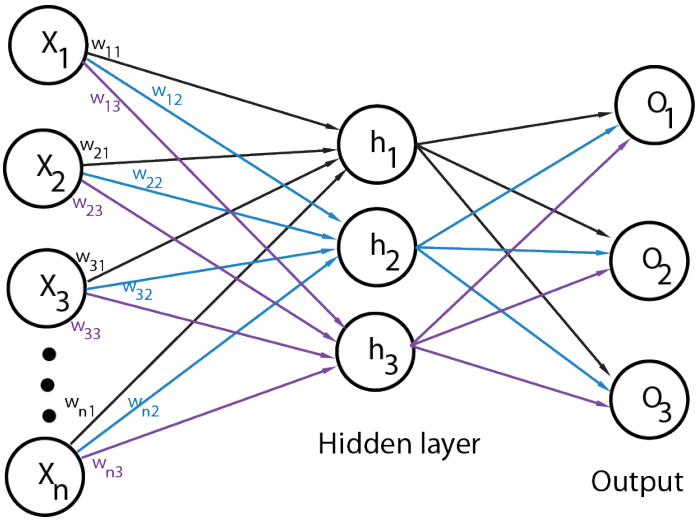


Figure 7.2: Multi-layer perceptron. A node layout with n input nodes, one layer of 3 hidden nodes and 3 output nodes are shown.

performed to test each of the algorithms. The classifier accuracies were fairly low for all algorithms, ranging from 25 to 42%. However, while the training time was higher for the MLP by as much as a factor of 10 (approximately 5 minutes for the MLP as compared to around 30 seconds or less for the other algorithms), its resulting average accuracy was noticeably higher than any of the others. As such, we decided to concentrate our effort on using the MLP for our application. The configuration we used for the MLP was a single hidden layer containing 24 nodes, where the node number was obtained as (attributes + classes)/2 for 44 attributes and 5 classes. We used the default values for the learning rate, 0.3, and the momentum, 0.2. Trials were run to evaluate the effect of adjusting both the learning rate and momentum but no significant gain was made. The alternative algorithms we included in this screening were as follows:

- J48: the Weka implementation of the C4.5 decision tree [114], which builds a tree based on the information entropy. Each split of the tree is determined by

the attribute with the highest normalized information gain.

- Random forest [115]: an ensemble learning algorithm that creates multiple decision trees and outputs the mean of the classification of these trees.

- AdaBoost [116]: a boosting algorithm that uses a "weak learner" such as a decision stump and through multiple iterations promotes those input instances that are mislabeled.

- Kstar: an algorithm where the class of a test instance is based upon the class of those training instances similar to it, as determined by a similarity function based on an entropy-based distance.

- SMO [117]: an algorithm which solves the quadratic programming problem used to train a support vector machines (SVM). SVM is a classifier that defines a hyperplane for classification based on maximizing the margin between the instances of each label.

7.4 Cross-validation

Cross-validation is a common training and testing strategy for machine learning algorithms. It is a method of testing the ability of the learning algorithm to generalize, which is to predict the correct label on an instance vector not seen during the training of the classifier. For example, a 10-fold cross-validation strategy involves randomly dividing the available set of input instances into ten evenly sized subsets. The resulting division is referred to as a partitioning of the input instances with each subset of the partition referred to as a fold, thus the name 10-fold for this example. Each fold of the partition is then used, in turn, as testing data, with the other nine folds used as the training data for the classifier to be tested. This allows all available instances to

be used as a test case. To increase the number of training and testing runs, multiple partitions can be created and used in training and testing.

In this study, we defined seven datasets from the SimDom benchmark database: ALL, FU, FN, CU, CN, RN, and RU. The dataset ALL contains instances for all 144,000 sequence sets in SimDom. Each of the datasets FU, FN, CU, CN, RN and RU contains instances for 24,000 sequence sets. To perform 10-fold cross-validation analysis, each of the seven datasets was randomly partitioned into 10 subsets without replacement. Nine of the ten subsets were used for training and the remaining one subset was used for testing. This process was repeated ten times generating in total 100 training/testing runs for each dataset. The average and standard deviation of the prediction accuracy were calculated for each dataset from these 100 trials.

7.5 Description of the "closest to optimal" alignment problem

In this section we will describe our underlying machine learning problem which involves determining the best quality alignment. We measure the accuracy of an alignment by CSS when compared to the "true" or "reference" alignment. As described in Section 3.2.3, CSS is the number of correctly aligned pairs combined with a shift penalty for misalignment and under/over alignment. Alignment problems involve sequence sets that vary in the degree of pairwise divergence (evolutionary distance). When sequences within a set are closely related, there is a high degree of agreement or consistency between the alignments generated by any of the alignment programs. In such cases, the choice of the best or better alignment for that set is less critical. However, as the divergence between the sequences increases, there tends to be more differences (or inconsistencies) between various alignments making the choice of which alignment to use more critical.

In previous evaluation studies (e.g., [16][24][20][21]) as well as in our own study described in Chapter 6, regardless of the metric used, average alignment accuracies were usually reported. However, while a program may achieve the highest average score, another alignment program(s) could generate a more accurate alignment depending on the sequences. In fact, in the evaluation performed in Chapter 6 (see Figure 6.1), we found that no single alignment program among the five examined consistently produced the most accurate alignment.

When making the choice of which alignment program to use for a group of sequences, it is more important to be able to choose the alignment program that performs best on that particular set of sequences as opposed to the program that performs the best on average across any sequence sets. We therefore developed an algorithm that assists the user in determining which alignment from the group of alignments is the closest to the optimal. This algorithm is called "**S**e**L**ecting an **A**lignment **P**rogram" (**SLAP**). It consists of training a multi-class classifier to predict which of the five alignments generated by the five alignment programs is closest to the optimal. In the next section, the details of the data model used in the **SLAP** algorithm are described.

7.6 Data model for the "closest to optimal" problem

In this section we will discuss how the results of the evaluation study done for the five alignment programs reported in Chapter 6 were used to generate a list of candidate attributes for the input vector, the selection process by which the attributes were chosen and combined to form an input vector that most accurately predicted the label.

In our discussion of the data model, we divide the attributes of the instance vector

into two groups: those associated with sequence sets and those based on alignments. The sequence-set based attributes are those that describe a group of sequences as they compare to one another, independent of an alignment. The alignment-based attributes are a set of values that define the property of a specific alignment (e.g., lengths, number of gaps). We used both the forward selection and backward elimination algorithms to determine which combination of attributes most accurately predicted the correct label of our problem when used with the Weka workbench implementation of the MLP. The detailed discussion of our development effort is divided as follows:

- Description of the input data structure.
- A discussion of the candidates for the sequence-set based attributes along with their selection.
- A discussion of the candidates for the alignment-based attributes.
- How we determined the best combination of attributes for an input instance vector.

7.6.1 Structure of the input data

Each instance is the ordered set of attributes associated with a single sequence set and reference alignment. By ordered, we mean that the attributes are listed in the same order for each instance. The label assigned to each instance is the name of the program that generated the alignment with the highest CSS for the sequence set. The five programs we used in this study are described in Section 2.5 (their abbreviations used are listed in Table 1). The frequency of the assigned labels for SimDom (for each guide-tree topology group) is shown in Table 7.1.

7.6.2 Sequence-set associated attributes

We will now discuss the numeric values that we believe capture different aspects of a sequence set to be used as candidate attributes for our data model. We will then perform an attribute selection procedure to determine which attributes generate the classifier with the best accuracy.

The relative performance of the alignment programs can be affected by several aspects of the sequences to be aligned. As described in Chapter 6, the number of sequences, the amount of divergence between the sequences, as well as how the sequences relate to each other (represented by the guide-tree topology used in the simulation) all affected the relative performance of the five programs we tested. The number of sequences can easily be incorporated as an integer attribute.

To capture the sequence divergence without the use of an alignment will require multiple attributes. One aspect of sequence divergence is the variation in sequence length with the sequence set. The higher the indel rate and/or the longer the time of evolutionary (evolution distance), the greater will be the variation in lengths of the sequences. To capture the detail of the variation of the sequence length, we will included the average, standard deviation, maximum, and minimum lengths of sequences in our list of candidate attributes. We also included the minimum number of gap positions needed to bring all sequences to the same length and the ratio of maximum sequence length to minimum sequence length.

Another aspect of the divergence of a sequence set is the variation in the sequence property within the set. This aspect of divergence can be captured numerically as the frequency of each of 20 amino acids in each individual sequence, obtaining the average and the standard deviation within the set. This attribute set consists of the 20 average and standard deviation of amino acid frequencies, resulting in an additional

Table 7.1: Frequency of the labels for each guide-tree topology group.The data is shown by topology group: FU, FN, CU, CN, RU, RN. The alignment having the highest CSS value is designated as the label for each sequence set.

		frequency of label				
group	# sets	MUSCLE	LINSI	PROB	CLTW2	OMEGA
FU	24000	0.206	0.332	0.127	0.277	0.058
FN	24000	0.229	0.365	0.092	0.271	0.043
CU	24000	0.185	0.235	0.110	0.436	0.035
CN	24000	0.258	0.363	0.095	0.225	0.059
RU	24000	0.220	0.345	0.093	0.306	0.036
RN	24000	0.288	0.364	0.082	0.229	0.037

40 candidate attributes.

These attributes along with their designated single-letter codes are as follows:

- number of sequences (A)
- maximum sequence length (B)
- minimum sequence length (D)
- average sequence length (C)
- standard deviation of sequence length (Z)
- ratio of maximum sequence length to minimum sequence length (F)
- percent of sequences under the average length (G)
- minimum percent of positions needed as gaps (E)
- average amino acid frequencies (20 values) (Q)
- standard deviations of amino acid frequencies (20 values) (S)

In total, these 48 attributes (designated with 10 codes) make up the candidate sequence-set based attributes of the multi-class problem. We noted that some of these attributes are highly correlated. We therefore performed the attribute selection process to remove redundant attributes or those that do not contribute significantly to the performance of the classifier. It would be also desirable if we could devise a model based solely on sequence-set based attributes because it would the computation needed to generate the alignment-based attributes (described in the next section).

We used the *forward selection* algorithm employing a ten-fold cross-validation test on the full SimDom dataset, to determine which of the attributes should be included in our input instance (Algorithm 4). The results showed that the best combination of sequence attributes was the subset AFS, which contains 22 attributes

- number of sequences (A)
- ratio of maximum sequence length to minimum sequence length (F)
- standard deviations for amino acid frequencies (20 values) (S)

The best accuracy for this sequence related attributes was 42.8%, which is rather low, indicating that the sequence-based attributes by themselves are not sufficient to separate the five classes.

7.6.3 Alignment-based attributes

The alignment-based attributes are those that capture the shape, conservation, and divergence of the alignments generated from each program. The first set of attributes is an estimate of the number of conserved areas are identified using column-wise information scores. We divide the alignment into segments and scan them looking for those that have average column-wise information scores higher than a given threshold.

Data: List of attributes
Result: Combination of attributes with highest accuracy
forall *Attribute codes* **do**
  Create a instances with single attribute and label
  Perform a single ten-fold cross-validation run
  Record attribute and accuracy in list sorted on accuracy
end
Initialize combo_1 with attribute on the top of sorted list **forall** *Attribute codes* **do**
  Add next attribute in list to combo
  Create a instances with all attribute in combo_1 and label
  Perform a single ten-fold cross-validation run
  if *Accuracy does not decrease* **then**
    Keep attribute in combo_1
  end
  else
    Remove attribute and add it to retry list
  end
end
Initialize combo_2 with 2nd highest attribute on first sorted list **forall** *Attribute codes in retry list* **do**
  Add next attribute in retry list to combo_2
  Create an instance with all attributes in combo_2 and label
  Perform a single ten-fold cross-validation run
  if *Accuracy does not decrease* **then**
    Keep attribute in combo_2
  end
  else
    Remove attribute
  end
end
forall *Attribute codes combo_1* **do**
  Add next attribute in combo_1 to combo_2
  Create instances with all attributes in combo_2 and label
  Perform a single ten-fold cross-validation run
  if *Accuracy does not decrease* **then**
    Keep attribute in combo_2
  end
  else
    Remove attribute
  end
end
Return the combo (between combo_1 and comb0_2) with the highest accuracy

Algorithm 4: Forward selection of sequence-set attributes using 10-fold cross-validation.

The standard deviation of the column-wise information scores is also examined to identify possible transition regions from a domain to a linker (or vice versa) segment. Algorithm 5 summarizes this protocol. We set the length of the segments and the thresholds based on trial runs using the reference alignments of SimDom, where the exact number of domains is known.

Data: MSA, information score threshold,columns per segment
Result: Calculated estimated number of conserved areas
forall *columns in the MSA* **do**
  Calculate the information score, I_i
end
Break alignment in segments of N columns
forall *segment in the MSA* **do**
  Find the average and standard deviation of I_i in segment
  Assign designation
  if *average column-wise information score is greater then threshold* **then**
    designation $\leftarrow C$ (for 'Conserved')
  end
  else
    designation $\leftarrow N$ (for 'Not conserved')
  end
end
initialize area list with current area $\leftarrow C$ designation of first segment
for *all other segments in the MSA* **do**
  if *segment area designation equal designation of current area* **then**
    add segment to current area.
  end
  else
    if *standard deviation for previous segment is higher than threshold* **then**
      start new area and give it the designation of current segment.
    end
    else
      add segment to current area.
    end
  end
end
number of conserved areas $\leftarrow$ number of areas with C designation

Algorithm 5: Estimation of the number of conserved segments within an MSA.

The result of Algorithm 5 applied to the alignment generated by each of the five alignment methods yielded a value (integer), adding 5 attributes to the model. The designated code for these attributes is CON.

The next set of attributes deals with the column-wise characteristics for each individual alignment. These are divided into two groups, gap characteristics and conservation characteristics:

- Gap characteristics (GP):
  - % columns with no gaps
  - % columns with 50% or less gaps (excluding columns with no gaps)
  - % columns with more than 50% gaps (excluding those occupied by a single non-gap residue)
  - % columns with only one non-gap position (only one sequence with an amino acid in the column).
  - % gaps in the alignment, which is given as (the total number of gaps in the alignment)/[(the number of sequences in the alignment) * (number of columns in the alignment)]
- Conservation characteristics (CN),
  - % columns that are completely conserved and no gaps
  - % columns with only two types of amino acids and no gaps
  - % columns with three types of amino acids and no gaps

Each of the 5 alignments will contain these 8 attributes, adding 40 attributes to the model.

The next group of attributes consists of two metrics used to capture the overall divergence of the alignment. They are the average pairwise protein identity (PRO) (Section 3.1.1) and the Information score (INF) (Section 3.1.2). This contributes two attributes for each of five alignments, adding ten attributes to the model.

The next set of attributes consist of the pair-wise computation of CSS, each of the five alignments against the other, adding 10 attributes.

The final set of attributes captures the finer-grained column-wise composition in each alignment. For 20 amino acids and 1 gap character the number of all possible unordered triplets is 1771 . By unordered we mean that, for example, ACF is treated as the same as CAF and FAC. This results in 8855 3-mer frequency values for all five alignments. To reduce the number of attributes, we used the following two approaches (only one set of them is included in the final set of attributes as described in Section 7.6.4).

The first approach (RAT) is to combine these 3-mer frequencies into seven generalized frequencies for each alignment. This would create a set of seven attributes for each of the five alignments, resulting in 35 attributes representing all five alignments. These generalized frequencies consist of 3-mers that are as follows:

- all gaps (occurs when three or more sequences have a gap)
- 2 gaps with any one amino acid
- 1 gap with 2 of the same of any amino acid
- 1 gap with any 2 different amino acids
- any 3 different amino acids
- any 2 same amino acids with one different

- any 3 of the same amino acids

The second approach (PM) uses principal component analysis and uses a smaller number of the principal components that recapture a specified amount of variance are used as the fewer number of converted attributes. To re-capture 90 percent of the variance of the original attributes, for example, as many as 544 principal components were needed, which resulted in more than 2500 attributes. This large number of attributes will greatly increase the amount of time needed to train the model. As such, it will have to show a marked improvement in prediction to justify the increase computational expense.

7.6.4 The full set of candidate attributes

The full set of candidate attributes (and the number of values) is as follows:

- number of taxa in the sequence set (1)
- percent of sequences over average (1)
- standard deviation of amino acid frequencies in the sequence set (20)
- number of conserved areas from each alignment (5)
- 8 gap and conservation characteristics from each alignment (40)
- information score from each alignment (5)
- average pairwise protein identity from each alignment (5)
- pair-wise CSS comparisons (10)
- 7 generalized column-wise 3-mer frequencies from each alignment (35), or

- up to 544 principle components from 3-mer frequencies from each alignment (over 2500)

We performed a search of the attribute space to find the subset that predicted the label most accurately. To reduce the complexity of this search, the attributes were added and removed in groups as listed above. We used both the forward selection and a hybrid multiple backward elimination with no bias added toward smaller attribute sets employing a ten-fold cross-validation test on the full SimDom dataset. The algorithm for a hybrid multiple backward elimination is given below in Algorithm 6. The results of the full attribute selection trial using the multi-start backward algorithm yielded the same combination as the forward selection algorithm for the full set of attributes. The final set of the attributes (and the number of values) consisted of:

- number of taxa in the sequence set (1)
- average protein identity (5)
- pairwise CSS from five alignments (10)
- generalized 3-mer frequencies from each alignment (35)

The final average accuracy using both one sequence-based attribute and 50 alignment-based attributes was 0.621 for a single ten-fold cross-validation run. This represents an increase of roughly 0.200 in accuracy over using just the sequence attributes (0.428, Section 7.6.2). There was also a significant reduction in the number of attributes from the full set of attributes, which eliminated a large amount of the calculation that the discarded attributes would require. This is especially true for the principal component transformation, which consisted of 450 attributes and required 8.5 hours per ten-fold

Data: Full set of attributes
Result: Selection of the alignment-based attributes using ten-fold cross-validation
Form instances from all attributes
Perform a ten-fold cross-validation recording score.
Initialize combination list, adding first combination of all attributes along with score.
Initialize empty reject list
while *combination list not empty* **do**
  remove next combination from list
  foreach *group of attributes in combination* **do**
    Remove a single attribute
    Form instances from all attributes
    Perform a ten-fold cross-validation recording score.
    if *accuracy goes up* **then**
      Add reduced combination to combination list if not already on list
    end
    else
      Add to reject list
    end
  end
end
Last combination off of combination list

Algorithm 6: Generation of a list of attributes that will train the most accurate classifier.

partition using the full SimDom database, as opposed to 38 minutes required for the 35-value generalized 3-mer frequencies. Use of the principle component attributes did not increase the accuracy and in some cases, there was a decrease in accuracy. Therefore, this set of attributes could be eliminated. It should be noted that we did not perform an exhaustive search of the attribute space, but more of a multi-start greedy approach. However, since the gain and loss of accuracy fell within a range of 4.5% with an overall increase of 1.5%, there is only a small likelihood that additional search will yield much difference in accuracy.

Chapter 8

Performance analysis of SLAP

In this chapter we will discuss the performance of **SLAP** to select the "closest to optimal" alignment. We outline the testing procedure and evaluate both the accuracy of the classifier and the resulting improvement in average alignment quality as measured by CSS. We then analyze the training/testing data to ascertain the reason for the results we obtain and the implication to the underlying goal: selecting the one alignment closest to the optimal out of the five. We also define a new metric, SLAP_RAT, which gives a better measure of the performance of **SLAP** in relation to the machine learning problem.

8.1 Procedure for testing SLAP

We tested the performance of **SLAP** in its ability to select the closest to the optimal alignment, using seven datasets from the simulated protein sequence benchmark database, SimDom. The seven test datasets are defined as follows:

- ALL - contains all 144,000 the sequence sets of the SimDom database
- FU - contains all 24,000 the sequence sets simulated with the FU guide trees.
- FN - contains all 24,000 the sequence sets simulated with the FN guide trees.
- CU - contains all 24,000 the sequence sets simulated with the CU guide trees.

- FU - contains all 24,000 the sequence sets simulated with the CN guide trees.
- CN - contains all 24,000 the sequence sets simulated with the RU guide trees.
- RU - contains all 24,000 the sequence sets simulated with the RN guide trees.

The dataset ALL covers a larger area of the instance space for the machine learning problem than the guide-tree specific datasets. Each of the smaller datasets covers a smaller portion of the total instance space. Our evaluation procedure consists of performing a 10-fold cross-validation test on ten unique partitioning of the dataset, resulting in 100 individual training/testing runs (see Section 7.4 for the details). We perform the same testing procedure on each of the seven test datasets and quantify the performance using the classifier accuracy metric, which is given as:

$$\text{Accuracy} = \frac{\text{correct labels}}{\text{all labels}}. \tag{8.1}$$

The resulting accuracies of **SLAP** on the seven test datasets are given in Table 8.1. The accuracies range from 0.524 for CN to 0.658 for CU, with the largest dataset ALL having an accuracy of 0.614. The first two observations are: 1) the accuracies seem to be low and 2) there is a large variation in the accuracies for the different test datasets.

Table 8.1: Accuracies of **SLAP** on the seven test dataset using 10-fold cross-validation analysis. SD: standard deviation.

	ALL	FU	FN	CU	CN	RU	RN
accuracy	0.614	0.637	0.633	0.658	0.524	0.626	0.564
(SD)	(0.004)	(0.012)	(0.008)	(0.012)	(0.011)	(0.006)	(0.011)

When we evaluate the accuracy of **SLAP** we must take into account that the **SLAP** classifier is a five-class classifier. Thus we consider the following two trivial classifiers: 1) the five-class classifier that simply selects a label at random and 2) the classifier that labels all test cases with the label of the class with the highest frequency in the dataset ALL. The first trivial classifier will achieve an accuracy of approximately 0.20. The second trivial classifier would achieve an accuracy of 0.338 which is the label frequency for LINSI, the highest for the dataset ALL (see Table 6.4). Compared to these values, the accuracies obtained by **SLAP** shown in Table 8.1 are much higher, indicating that we are achieving positive learning (learning in the direction we need) in selecting the "closest to optimal" alignment.

8.2 Improvement in average alignment quality

In this section, we evaluate the extent to which **SLAP** achieved the goal of the underlying machine learning problem, which is to improve the quality of the alignments obtained over using a single program on all sequences sets of the dataset.

We used CSS to determine the label for each instance in the test datasets. As described in Section 3.2.3, CSS is a measure of alignment accuracy where the higher the CSS, the higher the quality of the alignment is. We assign the program that created the alignment with maxCSS as the label for that sequence set. If **SLAP** had achieved 100% accuracy, the alignment with maxCSS for each sequence set would have been selected, resulting in the maximum average CSS for the dataset. In order to evaluate the improvement gained using **SLAP** compared to the maximum average CSS for each individual alignment program on each test dataset, we need to calculate the following quantities for the comparison:

- the maximum average CSS possible quality for each test group

- the average CSS using **SLAP** for each test group
- the average CSS using a single alignment program for each test group, for each alignment program

8.2.1 Calculating the maximum possible quality

We define BEST to be the average of all maxCSS using all five programs for each test dataset (see Section 6.2.4). BEST can be used to represent the maximum average CSS possible for a given group of sequence sets using a given group of alignment programs. In Table 8.2, the value for BEST for each test dataset is shown. A sizable difference in the BEST values can be seen from dataset to dataset. For example, between the datasets CN and RU, there is a difference of 0.155 in BEST values. This is pointed out to emphasize the variation present in this machine learning problem.

8.2.2 Calculating the average quality achieved by SLAP

The average quality obtained using **SLAP** CSS_{SLAP} is calculated by averaging the CSS from all the alignments predicted by **SLAP**. As shown in green in Table 8.2, CSS_{SLAP} varies by a large amount between the test groups, from as high as 0.938 in the CN group and down to 0.782 in the RN group.

8.2.3 Calculating the average quality for each alignment program

As described in Section 6.2.4, the average CSS for each alignment program ($\text{CSS}_{\text{program}}$) is calculated on each test dataset, which establishes a base for comparing the improvement achieved by **SLAP** over the use of a single alignment program. As shown in Table 8.2, the relative performance between the alignment programs shifts between the test groups (the highest $\text{CSS}_{\text{program}}$ within each test group is shown in blue). For

Table 8.2: Average CSSs for each test dataset. BEST is the average maxCSS and CSS_{SLAP} (shown in green) is the average CSS for the alignments predicted by **SLAP**. The average CSS for each alignment program on each sequence set is given with the largest value per of $CSS_{program}$ shown in blue.

	ALL	FU	FN	CU	CN	RU	RN
BEST	0.853	0.806	0.853	0.811	0.943	0.788	0.919
CSS_{SLAP}	0.848	0.801	0.849	0.806	0.938	0.782	0.915
CSS_{MUSCLE}	0.830	0.776	0.827	0.783	0.931	0.760	0.906
CSS_{LINSI}	0.833	0.782	0.832	0.784	0.930	0.764	0.905
CSS_{PROB}	0.805	0.754	0.801	0.762	0.904	0.733	0.878
CSS_{CLTW2}	0.832	0.779	0.829	0.799	0.924	0.762	0.902
CSS_{OMEGA}	0.808	0.760	0.805	0.760	0.911	0.730	0.881

example, in the CU group, CLTW2 has the highest CSS, which is larger by 0.015 than the next highest program, LINSI. However, in other groups, LINSI often shows the highest CSS value but only slightly ahead of the second best program.

8.2.4 Comparing the quality of alignments generated by SLAP with those generated by each single alignment program

Table 8.2., shows clearly that in all cases, CSS_{SLAP} is higher (closer to BEST) than any of CSS_{MUSCLE}, CSS_{LINSI}, CSS_{PROB}, CSS_{CLTW2} or CSS_{OMEGA}. Regardless of which alignment program achieved the highest CSS value, CSS_{SLAP} was smaller, indicating a gain in average quality. The difference between BEST and CSS_{SLAP} (ΔBP(SLAP)) across all groups are noticeably smaller than the differences obtained with any of the single programs by factors ranging from 2.4 to 11 (Table 8.3 and Figure 8.1)

The difference between CSS_{SLAP} and each CSS_p, defined as $\Delta SP(p)$, where p is the alignment program name, is shown in Table 8.4. $\Delta SP(p)$ ranges from between 0.007 to 0.052 across all datasets. Again, the program that achieves the lowest $\Delta SP(p)$ changes with the dataset used. Table 5.2 shows the results of direct comparisons of CSS values of the alignments obtained by SLAP and each program. Alignments chosen by SLAP

had clearly and significantly more often larger CSS values compared to those obtained by each individual program ($p < 10^{-15}$ for all comparisons by the one-tailed sign test). It should be noted that when comparing against a program p, if the maximum CSS is given by this program and SLAP correctly chose the alignment generated by this program, ΔSP(p) becomes 0. Therefore, both ΔSP(p)=0 and ΔSP(p)>0 should be counted as "success". However, in the implementation of the sign test we used, "ties" (ΔSP(p)=0) are not counted as "success". Even with such "unfair" comparisons, SLAP performed highly significantly better than any single program for all datasets.

These results showed that in spite of the relatively low accuracy of the prediction by the **SLAP** classifier (Table 8.1), clearly a significant improvement in average CSS was achieved based on the CSS values using **SLAP** over using individual programs.

Table 8.3: The difference between BEST and average CSS values using **SLAP** and five alignment programs. ΔBP(SLAP) is shown in green. The smallest difference (ΔBP(p)) when using a single program is shown in blue.

	ALL	FU	FN	CU	CN	RU	RN
ΔBP($SLAP$)	0.005	0.005	0.004	0.005	0.005	0.006	0.004
ΔBP($MUSCLE$)	0.023	0.030	0.025	0.028	0.013	0.028	0.013
ΔBP($LINSI$)	0.021	0.024	0.021	0.027	0.014	0.024	0.014
ΔBP($PROB$)	0.048	0.052	0.052	0.049	0.040	0.055	0.042
ΔBP($CLTW2$)	0.021	0.027	0.024	0.012	0.019	0.026	0.017
ΔBP($OMEGA$)	0.046	0.047	0.047	0.051	0.032	0.058	0.038

Table 8.4: ΔSP(p) values for seven test datasets. The smallest ΔSP(p) is shown in blue while the largest ΔSP(p) is shown in green.

	ALL	FU	FN	CU	CN	RU	RN
ΔSP(MUSCLE)	0.018	0.025	0.021	0.024	0.007	0.022	0.009
ΔSP(LINSI)	0.016	0.019	0.017	0.022	0.008	0.018	0.010
ΔSP(PROB)	0.043	0.047	0.048	0.045	0.034	0.049	0.038
ΔSP(CLTW2)	0.016	0.022	0.020	0.008	0.014	0.020	0.013
ΔSP(OMEGA)	0.041	0.041	0.043	0.046	0.027	0.052	0.034

sheet statics

Table 8.5: Pairwise comparisons of maxCSS(id) and CSS(id, p). CSS values are compared between the alignment chosen by SLAP and the one generated by each of the five programs. "equal", "SLAP", and "program" show the numbers of times of CSS(id, SLAP) = CSS(id, p), CSS(id, SLAP) > CSS(id, p), and CSS(id, SLAP) < CSS(id, p), respectively. The one-tailed sign test was performed using the SIGN.test function from the R library, "Basics Statistics and Data Analysis" (BSDA). For all comparisons, $p < 10^{-15}$.

ALL	equal	**SLAP**	program
MUSCLE	32,201	88,254	23,545
LINSI	65,182	60,419	18,399
CLTW2	46,589	83,144	14,267
PROB	10,922	119,740	13,338
OMEGA	3,387	127,280	13,333
FU	equal	**SLAP**	program
MUSCLE	3,958	16,508	3,534
LINSI	10,088	11,216	2,696
PROB	2,428	19,215	2,357
CLTW2	7,049	14,710	2,241
OMEGA	1,152	20,883	1,965
FN	equal	**SLAP**	program
MUSCLE	4,033	15,865	4,102
LINSI	12,432	9,308	2,260
PROB	1,861	20,033	2,106
CLTW2	6,476	15,219	2,305
OMEGA	527	21,508	1,965
CU	equal	SLAP	program
MUSCLE	3,923	16,699	3,378
LINSI	7,605	13,554	2,841
PROB	1,058	20,821	2,121
CLTW2	11,510	10,109	2,381
OMEGA	483	21,824	1,693
CN	equal	**SLAP**	program
MUSCLE	4,698	13,042	6,260
LINSI	14,680	6,139	3,181
PROB	2,165	19,227	2,608
CLTW2	7,317	14,035	2,648
OMEGA	1,210	19,420	3,370
RU	equal	**SLAP**	program
MUSCLE	3,168	16,381	4,451
LINSI	10,604	10,473	2,923
PROB	1,809	20,051	2,140
CLTW2	9,023	13,010	1,967
OMEGA	377	21,854	1,769
RN	equal	**SLAP**	program
MUSCLE	9,133	10,856	4,011
LINSI	10,524	8,781	4,695
PROB	1,771	19,977	2,252
CLTW2	6,070	15,109	2,821
OMEGA	854	20,864	2,282

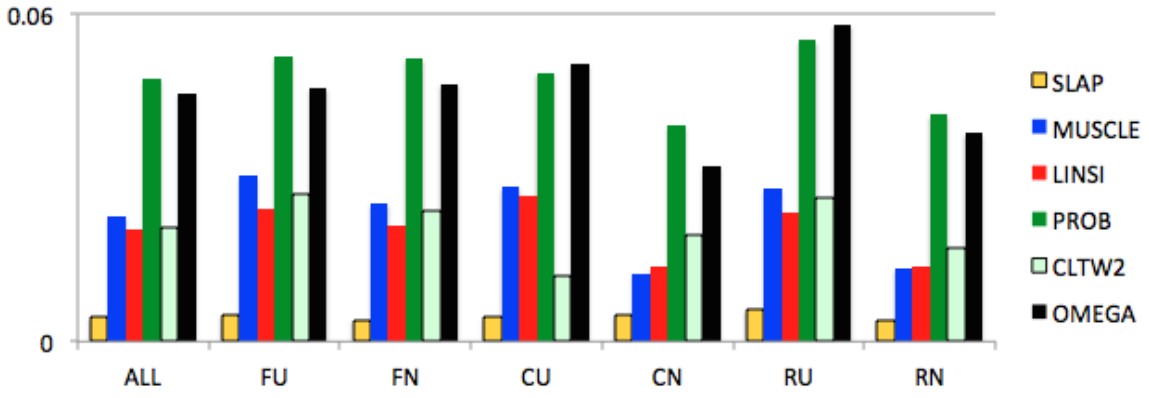


Figure 8.1: Difference in CSS values between BEST and CSS_{SLAP} as well as between the average CSS obtained by each program for each test group.

8.3 Effect of $\Delta 12$ on SLAP accuracy

In this section, we will discuss a probable cause for the low accuracy of the multi-class classifier used in **SLAP** , or more specifically, why the correct label is not more frequently predicted. Because we are using a simulated benchmark dataset, there is no ambiguity in the assignment of each label for each sequence set. However, there is a large variation in the CSS score on which the label is based. This variation is seen in Table 6.1, in both the range of CSS from sequence set to sequence set and in the variation in the CSS values between the five alignments generated from the same sequence set. This large variation is the nature of the multiple sequence alignment problem and represents the main challenge in determining optimality in the resulting alignments.

As we also presented in Section 6.3, for many sequence sets, differences in CSS values (e.g., $\Delta 12$) between the alignment programs are very small. Such small differences in CSS values can contribute to the low accuracy of the classifier. This means that in quite a few cases, the classifier is attempting to "split very fine hairs". To quantify this difference, we examine the variables $\Delta 12$, $\Delta 23$, $\Delta 34$, and $\Delta 45$ (defined in Section 6.2.4). Table 8.6 shows the average values of these Δ variables for each test dataset. Note that the standard deviation for these Δ variables is very large. In fact, they are of the same magnitude or larger than that of the average scores, indicating that there is often no difference in CSS values between the ranks.

As an initial evaluation of the effect that $\Delta 12$ might have on the accuracy of the classifier used in **SLAP**, we separated the test datasets into two disjoint subsets, instances that were correctly classified and instances that were not. We then calculated the average $\Delta 12$ for each of the two subsets for each dataset. As shown in Table 8.7, the average $\Delta 12$ for correctly predicted instances is almost twice that of the average

Table 8.6: Comparison of average Δ variables for each test dataset. SD: standard deviation of each variable; max.: the maximum value.

	ALL	FU	FN	CU	CN	RU	RN
Δ12	0.017	0.020	0.018	0.020	0.012	0.021	0.012
SD	0.024	0.026	0.027	0.024	0.020	0.025	0.020
max.	0.400	0.400	0.327	0.285	0.228	0.270	0.230
Δ23	0.014	0.016	0.015	0.014	0.011	0.017	0.012
SD	0.020	0.022	0.021	0.016	0.019	0.021	0.018
max.	0.340	0.280	0.238	0.179	0.209	0.255	0.340
Δ34	0.016	0.016	0.017	0.016	0.012	0.020	0.013
SD	0.023	0.022	0.025	0.021	0.020	0.026	0.021
max.	0.333	0.269	0.333	0.190	0.261	0.295	0.258
Δ45	0.015	0.018	0.017	0.014	0.013	0.018	0.012
SD	0.019	0.020	0.020	0.016	0.019	0.021	0.016
max.	0.231	0.247	0.229	0.225	0.236	0.332	0.160

Δ12 for incorrectly predicted instances for all test datasets. The same observation can be made of the standard deviation and to a lesser extent for the maximum values. This strongly suggests that our data model and learning algorithm combination have more difficulty in correctly predicting a label as Δ12 decreases.

Table 8.7: Average Δ12 when **SLAP** predictions were correct and incorrect. SD: standard deviation. max.: the maximum value.

Δ12		ALL	FU	FN	CU	CN	RU	RN
average	correct	0.022	0.026	0.024	0.025	0.017	0.025	0.016
	wrong	0.010	0.012	0.009	0.011	0.008	0.013	0.007
SD	correct	0.028	0.030	0.030	0.027	0.023	0.028	0.024
	wrong	0.014	0.015	0.014	0.013	0.015	0.016	0.013
max.	correct	0.400	0.400	0.327	0.285	0.228	0.270	0.230
	wrong	0.226	0.226	0.181	0.161	0.206	0.180	0.182

To further investigate the effect Δ12 has on accuracy, we divided the test datasets into four intervals based on their Δ12 values as enumerated in Table 8.8. The average accuracy for each group was then calculated with the results shown in Table 8.9. The average accuracy of the classifier increases by a considerable amount in the higher

Table 8.8: Number of test datasets included in each Δ12 interval group. For each interval listed, the upper limit is inclusive.

Δ12 intervals	ALL	FU	FN	CU	CN	RU	RN
> 0.020	40,698	8,056	7,091	7,989	4,514	8,387	4,661
0.01-0.02	23,366	4,247	3,811	4,709	2,736	4,555	3,308
0.005 - 0.01	19,223	3,361	3,110	3,726	2,597	3,380	3,049
≤ 0.005	60,713	8,336	9,988	7,576	14,153	7,678	12,982
dataset total	144,000	24,000	24,000	24,000	24,000	24,000	24,000

Table 8.9: Accuracy of **SLAP** for each dataset based on Δ12 interval. The average SLAP accuracy was calculated from all test instances with 12 values between the intervals listed where the upper limit is inclusive.

Δ12 intervals	ALL	FU	FN	CU	CN	RU	RN
> 0.020	0.822	0.830	0.860	0.871	0.721	0.801	0.790
0.01 - 0.020	0.762	0.771	0.803	0.800	0.670	0.741	0.732
0.005 - 0.01	0.649	0.727	0.762	0.748	0.639	0.699	0.694
≤ 0.005	0.507	0.508	0.517	0.494	0.509	0.500	0.491

Δ12 intervals. In the lowest interval (Δ12 ≤ 0.005), the average accuracy is the lowest for all datasets. This lowest interval group contains between 32% and 59% of the test data depending on the dataset. The average accuracy for both of the test datasets CU and RN drops below 0.50 in this lowest interval (Δ12 ≤0.005).

The datasets CN and RN have the largest proportions of test instances that fall within the lowest bracket Δ12 group (Δ12 ≤0.005), specifically 0.580 (14253/24000) and 0.54 (12,982/24000), respectively. They also have the lowest overall average accuracy, 0.524 and 0.564, respectively as shown in Table 8.1. On the other hand, CU, which has the smallest proportion of test instances in the lowest Δ12 group, 0.316 (7576/24000) has the highest overall accuracy, 0.658, among all test datasets. This shows that the proportion of test instances that have Δ12 ≤ 0.005 influences the classifier accuracy.

8.4 Alternative metric for measuring performance

We have shown that the accuracy for **SLAP** decreases as $\Delta 12$ decreases. As $\Delta 12$ decreases, the difference in quality between the top two alignments decreases resulting in the choice between the two becoming less critical. Thus, if $\Delta 12$ is sufficiently small, an incorrect prediction of the second highest CSS would have a negligible effect on the average quality. Looking at this from the viewpoint of a single sequence set, the closer the CSS are between the two top alignments, the less critical the choice between the two alignments is. We believe that this is the main reason that in spite of the low prediction accuracy (averaging around 0.610), **SLAP** was able to achieve a significant improvement in average alignment quality.

To better gauge the performance of **SLAP** to see how the predictions of **SLAP** affect the improvement of the average CSS, we introduce a new set of metrics. They are referred to collectively as the ratio variables and are defined as:

- SLAP_RAT - the ratio of the CSS from the alignments predicted by **SLAP** and the maxCSS
- RAT_MUSCLE - the ratio of the CSS from the alignment produced by MUSCLE and the maxCSS
- RAT_LINSI - the ratio of the CSS from the alignment produced by LINSI and the maxCSS
- RAT_PROB - the ratio of the CSS from the alignment produced by PROB and the maxCSS
- RAT_CLTW2 - the ratio of the CSS from the alignment produced by CLTW2 and the maxCSS

- RAT_MUSCLE - the ratio of the CSS from the alignment produced by OMEGA and the maxCSS

These can be formally expressed using the CSS measures defined in Section 8.1:

$$\text{SLAP\_RAT(id)} = \frac{\text{CSS}(id, SLAP)}{\text{maxCSS}(id)}, \tag{8.2}$$

$$\text{RAT\_MUSCLE(id)} = \frac{\text{CSS}(id, MUSCLE)}{\text{maxCSS}(id)}, \tag{8.3}$$

$$\text{RAT\_LINSI(id)} = \frac{\text{CSS}(id, LINSI)}{\text{maxCSS}(id)}, \tag{8.4}$$

$$\text{RAT\_PROB(id)} = \frac{\text{CSS}(id, PROB)}{\text{maxCSS}(id)}, \tag{8.5}$$

$$\text{RAT\_CLTW2(id)} = \frac{\text{CSS}(id, CLTW2)}{\text{maxCSS}(id)} \quad and \tag{8.6}$$

$$\text{RAT\_OMEGA(id)} = \frac{\text{CSS}(id, OMEGA)}{\text{maxCSS}(id)}, \tag{8.7}$$

where id is the identification for the specific sequence for which the ratio variable is being calculated. It should be evident that when the CSS(id, p) value is equal to the maxCSS(id), the associated ratio variable will equal 1. Table 8.10 illustrates an example of these ratio variables. In this example, **SLAP** predicted LINSI, which had the second highest CSS and as such, was not the label for the sequence set (OMEGA was the label). The calculation for SLAP_RAT for this sequence set is given by

Table 8.10: Example of the use of ratio variables using sequence set E32B2L8_1. In this example **SLAP** predicts the second highest CSS of the five produced by LINSI, when the highest CSS was from the alignment produced by OMEGA.

	CSS	ratio variable	difference with label
SLAP	0.63953	0.9994	0.0006
OMEGA	0.63989	1	0
LINSI	0.63953	0.9994	0.0006
PROB	0.60384	0.9437	0.0563
MUSCLE	0.58727	0.9178	0.0822
CLTW2	0.57319	0.8958	0.1042

$$\text{SLAP\_RAT} = \frac{\text{CSS}(\, E32B2L8\_1,\ LINSI\,)}{\text{maxCSS}(\, E32B2L8\_1\,)} = \frac{0.6396}{0.6399} = 0.9994 \qquad (8.8)$$

In this example, RAT_SLAP demonstrates that in spite of the prediction not agreeing with the label, it still predicted a program that had a CSS within 0.00036 of the maxCSS.

Table 8.11 summarizes the ratio variables for each test dataset. The ratio variables for specific programs correspond to the average performance of each alignment program when it is chosen for all sequence sets in the group. This example also shows the large difference between the CSS of the alignments in third, fourth and fifth place and those in the first and second demonstrating the gap in performance between the first two and the rest of the alignment programs. In spite of the low prediction accuracy from the classifier (between 0.58 and 0.64 as shown in Table 8.1), SLAP is again shown to predict the alignment that is on average within 99.1% of the maximum CSS.

8.5 Summary of the performance of SLAP with simulated datasets

We have evaluated the performance of the classifier used in **SLAP** by testing it on seven simulated test datasets taken from SimDom. We have found that the accuracy

Table 8.11: Ratio variables for all test datasets using SimDom. SLAP_RAT is shown in green font. The highest ratio of the single alignment programs is shown in blue font.

	ALL	FU	FN	CU	CN	RU	RN
SLAP_RAT	0.994	0.993	0.993	0.994	0.994	0.991	0.995
RAT_MUSCLE	0.968	0.955	0.955	0.960	0.986	0.958	0.983
RAT_LINSI	0.972	0.964	0.964	0.963	0.984	0.965	0.983
RAT_PROB	0.940	0.933	0.933	0.937	0.955	0.928	0.951
RAT_CLTW2	0.975	0.967	0.967	0.985	0.979	0.964	0.981
RAT_OMEGA	0.942	0.939	0.939	0.935	0.964	0.922	0.955

of this classifier, while being much better than that of random selection (0.20) or that of simply using the program (LINSI) that produces the label most frequently (0.33), is still between 0.524 and 0.658. We have also shown that in spite of this low accuracy, **SLAP** has achieved a significant improvement in alignment quality as measured by CSS_{SLAP}.

Furthermore, we have shown that the source of the low accuracy of the classifier is in part due to the large portion of each dataset having $\Delta 12$ values less than 0.005. By way of the newly defined ratio variables, we show that the alignments obtained by SLAP capture between 0.991 and 0.994 of the maximum CSS available. This is the bases for SLAP achieving the large improvement in average alignment quality in spite of the low classifier accuracy.

In essence, the behavior of **SLAP** corresponds to the needs of the underlying problem where we want to select the alignment that is closest to the optimal. When sequence sets are closely related, the alignments formed by the different alignment programs are close in accuracy and thus have similar CSS values. In these cases, the choice between the individual alignment is less critical than when there is a large difference between CSS values. We have shown that with the test cases where alignments are close, **SLAP** has low predictive accuracy.

However, when the sequence sets to be aligned contain sequences that are more distantly related, the alignments created by the five alignment programs tend to disagree more as the shift in relative performance is manifested. This represents instances where the choice between alignments is critical and where the most improvement on average quality can be made. These are also the instances on which the classifier has greater accuracy. This trend in the accuracy of **SLAP** results in a substantial improvement in average alignment in spite of the low classifier accuracy.

We have also shown by breaking SimDom into different test datasets the there is a shift in relative performance demonstrated by different alignment programs achieving the maxCSS. This illustrates the advantage of using **SLAP**.

8.6 Performance analysis of SLAP using non-simulated data

8.6.1 Non-simulated databases used

In this section, we will discuss the results of using **SLAP** on non-simulated benchmark datasets. The non-simulated benchmark databases we chose are BB3, which is the most frequently used benchmark database for alignment program evaluation studies, as well as HOM and OX3 (all three are discussed in Section 3.3). Table 8.12 shows the number of sequence sets we used from each of the datasets, as well as the frequency of the label for each alignment program.

The distribution of the label again shows the relative performance difference between the programs. Each dataset has a different alignment program that performed the best (LINSI, CLTW2 and PROB for BB3, HOM and OX3, respectively). It should also be noticed that in both BB3 and HOM there are only two programs that make up the majority of the labels: For the BB3 dataset, LINSI and OMEGA make up 99.7% of the labels. For the HOM dataset, CLTW2 and OMEGA make up 92.%

Table 8.12: Frequency of the labels for non-simulated benchmark datasets BB3, HOM and OX3. The frequency of the labels for each program for each dataset is given in parentheses.

	BB3	HOM	OX3
	count(freq.)	count(freq.)	count(freq.)
total seq. sets	608 (1.000)	402 (1.000)	399 (1.000)
MUSCLE	0 (0.000)	1 (0.002)	42 (0.105)
LINSI	379 (0.623)	1 (0.002)	31 (0.078)
PROB	2 (0.003)	27 (0.067)	198 (0.496)
CLTW2	0 (0.000)	191 (0.475)	51 (0.128)
OMEGA	227 (0.373)	182 (0.453)	77 (0.193)

of the labels. For the OX3 dataset, roughly 49.6% of the labels have been assigned to PROB while the remaining labels are distributed among the other four programs. The ratio of the lowest frequency to highest frequency in each database is 0.0000, 0.0052, and 0.1566 for BB3, HOM and OX3, respectively.

For comparison, SimDom has the label frequencies of 0.338, 0.291, 0.233, 0.094 and 0.04.5 for LINSI, CLTW2, MUSCLE, PROB, and OMEGA, respectively (Table 6.4). This represents a ratio of the lowest frequency to highest of 0.1323. However, with the total number of alignments being 144,000, the label with the lowest frequency (OMEGA) still has 6,440 instances, giving the learning algorithm much more information to train on, rendering it better able to identify the instance where OMEGA is the label.

We performed analysis using **SLAP** on ten ten-fold cross-validation tests for resulting in 100 training and testing runs for each non-simulated dataset. We used the same metrics we used with the simulated datasets for this analysis.

8.6.2 Accuracy of SLAP prediction on non-simulated datasets

Table 8.13 summarizes the accuracies of the **SLAP** classifier. The same trend as described for the simulated datasets are apparent: the accuracy increases as the $\Delta 12$

value increases. In both BB3 and HOM, the accuracies (0.851 and 0.727, respectively) are much higher than what was achieved with the simulated data. This increase in classifier accuracy can be explained by the larger average $\Delta 12$ for BB3 and HOM, (0.092 and 0.070, respectively) as opposed to the much smaller $\Delta 12$ (0.017) for SimDom. This resulted from a much lower portion of sequence in both BB3 and HOM having $\Delta 12 \leq 0.005$ (0.039 and 0.221, respectively). In the SimDom database, 0.421 of the sequence sets have $\Delta 12 \leq 0.005$. Furthermore, the fact that the label distribution in both BB3 and HOM effectively reduces the learning problem with these datasets to a simpler binary problem could also contribute to the higher accuracy in BB3 and HOM (Table 8.12). However, in OX3, where 59% of the sequence sets have $\Delta 12 \leq 0.005$ and the average $\Delta 12$ is 0.010, **SLAP** showed a much lower accuracy (0.450).

Table 8.13: Accuracies of **SLAP** with the average $\Delta 12$ value for non-simulated datasets. SD: standard deviation.

	BB3	HOM	OX3
Accuracy	0.851	0.727	0.452
SD	(0.044)	(0.075)	(0.091)
$\Delta 12$	0.092	0.070	0.010
SD	(0.114)	(0.116)	(0.019)
proportion of dataset with $\Delta 12 \leq 0.005$	0.039	0.221	0.590

8.6.3 Comparison of the average performance by SLAP against individual programs

In order to determine the improvement in the average alignment quality achieved by **SLAP** over any of the individual alignment programs, we compared BEST and $\mathrm{CSS}_{\mathrm{program}}$ obtained by **SLAP**and each of the five alignment programs.

As shown in Table 8.14, when using BB3, **SLAP** achieved the average CSS that was 0.033 higher than the highest performing individual program, OMEGA. **SLAP** also had a ΔBP of only 0.007 while OMEGA had a ΔBP of 0.040. It should be noted with BB3, while LINSI had a higher frequency of the label than OMEGA (Table8.12), $\text{CSS}_{\text{OMEGA}}$ was larger than the average CSS values obtained using other programs including LINSI. This resulted because when OMEGA did achieve maxCSS, there was a larger difference between CSS(id,OMEGA) and CSS(id,LINSI) than when LINSI achieved maxCSS.

Table 8.14: Average CSS values for non simulated datasets. ΔBP(p) and ΔSP(p) are shown in parentheses and square brackets, respectively. CSS_{SLAP} is shown in green and the largest CSS among those obtained by a single program is shown in blue.

	BB3			HOM			OX3		
	CSS	(ΔBP)	[ΔSP]	CSS	(ΔBP)	[ΔSP]	CSS	(ΔBP)	[ΔSP]
BEST	0.920			0.971			0.902		
CSS_{SLAP}	0.913	(0.007)		0.966	(0.005)		0.885	(0.017)	
CSS_{MUSCLE}	0.682	(0.238)	[0.231]	0.833	(0.138)	[0.133]	0.881	(0.020)	[0.004]
CSS_{LINSI}	0.867	(0.053)	[0.046]	0.830	(0.141)	[0.136]	0.873	(0.029)	[0.012]
CSS_{PROB}	0.724	(0.196)	[0.189]	0.847	(0.124)	[0.119]	0.885	(0.017)	[0.000]
CSS_{CLTW2}	0.654	(0.266)	[0.259]	0.913	(0.058)	[0.053]	0.879	(0.022)	[0.006]
CSS_{OMEGA}	0.880	(0.040)	[0.033]	0.963	(0.008)	[0.003]	0.876	(0.025)	[0.009]

When using HOM, **SLAP** again achieved a higher average CSS over the closest single alignment program, OMEGA. However, this was a much smaller improvement (ΔSP = 0.003) $\text{CSS}_{\text{OMEGA}}$. ΔBP for **SLAP** and OMEGA were 0.005 and 0.008, respectively. The improvement in alignment quality using **SLAP** with HOM was not as large as it was when using BB3. This could be attributed to the higher portion of datasets with $\Delta 12 \leq 0.005$ and therefore the lower average Δ12 value for the dataset. Although OMEGA was not the program with the highest label frequency for the HOM dataset it again achieved a higher average CSS than CLTW2, which was the highest performing alignment program for this dataset (Table 8.12).

Using OX3, the highest performing alignment program was PROB. **SLAP** achieve

higher ΔBP value than individual alignment programs, with the exception of PROB, where **SLAP** and PROB tied ($CSS_{SLAP} = CSS_{PROB}$). However, it is notable that while **SLAP** did not improve the average CSS, it also did not decrease it. This lower improvement using **SLAP** is due to the much lower classifier accuracy (0.451), which can be linked to the low average Δ12 (0.001) resulting from the higher portion of sequence sets that have $\Delta 12 \leq 0.005$.

The average CSS values across all datasets show a large shift in relative performance within and between the datasets. The most extreme example is between LINSI and CLTW2. In BB3, LINSI had the second highest average CSS with a ΔBP = 0.053, while CLTW2 had the lowest CSS with ΔBP = 0.266. However, in HOM, LINSI had the lowest average CSS with ΔBP = 0.141, while CLTW2 had the highest average CSS with ΔBP = 0.008. In OX3, both LINSI and CLWT2 had comparable CSS and ΔBP, although neither achieved the best CSS, which was achieved by PROB. This once again shows that different alignment programs perform differently on different sequence sets.

A shift in the relative performance of the alignment programs can most effectively be seen in Figure 8.2. In BB3, the definite improvement in alignment quality when using **SLAP** can be seen in the small difference from BEST. In HOM, there is still a distinct improvement, although the best single program, OMEGA, is not as far behind **SLAP**. In the dataset OX3, the best performing program, PROB, is tied with **SLAP** while the difference with other programs is also small. This shift in relative performance demonstrates the advantage of using **SLAP** to select the best alignment.

Table 8.15 shows the results of direct pairwise comparisons of CSS values between the alignments obtained by **SLAP** and each of the individual programs. Significantly more often alignments chosen by **SLAP** had larger CSS values compared to those

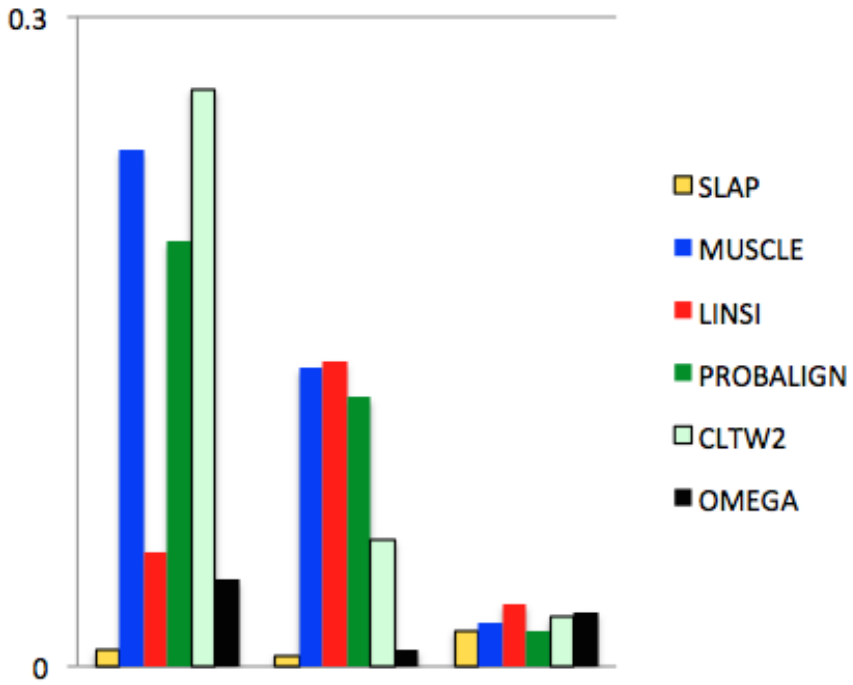


Figure 8.2: Comparison of the average ΔBP between SLAP and individual programs. See Table 8.14 for the actual ΔBP values.

obtained by each individual program ($p < 10^{-15}$) for all comparisons by the one-tailed sign test) except for the comparison against PROB when used with the OX3 database. As noted before, the implementation of the sign test we used does not count "ties" ($\Delta SP(p) = 0$) as "success", and PROB has the largest number of ties especially with the OX3 datasets. If both $\Delta SP(p) = 0$ and $\Delta SP(p) = 0$ are counted as "success", all success rates become comparable including PROB. Note also the small average $\Delta 12$ for OX3 (Table 8.13). Because the amount of improvement that is possible using **SLAP** is based on the extent of the relative performance difference, when there is not a large shift in relative performance, indicated by low $\Delta 12$, using **SLAP** will not result in a large improvement in average CSS.

Table 8.15: Pairwise comparisons of CSS(id, SLAP) and CSS(id, p) values. CSS values are compared between the alignment chosen by **SLAP** and the one generated by each of the five programs. "equal", "SLAP", and "program" show the numbers of times of CSS(id, SLAP) = CSS(id, p), CSS(id, SLAP) > CSS(id, p), and CSS(id, SLAP) < CSS(id, p), respectively. "p-value" is based on the one-tailed sign test, which was performed using the SIGN.test function from the R library BSDA.

BB3	p-value	equal	success	program
MUSCLE	3.331E-16	20	6041	19
LINSI	2.200E-16	3849	1808	423
PROB	3.331E-16	20	6034	26
CLTW2	3.331E-16	20	6041	19
OMEGA	2.200E-16	2251	3367	462

HOM	p-value	equal	success	program
MUSCLE	2.200E-16	247	3697	76
LINSI	2.200E-16	182	3779	59
PROB	2.200E-16	383	3558	79
CLTW2	2.200E-16	2032	1484	504
OMEGA	2.200E-16	2215	1345	460

OX3	p-value	equal	success	program
MUSCLE	2.200E-16	1287	1845	858
LINSI	2.200E-16	1035	2085	870
PROB	1.000E+00	2637	603	750
CLTW2	2.200E-16	1173	1887	930
OMEGA	2.200E-16	1499	1469	1022

8.6.4 Ratio variables using SLAP

We next calculated the ratio variables for each benchmark dataset. The results are shown in Table 8.16. These results mirror the trends in the $\text{CSS}_{program}$ values, with the exception that **SLAP** achieved a higher value for SLAP_RAT than that of RAT_PROB when using OX3. This augments the observation that while there was not an improvement in average CSS, there is also not a decrease in average CSS.

Table 8.16: Ratio variables for all non-simulated datasets. SLAP_RAT values are shown in green font. The highest ratio value for the individual alignment programs is shown in blue font.

	BB3	HOM	OX3
SLAP_RAT	0.991	0.995	0.975
RAT_MUSCLE	0.729	0.854	0.970
RAT_LINSI	0.939	0.852	0.952
RAT_PROB	0.778	0.869	0.972
RAT_CLTW2	0.699	0.937	0.971
RAT_OMEGA	0.955	0.992	0.959

8.6.5 Summary of the performance of SLAP with non-simulated datasets

SLAP achieved a significant improvement overall single alignment programs when used with the non-simulated databases BB3 and HOM. The accuracy (0.851 and 0.727, respectively) was higher than those (0.61) obtained with the simulated datasets. The improvement in terms of CSS values ranged from 0.033 to 0.259 in BB3 and between 0.003 and 0.136 in HOM. In both BB3 and HOM, the average $\Delta 12$ (0.092 and 0.070, respectively) was higher than in SimDom, and the proportions of sequence sets in BB3 and HOM that had $\Delta 12 < 0.005$ (3.9% and 22.1%, respectively) were much less than those with SimDom. These two factors contributed to the highest classifier accuracy achieved by **SLAP**. In fact, in BB3, which had a highest average $\Delta 12$ and smallest portion of sequence sets with $\Delta 12 < 0.005$, **SLAP** achieved both the higher classi er accuracy and the highest improvement in average CSS among the non-simulated datasets tested.

When used with OX3, **SLAP** achieved only a 0.452 accuracy. However, it showed significant improvement over four of the alignment programs, and tied with PROB. OX3 had an average $\Delta 12$ of 0.0100, which is significantly lower than SimDom. OX3 also had a much higher proportion of sequence sets with $\Delta 12 < 0.005$ (59.1%). The

results on these three non-simulated databases support the hypothesis that low $\Delta 12$ contributes to the low **SLAP** classifier accuracy.

8.6.6 Discussion

The positive results achieved by **SLAP** on both the simulated database (SimDom) and the non-simulated databases (BB3, HOM and OX3) indicate that a significant improvement in average CSS can be achieved by training a multi-class classifier to select the alignment closest to the optimal. The extent of improvement that can be achieved depends on two factors: the accuracy of the **SLAP** classifier and the amount of difference in CSS(id, p) for each sequence set. This is important in that if **SLAP** had 100% accurate prediction, the amount of improvement for a specific sequence set is limited by the shift in relative performance between the alignment programs. For, example, in the case where all alignments agree, there would be no improvement in CSS whether **SLAP** predicted the correct label or not. The best combination of programs to bring together to be used with **SLAP**, therefore, would be those that perform well overall, but can strongly outperform other programs on specific types of sequence sets. For instance, we have observed that while LINSI and MUSCLE tend to outperform CLTW2 on more closely related sequence sets, CLTW2 strongly outperforms both LINSI and MUSCLE when the sequences are more divergent.

In light of these two factors: **SLAP** accuracy and actual performance difference between programs, the following areas should be investigated further to improve the performance of **SLAP**:

- Use different learning algorithm. We feel that the use of deep learning algorithms would help achieve a higher accuracy. The work done with convolution neural networks (CNN) on the problem of image recognition has achieved high

levels of success. In the image recognition program, the image is represented as an array of pixel color values. Since an MSA can very easily be represented as an array of amino acids, it is possible that comparing two alignments can make use of the advances in image recognition algorithms. Also, much work has been done to create deep learning algorithms that make use of more efficient processing of data arrays using the GPU.

- Use different combinations of alignment program. For this initial work, we selected a group of five programs based on their previous performance studies and on their availability. More programs should be investigated for our suite of alignment programs, either to take the place of one currently being used or to increase the number of programs used with **SLAP** . This would allow the more recent advances made in alignment programs to be incorporated into the **SLAP** method.

- Use different quality metric. We used CSS as our measure of accuracy and hence "quality" between two alignments. There are other metrics that are frequently used in alignment program evaluation techniques. Therefore, it would be of interest to examine how employing these metrics would affect the performance of **SLAP**.

- Use alternative objective. The objective of **SLAP** was to select the single alignment that had the highest CSS for a specific sequence set. However, as we pointed out, when the difference between the highest and the second highest program is very small, it is difficult for **SLAP** to distinguish between them. As such, a series of different machine learning problems could be designed to assist in selecting the best MSA. These include:

  - use the ANN to rank the five programs.
  - create a binary problem to choose between two alignments for the highest quality alignment and use multiple binary classifiers in a tournament selection process to see which of the group would be the best.
  - create a binary problem to answer the question, "is the difference between the five alignments greater than 0.005."

The past attempts at a solution to this problem were approached from a bioinformatics point of view, that is, to take the plethora of biologically relevant information on a group of sequences and to organize it to select the alignment program that will produce the best alignment. The approach in the development of our solution to this problem was taken from the computer science point of view. By this, we mean we used a systematic approach based on an understanding of the computational tools available and on a full analysis of the problem. An overview of this systematic approach is as follows:

- From the tools that can be used to classify entities, we identified the process of machine learning to train a multi-class classifier to be the heart of our solution.
- We addressed the factors needed for a machine learning algorithm to successfully train a classifier that will select the alignment closest to the optimum: the amount of training data, scope of training data, discerning attributes in the data model, and error free labeling.
- We determined that there were no databases that had sufficient data with sufficient scope as well as error-free labeling, on which to test our concept. To remedy this, using the most advanced protein sequence evolution simulator, which allowed for the three basic evolution events as well as the ability to model

domains, we generated a large-scale benchmark database with sufficiently divergent scope.

- Previous studies described only general trends based on the average alignment accuracy scores, and their results varied significantly. Therefore, we performed our own performance evaluation and developed a protocol for tracking not only average alignment quality, but also the performance difference between alignment programs, the extent of this performance difference, the maximum possible average quality, and the maximum improvement achievable from each alignment program. Using this system, we identified the factors that had a correlation to the best alignments which included alignment based attributes.

- After we formed an input vector based on the results of our performance evaluation test, we systematically tested the combination of attributes to find those that resulted in the highest classifier accuracy and eliminated those that did not.

- Although the resulting classifier accuracy seemed relatively low, when we analyzed the results in terms of the underlying problem, which is to improve average alignment quality, we found that the results of our solution were in line with this objective. We, in fact, achieved 99.1 - 99.5% of the alignment quality available in spite of the low classifier accuracy.

- With the proof of concept successfully performed on our simulated data, we then tried our solution on non-simulated benchmark data and achieved as much as 99.1% of the possible quality, increasing the average accuracy by 0.033 (in terms of CSS).

Using this systemized approach to the problem of selecting the alignment "closest to the optimal", we achieve a high degree of success.

8.6.7 Conclusions

In the development and testing of **SLAP** we demonstrated the following:

- It is possible to harness the relative performance difference between alignment programs to improve the alignment quality for a set of data by selecting the alignment program that created the alignment closest to the optimal.

- In non-simulated datasets where there is a distinct shift in the relative performance marked by a high average $\Delta 12$, as was the case with BB3 and HOM, **SLAP** can achieve an improvement in the average quality of the alignments selected. However, when there is not a sizable shift in relative performance resulting in low average $\Delta 12$ value, as was the case with OX3, **SLAP** will not degrade the alignment quality generated from the highest single program.

- Simulated data, specifically SimDom, created to mimic biologically realistic protein sequences with one or more functionally constrained areas, can be used to develop a method for selecting an alignment closest to the optimal. The advantages of confidence in instance label and the presence of a true alignment against which to measure the most accurate alignment, as well as the large number and scope of the sequence sets to be used as training data, allowed for the best attributes to be selected when designing the data model of the **SLAP**classifier.

- The success **SLAP** achieved on the non-simulated data sets for BB3 and HOM, along with the neutral results it achieved on OX3, demonstrate that techniques

developed using simulated data could be used with success on non-simulated data.

Chapter 9

Conclusions

Because the MSA is the basis of so many bioinformatics studies, it needs to reflect an accurate relation between the sequences it contains. With the increasing use of and dependency on automated alignment programs, it is important to assess the accuracy of a resulting alignment to ensure that it is a quality alignment. Many evaluation studies have been performed to compare the currently available alignment programs. This is an attempt to determine which program creates the alignments with the highest average accuracy. However, these studies show only average quality results and do not emphasize the shift in the performance of the programs on different sequence sets.

The overall objective of this work is to improve the quality of MSAs. We approached this on two fronts: 1) developing SuiteMSA, a user-friendly program that offers multiple alignment viewers to allow the user to visually and quantitively assess MSAs and 2) by developing **SLAP**, an algorithm that will select the alignment closest to the optimal from a group of alignments created from the same sequence set.

9.1 Summary of SuiteMSA development

Assessment of MSAs can be as simple as examining the alignment visually to assess its shape (gap pattern) or to see how the alignment agrees with any supplemental data

available on the sequences contained in the alignment, such as secondary structure, transmembrane predictions, accessibility, etc. We observed the following situation: in spite of the many programs available that display, allow for editing of and preparing images of MSAs for publication, there were no user-friendly, visual inspection tools to assist in the assessment of alignments by either comparing multiple alignments on a column-by-column basis or by examining a single MSA with the visual display of functional information such as secondary structure information, transmembrane prediction, solvent accessibility, etc.

We approached the problem by designing and implementing SuiteMSA as a visual assessment tool for MSAs that allows for the display of auxiliary data as well as displaying column-wise values of basic metrics used in the assessment of alignments such as CS, SPS, CSS and information score on a column-wise basis. SuiteMSA provides three unique alignment viewers.

- The **MSAviewer** allows for the viewing, editing, and assessing of a single alignment assisted by the display of secondary structure or transmembrane prediction data correlated with the sequences in the alignment.

- The **MSAcomparator** allows for the viewing, comparing and assessing of two MSAs with a varying amount of detail. Column-wise values, as well as the alignment averages, for commonly used metrics such as CSS, SPS, CS, and information score can be calculated and displayed.

- The **Pixel plot** allows for the viewing, comparing and assessing of multiple MSAs (2 or more) in a novel format developed specifically for the panoramic view of long alignments, the number of which is limited only by system resources such as monitor size and RAM. It allows a visualization of a larger portion of the alignment to be shown at one time, emphasizing the gap patterns. Color

schemes related to secondary structure and transmembrane prediction data can be applied to aid the visual inspection of MSAs.

SuiteMSA gives researchers the utilities, the type of which was not previously available, to perform visual quality assessment on MSAs. I have received communications from researchers and professors of bioinformatics for requests of assistance, additional features and general feedback. These communications reveal that SuiteMSA has been used in lectures discussing multiple sequence alignments and the differences between the alignments created by the various alignment programs. Students are also using SuiteMSA in their projects related to bioinformatics. SuitMSA has been published [6][7].

9.2 Summary of the SLAP "closest to optimal" classifier development

Our thesis is that by choosing the best alignment out of a group of alignments depending on properties of sequences, as opposed to simply using the alignment from the program with the highest average score across any types of sequences, would allow us to obtain the most accurate alignment for a variety of sequences sets

Our approach is novel in that it is based on the demonstrated phenomenon that alignment programs perform better or worse than others depending on the sequence set. When alignment accuracy is critical, we advocate taking advantage of strengths of various alignment programs for different alignment problems to provide the most accurate alignment. The goal is to select the best alignment from a group of alignments. This is in contrast to selecting an alignment program based the output of tie-prone binary classifiers for efficiency as done in AlexSys. Our strategy also has an advantage over limiting the choice to just two base alignment programs as done in AQUA.

9.2.1 Tasks accomplished and concerns addressed

1. Development of the simulated protein sequence database, SimDom. The concerns this addressed are as follow

   - *Existing non-simulated benchmarks are limited in size.* For non-simulated protein sequences, reference alignments are usually inferred based on the structural alignments. Due to the small number of proteins whose 3D structures are available, this limits the number of reference alignments. The largest non-simulated database, BB3, has only 603 alignments that contain more than two sequences. In contrast, SimDom currently has 144,000 true reference alignments, which can be easily expanded when needed.
   - *Existing non-simulated benchmarks are limited in scope.* Because they rely on the existing protein 3D structures, the scope of protein sequences represented by non-simulated benchmark databases is limited, and they can not provide full assessment capability for protein alignment programs. Especially a very small, if any, number of muti-domain proteins are included in the non-simulated reference databases. SimDom uses 1750 domain models in 1000 protein sequence architecture, each of which is simulated using 144 evolutionary scenarios. This provides a much larger scope of evolution patterns and sequence properties on which to evaluate alignment programs.
   - *Existing non-simulated benchmarks have no true alignments.* Non-simulated databases contain reference alignments that are educated approximations based on structural alignments. They are not guaranteed to be true alignments. The confidence is even weaker for the sections of the alignment where no 3D-strutural information is available.

- *Existing simulated databases are limited in biological realism.* The simulation program we used, REvolver, uses profile HMMs to capture functional constraints under evolution in the protein domain regions. REvolver has been shown to maintain 95% of the domain identity over time [97]. Therefore, the simulated protein sequences included in SimDom should maintain biologically realistic protein sequences.

2. Performance evaluation of alignment programs using a protocol that can be performed using any metric across all datasets.

    - *The alignment program with the highest average score does not always produce the highest scoring alignment.* We established that a relative performance shift occurs between sequence sets. We showed that not only does no single alignment program outperform all others programs, but no single alignment program consistently outperform any other program. This degree of performance shift has until now not been noted and forms the basis of our concept of choosing the best alignment for a selection of alignment formed by various alignment programs.
    - *The extent of the shift in performance between alignment programs was not tracked.* We developed a set of metrics that can quantify this shift.
        - The value of BEST establishes, quantitatively, the maximum possible average quality that is available for a specific dataset using a specific group of alignment programs.
        - ΔBP establishes the degree to which improvement can be made over the use of any single alignment program in our suite.
        - Δ12 identifies those sequence set and alignment characteristics that can serve as indicators of a shift in relative performance

3. Development of the data model and the **SLAP** (**S**e**L**ecting an **A**lignment **P**rogram) classifier.

    - *Data model needed attributes that discern the best alignment.* Using the results of the performance analysis of multiple alignment programs described above, the data model was developed. To maximize the classifier accuracy, large-scale simulated datasets from SimDom were used. Attributes of this data model included both sequence set characteristics as well as characteristics from alignments

    - ***SLAP****should generate a higher average score than any single alignment program.* Trained and tested **SLAP** on the various data sets from the database SimDom and using the alignment program evaluation protocol, analyzed the performance of the multi-class classifier which showed, even with an average accuracy of the multi-class classier **SLAP** is around 0.61, there was still an improvement of as much as 0.052 in average CSS.

    - *Traditional classifier accuracy did not adequately quantify* ***SLAP****'s performance.* We defined a new metric, SLAP_RAT that showed that, in spite of the relatively low prediction accuracy, **SLAP** achieved 99.1% or higher of the maximum accuracy possible between all five alignment programs, in spite of the relatively low prediction accuracy.

    - *SimDom was developed using simulated data.* We trained and tested **SLAP** on various non-simulated benchmark databases. For BB3, the most cited benchmark database for evaluation studies, on which **SLAP** achieved a classifier accuracy on average of .851. It resulted in an overall improvement of 0.033 in average CSS, which represents 99.1% of the maximum CSS. The results **SLAP** achieved on non-simulated benchmark datasets

demonstrated that a method developed on a simulated database can be transferred to non-simulated benchmark datasets with higher levels of success than any previous approach.

9.3 List of contributions

The overall objective of this study was to improve the quality of multiple sequence alignments. Our contributions towards this end are as follows:

- **Development of the SLAP classifier**. We provided a classifier to help identify alignment closest to the optimal. It achieved a substantial improvement in average alignment accuracy for both simulated and non-simulated protein sequences.

- **Development of SuiteMSA**. We provided a user-friendly program that contains three novel alignment viewers to assist in the visual assessment of MSA quality. These three viewers are as follows:

    1. MSAviewer. It allows graphical assessments of MSAs using functional information such as secondary structures and transmembrane prediction
    2. MSAcomparator. It allows the detailed comparison between two alignments with column-wise alignment quality and conservation scores.
    3. Pixel plot. It allows a large-scale visual comparison among multiple MSAs. It can incorporate functional information such as secondary structures and transmembrane prediction as color schemes.

- **Development of the SimDom database**. We created a large-scale simulated protein alignment benchmark database, SimDom. It specifically designed for

the evaluation of alignment programs for maximum quality. SimDom has the following advantages:

1. Availability of true (error free) reference alignments for alignment quality evaluation
2. Inclusion of a large (1000) protein types incorporating 1750 different domain models
3. Inclusion of 144 individual evolutionary scenarios per protein type
4. Inclusion of a total of 144,000 sequence sets with true alignments
5. Possibility to be easily supplemented by additional simulations

- **Development and performance of an alignment program evaluation protocol**. We developed an alignment program evaluation protocol that included a system of measures to document relative performance differences in quality between the alignment programs evaluated. Using this protocol we showed that the shift in relative performance was much larger than indicated in previous evaluation studies where only the average quality values had been used. We also revealed trends that have not been seen previously (i.e., CLTW2 can generate much better MSAs from highly divergent sequence sets compared to other MSA methods).

- **Positive impacts for downstream analyses**. Successful selection of the alignment closest to the optimum will allow for better results from downstream analyses. Therefore, the results described in this dissertation will contribute to the improvement of various bioinformatics and molecular evolutionary analyses.

9.4 Future work

This work has several natural extensions. They can be divided into four areas:

- Package **SLAP** in a user friendly program for use by the bioinformatics community.
- Explore the use of deep learning [118] to train a classifier of higher capability to improve the accuracy of selecting the most accurate alignment can be improved. Using a convolution style deep learner with software that make use of the graphics processing unit, it might be possible to increase the accuracy of the classifier.
- Augmenting SimDom with additional sequence sets by:
  - Adding alignments to the SimDom database to mimic existing non-simulated benchmarks databases. This is to allow for a variety of evolutionary scenarios included in the existing benchmarks. It should give additional evidence of over-training or over-specialization to the benchmark. This will also demonstrate the validity of training a classifier on simulated data for testing on non-simulated data.
  - Adding alignments to the SimDom database for sequences sets with additional sequence set characteristics, such as creating subgroups to contain orphaned or divergent sequences or having long insertions between domains. Such cases are available in BAliBASE.
  - Incorporating the simulated data with the actual protein sequences to fill in the gaps of the instance space. By using a series of binary and multi-class

classifiers, improve the accuracy of the selection of the alignment closest to the optimum.

- Adding sequence sets that are not represented in the current SimDom datasets, such as transmembrane proteins.
- Incorporating new developments in the field, such as including two new alignment program QuickProbs 2 [119] and PnPProbs [120] as well investigating new methods for scoring the accuracy of an MSA such as LEON_BIS [121], evalmsa [122] and secondary structure prediction accuracy (SSPA) [123].

Bibliography

[1] P. Markova-Raina and D. Petrov, "High sensitivity to aligner and high rate of false positives in the estimates of positive selection in the 12 drosophila genomes," *Genome Res*, vol. 21, no. 6, pp. 863–74, 2011.

[2] D. A. Morrison and J. T. Ellis, "Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18s rDNAs of apicomplexa," *Mol Biol Evol*, vol. 14, no. 4, pp. 428–41, 1997.

[3] G. P. Raghava, S. M. Searle, P. C. Audley, J. D. Barber, and G. J. Barton, "OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy," *BMC Bioinformatics*, vol. 4, p. 47, 2003.

[4] D. T. Jones, "Progress in protein structure prediction," *Curr Opin Struct Biol*, vol. 7, no. 3, pp. 377–87, 1997.

[5] D. A. Morrison, "Why would phylogeneticists ignore computerized sequence alignment?" *Syst Biol*, vol. 58, no. 1, pp. 150–8, 2009.

[6] C. L. Anderson, C. L. Strope, and E. N. Moriyama, "SuiteMSA: visual tools for multiple sequence alignment comparison and molecular sequence simulation," *BMC Bioinformatics*, vol. 12, p. 184, 2011.

[7] C. Anderson, C. Strope, and E. Moriyama, "Assessing multiple sequence alignments using visual tools," in *Bioinformatics - Trends and Methodologies*, M. A. Mahdavi, Ed. InTech, 2011.

[8] R. Esquivel, M. Molina-Espritu, F. Salas, J. S. Dehesa, and J. Dobado, "Decoding the building blocks of life from the perspective of quantum information," in *Advances in Quantum Mechanics*, P. Bracken, Ed. INTECH, 2013, ch. 27.

[9] "Boundless biology textbook," 2016 (accessed 12-17-2016), www.boundless.com/biology/textbooks/boundless-biology-textbook/biological-macromolecules-3/proteins-56/protein-structure-304-11437/.

[10] D. R. Maddison and K.-S. Schulz, "The tree of life web project," 2007 (accessed 11-21-2016), www.tolweb.org.

[11] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc Natl Acad Sci U S A*, vol. 89, no. 22, pp. 10 915–9, 1992.

[12] M. Dayhoff, R. M. Schwartz, and B. C. Orcutt, "A model of evolutionary change in proteins," *Atlas of Protein Sequence and Structure*, vol. 5, no. 3, pp. 555–65, 1978.

[13] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J Mol Biol*, vol. 48, no. 3, pp. 443–53, 1970.

[14] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J Mol Biol*, vol. 147, no. 1, pp. 195–7, 1981.

[15] D. F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *J Mol Evol*, vol. 25, no. 4, pp. 351–60, 1987.

[16] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res*, vol. 32, no. 5, pp. 1792–7, 2004.

[17] K. Katoh, K. Kuma, T. Miyata, and H. Toh, "Improvement in the accuracy of multiple sequence alignment program MAFFT," *Genome Inform*, vol. 16, no. 1, pp. 22–33, 2005.

[18] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins, "Clustal W and Clustal X version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–8, 2007.

[19] K. Katoh and D. M. Standley, "MAFFT multiple sequence alignment software version 7: improvements in performance and usability," *Mol Biol Evol*, vol. 30, no. 4, pp. 772–80, 2013.

[20] U. Roshan and D. R. Livesay, "Probalign: multiple sequence alignment using partition function posterior probabilities," *Bioinformatics*, vol. 22, no. 22, pp. 2715–21, 2006.

[21] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson, and D. G. Higgins, "Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega," *Mol Syst Biol*, vol. 7, p. 539, 2011.

[22] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res*, vol. 22, no. 22, pp. 4673–80, 1994.

[23] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Mol Biol Evol*, vol. 4, no. 4, pp. 406–25, 1987.

[24] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic Acids Res*, vol. 30, no. 14, pp. 3059–66, 2002.

[25] R. Sokal and C. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Science Bulletin*, vol. 38, p. 14091438, 1958.

[26] R. C. Edgar, "MUSCLE: a multiple sequence alignment method with reduced time and space complexity," *BMC Bioinformatics*, vol. 5, p. 113, 2004.

[27] ——, "Local homology recognition and distance measures in linear time using compressed amino acid alphabets," *Nucleic Acids Res*, vol. 32, no. 1, pp. 380–5, 2004.

[28] M. Kimura, *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983.

[29] G. H. Gonnet, M. A. Cohen, and S. A. Benner, "Exhaustive matching of the entire protein sequence database," *Science*, vol. 256, no. 5062, pp. 1443–5, 1992.

[30] G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins, "Sequence embedding for fast construction of guide trees for multiple sequence alignment," *Algorithms Mol Biol*, vol. 5, p. 21, 2010.

[31] J. Soding, "Protein homology detection by HMM-HMM comparison," *Bioinformatics*, vol. 21, no. 7, pp. 951–60, 2005.

[32] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler, "Hidden markov models in computational biology. applications to protein modeling," *J Mol Biol*, vol. 235, no. 5, pp. 1501–31, 1994.

[33] R. Durbin, *Biological sequence analysis : probabalistic models of proteins and nucleic acids*. Cambridge, UK New York: Cambridge University Press, 1998.

[34] E. L. Sonnhammer, S. R. Eddy, and R. Durbin, "Pfam: a comprehensive database of protein domain families based on seed alignments," *Proteins*, vol. 28, no. 3, pp. 405–20, 1997.

[35] M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn, "The Pfam protein families database," *Nucleic Acids Res*, vol. 40, no. Database issue, pp. D290–301, 2012.

[36] R. D. Finn, P. Coggill, R. Y. Eberhardt, S. R. Eddy, J. Mistry, A. L. Mitchell, S. C. Potter, M. Punta, M. Qureshi, A. Sangrador-Vegas, G. A. Salazar, J. Tate, and A. Bateman, "The Pfam protein families database: towards a more sustainable future," *Nucleic Acids Res*, vol. 44, no. D1, pp. D279–85, 2016.

[37] B. Schuster-Bockler, J. Schultz, and S. Rahmann, "HMM logos for visualization of protein families," *BMC Bioinformatics*, vol. 5, p. 7, 2004.

[38] S. R. Eddy, "Profile hidden Markov models," *Bioinformatics*, vol. 14, no. 9, pp. 755–63, 1998.

[39] "UniProt: the universal protein knowledgebase," *Nucleic Acids Res*, vol. 45, no. D1, pp. D158–D169, 2016.

[40] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree: computing large minimum evolution trees with profiles instead of a distance matrix," *Mol Biol Evol*, vol. 26, no. 7, pp. 1641–50, 2009.

[41] S. Guindon, J. F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, "New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0," *Syst Biol*, vol. 59, no. 3, pp. 307–21, 2010.

[42] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree 2–approximately maximum-likelihood trees for large alignments," *PLoS One*, vol. 5, no. 3, p. e9490, 2010.

[43] E. Camon, D. Barrell, C. Brooksbank, M. Magrane, and R. Apweiler, "The gene ontology annotation (GOA) project–application of GO in SWISS-PROT, TrEMBL and InterPro," *Comp Funct Genomics*, vol. 4, no. 1, pp. 71–4, 2003.

[44] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucleic Acids Res*, vol. 28, no. 1, pp. 235–42, 2000.

[45] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures," *J Mol Biol*, vol. 247, no. 4, pp. 536–40, 1995.

[46] N. J. Mulder and R. Apweiler, "The InterPro database and tools for protein domain analysis," *Curr Protoc Bioinformatics*, vol. Chapter 2, p. Unit 2 7, 2008.

[47] J. D. Thompson, F. Plewniak, R. Ripp, J. C. Thierry, and O. Poch, "Towards a reliable objective function for multiple sequence alignments," *J Mol Biol*, vol. 314, no. 4, pp. 937–51, 2001.

[48] I. Sela, H. Ashkenazy, K. Katoh, and T. Pupko, "GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters," *Nucleic Acids Res*, vol. 43, no. W1, pp. W7–14, 2015.

[49] S. Y. Chung and S. Subbiah, "A structural explanation for the twilight zone of protein sequence homology," *Structure*, vol. 4, no. 10, pp. 1123–7, 1996.

[50] R. K. Shultzaberger and T. D. Schneider, "Using sequence logos and information analysis of Lrp DNA binding sites to investigate discrepancies between natural selection and selex," *Nucleic Acids Res*, vol. 27, no. 3, pp. 882–7, 1999.

[51] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, p. 379423, 1948.

[52] J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins, "The CLUSTAL X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools," *Nucleic Acids Res*, vol. 25, no. 24, pp. 4876–82, 1997.

[53] R. C. Edgar, "Quality measures for protein alignment benchmarks," *Nucleic Acids Res*, vol. 38, no. 7, pp. 2145–53, 2010.

[54] C. Kemena and C. Notredame, "Upcoming challenges for multiple sequence alignment methods in the high-throughput era," *Bioinformatics*, vol. 25, no. 19, pp. 2455–65, 2009.

[55] J. D. Thompson, F. Plewniak, and O. Poch, "A comprehensive comparison of multiple sequence alignment programs," *Nucleic Acids Res*, vol. 27, no. 13, pp. 2682–90, 1999.

[56] J. M. Sauder, J. W. Arthur, and J. Dunbrack, R. L., "Large-scale comparison of protein sequence alignment algorithms with structure alignments," *Proteins*, vol. 40, no. 1, pp. 6–22, 2000.

[57] M. Cline, R. Hughey, and K. Karplus, "Predicting reliable regions in protein sequence alignments," *Bioinformatics*, vol. 18, no. 2, pp. 306–14, 2002.

[58] T. Lassmann and E. L. Sonnhammer, "Automatic assessment of alignment quality," *Nucleic Acids Res*, vol. 33, no. 22, pp. 7120–8, 2005.

[59] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington, "HOMSTRAD: a database of protein structure alignments for homologous families," *Protein Sci*, vol. 7, no. 11, pp. 2469–71, 1998.

[60] P. I. de Bakker, A. Bateman, D. F. Burke, R. N. Miguel, K. Mizuguchi, J. Shi, H. Shirai, and T. L. Blundell, "HOMSTRAD: adding sequence information to structure-based alignments of homologous protein families," *Bioinformatics*, vol. 17, no. 8, pp. 748–9, 2001.

[61] L. A. Stebbings and K. Mizuguchi, "HOMSTRAD: recent developments of the homologous protein structure alignment database," *Nucleic Acids Res*, vol. 32, no. Database issue, pp. D203–7, 2004.

[62] L. Lo Conte, S. E. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin, "SCOP database in 2002: refinements accommodate structural genomics," *Nucleic Acids Res*, vol. 30, no. 1, pp. 264–7, 2002.

[63] B. Boeckmann, A. Bairoch, R. Apweiler, M. C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider, "The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003," *Nucleic Acids Res*, vol. 31, no. 1, pp. 365–70, 2003.

[64] J. D. Thompson, F. Plewniak, and O. Poch, "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs," *Bioinformatics*, vol. 15, no. 1, pp. 87–8, 1999.

[65] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark," *Proteins*, vol. 61, no. 1, pp. 127–36, 2005.

[66] R. B. Russell and G. J. Barton, "Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels," *Proteins*, vol. 14, no. 2, pp. 309–23, 1992.

[67] I. Van Walle, I. Lasters, and L. Wyns, "SABmark–a benchmark for sequence alignment that covers the entire known fold space," *Bioinformatics*, vol. 21, no. 7, pp. 1267–8, 2005.

[68] T. V. Astakhova, M. N. Lobanov, I. V. Poverennaia, M. A. Roitberg, and V. V. Iakovlev, "Verification of the PREFAB alignment database," *Biofizika*, vol. 57, no. 2, pp. 205–11, 2012.

[69] J. D. Thompson, J. C. Thierry, and O. Poch, "RASCAL: rapid scanning and correction of multiple sequence alignments," *Bioinformatics*, vol. 19, no. 9, pp. 1155–61, 2003.

[70] S. Chakrabarti, C. J. Lanczycki, A. R. Panchenko, T. M. Przytycka, P. A. Thiessen, and S. H. Bryant, "Refining multiple sequence alignments with conserved core regions," *Nucleic Acids Res*, vol. 34, no. 9, pp. 2598–606, 2006.

[71] K. Bucka-Lassen, O. Caprani, and J. Hein, "Combining many multiple alignments in one improved alignment," *Bioinformatics*, vol. 15, no. 2, pp. 122–30, 1999.

[72] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *J Mol Biol*, vol. 302, no. 1, pp. 205–17, 2000.

[73] I. M. Wallace, O. O'Sullivan, D. G. Higgins, and C. Notredame, "M-coffee: combining multiple sequence alignment methods with t-coffee," *Nucleic Acids Res*, vol. 34, no. 6, pp. 1692–9, 2006.

[74] P. W. Collingridge and S. Kelly, "Mergealign: improving multiple sequence alignment performance by dynamic reconstruction of consensus multiple sequence alignments," *BMC Bioinformatics*, vol. 13, p. 117, 2012.

[75] J. Muller, C. J. Creevey, J. D. Thompson, D. Arendt, and P. Bork, "AQUA: automated quality improvement for multiple sequence alignments," *Bioinformatics*, vol. 26, no. 2, pp. 263–5, 2010.

[76] M. R. Aniba, O. Poch, A. Marchler-Bauer, and J. D. Thompson, "AlexSys: a knowledge-based expert system for multiple sequence alignment construction and analysis," *Nucleic Acids Res*, vol. 38, no. 19, pp. 6338–49, 2010.

[77] K. Katoh and H. Toh, "Recent developments in the MAFFT multiple sequence alignment program," *Brief Bioinform*, vol. 9, no. 4, pp. 286–98, 2008.

[78] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "Probcons: Probabilistic consistency-based multiple sequence alignment," *Genome Res*, vol. 15, no. 2, pp. 330–40, 2005.

[79] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch, "A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives," *PLoS One*, vol. 6, no. 3, p. e18093, 2011.

[80] M. Gouy, S. Guindon, and O. Gascuel, "SeaView version 4: A multiplatform graphical user interface for sequence alignment and phylogenetic tree building," *Mol Biol Evol*, vol. 27, no. 2, pp. 221–4, 2010.

[81] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins, "Clustal W and Clustal X version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–8, 2007.

[82] "Se-Al," 2002 (accessed 12-21-2016), http://tree.bio.ed.ac.uk/software/seal/.

[83] A. M. Waterhouse, J. B. Procter, D. M. Martin, M. Clamp, and G. J. Barton, "Jalview version 2–a multiple sequence alignment editor and analysis workbench," *Bioinformatics*, vol. 25, no. 9, pp. 1189–91, 2009.

[84] A. Larsson, "AliView: a fast and lightweight alignment viewer and editor for large datasets," *Bioinformatics*, vol. 30, no. 22, pp. 3276–8, 2014.

[85] T. Shafee and I. Cooke, "AlignStat: a web-tool and R package for statistical comparison of alternative multiple sequence alignments," *BMC Bioinformatics*, vol. 17, no. 1, p. 434, 2016.

[86] A. Loytynoja and N. Goldman, "webPRANK: a phylogeny-aware multiple sequence aligner with interactive alignment browser," *BMC Bioinformatics*, vol. 11, p. 579, 2010.

[87] K. Tamura, J. Dudley, M. Nei, and S. Kumar, "MEGA4: Molecular evolutionary genetics analysis (MEGA) software version 4.0," *Mol Biol Evol*, vol. 24, no. 8, pp. 1596–9, 2007.

[88] C. L. Strope, K. Abel, S. D. Scott, and E. N. Moriyama, "Biological sequence simulation for testing complex evolutionary hypotheses: indel-Seq-Gen version 2.0," *Mol Biol Evol*, vol. 26, no. 11, pp. 2581–93, 2009.

[89] A. Kahler and H. Sticht, "A modeling strategy for g-protein coupled receptors," *AIMS Biophysics*, vol. 3, no. 2, pp. 211–231, 2016.

[90] V. A. Simossis and J. Heringa, "PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information," *Nucleic Acids Res*, vol. 33, no. Web Server issue, pp. W289–94, 2005.

[91] J. Pei and N. V. Grishin, "PROMALS: towards accurate multiple sequence alignments of distantly related proteins," *Bioinformatics*, vol. 23, no. 7, pp. 802–8, 2007.

[92] D. M. Hillis, "Approaches for assessing phylogenetic accuracy," *Systematic Biology*, vol. 44, no. 1, pp. 3–16, 1995.

[93] D. T. Jones, W. R. Taylor, and J. M. Thornton, "The rapid generation of mutation data matrices from protein sequences," *Comput Appl Biosci*, vol. 8, no. 3, pp. 275–82, 1992.

[94] J. Stoye, D. Evers, and F. Meyer, "Rose: generating sequence families," *Bioinformatics*, vol. 14, no. 2, pp. 157–63, 1998.

[95] A. Pang, A. D. Smith, P. A. Nuin, and E. R. Tillier, "SIMPROT: using an empirically determined indel distribution in simulations of protein evolution," *BMC Bioinformatics*, vol. 6, p. 236, 2005.

[96] W. Fletcher and Z. Yang, "INDELible: a flexible simulator of biological sequence evolution," *Mol Biol Evol*, vol. 26, no. 8, pp. 1879–88, 2009.

[97] T. Koestler, A. von Haeseler, and I. Ebersberger, "REvolver: modeling sequence evolution under domain constraints," *Mol Biol Evol*, vol. 29, no. 9, pp. 2133–45, 2012.

[98] P. A. Nuin, Z. Wang, and E. R. Tillier, "The accuracy of several multiple sequence alignment programs for proteins," *BMC Bioinformatics*, vol. 7, p. 471, 2006.

[99] C. L. Strope, S. D. Scott, and E. N. Moriyama, "indel-Seq-Gen: a new protein family simulator incorporating domains, motifs, and indels," *Mol Biol Evol*, vol. 24, no. 3, pp. 640–9, 2007.

[100] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J Phys Chem*, vol. 81, pp. 2340–2361, 1977.

[101] S. A. Benner, M. A. Cohen, and G. H. Gonnet, "Empirical and structural models for insertions and deletions in the divergent evolution of proteins," *J Mol Biol*, vol. 229, no. 4, pp. 1065–82, 1993.

[102] M. R. Aniba, O. Poch, and J. D. Thompson, "Issues in bioinformatics benchmarking: the case study of multiple sequence alignment," *Nucleic Acids Res*, vol. 38, no. 21, pp. 7353–63, 2010.

[103] S. Iantorno, K. Gori, N. Goldman, M. Gil, and C. Dessimoz, "Who watches the watchmen? an appraisal of benchmarks for multiple sequence alignment," *Methods Mol Biol*, vol. 1079, pp. 59–73, 2014.

[104] K. Katoh, K. Kuma, H. Toh, and T. Miyata, "MAFFT version 5: improvement in accuracy of multiple sequence alignment," *Nucleic Acids Res*, vol. 33, no. 2, pp. 511–8, 2005.

[105] M. T. Pervez, M. E. Babar, A. Nadeem, M. Aslam, A. R. Awan, N. Aslam, T. Hussain, N. Naveed, S. Qadri, U. Waheed, and M. Shoaib, "Evaluating the accuracy and efficiency of multiple sequence alignment methods," *Evol Bioinform Online*, vol. 10, pp. 205–17, 2014.

[106] P. A. Nuin, Z. Wang, and E. R. Tillier, "The accuracy of several multiple sequence alignment programs for proteins," *BMC Bioinformatics*, vol. 7, p. 471, 2006.

[107] B. Rost, "Twilight zone of protein sequence alignments," *Protein Eng*, vol. 12, no. 2, pp. 85–94, 1999.

[108] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res*, vol. 22, no. 22, pp. 4673–80, 1994.

[109] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch, "A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives," *PLoS One*, vol. 6, no. 3, p. e18093, 2011.

[110] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, 1998.

[111] A. Sharma and D. R. Chaudhar, "Character recognition using neural network," *IJETT*, vol. 4, no. 4, 2013.

[112] T. M. Mitchell, *Machine Learning*, ser. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997.

[113] I. H. Witten, E. Frank, and M. A. Hall, *Data mining : practical machine learning tools and techniques*, 3rd ed., ser. Morgan Kaufmann series in data management systems. Burlington, MA: Morgan Kaufmann, 2011.

[114] J. R. Quinlan, *C4.5 : programs for machine learning*, ser. The Morgan Kaufmann series in machine learning. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

[115] M. Pal, "Random forest classifier for remote sensing classification," *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217–222, 2004.

[116] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, p. 119139, 1997.

[117] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machine," *Microsoft research*, 1998.

[118] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Brief Bioinform*, 2016.

[119] A. Gudys and S. Deorowicz, "QuickProbs 2: Towards rapid construction of high-quality alignments of large protein families," *Sci Rep*, vol. 7, p. 41553, 2017.

[120] Y. Ye, T. W. Lam, and H. F. Ting, "PnpProbs: a better multiple sequence alignment tool by better handling of guide trees," *BMC Bioinformatics*, vol. 17 Suppl 8, p. 285, 2016.

[121] R. Vanhoutreve, A. Kress, B. Legrand, H. Gass, O. Poch, and J. D. Thompson, "LEON-BIS: multiple alignment evaluation of sequence neighbours using a bayesian inference system," *BMC Bioinformatics*, vol. 17, no. 1, p. 271, 2016.

[122] A. Chiner-Oms and F. Gonzalez-Candelas, "EvalMSA: A program to evaluate multiple sequence alignments and detect outliers," *Evol Bioinform Online*, vol. 12, pp. 277–284, 2016.

[123] Q. Le, F. Sievers, and D. G. Higgins, "Protein multiple sequence alignment benchmarking through secondary structure prediction," *Bioinformatics*, 2017.