Multiple Sequence Alignment Using External Sources Of Information

PhD Thesis

in partial fulfilment of the requirements for the degree
"Doctor of Philosophy (PhD)"

within the doctoral program Computer Science (PCS) of the
Georg-August University School of Science (GAUSS)
Faculty of Microbiology and Genetics

Submitted by

Layal Yasin

From Jeddah

Göttingen,2016

Date of the oral examination:
28. January 2016

Acknowledgments

First and foremost, it is with immense gratitude that I acknowledge the assistance and help of my Supervisor Prof. Dr. Burkhard Morgenstern throughout the whole period of my PhD studies. You have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Without your guidance and persistent help this dissertation would have been impossible.

I would like to thank my Thesis committee members Prof Dr. Carsten Damm and Prof. Dr Kifah Tout for your brilliant comments and suggestions, and for the fruitiful discussions we had during the thesis committee meetings held.

I wish to express my sincere gratitude to Dr. Eduardo Corel for the productive collaboration we did together. You made it easier for me to get integrated into the research field during the early stages of my PhD through teaching me the needed skills and techniques.

I wish to thank our secretary Britta Leinemann for making my stay in Germany easier and for providing me with unconditional help.

It gives me great pleasure in acknowledging the support and help of my colleagues Dr. Thomas Lingner, Dr. Youssef El Hajj Chehade, Zaher Yamak, Rayan Daou and Kathrin Asshauer.

I am indebted to our IT administrator Rasmus Steinkamp for always being ready to help with any IT-related issues and with setting up the webservers I was working on during my study. Special and unlimited thanks to my family overseas in Lebanon and Jordan. Words cannot express how grateful I am to you for your prayers and unlimited care.

Last but not least, I owe my deepest gratitude to my beloved husband Tarik, your continuous warm support and invaluably constructive criticism had improved the quality of my work and made it easier to finish out with such a rich output.

Multiple sequence alignment using external sources of information

Abstract:

Multiple sequence alignment is an alignemnt of three or more protein or nucleic acid sequences. The alignment area has always been of much interest for researchers, this is due to that fact that many scientific researchs depend in their workflow on sequence alignemnts. Thus, having an alignment of high quality is of high importance. Much work has been done and is still carried in this field to help improving the quality of alignments. Many approaches have been developed so far for performing pairwise and multiple sequence alignments, yet, most of those approaches rely basically on the sequences to be aligned as their only input. Recently, some approaches began to incorporate additional sources of information in the alignment process, the sources of external data can come from user knowledge or online databases. This data, when integrated in the workflow of the alignment programs, may add new constraints to the produced alignment and improve its quality by making it biologically more meaningful. In this thesis, I will introduce new approaches for multiple sequence alignment which use the alignment software DIALIGN along with external information from databases, where useful information is extracted and then integrated in the alignment process. By testing those approaches on benchmark databases, I will show that using additional data during alignemnt produced better results than using DIALIGN alone without any external input other than the sequences to be aligned.

Keywords: Multiple sequence alignment, PFAM, PROSITE, Protein-Domains, Patterns, Profile hidden markov models

Contents

Chapter 1

Introduction

1.1 Assignment formulation

The research projects in genome sequencing and related fields are producing huge amounts of biological data daily. This data is deposited in public and private databases in a structured and searchable form. At present, hundreds of free public databases such as *PFAM* [47] and *PROSITE* [16] are available.

Many multiple sequence alignment softwares accept, as their only input, the sequences to be aligned without employing any other source of external information in the alignment process. But when such softwares make use of the data available in public databases, the quality of the produced alignments will be improved. This happens because constraints will be extracted from the used data and employed during the alignment calculation.

In this thesis, I will present new approaches which fall under the category of incorporating external information in the alignment process.

The tasks of this thesis are clearly summarized by the following points:

- Short survey about the various pairwise and multiple sequence alignment methods.

- Performing sequence alignments using external information from *PFAM* and *PROSITE* databases and then testing those two approaches on the two benchmark databases: BAliBASE and SABmark.

- Implementing a webserver for multiple sequence alignments using external information from *PFAM* database.

- Implementing an approach for aligning alignments with unaligned sequences.

- Implementing a webserver for aligning alignments with unaligned sequences.

1.2 Structure of the thesis

Chapter 1 shows the structure of the thesis in addition to a summery of all the projects presented in this thesis. Moreover, a list of the published, unpublished manuscripts and posters that have been written during the course of the thesis is presented in this chapter.

Chapter 2 covers a brief introduction to DNA, RNA and proteins.

Chapter 3 provides a detailed introduction about sequence alignment. Various methods for performing pairwise and multiple sequence alignments will be mentioned, in addition to some tools which implement those methods. Furthermore, the different versions of the alignment software *DIALIGN* will be discussed.

Chapter 4 describes in detail the four projects presented in this thesis: *DIALIGN-PFAM*, *DIALIGN-PROSITE*, webserver for *DIALIGN-PFAM*, ”aligning alignments with unaligned sequences”. The algorithms for those approaches are outlined. Besides, the results of testing *DIALIGN-PFAM*, *DIALIGN-PROSITE* on benchmark databases will be presented.

Chapter 5 mentions a general conclusion and suggests future perspectives.

1.3 Thesis projects summary

Four main projects have been implemented during the PhD period and are presented in this thesis:

1.3.1 DIALIGN-PFAM

Using external sources of information in the alignment process will certainly improve the performance of the alignment programs by producing biologically more meaningful and correct alignments. *DIALIGN-PFAM* [29], one of the latest versions of *DIALIGN*, incorporates additional information from *PFAM* database to improve its output.

The paper entitled ”Using protein-domain information for multiple sequence alignments” [29] explains the first implementation of the algorithm behind *DIALIGN-PFAM*. A manuscript entitled ”Multiple sequence alignment using information derived from *PFAM* and *PROSITE* databases” provides a detailed description of an improved version of the algorithm in addition to the testing results on BAliBASE and SABmark.

1.3.2 DIALIGN-PFAM webserver

DIALIGN-PFAM webserver is an interactive version of *DIALIGN-PFAM*. It allows users to participate in some of the steps in the workflow of this tool.

The paper entitled "DIALIGN at Gobics-multiple sequence alignment using various sources of external information" [30] speaks briefly about the *DIALIGN-PFAM* webserver, in addition to a brief overview on the previous versions of *DIALIGN* webservers.

1.3.3 DIALIGN-PROSITE

Similar to *DIALIGN-PFAM*, the idea behind *DIALIGN-PROSITE* is the integration of external information from *PROSITE* database in the alignment process.

The manuscript entitled "Multiple sequence alignment using information derived from *PFAM* and *PROSITE* databases" explains in more details the algorithm behind *DIALIGN-PROSITE* and shows the results of testing this approach on BAliBASE and SABmark.

1.3.4 Aligning Alignments with Unaligned Sequences

This project is considered as an enhancement done to the anchoring option of *DIALIGN*. This newly developed functionality permits users to align an already existing alignment with a set of alignments or unaligned sequences. Users can either choose to keep the input alignment/s fixed in the final alignment, or just keep certain blocks in the input alignment/s fixed in the final alignment.

In the second case, the user has to specify the start and end positions, with respect to the alignment, of the blocks he wishes to keep fixed in the final multiple sequence alignment.

Afterward, the smallest possible set of anchor points is extracted from these blocks and input later on to *DIALIGN* along with the input sequences in order to produce the final multiple sequence alignment.

In the manuscript entitled "Multiple sequence alignment using partial-alignments as anchor points", a webserver for the previously mentioned option is presented. Users can see visually the input alignments on the screen and select blocks (partial alignments) interactively by pressing on the start and end positions for each block the user wishes to keep fixed in the final multiple sequence alignment.

In case the user wishes the whole alignment to be kept fixed in the final alignment then no blocks shall be selected; thus, the whole alignment will be considered as one large block.

1.4 List of published and unpublished manuscripts

1. **Al-Ait, L., Corel, E., Morgenstern, B.: Using protein-domain information for multiple sequence alignment. In Preceedings of the IEEE 12th International Conference on Bioinformatics and BioEngineering (BIBE 12) 163-168 (2012).**
    - Status: published.
2. **Al-Ait, L., Yamak, Z., Morgenstern, B.: DIALIGN at GOBICS-multiple sequence alignment using various sources of external information. Nucleic Acids Research 41, W3-W7 (2013).**
    - Status: published.
3. **Multiple sequence alignment using information derived from *PFAM* and *PROSITE* databases**
    - Status: unpublished.
4. **Multiple sequence alignment using partial-alignments as anchor points**
    - Status: unpublished.

1.5 Posters

1. **Integrating protein domain prediction in multiple sequence alignment**

    Presented at ISMB/ECCB 2011, Vienna.
2. **Multiple protein alignment using protein-domain information**

    Presented at GCB 2014, Goettingen.

Chapter 2

Biological Background

2.1 Nucleic acid

2.1.1 DNA

DNA stands for deoxyribonucleic acid. It is considered to be the blueprint of organisms since it contains all the necessary information for a cell growth and division. The chemical structure of DNA is a regular backbone of 2'-deoxyriboses, joined by 3'-5' phosphodiester bonds. Information carried by a certain DNA molecule is represented by a series of four chemical bases(Figure 2.1):

- The purines adenine 'A' and guanine 'G'
- The pyrimidines cytosine 'C' and thymine 'T'

DNA bases pair with each other: the adenine nucleotide on one DNA strand can bind with the thymine nucleotide on the other strand via hydrogen bonds to form units called base pairs. On the other hand, the guanine and cytosine can base pair with each other. These base pairs A-T and C-G when stacked over each other via hydrophobic interactions will form what is known as a 'Chromosome'. The latter is a thread-like molecule that carries hereditary information on two DNA helix strands.

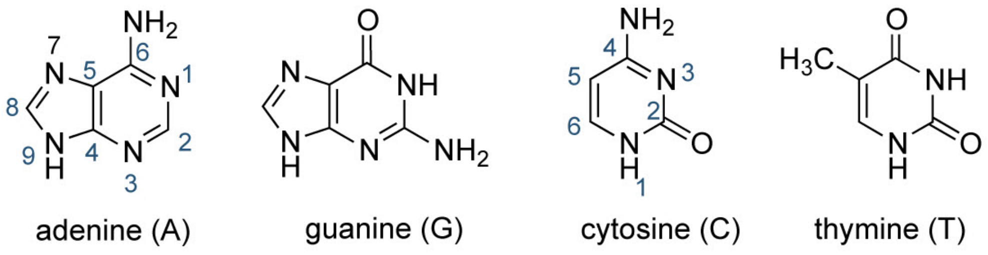


Figure 2.1: Chemical structures of the heterocyclic bases of DNA.
Source: "www.atdbio.com"

2.1.2 RNA

RNA stands for ribonucleic acid. Its chemical structure is similar to that of DNA. The backbone of RNA uses riboses rather than 2'-deoxyriboses, and the methyl group on the thymine is absent. There are three main types of RNA:

- Messenger RNA (mRNA)
- Transfer RNA (tRNA)
- Ribosomal RNA (rRNA)

rRNA and tRNA are parts of protein synthesizing engine, and mRNA is a template for protein synthesis.

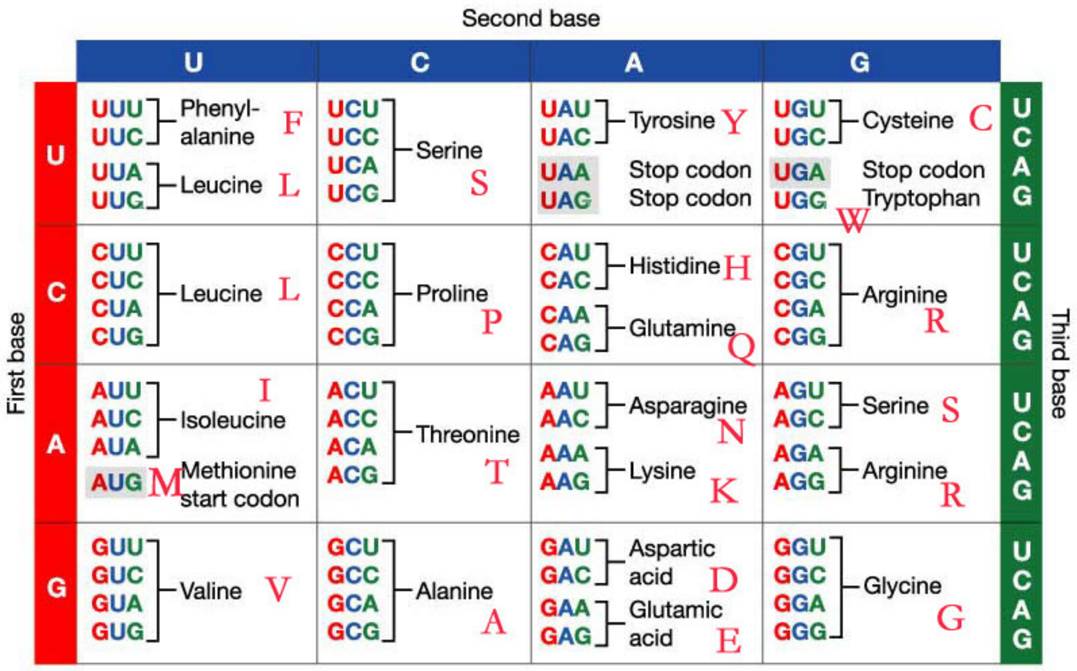
2.2 Proteins

2.2.1 Definition

Proteins are among the most complex molecules known so far. A protein molecule is made of a sequence of amino acids. Amino acids are the building blocks of all proteins. There are 20 amino acids, each one is characterized by its own chemical properties. Most of the amino acids have a structure based on a single carbon atom to which is attached four different groups of atoms: an amine group, a carboxylic acid group, a hydrogen atom and a variable group which is unique for every amino acid. Every three bases of RNA codes for an amino acid, (see figure 2.2 for more details).

2.2.2 Protein structure

A protein sequence is composed of a chain of amino acids. This chain is called a polypeptide chain and is considered to be the primary structure of a given protein sequence. The primary structure describes the unique order in which amino acids are linked together. Protein also have a secondary structure. The most common ones are alpha helices and beta sheets. The alpha helix structure resembles a coiled spring while the beta sheet structure appears to be folded out like a sheet. Moreover, proteins have a tertiary structure which refers to the comprehensive 3-D structure of the polypeptide chain of the protein.



	Second base: U	C	A	G	
First base					Third base
U	UUU UUC Phenyl-alanine F UUA UUG Leucine L	UCU UCC UCA UCG Serine S	UAU UAC Tyrosine Y UAA Stop codon UAG Stop codon	UGU UGC Cysteine C UGA Stop codon UGG Tryptophan W	U C A G
C	CUU CUC CUA CUG Leucine L	CCU CCC CCA CCG Proline P	CAU CAC Histidine H CAA CAG Glutamine Q	CGU CGC CGA CGG Arginine R	U C A G
A	AUU AUC AUA Isoleucine I AUG M Methionine start codon	ACU ACC ACA ACG Threonine T	AAU AAC Asparagine N AAA AAG Lysine K	AGU AGC Serine S AGA AGG Arginine R	U C A G
G	GUU GUC GUA GUG Valine V	GCU GCC GCA GCG Alanine A	GAU GAC Aspartic acid D GAA GAG Glutamic acid E	GGU GGC GGA GGG Glycine G	U C A G

Figure 2.2: Triplet codes for each Amino Acid. Source: "www.biogem.org"

2.2.3 Transcription and translation

Important biological events link DNA, RNA and proteins. These events are transcription and translation. Transcription is the process of transforming DNA to messenger RNA (mRNA). At this stage, the mRNA constitutes of expressed regions (exons) and unexpressed regions (introns). The latter are spliced out by a splicing machinary. In the case of eukaryotes, transcription takes place in the cell nucleus by an enzyme named RNA Polymerase. The mRNA is then transferred to the cytoplasm where it meets the ribosome. fterwards, translation takes place. During this process, mRNA along with tRNA are used by the ribosome to produce proteins.

2.2.4 Protein functions

Proteins serve various functions in the body. They are involved in almost all cell functions. Every protein has a specific function. The antibodies for instance are involved in defense against antigens. Enzymes catalyze biochemical reactions, for example, lactase enzyme breaks down the lactose sugar found in milk. Pepsin, another enzyme, is a digestive enzyme that works in the stomach to break down proteins in food. Other proteins act as transcription factors that turn genes on and off. Other proteins, the hormonal proteins, help in coordinating certain bodily activities. For instance, insulin regulates glucose metabolism by controlling the blood-sugar concentration; oxytocin simulates contractions in females during childbirth while somatotropin is a growth hormone that simulates protein production in muscle cells.

The structure of a protein helps in identifying its function. For instance, collagens have a long coiled helical shape which looks like a rope. This structure is great for providing support for connective tissues such as tendons and ligaments. For another example, see figure 2.3.

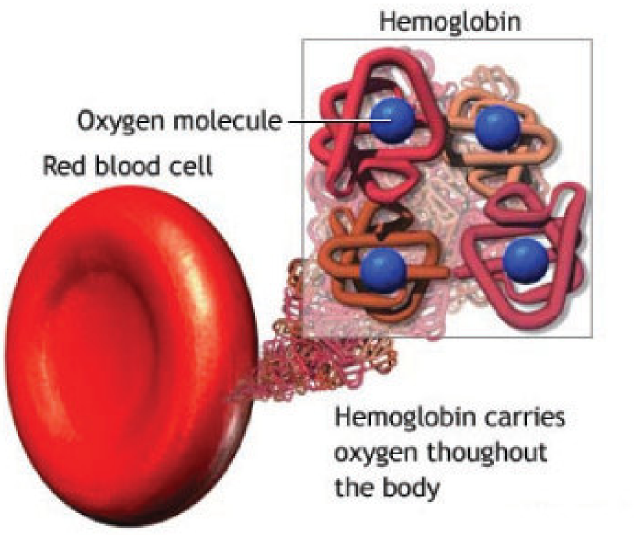


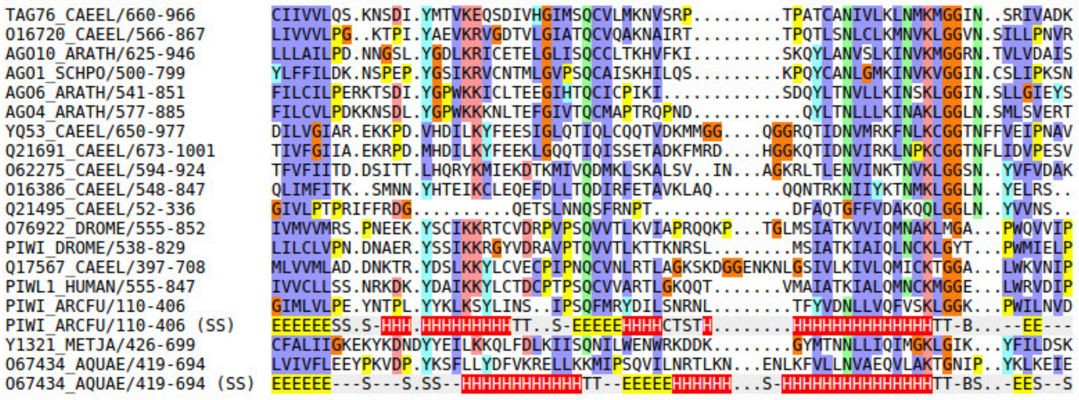
Figure 2.3: Hemoglobin is a folded and compact protein with a spherical shape that is mostly useful for maneuvering through blood vessels and transporting oxygen. Source: "rpi-cloudreassembly.transvercity.net"

2.2.5 Protein domains and families

A protein domain is a part of a protein sequence that has a specific function or interaction, can fold and exist independently of the rest of the protein sequence. A single protein domain's length ranges from 25 up to 500 amino acids. Most often, it is conserved through evolution. A certain domain may appear in a variety of different proteins. On the other hand, many proteins consist of several structural domains. For instance, Src homology 3 (SH3) is a protein domain that is involved in protein-protein interactions. These domains occur in a diverse range of proteins with different functions. An example of a protein which contains three copies of the

SH3 domain is the cytoplasmic protein Nck. lilo put reference to this protein

A protein family is a set of proteins which share common functions, similar structures and a common evolutionary ancestor. PFAM [47] is an important example of a database which contains a large number of protein domains and families, presented as profile hidden markov models [18] and multiple sequence alignments (For an example of a protein family, see figure 2.4).



```
TAG76_CAEEL/660-966         CIIVVLQS.KNSDI.YMTVKEQSDIVHGIMSQCVLMKNVSRP.........TPATCANIVLKLNMKMGGIN..SRIVADK
O16720_CAEEL/566-867        LIVVVLPG..KTPI.YAEVKRVGDTVLGIATQCVQAKNAIRT.........TPQTLSNLCLKMNVKLGGVN.SILLPNVR
AGO10_ARATH/625-946         LLLAILPD.NNGSL.YGDLKRICETELGLISQCCLTKHVFKI.........SKQYLANVSLKINVKMGGRN.TVLVDAIS
AGO1_SCHPO/500-799          YLFFILDK.NSPEP.YGSIKRVCNTMLGVPSQCAISKHILQS.........KPQYCANLGMKINVKVGGIN.CSLIPKSN
AGO6_ARATH/541-851          FILCILPERKTSDI.YGPWKKICLTEEGIHTQCICPIKI............SDQYLTNVLLKINSKLGGIN.SLLGIEYS
AGO4_ARATH/577-885          FILCVLPDKKNSDL.YGPWKKKNLTEFGIVTQCMAPTRQPND..........QYLTNLLLKINAKLGGLN.SMLSVERT
YQ53_CAEEL/650-977          DILVGIAR.EKKPD.VHDILKYFEESIGLQTIQLCQQTVDKMMGG....QGGRQTIDNVMRKFNLKCGGTNFFVEIPNAV
Q21691_CAEEL/673-1001       TIVFGIIA.EKRPD.MHDILKYFEEKLGQQTIQISSETADKFMRD....HGGKQTIDNVIRKLNPKCGGTNFLIDVPESV
O62275_CAEEL/594-924        TFVFIITD.DSITT.LHQRYKMIEKDTKMIVQDMKLSKALSV..IN...AGKRLTLENVINKTNVKLGGSN..YVFVDAK
O16386_CAEEL/548-847        QLIMFITK..SMNN.YHTEIKCLEQEFDLLTQDIRFETAVKLAQ.......QQNTRKNIIYKTNMKLGGLN..YELRS..
Q21495_CAEEL/52-336         GIVLPTPRIFFRDG..........QETSLNNQSFRNPT.............DFAQTGFFVDAKQQLGGLN..YVVNS..
O76922_DROME/555-852        IVMVVMRS.PNEEK.YSCIKKRTCVDRPVPSQVVTLKVIAPRQQKP...TGLMSIATKVVIQMNAKLMGA...PWQVVIP
PIWI_DROME/538-829          LILCLVPN.DNAER.YSSIKKRGYVDRAVPTQVVTLKTTKNRSL........MSIATKIAIQLNCKLGYT...PWMIELP
Q17567_CAEEL/397-708        MLVVMLAD.DNKTR.YDSLKKYLCVECPIPNQCVNLRTLAGKSKDGGENKNLGSIVLKIVLQMICKTGGA...LWKVNIP
PIWL1_HUMAN/555-847         IVVCLLSS.NRKDK.YDAIKKYLCTDCPTPSQCVVARTLGKQQT.......VMAIATKIALQMNCKMGGE...LWRVDIP
PIWI_ARCFU/110-406          GIMLVLPE.YNTPL.YYKLKSYLINS..IPSQFMRYDILSNRNL........TFYVDNLLVQFVSKLGGK...PWILNVD
PIWI_ARCFU/110-406 (SS)     EEEEEESS.S-HHH.HHHHHHHHHTT..S-EEEEEHHHHCTSTH........HHHHHHHHHHHHHHTT-B...--EE---
Y1321_METJA/426-699         CFALIIGKEKYKDNDYYEILKKQLFDLKIISQNILWENWRKDDK........GYMTNNLLIQIMGKLGIK...YFILDSK
O67434_AQUAE/419-694        LVIVFLEEYPKVDP.YKSFLLYDFVKRELLKKMIPSQVILNRTLKN...ENLKFVLLNVAEQVLAKTGNIP..YKLKEIE
O67434_AQUAE/419-694 (SS)   EEEEEE---S---S.SS--HHHHHHHHHHHHTT--EEEEEHHHHHH...S-HHHHHHHHHHHHHHHTT-BS..-EES--S
```

Figure 2.4: Piwi is a family of protein sequences, its name stands for: P-element included wimpy testis. Piwi is a class of genes which play crucial roles during germline development and gematogenesis of many metazoan species. This family contains 2067 protein sequences. This figure illustrates a section of the seed alignment for the family which constitutes of 18 sequences. A seed alignment of a certain family is a small subset of sequences from the complete set of sequences contained within the family. The sequences in the seed alignment are representative members of the family they belong to. Source: "pfam.xfam.org"

2.3 Types of mutations through evolution

Mutations are permanent changes that happen to the nucleotides of a DNA sequence. Two factors play important role in causing DNA mutations:

- External factors, i.e. environmental factors such as radiation.
- Native factors which are errors that occur during DNA replication.

2.3.1 Types of mutations

There are several types of mutations, they are listed below with examples. The mutated segments in the sequences are highlighted in yellow.

- Insertion: a mutation where extra base pairs are inserted into the DNA sequence. Example:

  Original sequence ACG GGC TTA ATA ATG

  Mutated sequence ACG GGC TT A TA A ATA ATG

- Deletion: a mutation in which one or more nucleotides are deleted from the DNA sequence. Example:

  Original sequence ACG GGC T TA T AA ATA ATG

  Mutated sequence ACG GGC TAA ATA ATG

- Translocation: a section of DNA is exchanged between two or more non-homologous chromosomes (for an example, see figure 2.5).

- Inversion: The order of a segment of nucleotides is reversed.

2.3.1.1 Frameshift

A frameshift is an insertion or deletion of a number of nucleotides. The added or deleted segment is not a multiple of three, as a result, the reading frame is altered. Example:

Original sequence ACGGGCT TATT AAATAAT

Mutated sequence ACGGGCTAAATAAT

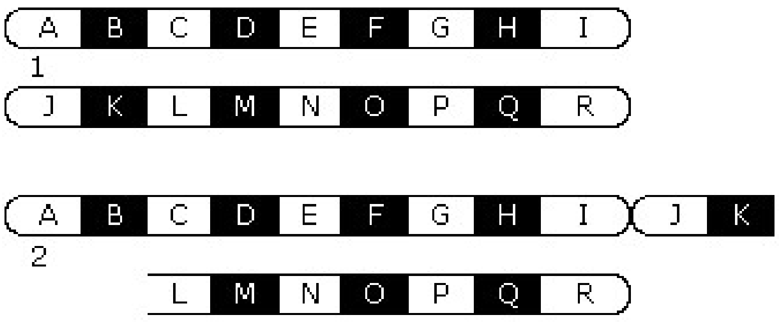


Figure 2.5: Example of a translocation of the two genes J and K from one chromosome to th other. Source: "biology-online.org"

2.3.1.2 Point mutations

A point mutation is a single base change in a DNA sequence. A point mutation may be silent, missense, or nonsense.

- Silent: since amino acids are encoded by more than one codon, a mutation can silent when the change in the DNA sequence does not change the coded protein sequence. Example:

  Original sequence CAA GGC TAA TAA

  Mutated sequence CA G GGC TAA TAA

  In both sequences, the first codon codes for the Glutamine amino acid

- Missense: a mutation in one nucleotide which changes the codon to a different amino acid. EXample:

  Original sequence CAA GGC TAA TAA

  Mutated sequence CA C GGC TAA TAA

  The first codon in the original sequence codes for Glutamine while the first codon in the mutated sequence codes for Histadine.

- Nonsense: a mutation in one nucleotide which results in a STOP codon. Example:

  Original sequence CAA GGC TTA TAAT

  Mutated sequence CAA GGC T A A TAAT

  The third codon in the original sequence codes for Leucine while in the mutated sequence it codes for the STOP codon.

Chapter 3

Sequence Alignment

Sequence alignment is one of the major research subjects in the bioinformatics field. It accepts as its input two or more protein or nucleic acid sequences, then identifies using some measures the regions of the sequences that are similar, and finally outputs the homologous positions aligned in columns. An alignment displays the residues for each sequence on a single line, with gaps ”-” inserted such that homologous residues appear in the same column.

Let $Al = \{\alpha_1 \ldots \alpha_n\}$ be an alphabet of size n containing the characters that may constitute any given sequence. For DNA sequences, $Al = \{A, C, G, T\}$. For protein sequences, Al constitutes mainly of the twenty amino acids. A sequence can be denoted as $S = \alpha_1\alpha_2 \ldots \alpha_g$ where g is the length of the sequence. An alignment of k sequences $S_1, S_2, \ldots S_k$ is the set of sequences $S'_1, S'_2, \ldots S'_k$ where S_1 is transformed to S'_1, S_2 is transformed to $S'_2 \ldots$ and S_k is transformed to S'_k by inserting gaps in the original sequences in certain positions allowing the new produced sequences to share more similarity.

3.1 Similarity versus homology

The term 'Homologous' in the sequence alignment context is meant to be on the structural and evolutionary level. Sequence alignment always try to visualize the relationships between residues in a collection of evolutionarily or structurally related sequences.

Strong similarity between two sequences presents a strong argument for their homology and provides an evidence that the two sequences have a common ancestor. Thus, sequences of related proteins or genes are similar, in a sense that one could align the sequences such that many corresponding residues match.

On the other hand, if a set of sequences are homologous, then they should not necessarly share a noticeable similarity .

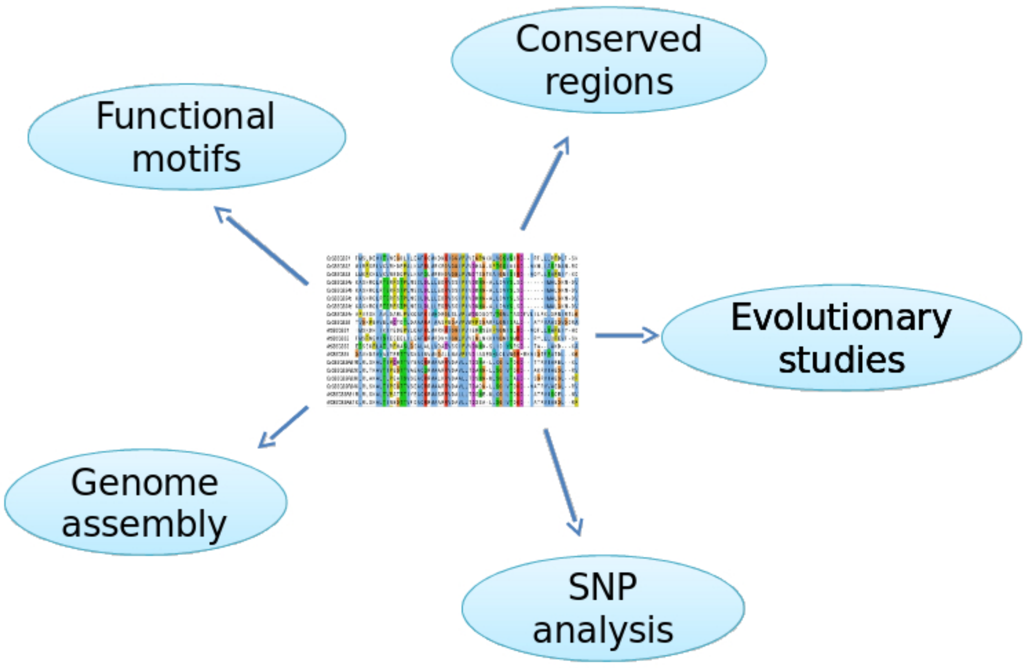


Figure 3.1: Some of the applications of sequence alignment.

3.2 Applications Of multiple sequence alignment

An alignment provides a closer view on the underlying evolutionary, structural, or functional constraints characterizing the sequences involved in the alignment.

Sequence alignment is a critical step towards sequence comparison. It is useful in discovering structural information and helps in detecting functional relationships between related species.

Alignment, as a single task, is of little interest for most researchers. Most often it is used as a transitional step to reach deeper areas of study (Figure 3.1). Sequence alignment enables researchers to identify conserved regions and functional motifs, facilitates evolutionary and phylogenetic studies [40, 41], aids in structure prediction [42, 43] and characterization of protein families [44, 46, 47]. The quality of sequence alignments plays a major role in the analyses process of protein sequences [48]. Thus, it is important to obtain and use high quality and biologically meaningful alignments. This is why sequence alignment is an active area of research.

3.3 Global verses Local alignment

A global alignment in general tries to align a pair of sequences starting from the first pair of nucleotides in both sequences till the last pair. Global alignments are used when the sequences to be aligned share similarity along their full length.

On the other hand, local alignments tries to align only some specific parts of the sequences which share a significant similarity according to some measure (Figure 3.2).

It seems that local alignment shall always be used, however, it may be difficult to spot an overall similarity if one uses only local alignment. It might also appear that finding an optimal local alignment is more complex than finding an optimal global alignment since in the first case, the start and end positions of the sequences involved in the alignment must be found. Nevertheless, only a constant factor more calculation is necessary.

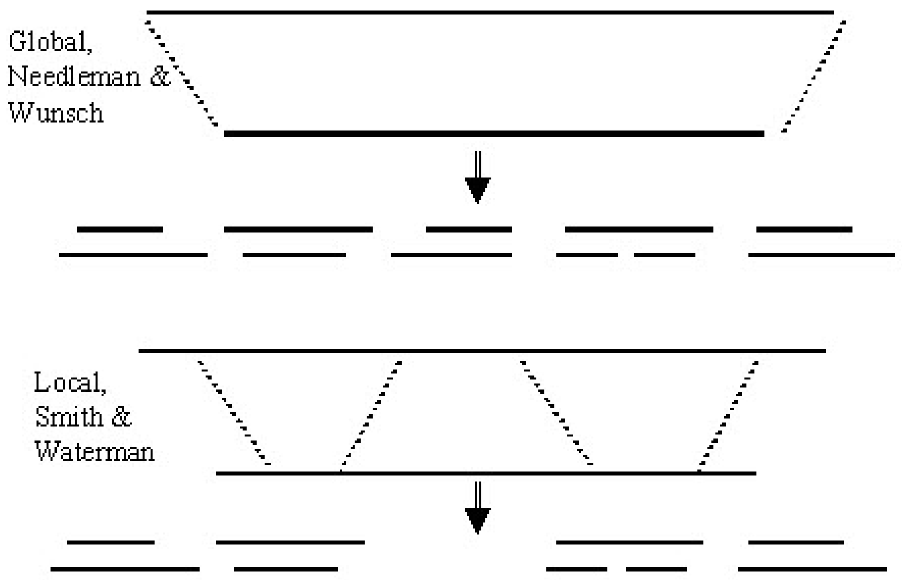


Figure 3.2: Global alignment (upper figure) v.s. local alignment (lower figure). To align sequences globally, the Needleman & Wunch algorithm [1] can be used while the Smith & Waterman algorithm [5] can be used to align sequences locally. Source: David Gibert, 2013 [Sequence Comparison].

3.4 Scoring scheme

Before calculating an alignment, two important factors should be defined: the objective function and the optimization algorithm. The objective function defines how the quality of a certain alignment is determined. Generally, the quality of an alignment depends on its score such that alignments with higher scores have better quality. On the other hand, the optimization algorithm defines the method used to calculate the alignment.

A scoring scheme is adopted by almost all alignment methods in order to evaluate an alignment. Many alignment methods uses a scoring scheme that consists of character substitution scores plus penalties for gaps(explained in the next section).

A simple scoring scheme is represented by a ”+1” for a match and a ”−1” or ”0” for a mismatch. This scheme is not very helpful when it comes to identifying the biologically meaningful alignments since more factors should be taken into consideration when assigning substitution scores. For example, in the case of protein sequence alignment, the score of substituting two amino acids that are chemically similar should be higher than that of two chemically non-similar ones.

3.4.1 Protein substitution matrices

The protein substitution matrices are used to score protein sequence alignments. They offer a substitution score for each pair of amino acids. The most widely used substitution matrices for protein alignment are PAM [2] and BLOSUM [3]. The scores presented in these matrices are derived from the analysis of known alignments of evolutionary related proteins.

3.4.1.1 PAM

PAM matrices were developed by Margaret Dayhoff and co-workers. PAM stands for Point Accepted Mutations. The scores presented in PAM are derived from alignments of very similar sequences of at least 85% identity. The expression ”Accepted point Mutation” refers to the case where a single amino acid is replaced with another one such that this process is accepted by natural selection.

3.4.1.2 BLOSUM

The BLOSUM matrices were created by Henikoff and Henikoff. BLOSUM stands for BLOck SUbstitution Matrix.

BLOSUM matrices are based on local alignments of distantly related sequences. In summary, a BLOSUM matrix is created by the following procedure:

- Short gap-free multiple alignments are gathered.
- In every alignment, similar sequences (according to some threshold value of sequence identity) are clustered into groups.
- For every pair of amino acids between the already produced groups, substitution frequencies are determined. The BLOSUM matrix is then calculated using those frequencies.

The number found next to any BLOSUM matrix name reflects the threshold identity percentage of the sequences clustered in the groups.

3.5 Gaps

Gaps can be considered as artificial insertions into sequences to move similar segments of sequences into alignment. For instance, consider the following simple pairwise alignment:

S_1 : VSAAP- EEM
S_2 : VSAAPYEEM

A gap, which is represented by a dash symbol ”-”, is inserted after the fifth residue in S_1. This can imply two possibilities:

- First possibility: an amino acid residue has been deleted from S_1.
- Second possibility: an amino acid residue has been inserted in S_2

Regarding the first possibility, when a residue seems to be deleted, a dash is inserted instead. With respect to the second possibility, if it seems that a residue is inserted, then a gap is inserted in the other unaugmented sequence.

In any given alignment, inserting gaps will lower the score of this alignment, this is due to what is called a ”gap penalty”. A gap penalty can be affine, constant or linear, where the last two are considered to be special cases of the affine gap penalty.

3.5.1 Constant

A constant penalty is given to every gap. This method does not depend on the gap length. For instance, a gap penalty of -2 and a gap of length 10 will result in a total gap penalty of -2 (the gap length is not taken into consideration).

3.5.2 Linear

This gap penalty depends linearly on the gap length. For example, a gap penalty of -2 and a gap of length 10 will result in a total gap penalty of -2 x $10 = -20$.

3.5.3 Affine

The affine gap penalty depends on a linear function to calculate the gap cost. One of the terms in this function depends on the length of the gap while the other does not. The form of the gap penalty is:

$$\text{Gap penalty} = X + (Y.L)$$

Where X is the gap opening penalty, Y is the extension penalty and L is the length of the gap. As an example, a gap penalty of -2, a gap extension penalty of -1 and a gap of length 10 will result in a total gap penalty of $-2 + (-1$ x $9) = -11$.

3.6 Pairwise sequence alignment

3.6.1 Dynamic programming

The idea behind using dynamic programming in sequence alignment is to build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences. Dynamic programming was first used to calculate global alignments with the Needleman-Wunch algorithm [1]. A local alignment version is the Smith-Waterman algorithm [5]. For a pair of sequences, both algorithms take nxm for space and time complexity where n is the length of the first sequence and m is the length of the second one.

3.6.1.1 Needleman-Wunsch

The Needleman-Wunsch algorithm is a classical algorithm which uses dynamic programming to find the optimal global alignment between two sequences using already calculated optimal alignments of shorter subsequences.

A scoring function is used in order to score the various alignments, and the alignment which gets the highest score is considered to be the optimal alignment. The score of matches is calculated using a substitution matrix (such as PAM [2] or BLOSUM [3]). On the other hand, gaps penalize the alignment score with a given gap penalty. For the Needleman-Wunsch algorithm, linear gap penalty is used.

This algorithm produces reasonable alignments when the input sequences are closely related. Furthermore, detecting similar regions between sequences with small overall similarity is not so manageable, especially that the resulting alignments depend on a set of user-defined input parameters.

To briefly explain the Needleman-Wunch algorithm, consider a pair of sequences: $S = s_1 \dots s_n$ and $S' = s'_1 \dots s'_m$. Construct a matrix M with $n+1$ columns and $m+1$ rows. $M(i,j)$ is the score of the best alignment for the subsequences $s_1 \dots s_i$ from S and $s'_1 \dots s'_j$ from S'.

In the first stage, M is initialized as follows:

- Initialize $M(0,0)$ with 0;
- Along the top row where $j = 0$, initialize $M(i,0)$ with $-i.d$, where d is the gap penalty and $0 < i \leq n$;
- Along the first column where $i = 0$, initialize $M(0,j)$ with $-j.d$, where d is the gap penalty and $0 < j \leq m$;

M is then filled recursively from top left to bottom right. In order to find the value of $M(i,j)$, three values should be known:

- $M(i-1, j-1)$
- $M(i, j-1)$
- $M(i-1, j)$

An alignment up to s_i and s'_j can be performed in three different ways:

- s_i and s'_j are aligned together
- s_i is aligned to a gap
- s'_j is aligned to a gap

Since $M(i,j)$ is the score of the best alignment up to s_i and s'_j, it is defined to be the maximum of the following three different values:

$$M(i,j) = max \begin{cases} M(i,j-1) - d \\ M(i-1,j) - d \\ M(i-1,j-1) + C(s_i, s'_j)\} \end{cases}$$

where $C(a_i, b_j)$ is the substitution score of the two residues s_i and s'_j and d is the gap penalty. $M(i,j-1) - d$ is the score of the best alignment of $s_1 \ldots s_i$ and $s'_1 \ldots s'_{(j-1)}$ minus a gap penalty, since s'_j is aligned to a gap. $M(i-1,j) - d$ is the score of the best alignment of $s_1 \ldots s_{(i-1)}$ and $s'_1 \ldots s'_j$ minus a gap penalty, since s_i is aligned to a gap. $M(i-1,j-1) + C(s_i, s'_j)$ is the score of the best alignment of $s_1 \ldots s_{(i-1)}$ and $s'_1 \ldots s'_{(j-1)}$ plus the substitution score of s_i and s'_j since these two residues are aligned together.

Whenever a certain $M(i,j)$ is calculated, a pointer to the previous cell in the matrix where $M(i,j)$ was derived from is kept.

M is filled recursively until all the values from $M(1,1)$ to $M(n,m)$ are calculated.

$M(n,m)$ carries the score of the optimal global alignment between S and S'. In order to get this optimal global alignment, a traceback is performed where the alignment is built in reverse starting from the last cell in matrix M and following the already stored pointers until the first cell $M(0,0)$ is reached. Going back from $M(i,j)$ to the previous cell can be done in three ways:

- Case 1: going back from $M(i,j)$ to $M(i-1,j-1)$
- Case 2: going back from $M(i,j)$ to $M(i,j-1)$
- Case 3: going back from $M(i,j)$ to $M(i-1,j)$

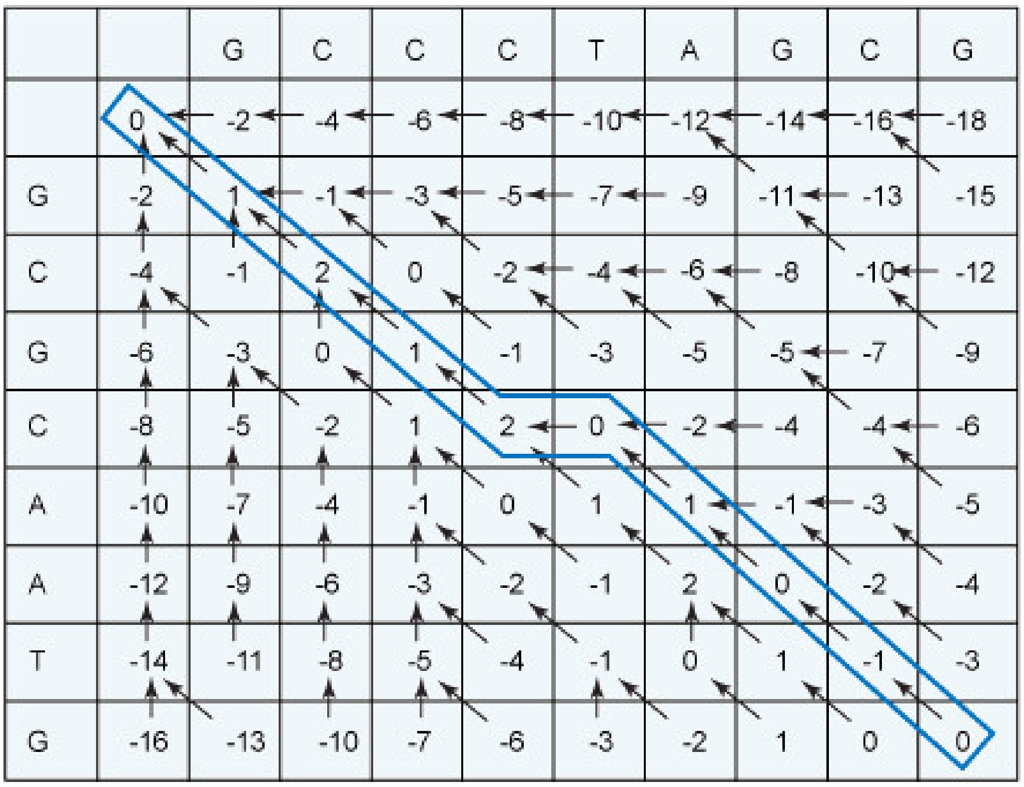
This is determined by the pointer that have been saved previously which shows from which cell $M(i,j)$ has been derived from.

Depending on which previous cell is reached, a pair of symbols is added to the front of the global alignment that is being calculated:

- s_i and s'_j are added if Case 1 applies.
- s'_j and a gap '$-$' are added if Case 2 applies.
- s_i and a gap '$-$' are added if Case 3 applies.

Backtracking terminates when $M(0,0)$ is reached (Figure 3.3).

The time and space complexity of the Needleman-Wunsch algorithm is $O(n.m)$.



		G	C	C	C	T	A	G	C	G
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
G	-2	1	-1	-3	-5	-7	-9	-11	-13	-15
C	-4	-1	2	0	-2	-4	-6	-8	-10	-12
G	-6	-3	0	1	-1	-3	-5	-5	-7	-9
C	-8	-5	-2	1	2	0	-2	-4	-4	-6
A	-10	-7	-4	-1	0	1	1	-1	-3	-5
A	-12	-9	-6	-3	-2	-1	2	0	-2	-4
T	-14	-11	-8	-5	-4	-1	0	1	-1	-3
G	-16	-13	-10	-7	-6	-3	-2	1	0	0

Figure 3.3: Filled-in Needleman-Wunsch table with traceback. Source: http://www.ibm.com/developerworks/library/j-seqalign/.

3.6.1.2 Smith-Waterman

It may often be the case that two sequences share some similarity in certain parts and not through the entire length of the sequence. This is achieved through aligning the pair of sequences locally. The Smith-Waterman algorithm finds the best local alignment between a pair of sequences.

Just like the Needleman-Wunsch algorithm, the Smith-Waterman uses dynamic programming to find the optimal local alignment between two sequences using already calculated optimal alignments of shorter subsequences.

A scoring function is used in order to score the various alignments, and the local alignment in the alignment matrix which gets the highest score is considered to be the optimal local alignment.

The Smith-Waterman algorithm is very similar to that of the Needleman-Wunch, but with some differences.

The first difference lies in the way matrix M is initialized. The top row and the first column of M are fillied with 0 instead of $-i.d$ and $-j.d$ as it is the case in the

Needleman-Wunsch algorithm.

When it comes to finding the value of $M(i,j)$, a new value '0' is added to the previous three values from which $M(i,j)$ take the maximum of:

$$M(i,j) = max \begin{cases} M(i,j-1) - d \\ M(i-1,j) - d \\ M(i-1,j-1) + C(s_i, s'_j) \\ 0\} \end{cases}$$

Taking the option 0 is equivalent to starting a new alignment. When the best alignment up to some point has a negative score, then it is certainly better not to extend the old alignment but to start with a new one. Random long matches have a negative score, otherwise they will be favored more than strong local matches of shorter length.

Since the aim of the algorithm is to find the best local alignment, this alignment can start and end anywhere in M without the need to extend from $M(0,0)$ till $M(n,m)$. As a result, negative values are not considered.

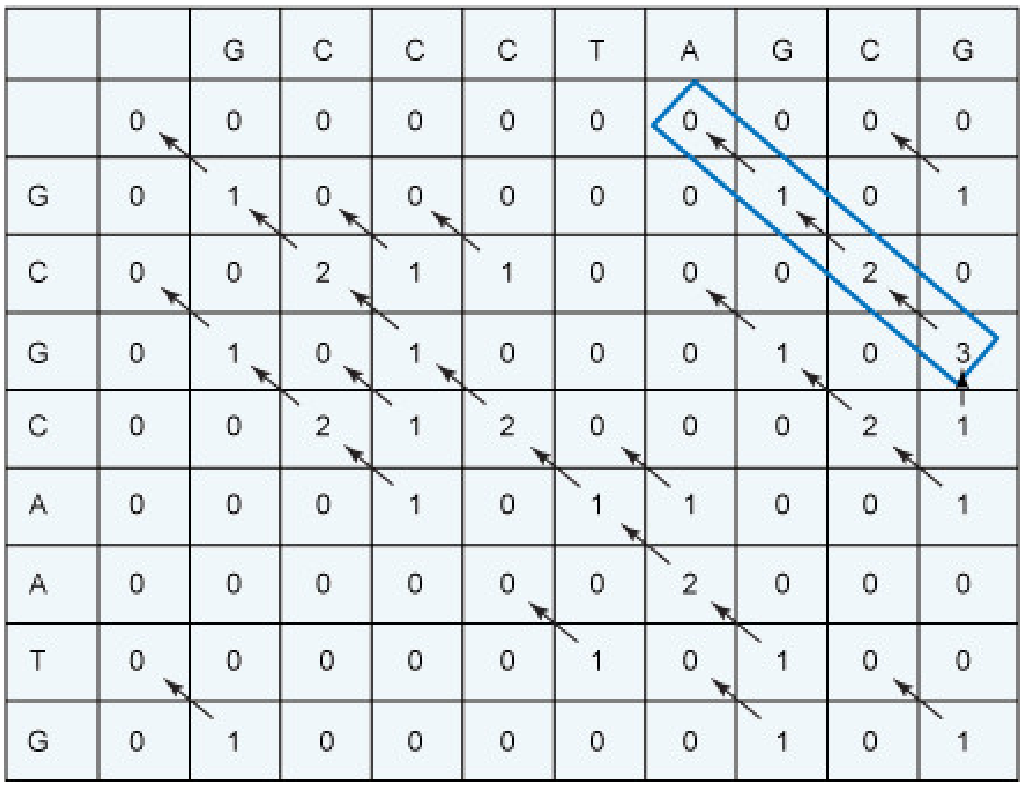
For backtracking, instead of starting at $M(n,m)$, the process starts at the cell having the highest score M_h. Then it goes backward following the path of the pointer stored in the current cell as it is the case in the Needleman-Wunsch algorithm. This continues until a cell with value 0 (which corresponds to the start of the local alignment) is reached (Figure 3.4). The score of the obtained local alignment is equal to M_h.

A time complexity of $O(nm)$ is required to align a pair of sequences of length n and m.

3.6.2 BLAST

When it comes to aligning large sequences together, dynamic programming becomes too slow for this task. Thus, heuristic methods are preferably used in this case. The K-tuple methods are in general more effective than the two previously mentioned ones. They are heuristic methods based on shared tuples (words) of length K between a pair of sequences. Such methods do not guarantee to find the optimal alignment solution.

BLAST [6] which stands for Basic Local Alignment Search Tool is a widely used tool that uses string matching algorithms in order to accomplish its tasks. BLAST



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

Figure 3.4: Filled-in Smith-Waterman table with traceback. Source: http://www.ibm.com/developerworks/library/j-seqalign/.

allows to search a query sequence against a library or a database of sequences, and outputs a set of sequences that resemble the input sequence.

Depending on the query sequence, different types of BLAST exist, these include: nucleotide blast, protein blast, blastx, tblastn, tblastx, ... etc.

BLAST uses heuristic algorithms to calculate alignments:

1. BLAST divides the query sequence into tuples (a tuple is a series of characters) with a certain fixed length. By default, the length is equal to three for protein sequences and eleven for nucleotide sequences. A sliding window is used to break a given sequence into tuples. Those tuples are then compared against sequences in a database (hit sequences).

2. Blast locates all common tuples between the query sequence and the hit sequence(s). Only those matches having a score higher than a certain score threshold T are considered. The score of a match is calculated using a substitution matrix.

3. After obtaining all the possible matches, BLAST extends them in both directions in an attempt to generate an alignment. With every extension, the score of the alignment is either increased or decreased. The extension continues as long as the score of the alignment does not drop beneath the maximal score obtained so far. Otherwise, the alignment ceases to extend. This will prevent poor alignments to be included in the final result. Moreover, segments in the alignment with score less than T are discarded.

3.6.2.1 BLAST verses the Smith-Waterman algorithm

When it comes to database similarity searches, the Smith-Waterman and the BLAST algorithms are the most widely used algorithms within this field.

By using the Smith-Waterman algorithm, one can be sure that optimal local alignment(s) between a given query sequence and database sequence is calculated. It is more accurate than BLAST meaning that it does not miss any information. Due to this, the Smith-Waterman algorithm is very time-consuming and computer power intensive.

When it comes to BLAST, and due to the fact that it is a heuristic algorithm, it does not guarantee to find the best results, as it misses the hard-to-find matches between the query and the target sequences. However, the algorithm is fast compared to Smith-Waterman.

3.7 Multiple sequence alignment methods

Multiple Sequence Alignment is an alignment of three or more protein or nucleic acid sequences that help in identifying regions of homology between the input sequences. This will help in a following step to study more the evolutionary relationships between the input sequences.

3.7.1 Progressive alignment

Aligning large number of protein sequences may need to be accomplished using heuristic methods rather than the optimal methods that consume an exponential runtime. The mostly adopted heuristic strategy when aligning protein sequences is the progressive alignment method. This method produces sensible alignments with an efficient running time.

The main idea behind any progressive alignmnet approach is building a guide tree out of a set of sequences and then aligning those sequences according to the order proposed by the tree.

The Feng-Doolittle progressive alignment method [31] is one of the first progressive alignment methods. Many of the current widely used alignment tools adopt the algorithm of Feng-Dolittle.

For a set of n sequences, progressively aligning those sequences using the Feng-Dolittle requires performing the following main steps:

1. Distances are calculated between each pair of sequences. A $n(n-1)/2$ distances matrix is produced.

2. Using a clustering algorithm, a guide tree is constructed from the distance matrix of step 1.

3. Child nodes are aligned according to the order provided by the guide tree. The two child nodes could be two sequences, two alignments, or an alignment and a sequence.

The algorithm stops when all the nodes in the guide tree have been processed.

Regarding step 1, the Feng-Dolittle algorithm uses the following formula to calculate the distances:

$$D = -logS_{eff} = -log(S_{obs} - S_{rand})/(S_{max} - S_{rand})$$

S_{obs}: The observed pairwise alignment score.

S_{max}: The maximum score, which is the average score of aligning either sequence to itself.

S_{rand}: Score of aligning two random sequences having the same length and residue composition.

Regarding step 2, UPGMA and NJ clustering methods are suitable for this task.

Regarding step 3, three types of alignments may exist:

- When the two child nodes are represented by two sequences, the usual pairwise dynamic algorithm is used to align those pair of sequences.

- When the two child nodes are represented by an alignment and a sequence, then this sequence is pairwisely aligned with every other sequence in the alignment. The pairwise alignment having the highest score will define how the sequence will be aligned to the alignment.

- When the two child nodes are represented by two alignments, all possible pairwise alignments between the sequences of the two groups are performed. The pairwise alignment having the highest score will define how the two alignments are aligned together.

The disadvantage of progressive alignment methods in general is the inability to recover from errors made in earlier steps. This is because once an alignment is produced in each step, it is kept fixed and cannot be altered.

PRRP aligns sequences progressively according to a predicted evolutionary tree, and periodically reassesses both the evolutionary tree and the alignment under construction.

PIMA, which stands for Pattern-induced multi-sequence alignment, is an alignment method which uses the progressive alignment technique to align protein sequences. Its workflow can be summarized by the following steps:

1. PIMA first calculates all possible pairwise local alignments between each pair of the input sequences and a distance matrix is then produced.

2. WPGMA (Weighted pair Group Method using Arithmetic averaging) is then used to build a guide tree.

3. Sequences from the leafs of the guide tree are progressively aligned and a pattern is generated for each pairwise alignment.

3.7.1.1 Iterative alignment

The iterative alignment methods are considered as an extension of the progressive alignment methods. Since alignments when produced using a progressive alignment method are fixed and cannot be changed, the iterative methods came to provide a solution for this point. The algorithm behind iterative alignment methods can be summarized by the following steps:

1. An initial alignment is calculated.

2. One sequence is taken from the alignment and re-aligned to the profile of the remaining sequences. Only cases where the score is being optimized are considered, this means that the overall score is increased or stays the same.

3. Step 2 is repeated by choosing another sequence and re-aligning it to the profile of the remaining aligned sequences until the alignment does not change.

The iterative refinement methods are able to generate excellent alignments, but require more computing resources than progressive alignment methods.

FAlign [32] combines the two algorithms: progressive and iterative refinement to align protein sequences.

Another iterative multiple sequence alignment tool is MUSCLE [38] (multiple sequence comparison by log-expectation). The alignment process in MUSCLE is done on three stages (Figure 3.5).

Stage 1: Draft progressive Distances between sequences are estimated using K-tuple distances. Afterwards, UPGMA is used for clustering using the already produced distances. Then a progressive alignment is calculated.

Stage 2: Improved progressive Since the K-tuple distance measure results in suboptimal tree, the tree is re-estimated in this stage using Kimura distances. Kimura is more accurate than K-tuple. It requires an alignment as an input, so, the already produced alignment in the previous stage is used and new distances are calculated. UPGMA is used again for clustering and a new progressive alignment is produced.

Stage 3: Final stage In the last stage, an edge is chosen from the tree produced in the previous stage. Edges are visited in order of decreasing distance from the root. This edge is deleted creating two sub alignments. A profile is then created for each of the produced sub-alignments. Next, the profiles are aligned together and the resultant multiple sequence alignment's SP score is calculated. If the score gets worse, the alignment is discarded. If the SP score improves, the alignment is kept and all the steps of choosing an edge up to aligning the two profile alignments are repeated till convergence is achieved or till a user specified SP score is reached.

3.7.1.2 Profile alignment

When given a multiple sequence alignment, mush useful information can be extracted from the alignment and used later on when a new sequence(s) needs to be aligned to this already existing alignment.This important information constitutes what is known to be a profile, which is a table containing position-specific symbol comparison values and gap penalties.

An alignment of two profiles is a multiple sequence alignment obtained by inserting complete columns of gaps into the first profile or the second one without changing the alignment of any of the two profiles.

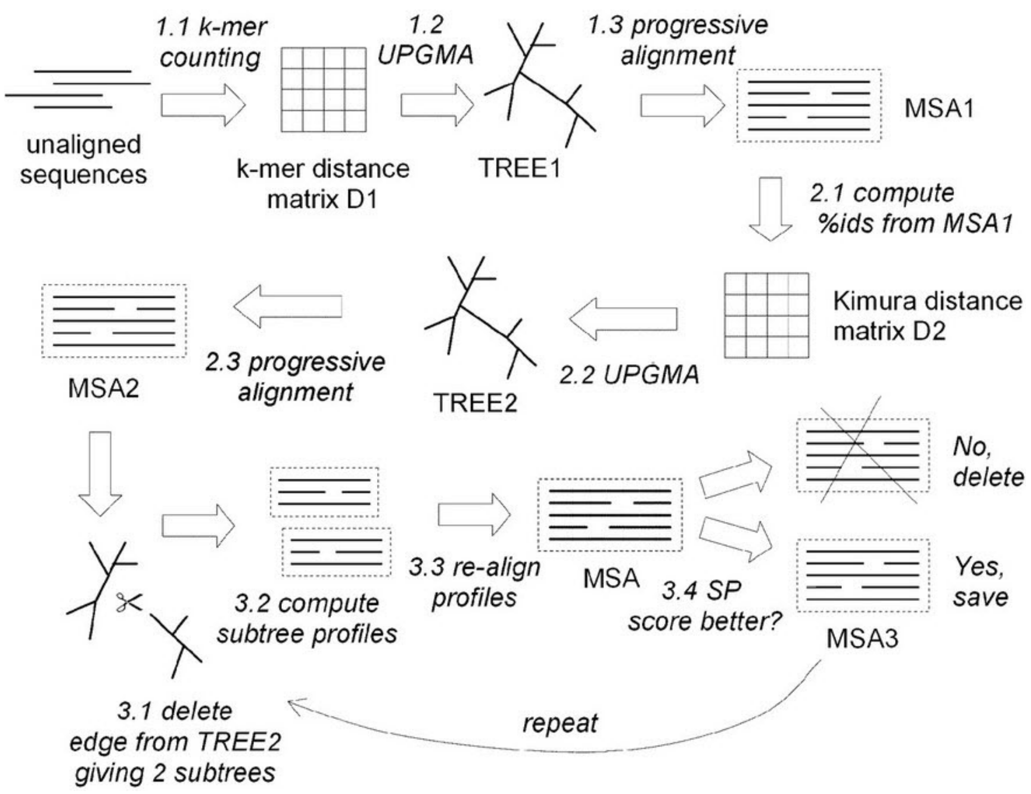

1.1 k-mer counting
1.2 UPGMA
1.3 progressive alignment
unaligned sequences
k-mer distance matrix D1
TREE1
MSA1
2.1 compute %ids from MSA1
Kimura distance matrix D2
2.2 UPGMA
TREE2
2.3 progressive alignment
MSA2
No, delete
Yes, save
3.2 compute subtree profiles
3.3 re-align profiles
MSA
3.4 SP score better?
MSA3
3.1 delete edge from TREE2 giving 2 subtrees
repeat

Figure 3.5: MUSCLE algorithm overview. [Source: Fig. 2 in PMID: 15034147].

CLUSTALW [8] is a profile-based progressive alignment tool. It allows new sequences to be added to an existing alignment without modifying it. Alignments in ClustalW are calculated over three stages:

1. Pairwise sequence alignments are produced for all possible pairs of sequences. Out of those alignments, a distance matrix is produced. The distances are calculated using a fast approximate method [50].

2. A guide tree is calculated from the distance matrix using neighbor-joining [49].

3. The sequences are progressively aligned respecting the order presented in the guide tree.

Other profile methods include PROMALS [9] (PROfile Multiple Alignment with predicted Local Structure). The workflow of PROMALS can be summarized by the following seven steps (Figure 3.6).

- K-tuples method is used to build a guide tree in order to set the alignment order. Building this guide tree requires performing the following steps:
  - K-tuple are identified for each sequence (a K-tuple is a contiguous subsequence of length K).
  - Pairwise distances between sequences are derived from the fraction of K-tuple in common between a given pair of sequences. A distance matrix is produced.
  - UPGMA is used to cluster the distance matrix. A guide tree is produced.
- Highly similar sequences are progressively aligned with a weighted sum-of-pairs measure of BLOSUM62 scores. Two neighboring groups are aligned in this step only if they have an average sequence identity that is higher than a certain threshold (the default threshold is equal to 60). Pre-aligned groups that are relatively divergent from each other are produced in this step.
- Representative sequences are selected from each pre-aligned group. A representative sequence is the longest sequence in a given group.
- The selected sequences from the previous step are processed by PSI-BLAST [10] which will search for homologous sequences from the UNIREF90 database [12]. Hits which have less than 20% identity are removed. PSIPRED [13] is then used to predict the secondary structures using the PSI-BLAST checkpoint file that is produced after the third iteration.
- Using the alignments produced by PSI-BLAST and the secondary structures produced by PSIPRED, profiles are produced and a matrix of posterior probabilities of matches between positions is obtained by forward and backward algorithms of a profile-profile hidden Markov model [9]. Out of those matrices, the scores are calculated [7].
- The scores are used to align the representatives progressively.
- The produced alignments, along with the pre-aligned groups that were obtained in the first step are merged together.
- Gap placement is refined in the alignment produced from the previous step to make the gap patterns more realistic. In order to explain how this is done, two terms should be defined:
  - Core block: a set of consecutive positions with gap content less than 0.5 at each position.

– Gappy region: a set of consecutive positions with gap contents no less than 0.5 at each position.

In the gap refinement stage, continuous gap characters are introduced in between the [l/2]th residue and the (l[l/2])th residue in all the gappy segments, where l is the number of amino acid residues in a given gappy region. Gappy segments in the N- or C-terminus regions are treated in a different way, where a group of continuous gap characters is inserted at the beginning of the sequence or at its end.

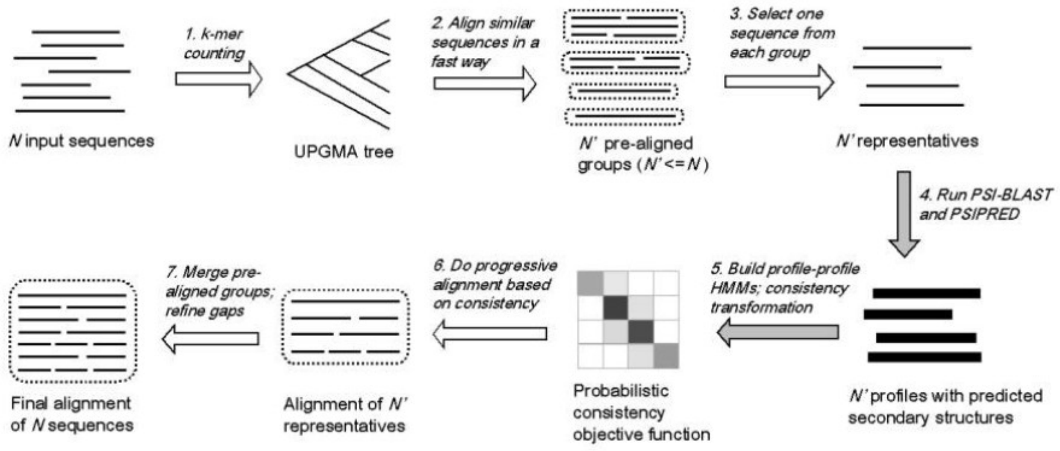

N input sequences
1. k-mer counting
UPGMA tree
2. Align similar sequences in a fast way
N' pre-aligned groups (N'<=N)
3. Select one sequence from each group
N' representatives
4. Run PSI-BLAST and PSIPRED
N' profiles with predicted secondary structures
5. Build profile-profile HMMs; consistency transformation
Probabilistic consistency objective function
6. Do progressive alignment based on consistency
Alignment of N' representatives
7. Merge pre-aligned groups; refine gaps
Final alignment of N sequences

Figure 3.6: Flowchart of PROMALS multiple sequence alignment procedure. Source: [9]

PSI-BLAST (Position-Specific Iterative Basic Local Alignment Search Tool) [10] is another profile alignment tool which provides a mean for detecting distant relationships between proteins. After obtaining a multiple sequence alignment of sequences detected using protein-protein BLAST, PSI-BLAST constructs a position-specific scoring matrix (PSSM) or profile which will be used to further search the database again for new matches. The matrix gets updated at each iteration with the newly detected sequences.

3.8 Tools that integrate external information in the alignment process

One of the alignment tools which make use of the publicly available data is COBALT [14] which is a constraint based alignment tool. It derives information from different sources and then incorporates it in the multiple sequence alignment process. One of the sources that COBALT uses to extract information is databases. COBALT searches databases and extract pairwise constraints. Those databases are the conserved domain database (CDD) [15] and PROSITE protein-motif database. This approach has proved to improve COBALT's alignment quality.

Another example is DBclustal [51], which is a web application that allows to include external information derived from database searches, more precisely from protein BLAST searches, in the alignment process.

T-coffee [22] is another perfect example. Its name stands for Tree based Consistency Objective Function For alignmEnt Evaluation. The basic idea behind T-coffee consists of combining global and local sequence information. The workflow of T-coffee (Figure 3.7) can be divided into five main steps:

- In the first step, T-coffee generates two primary libraries. Those libraries contain pairwise alignments. The first library contains global pairwise alignments for every pair of sequences. Those alignments are calculated by ClustalW [8]. The second library contains local alignments for each pair of sequences. Any given local alignment consists of the top ten scoring non-intersecting local alignments calculated by Lalign [11].

- T-coffee assigns a weight score for every pair of residues in the pairwise alignments included in both libraries. This weight is equal to the average identity between the matched residues in the complete alignment from which this pair comes.

- In the third step, both libraries are combined together into one library. Residue pairs which are common in both libraries are added to the new library as a single entry with a weight equal to the sum of weights of the two original residue pairs. Otherwise, an entry is created for any pair that exists only once in any of the two libraries.

- In the library, a weight is assigned to each pair of residues. The weight value depends directly on the number of sequences supporting the alignment of that pair: the more sequences involved, the higher the weight is. Afterwards, this pair with the new weight is added to an "extended library". This whole process is called library extension.

- Pairwise alignments are used to produce a distance matrix. Using neighbor joining, a guide tree is built which will guide the progressive alignment process later on. The closest pair of sequences in the tree are aligned first using dynamic programming. The weights in the extended library are used to align the sequences. The produced alignment is fixed and the positions where gaps are introduced cannot be altered. Afterwards, the closest pairs of a given entity is aligned together, where an entity can be a sequence or a group of sequences. Thus, either a sequence is aligned with a sequence, a sequence is added to an existing group of aligned sequences, or two groups of aligned sequences are joined together. Then the next closet pair of sequences is aligned, or a sequence is added to the existing alignment of the first two sequences. This continues until all the sequences have been aligned togetehr.

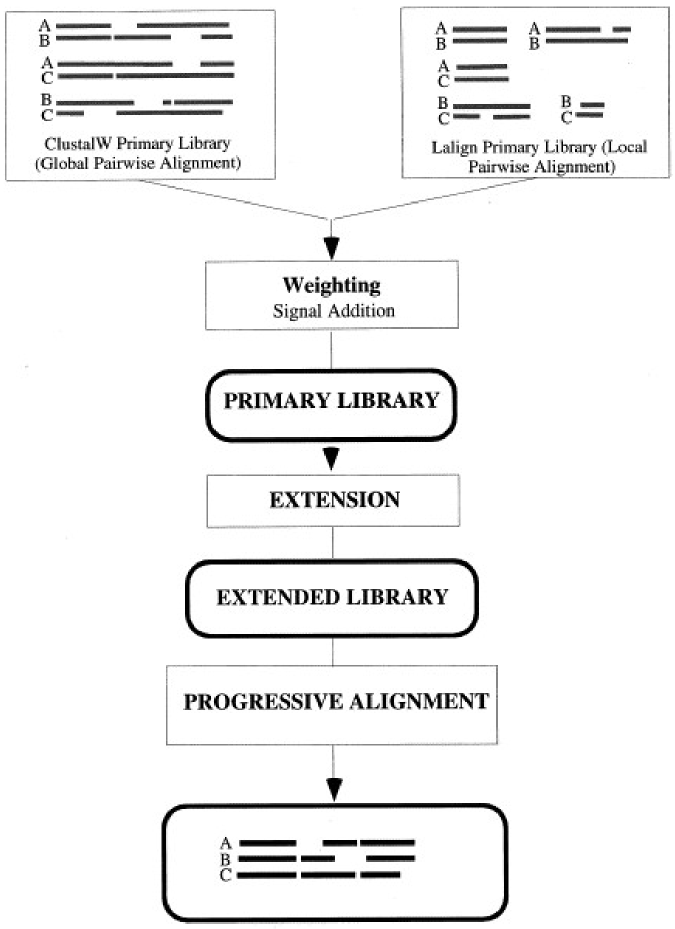

ClustalW Primary Library (Global Pairwise Alignment)
Lalign Primary Library (Local Pairwise Alignment)
Weighting
Signal Addition
PRIMARY LIBRARY
EXTENSION
EXTENDED LIBRARY
PROGRESSIVE ALIGNMENT

Figure 3.7: T-Coffee workflow: step 1: Generating primary libraries for alignments. Step 2: Deriving library weights. Step 3: Combining libraries into single primary library. Step 4: Extending the library. Step 5: Using the extended library for progressive alignment [Source: Fig. 1 in PMID: 10964570]

CLUSTAL Omega [17] is the latest addition to the CLUSTAL family. In order to calculate multiple sequence alignments, Cluctsl Omega first uses a modified version of mBED [53] in order to create a guide tree. Next, it aligns the sequences using the HHalign package [54]. Using Clustal Omega, one can also align new sequences to an existing alignment, or use an already existing alignment to help align new sequences. Clustal Omega has a new feature which allows the incorporation of external information in the alignment process. Using this option, users can add as input, in addition to the sequences to be aligned, a profile HMM that is derived from an alignment of sequences which are homologous to the input set of sequences. The latter will be aligned to the profile to help align them to the rest of the sequences.

Another tool which allows the inclusion of an external sources of information in the alignment process is *DIALIGN* using its anchoring option. This option, which is explained in more details in the *DIALIGN* section, allows users to integrate their own knowledge in the alignment process such that, if the user already knows that certain regions in the input sequences are functionally or evolutionary related and aligning them together will certainly improve the quality of the produced alignment, he/she can input those regions to *DIALIGN* in the form of anchor points. *DIALIGN* will first align all the regions specified by the anchor points and then align the rest of the sequences.

The new approaches presented in this thesis are also based on the idea of integrating external information in the alignment process for the sake of improving the alignment quality. Those new approaches can be considered as new functionalities added to *DIALIGN*. The reason behind choosing *DIALIGN* as a base for our new approaches is the fact that alignments in *DIALIGN* are composed of fragments; we took advantage of this point specifically. For example, fragments with segments that share a common protein domain or pattern are given a higher score.

3.9 Evaluating sequence alignment methods

In order to perform a comprehensive evaluation of an alignment method, benchmark databases containing accurate reference alignments are needed. Two of the widely used benchmark databases for this purpose are BAliBASE [34] and SABmark [35]. The following sections shall give a brief overview about these databases.

3.9.1 BAliBASE

BAliBASE [34] is a database containing manually refined multiple sequence alignments. These alignments are used as reference alignments for the evaluation of sequence alignment tools.

The BAliBASE database contains six main datasets, where each has its own distinguishing characteristics:

RV11 contains 38 families with sequence identity less than 20%.

RV12 contains 44 families with sequence identity between 20% and 40%.

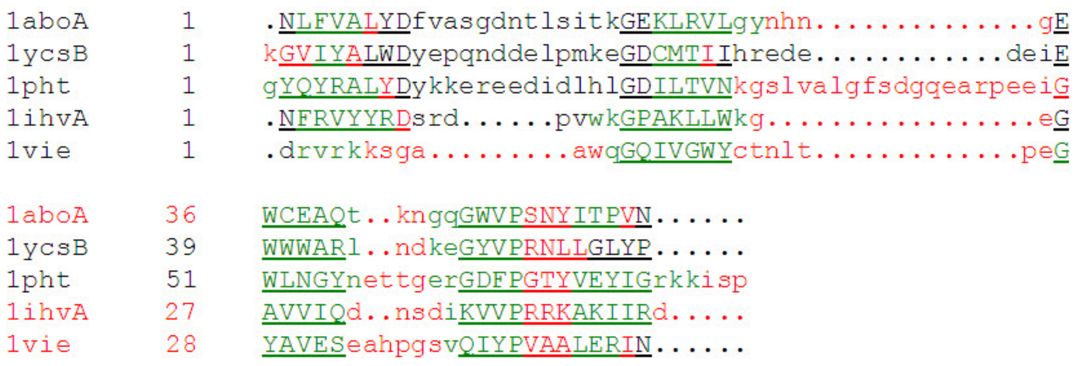
RV20 contains 41 families with sequence identity more than 40%.

RV30 contains 30 families which include some highly diverged sequences.

RV40 contains 49 families with large N/C terminal extensions.

RV50 contains 16 families with large internal insertions.

Each reference alignment in BAliBASE contains a number of core blocks that are considered to be reliably aligned (Figure 3.8). In order to calculate the scores of the alignments produced by any alignment approach, the application bali score provided by BAliBASE 3.0 is used.



```
1aboA      1    .NLFVALYDfvasgdntlsitkGEKLRVLgynhn..............gE
1ycsB      1    kGVIYALWDyepqnddelpmkeGDCMTIIhrede............deiE
1pht       1    gYQYRALYDykkereedidlhlGDILTVNkgslvalgfsdgqearpeeiG
1ihvA      1    .NFRVYYRDsrd......pvwkGPAKLLWkg.................eG
1vie       1    .drvrkksga.........awqGQIVGWYctnlt.............peG

1aboA     36    WCEAQt..kngqGWVPSNYITPVN......
1ycsB     39    WWWARl..ndkeGYVPRNLLGLYP......
1pht      51    WLNGYnettgerGDFPGTYVEYIGrkkisp
1ihvA     27    AVVIQd..nsdiKVVPRRKAKIIRd.....
1vie      28    YAVESeahpgsvQIYPVAALERIN......
```

Figure 3.8: An example of a reference alignment composed of five protein sequences from BAliBASE. The red color corresponds to segments having an alpha helix secondary structure. The green color corresponds to segments a beta strand secondary structure. The underlined segments represents the core blocks which BAliBASE uses in order to calculate the SP and TC scores when evaluating alignments.

Two scoring schemes were used to evaluate a test alignment with a reference alignment of the same sequences:

Sum-of-pairs (SP) is the percentage of residue pairs in the core blocks of the reference alignment that are also correctly aligned in the test alignment.

True-columns score (TC) is the percentage of columns in the core blocks of the reference alignment that are also correctly aligned in the test alignment.

Since most aligners work very good on benchmark databases where the sequences share medium to high similarity, it is preferable also to test the aligners on databases that focus on sequences with low to intermediate similarity. SABmark database can be used for this task.

3.9.2 SABmark

SABmark [35] is an automatically generated benchmark database for multiple protein alignment containing sequences from the SCOP [36] database. SABmark is composed of two large sets:

The twilight zone contains 209 groups of single-domain sequences. Sequences in this set share less than 25% identity.

The superfamilies set contains 425 groups of single-domain sequences. Sequences in this set share about 50% identity.

Two scoring schemes were used for testing against SABmark:

fp score which is equivalent to the SP score used in BAliBASE.

fm score [39] which is defined as the number of residue pairs that are correctly aligned in the test alignment divided by the total number of residue pairs aligned in the test alignment.

3.10 DIALIGN

3.10.1 DIALIGN 1

DIALIGN [19] is a tool for performing pairwise and multiple sequence alignments. In *DIALIGN*'s concept, an alignment, whether pairwise or multiple, is represented by a collection of fragments, where a fragment is a pair of equal length segments from two distinct sequences (Figure 3.9). A fragment is also called a diagonal, that is where the name *DIALIGN* comes from: DIAgonal ALIGNment. The fragments constituting a given alignment should be consistent, i.e. the order of the residues in any sequence in the alignment should be respected (Figure 3.10 and 3.11). Consistency of fragments in a pairwise alignment is met if for any two fragments between a pair of sequences, the end position of one of them is strictly smaller than the respective starting point of the other one. When it comes to multiple sequence alignments, the consistency concept is more complicated [19]. Every fragment is assigned a weight score [27] which is based on the probability of its random occurrence.

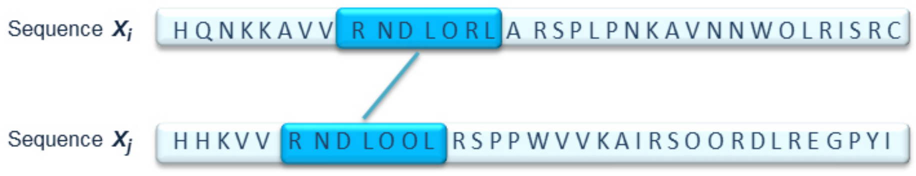

Sequence X_i
H Q N K K A V V R ND L O R L A R S P L P N K A V N N W O L R I S R C
Sequence X_j
H H K V V R ND L O O L R S P P W V V K A I R S O O R D L R E G P Y I

Figure 3.9: An example of a fragment which consists of a pair of segments from sequences X_i and X_j. The length of the fragment is 7. The first segment from X_i contains the subsequence:$RNDLORL$. The second segment from X_j contains the subsequence:$RNDLOOL$.

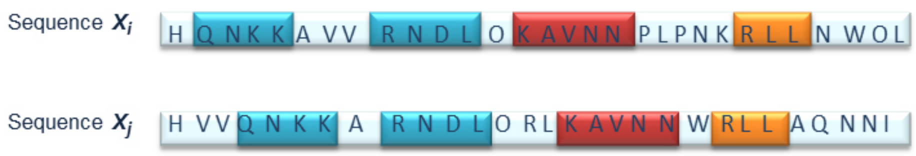

Sequence X_i
H Q N K K A V V R N D L O K A V N N P L P N K R L L N W O L
Sequence X_j
H V V Q N K K A R N D L O R L K A V N N W R L L A Q N N I

Figure 3.10: The figure shows a set of four consistent fragments, two blue, one red and one orange fragment.

DIALIGN 1 uses a weight function in order to give weights for all possible fragment. Denote by F a fragment of length L_F. Let S_F be the sum of similarity scores of all residue pairs of the two segments of F. For protein sequences, the BLOSUM matrix [3] can be used to calculate the similarity scores, while for DNA

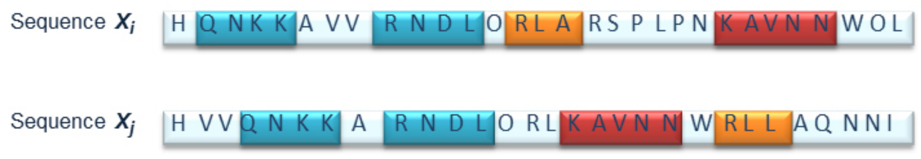

Sequence X_i
H Q N K K A V V R N D L O R L A R S P L P N K A V N N W O L
Sequence X_j
H V V Q N K K A R N D L O R L K A V N N W R L L A Q N N I

Figure 3.11: The figure shows a set of four fragments. The orange and the red fragments are inconsistent with the rest of the fragments since starting positions of the two segments of the orange fragment (or the red fragment) are not strictly less than those of the red fragment (or orange fragment).

sequences, the similarity scores are 1 for matches and 0 for mismatches. Denote by $P_1(L_F, S_F)$ the probability that a random fragment of length L_F has at least the same sum S_F of similarity scores. The weight of F, $w_1(F)$ is then defined by:
$w_1(F) = -logP_1(L_F, S_F)$

DIALIGN 1 tries to find a set of fragments with a maximum sum of weights. Since an alignment in this case might consist of short noisy fragments, a new constraint is introduced. This constraint is a weight threshold T that is used in a way such that all fragments having a weight less than T are discarded.

An alignment produced by *DIALIGN* is defined to be a one having the highest score. The score which *DIALIGN* tries to maximize is defined by the sum of the weights of all the fragments which constitute the alignment.

The main difference between *DIALIGN* and the traditional alignment approaches is the scoring scheme. Most of the traditional methods sum up substitution scores for aligned residues and subtract gap penalties while *DIALIGN* scoring method for alignments depends on the $P-value$ of local sequence similarities. Thus, only segments of sequences that share some significant similarity are aligned, leaving the rest of the sequences unaligned. This justifies the use of *DIALIGN* for comparative genomics [52] since when dealing with genomic sequences, islands of conserved homologies may be separated from non related parts of the sequence. *DIALIGN* has also been used to find protein-coding genes in eukaryotes [24, 25]. Also, *DIALIGN* has been used in studies to analyze protein families [23].

3.10.2 DIALIGN 2

DIALIGN 2.2 [26] is the standard version of the program. An optimal pairwise alignment in *DIALIGN* is defined to be a chain of consistent fragments having the highest sum of fragments scores. To obtain this optimal chain, a simple and space efficient fragment chaining algorithm is used [33].

DIALIGN 2 introduced a new weight function [27]. Using the same notations defined in the section '*DIALIGN 1*', consider the probability $P_2(L_F, S_F)$ now to be the probability of finding any fragment of length L_F whose sum of individual similarity values is at least as large as S_F somewhere within the comparison matrix of two random sequences of the same length as the original sequences. Using this new weight function has reduced the problem of including noisy short fragments in the alignments produced by *DIALIGN*.

3.10.3 Anchoring option of DIALIGN

It often happens that the user has previous knowledge about the sequences in hand and already knows that certain regions in the sequences are functionally or evolutionary related and, if aligned together, will certainly produce a better and biologically more meaningful alignment. Using the Anchoring option of *DIALIGN* [20, 21], this is possible to do. The user can use this option to force the program to align certain regions in the sequences (Figure 3.12). Those regions are referred to as anchor points. An anchor point is a pair of equal-length segments from two distinct sequences. It is characterised by the following six parameters:

- A number which corresponds to the first involved sequence
- A number which corresponds to the second involved sequence
- The starting position in the first sequence
- The starting position in the second sequence
- The length of the anchor point
- The score of the anchor point

Anchoring can be applied to an alignment in two different ways:

- Strong anchoring
- Weak anchoring

Using the strong anchoring option, *DIALIGN* forces the alignment of the segment pairs defined by the anchor points, provided that the anchors are consistent. Afterwards, the rest of the sequences which were not aligned by the anchor points will be aligned by *DIALIGN*.

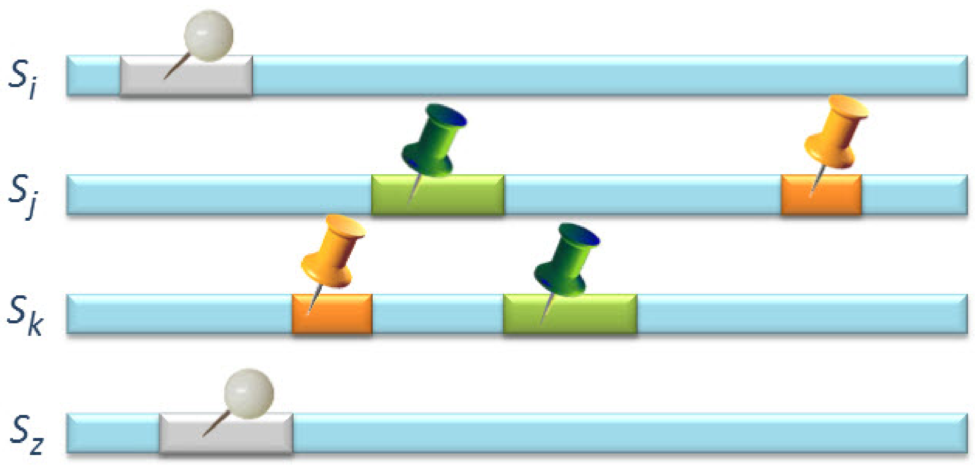


Figure 3.12: The white, orange and green pairs of segments represent three anchor points. Anchoring those three pairs works like spinning those parts together and insuring that they are kept aligned together in the final multiple sequence alignment given that they are consistent, otherwise, *DIALIGN* will greedily select a consistent set of anchor points.

When it comes to the weak anchoring option, *DIALIGN* does not force the alignment of the entire segments provided by the anchor points, but only to do some restrictions in the output alignment. For example, if a pair of residues from the first and second sequences are anchored together, this implies that positions strictly to the left of the first residue from the first sequence are aligned to residues which are strictly to the left of the second position in the second sequence. The same applies to positions to the right of the two residues. Aligning the pair of residues is not strict as mentioned before, it depends on the degree of the similarity between both residues.

Each anchor point is given a weight score by the user. This would prioritise anchors which are biologically more meaningful from the user's point of view. In order to obtain an alignment respecting the constraints defined by the user through the chosen anchor points, *DIALIGN* would first align the regions specified by those anchors, and then align the rest of the sequences. It is very possible that the anchors are not consistent with each other. In this case, a greedy selection of anchors is performed. The anchor with the highest weight is introduced first to the alignment. The next high scoring anchor is introduced into the alignment given that it is consistent with the first anchor, otherwise it is discarded. The next high scoring anchor is chosen and the process is repeated until all the anchor points have been processed.

3.10.4 DIALIGN TX

DIALIGN-TX [28] is a substantial improvement of the previous version of *DIALIGN*. It combines the greedy algorithm used in the previous version with a classical progressive alignment approach to improve the quality of the alignment produced.

Chapter 4

Thesis projects

4.1 DIALIGN-PFAM

4.1.1 Preliminary information

4.1.1.1 Pfam database

The *PFAM* [47] database is one of the most important collections of information for classifying proteins. The database categorizes 75 % of known proteins to form a library of protein families. *PFAM* contains more than 13000 manually curated protein families, where a family is a set of evolutionary-related protein sequences. Every family in *PFAM* is represented by three important files:

1. Seed alignment
2. A profile HMM
3. A multiple sequence alignment

The seed alignment is a manually refined alignment of a representative set of sequences. The profile HMM is built from the seed alignment. The full multiple sequence alignment is generated automatically by searching the swissprot database for detectable members and aligning them to the profile HMM.

Two types of families can be distinguished in *PFAM*: high quality manually curated *PFAM-A* families and automatically generated *PFAM-B* families.

4.1.1.2 Profile hidden markov models

A profile HMM is a linear state machine consisting of a series of nodes. In order to model sequences, each node must have three states:

- Match state
- Insert state
- Delete state

Two types of probabilities are associated with a profile HMM:

- Transition probability: the probability of transitioning from one state to another. A transition might be: match $\rightarrow$ match, match $\rightarrow$ insert, match $\rightarrow$ delete, insert $\rightarrow$ match, etc.
- Emissions probability: it is based on the probabilities of residues existing in that position in the alignment.

Figure 4.1 shows a profile HMM model.

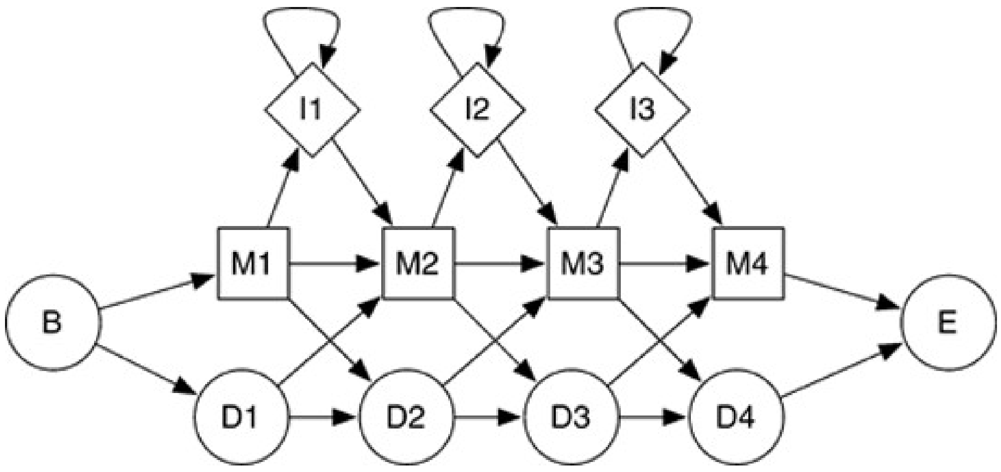


Figure 4.1: General structure of a profile HMM. B: begin state, D: delete state, M: match state, I: insert state, E: end state. Image source: http://cse-wiki.unl.edu/wiki/index.php/Machine_Learning_in_sequence_search

Dynamic programming methods are used to align a sequence to a profile HMM. In order to calculate such an alignment, the most probable path that any gievn sequence might take in a profile HMM is calculated. This is done using the transition and emission probabilities.

4.1.1.3 HMMER

HMMER [45] is a free distributable implementation of profile HMM software for protein sequence analysis. HMMER is used to align protein sequences. It is also used to search a certain sequence against a database of sequences for possible homologs. The HMMER suite contains a collection of useful programs including:

Hmmscan: used to search sequences against collections of profiles. For each sequence S from the input set of sequences, hmmscan searchs S against a target database of profiles and outputs ranked lists of the profiles with the most significant matches to S.

Hmmpress: used to press a database before it can be searched with Hmmscan.

4.1.2 DIALIGN-PFAM algorithm

Sequences belonging to a given *PFAM* family will most certainly share the same evolutionary history, this drove us to come up with our approach *DIALIGN-PFAM* which is based on the idea that segments of sequences matching the same protein domain should preferably be aligned together. The steps that were done can be summarized by the following:

1. Scanning sequences against *PFAM*
2. Building single-domain alignments of *PFAM* matches
3. Anchor points extraction
4. Using single-domain alignments to produce the final multiple sequence alignment

4.1.2.1 Scanning sequences against PFAM

First of all, HMMbuild is used to transfer PFAM into an HMM database. HMMpress is used afterwards in order to compress and index the already produced database. Then, HMMscan is used to scan every sequence from the input set of sequences for possible matches to protein domains in PFAM database.

Hmmscan gives a score to each match a sequence has to a given protein domain from PFAM. In order to decide which hits are taken into consideration by our algorithm, two threshold values for E-values of HMMER hits are used:

1. E_m: associated with matched models
2. E_d: associated with matched domains

The first threshold E_m ensures that only models with E-value less than E_m are taken into consideration. All the models which satisfy the first threshold condition are filtered again with the second threshold E_d for domains. All the domains with E-values less than E_d will be considered by *DIALIGN-PFAM* for the later processing steps. Domains which fail to satisfy the second filter are discarded. *DIALIGN-PFAM* uses the default values of 5x10^{-3} for E_m and 10^{-4} for E_d .

4.1.2.2 Building single domain alignments of PFAM matches

While scanning, segments from sequences matching *PFAM* domains are assembled in multiple lists $M_1 \dots M_p$ (p is the total number of domains which the input sequences have matches to) where each list is associated with one single domain. Figure 4.2 shows an example of such a list which contains three sequences that have matches to the same domain. Figure 4.3 shows the actual alignment of the sequences to the protein domain match consensus.

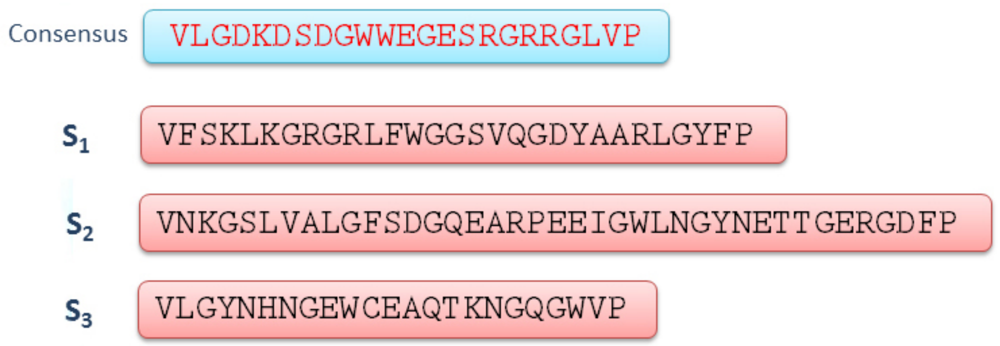
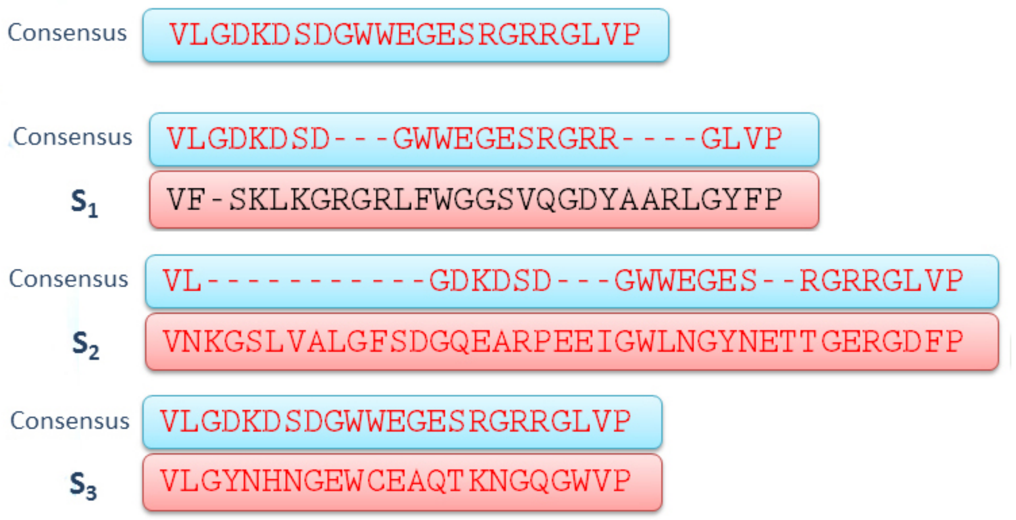

Consensus
VLGDKDSDGWWEGESRGRRGLVP
S1
VFSKLKGRGRLFWGGSVQGDYAARLGYFP
S2
VNKGSLVALGFSDGQEARPEEIGWLNGYNETTGERGDFP
S3
VLGYNHNGEWCEAQTKNGQGWVP

Figure 4.2: In this example, sequences S_1, S_2 and S_3 are found to have a match to the same *PFAM* protein domain. The consensus of the domain is shown in BLUE.

After scanning is finished, for every list M_i where $0 < i < p$, gaps are introduced (See figure 4.4) within the segments of M_i and a sub-alignment is obtained (See figure 4.5). These alignments are called single domain alignments *SDA*. As the name indicates, an *SDA* contains segments from sequences matching a specific protein domain. For each column in an *SDA*, the corresponding positions in the involved segments are required to match the same position of the *PFAM* domain that is associated with this block. Figure 4.5 shows an example of an *SDA*.

Multiple copies of the same protein domain may also exist in the same sequence. In our approach, *DIALIGN-PFAM* considers only the cases where a protein domain is present only once at any given sequence. Thus, when constructing a domain block associated with domain D, all sequences which contain more than one copy of D are ignored. The job of extracting sequence similarity from those ignored sequences segments is handed to the standard version of *DIALIGN*.

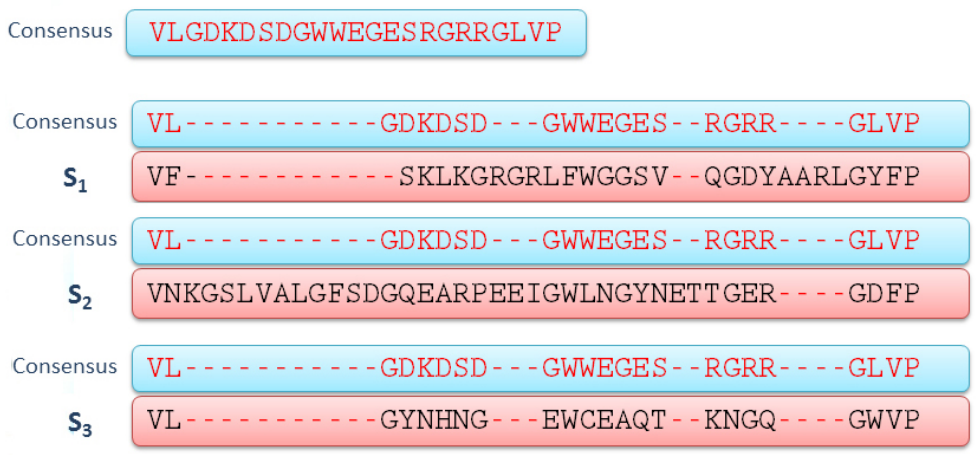
In the previous version of *DIALIGN-PFAM* [29] a domain block is considered to be a set of gap-free equal-length segments from distinct sequences matching the same segment of a *PFAM* protein domain. Thus, for each *PFAM* domain matched



```
Consensus  VLGDKDSDGWWEGESRGRRGLVP

Consensus  VLGDKDSD---GWWEGESRGRR----GLVP
S1         VF-SKLKGRGRLFWGGSVQGDYAARLGYFP

Consensus  VL-----------GDKDSD---GWWEGES--RGRRGLVP
S2         VNKGSLVALGFSDGQEARPEEIGWLNGYNETTGERGDFP

Consensus  VLGDKDSDGWWEGESRGRRGLVP
S3         VLGYNHNGEWCEAQTKNGQGWVP
```

Figure 4.3: The alignments of sequencs S_1, S_2 and S_3 to the matched protein domain consensus.

by the input sequences, we might have multiple domain blocks matching different parts of the protein domain and not one single domain block as it is the case in the current improved version of the algorithm.



```
Consensus  VLGDKDSDGWWEGESRGRRGLVP

Consensus  VL-----------GDKDSD---GWWEGES--RGRR----GLVP
S1         VF------------SKLKGRGRLFWGGSV--QGDYAARLGYFP

Consensus  VL-----------GDKDSD---GWWEGES--RGRR----GLVP
S2         VNKGSLVALGFSDGQEARPEEIGWLNGYNETTGER----GDFP

Consensus  VL-----------GDKDSD---GWWEGES--RGRR----GLVP
S3         VL-----------GYNHNG---EWCEAQT--KNGQ----GWVP
```

Figure 4.4: Gaps are introduced (within the sequences) in order to get the same alignment among all the sequences to the protein domain consensus.

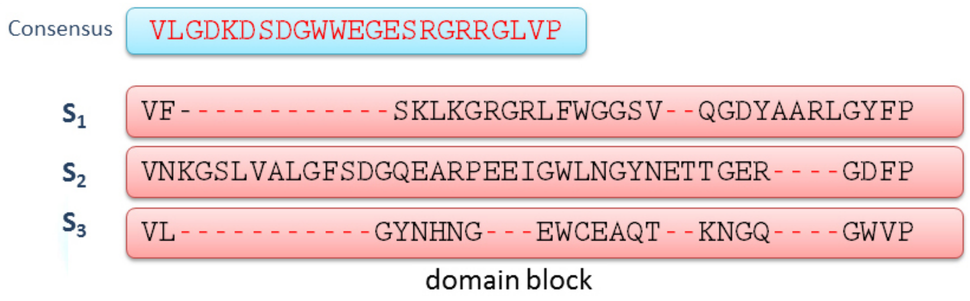

Consensus
VLGDKDSDGWWEGESRGRRGLVP
S1
VF-----------SKLKGRGRLFWGGSV--QGDYAARLGYFP
S2
VNKGSLVALGFSDGQEARPEEIGWLNGYNETTGER----GDFP
S3
VL-----------GYNHNG---EWCEAQT--KNGQ----GWVP
domain block

Figure 4.5: An SDA that contains three segments from three distinct sequences.

4.1.2.3 Anchor points extraction

The goal of this step is to extract one of the smallest possible sets of anchor points 'A' from an SDA such that if the sequences segments constituting the SDA are input to DIALIGN to be aligned along with set A, the same exact SDA will be produced. 'A' should not contain any redundant anchor points, and by redundant we mean an anchor point that has no additional effect on the produced alignment whether it is kept or removed from the list of anchor points (Figure 4.6).

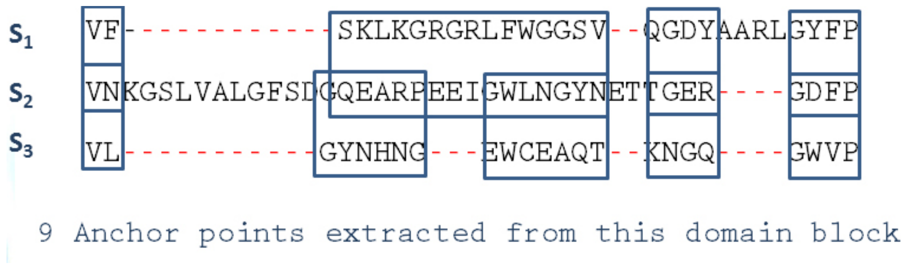

S1
VF-----------SKLKGRGRLFWGGSV--QGDYAARLGYFP
S2
VNKGSLVALGFSDGQEARPEEIGWLNGYNETTGER---GDFP
S3
VL-----------GYNHNG---EWCEAQT--KNGQ---GWVP
9 Anchor points extracted from this domain block

Figure 4.6: Anchor points extraction from an SDA

4.1.2.4 Using single-domain alignments to produce the final multiple sequence alignment

The anchor points produced from the previous step are input to *DIALIGN*. Through the anchoring option of *DIALIGN*, the anchor points are aligned first, then the rest of the sequences are aligned by *DIALIGN* respecting the constraints defined by those anchor points.

4.1.3 Requirements

In order to run *DIALIGN-PFAM*, the following requirements are needed:

- Java 1.3.1 or higher
- *PFAM* database release 27.0 or higher
- HMMER v3.1b1 or higher
- *DIALIGN* 2.2.1

4.1.4 Documentation of the main functions

The following section speaks about the main functions used in the *DIALIGN-PFAM* code.

1. $processFasta(String \quad fileName)$

   This function will check if the file which contains the input set of sequences is in FASTA format. The function takes as a parameter the name of the file containing the sequences to be processed (the name includes the path to the file).

2. $processSequences(String \quad fileName, LinkedList \quad names, String \quad savingDirectory)$

   This function will read the input sequences and then write each one of them into a separate file. *processSequences* takes as an input the following:

   - The file name of the input sequences (including the path).
   - A list where the name of the input sequences will be written in.
   - The path to the directory where the single sequences files will be saved.

3. $scanPfam(String \quad fileName, double \quad E1, double \quad E2, String \quad hmmerPath,$ $String \quad pfamPath, String \quad savingDirectory, int \quad numOfSequences, LinkedList \quad names)$

   This function will do two important steps:

1st step: all the input sequences are scanned against *PFAM* and matches to *PFAM* domains are obtained.

2nd step: all the segments in the output set of matches are distributed into several lists, where each list is associated with one protein domain. For example, if a certain segment has a match to *Piwi* domain, then this segment is put in a list which contains segments matching the *Piwi* domain only.

This function takes as an input the following:

- The name of the file containing the input sequences (the name includes the path).
- The two E-value thresholds used by HMMER to scan *PFAM*.
- The path to HMMER directory.
- The path to *PFAM-A* database.
- The path to the directory where the output of this function will be saved.
- The number of input sequences.
- A list containing the names of the sequences.

4. $extractFasta(String \quad fileName, LinkedList \quad Sequences)$

   This function will extract the segments from a protein domain list and will call the next function *processEachDomain* in order to process this set of sequences.

   This function takes as an input the following parameters:

   - The name of the file containing the segments of sequences matching a certain *PFAM* domain (the name includes the path).
   - A list where this function will put the extracted sequences in.

5. $processEachDomain(String \quad savingDirectory, LinkedList \quad sequences)$

   This function will process each protein domain list produced by the previous function *extractFasta* separately. It will introduce gaps in the segments of each list in order to get a sub-alignment. For more details, check the algorithm of *DIALIGN-PFAM*.

   The function takes the following as parameters:

   - The path to the directory where the output of this function will be saved.

- The list produced by the previous function which contains the segments of a certain protein domain list.

6. $extractAnchors(String \quad seqOne, String \quad seqTwo, int \quad seqOneId, int \quad seqTwoId)$

   This function is applied for every consecutive pair of sequences in an SDA file, for instance, if the file contains 5 sequences S_1 to S_5, then the function *extractAnchors* will process:

   - S_1 and S_2
   - S_2 and S_3
   - S_3 and S_4
   - S_4 and S_5

   The function *extractAnchors* takes as input four parameters:

   - The amino acid residues of the first sequence.
   - The amino acid residues of the second sequence.
   - The id of the first sequence.
   - The id of the second sequence.

   Every anchor point extracted by this fucntion is added to one single file that will contain all the anchor points extracted from all the existing *SDA* files.

7. $runDialign(int \quad flag, String \quad fileName)$

   The final step is to run the alignment program *DIALIGN*. The function *runDialign* takes as input two parameters. The first one is an integer called *flag*. Giving *flag* a value of 2 will order *DIALIGN* to run using its anchoring option. In this way, we can use the anchors file which contains the anchor points extracted from the single domain alignments produced in the previous steps.

   The second parameter is a String which specifies the name of the file (including its path) which contains the sequences to be aligned.

   The output of this step is a file containing the final multiple sequence alignment produced by *DIALIGN*.

4.1.5 DIALIGN-PFAM webserver

A webserver for *DIALIGN-PFAM* has been implemented [30]. Using the webserver, the user can participate more in the alignment process by being involved in the various steps of the *DIALIGN-PFAM* algorithm.

4.1.5.1 Input/Output

DIALIGN-PFAM accepts as its only input a file in FASTA format. This file should contain a set of unaligned protein sequences. Before running *DIALIGN-PFAM*, the user can alter the values of the two thresholds E_m and E_d; knowing that default values are already provided. Since the process of scanning *PFAM* may be time consuming, the user is provided with a URL that can be checked at anytime later. This URL leads to the results page. Moreover, another URL is given to the user in order to retrieve the final MSA. This is useful mostly in cases when the final alignment process by *DIALIGN* consumes a lot of time. The final MSA is downloadable from *gobics.de* and will be stored for a period of one week.

4.1.5.2 Workflow

The workflow of the *DIALIGN-PFAM* webserver can be summarized by the following steps:

1. Scanning sequences against *PFAM*

This step is already mentioned previously when describing the algorithm of *DIALIGN-PFAM*. Figure 4.7 shows an example of a HMMER output when scanning *PFAM*.

2. Domain blocks construction

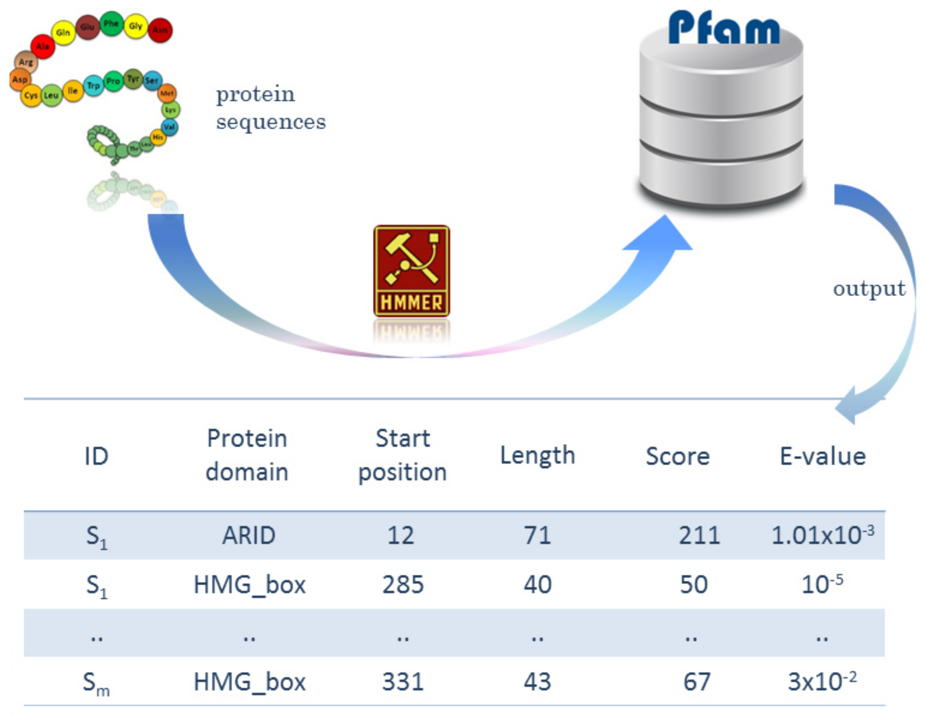
This step is already mentioned previously when describing the algorithm of *DIALIGN-PFAM*.

3. Manual inspection of domain blocks

After all the domain blocks have been extracted, the user has the option to view those blocks in two ways:

- Local view
- Global view

In the local view, only the single segments constituting a given block are shown; note that these segments may contain gaps. The global view shows the non-aligned



ID	Protein domain	Start position	Length	Score	E-value
S_1	ARID	12	71	211	1.01×10^{-3}
S_1	HMG_box	285	40	50	10^{-5}
..	..	..	..	..	..
S_m	HMG_box	331	43	67	3×10^{-2}

Figure 4.7: An example of an output from scanning protein sequences against *PFAM* using HMMER.

full input sequences which has segments matching a certain domain. Those segments are highlighted to make it easier for the user to clearly see where are those segments located exactly in the input sequences.

By default, all constructed blocks will be used in the next step in which the final multiple sequence alignment is calculated. The user can accept this default case or choose a certain number of blocks, from the list of all constructed blocks, to be used in the next step.

4. Anchor points extraction

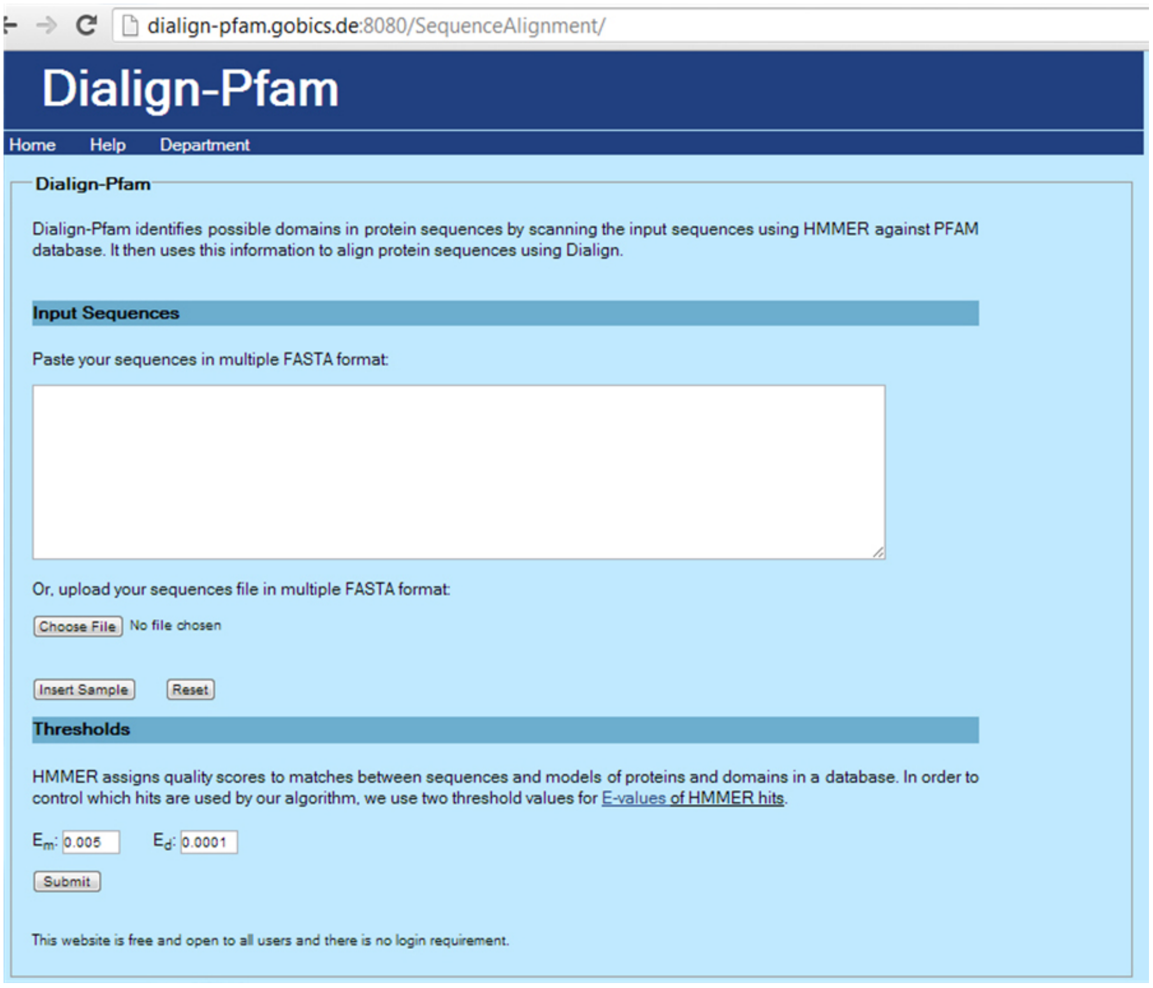
This step is already mentioned previously when describing the algorithm of *DIALIGN-PFAM.*

5. Alignment by *DIALIGN*

This step is already mentioned previously when describing the algorithm of *DIALIGN-PFAM.*

4.1.5.3 Example

On the home page of *DIALIGN-PFAM* (Figure 4.8), a file in FASTA format containing a set of seven protein sequences (Figure 4.9) is uploaded. The first output of running *DIALIGN-PFAM* is represented in Figure 4.10.


dialign-pfam.gobics.de:8080/SequenceAlignment/

Dialign-Pfam

Home Help Department

Dialign-Pfam

Dialign-Pfam identifies possible domains in protein sequences by scanning the input sequences using HMMER against PFAM database. It then uses this information to align protein sequences using Dialign.

Input Sequences

Paste your sequences in multiple FASTA format:

Or, upload your sequences file in multiple FASTA format:

Choose File No file chosen

Insert Sample Reset

Thresholds

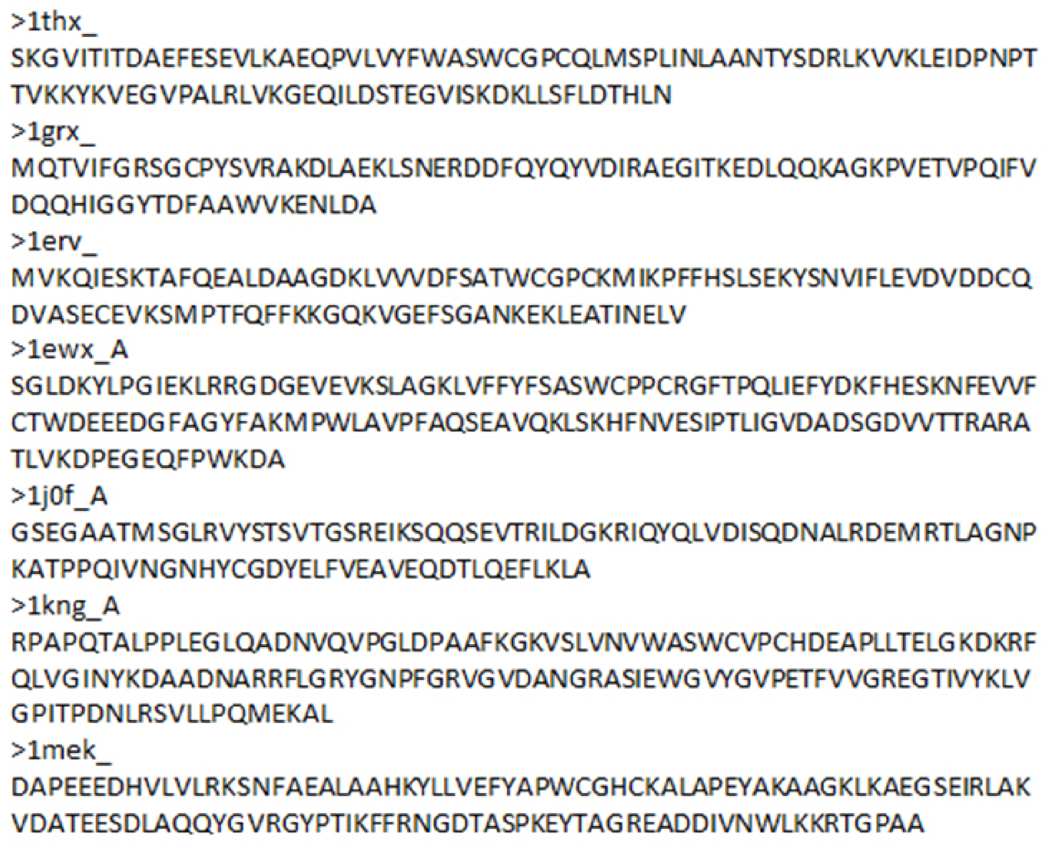
HMMER assigns quality scores to matches between sequences and models of proteins and domains in a database. In order to control which hits are used by our algorithm, we use two threshold values for E-values of HMMER hits.

E_m: 0.005 E_d: 0.0001

Submit

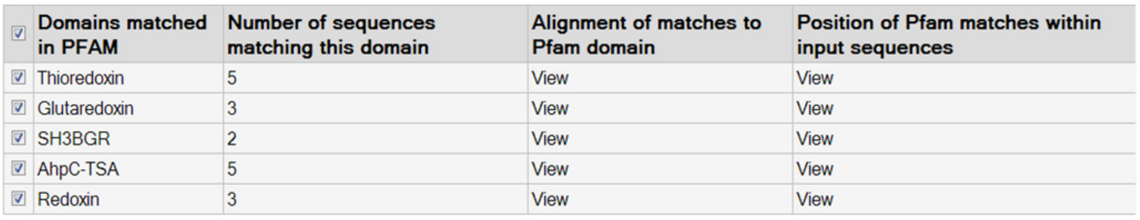
This website is free and open to all users and there is no login requirement.

Figure 4.8: The home page of *DIALIGN-PFAM*. The sequences files should be uploaded in this page. Moreover, the two threshold values should be specified.



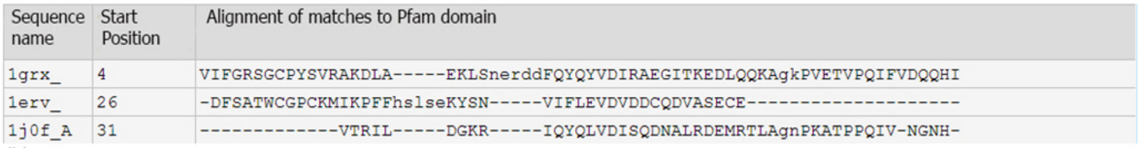
```
>1thx_
SKGVITITDAEFESEVLKAEQPVLVYFWASWCGPCQLMSPLINLAANTYSDRLKVVKLEIDPNPT
TVKKYKVEGVPALRLVKGEQILDSTEGVISKDKLLSFLDTHLN
>1grx_
MQTVIFGRSGCPYSVRAKDLAEKLSNERDDFQYQYVDIRAEGITKEDLQQKAGKPVETVPQIFV
DQQHIGGYTDFAAWVKENLDA
>1erv_
MVKQIESKTAFQEALDAAGDKLVVVDFSATWCGPCKMIKPFFHSLSEKYSNVIFLEVDVDDCQ
DVASECEVKSMPTFQFFKKGQKVGEFSGANKEKLEATINELV
>1ewx_A
SGLDKYLPGIEKLRRGDGEVEVKSLAGKLVFFYFSASWCPPCRGFTPQLIEFYDKFHESKNFEVVF
CTWDEEEDGFAGYFAKMPWLAVPFAQSEAVQKLSKHFNVESIPTLIGVDADSGDVVTTRARA
TLVKDPEGEQFPWKDA
>1j0f_A
GSEGAATMSGLRVYSTSVTGSREIKSQQSEVTRILDGKRIQYQLVDISQDNALRDEMRTLAGNP
KATPPQIVNGNHYCGDYELFVEAVEQDTLQEFLKLA
>1kng_A
RPAPQTALPPLEGLQADNVQVPGLDPAAFKGKVSLVNVWASWCVPCHDEAPLLTELGKDKRF
QLVGINYKDAADNARRFLGRYGNPFGRVGVDANGRASIEWGVYGVPETFVVGREGTIVYKLV
GPITPDNLRSVLLPQMEKAL
>1mek_
DAPEEEDHVLVLRKSNFAEALAAHKYLLVEFYAPWCGHCKALAPEYAKAAGKLKAEGSEIRLAK
VDATEESDLAQQYGVRGYPTIKFFRNGDTASPKEYTAGREADDIVNWLKKRTGPAA
```

Figure 4.9: An example of a protein sequences file that can be uploaded to *DIALIGN-PFAM* webserver. The format of the file is the FASTA format.



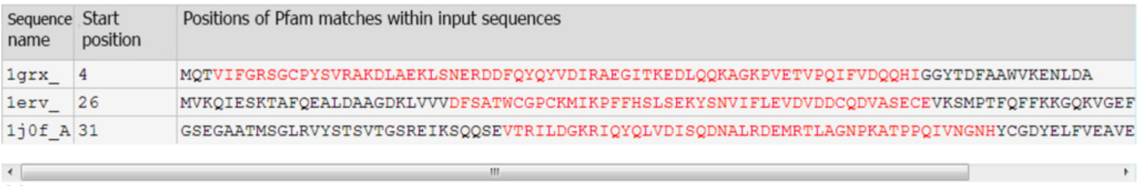
✓	Domains matched in PFAM	Number of sequences matching this domain	Alignment of matches to Pfam domain	Position of Pfam matches within input sequences
✓	Thioredoxin	5	View	View
✓	Glutaredoxin	3	View	View
✓	SH3BGR	2	View	View
✓	AhpC-TSA	5	View	View
✓	Redoxin	3	View	View

(a)



Sequence name	Start Position	Alignment of matches to Pfam domain
1grx_	4	VIFGRSGCPYSVRAKDLA-----EKLSnerddFQYQYVDIRAEGITKEDLQQKAgkPVETVPQIFVDQQHI
1erv_	26	-DFSATWCGPCKMIKPFFhslseKYSN-----VIFLEVDVDDCQDVASECE-------------------
1j0f_A	31	-------------VTRIL-----DGKR-----IQYQLVDISQDNALRDEMRTLAgnPKATPPQIV-NGNH-

(b)



Sequence name	Start position	Positions of Pfam matches within input sequences
1grx_	4	MQTVIFGRSGCPYSVRAKDLAEKLSNERDDFQYQYVDIRAEGITKEDLQQKAGKPVETVPQIFVDQQHIGGYTDFAAWVKENLDA
1erv_	26	MVKQIESKTAFQEALDAAGDKLVVVDFSATWCGPCKMIKPFFHSLSEKYSNVIFLEVDVDDCQDVASECEVKSMPTFQFFKKGQKVGEF
1j0f_A	31	GSEGAATMSGLRVYSTSVTGSREIKSQQSEVTRILDGKRIQYQLVDISQDNALRDEMRTLAGNPKATPPQIVNGNHYCGDYELFVEAVE

(c)

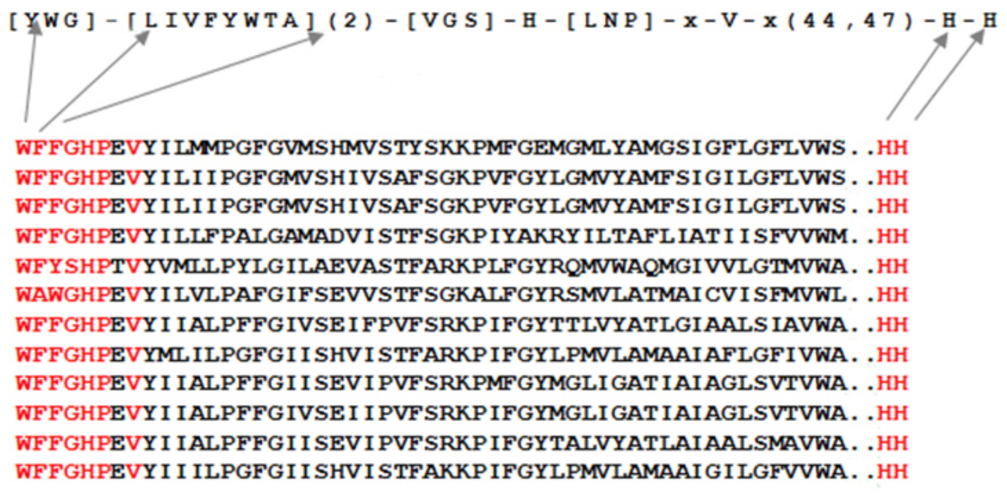
Figure 4.10: An example of running DIALIGN-PFAM with an input file containing seven protein sequences. (a) shows an output produced after running HMMER to scan each of the input sequences against PFAM. Each line in table (a) corresponds to a matched PFAM domain. The first column shows the domain name. The second column indicates how many of the input sequences had a matche to the domain in column one. For example, the first line shows that five sequences had matches to 'Thioredoxin' domain, the second line shows that three sequences had matches to 'Glutaredoxin', ... etc. The third and forth columns allow the user to view the matches of a specific row either locally (third column) or globally (forth column). The checkboxes on the left-hand side can be used to select/deselect matches to *PFAM* domains as anchor points for the final MSA calculated by our program. By default, all matches are selected. (b) shows a multiple sequence alignment of segments from the input sequences matching a given PFAM domain (so-called 'local view'). This is obtained by clicking on 'view' in the third column of (a). (c) is obtained by clicking on 'view' in the forth column of (a).It shows the positions of the matching segments to a certain PFAM domain in the respective full input sequences (so-called 'global view'). Segments matched by HMMER to the corresponding *PFAM* domain are marked in red. Source:[30]

4.2 DIALIGN-PROSITE

4.2.1 Preliminary information

4.2.1.1 PROSITE database

PROSITE [16] is a database of protein domains and families in addition to associated patterns and profiles that help in identifying whether an input sequence belongs to a certain protein family or not. It contains 1308 patterns and 1039 profiles. *PROSITE* is formulated in a way that allows tools to identify rapidly to which protein family a certain sequence belongs to, or what functional sites does it contain. This database has been used by our new approach *DIALIGN-PROSITE* . See figure 4.11 for an example of a pattern along with some sequences which have a match to this pattern.



```
[YWG]-[LIVFYWTA](2)-[VGS]-H-[LNP]-x-V-x(44,47)-H-H

WFFGHPEVYILMMPGFGVMSHMVSTYSKKPMFGEMGMLYAMGSIGFLGFLVWS..HH
WFFGHPEVYILIIPGFGMVSHIVSAFSGKPVFGYLGMVYAMFSIGILGFLVWS..HH
WFFGHPEVYILIIPGFGMVSHIVSAFSGKPVFGYLGMVYAMFSIGILGFLVWS..HH
WFFGHPEVYILLFPALGAMADVISTFSGKPIYAKRYILTAFLIATIISFVVWM..HH
WFYSHPTVYVMLLPYLGILAEVASTFARKPLFGYRQMVWAQMGIVVLGTMVWA..HH
WAWGHPEVYILVLPAFGIFSEVVSTFSGKALFGYRSMVLATMAICVISFMVWL..HH
WFFGHPEVYIIALPFFGIVSEIFPVFSRKPIFGYTTLVYATLGIAALSIAVWA..HH
WFFGHPEVYMLILPGFGIISHVISTFARKPIFGYLPMVLAMAAIAFLGFIVWA..HH
WFFGHPEVYIIALPFFGIISEVIPVFSRKPMFGYMGLIGATIAIAGLSVTVWA..HH
WFFGHPEVYIIALPFFGIVSEIIPVFSRKPIFGYMGLIGATIAIAGLSVTVWA..HH
WFFGHPEVYIIALPFFGIISEVIPVFSRKPIFGYTALVYATLAIAALSMAVWA..HH
WFFGHPEVYIIILPGFGIISHVISTFAKKPIFGYLPMVLAMAAIGILGFVVWA..HH
```

Figure 4.11: A multiple sequence alignment of twelve protein sequences sharing one common pattern of an enzyme called Cytochrome Oxidase I. The pattern signature is represented in the first line. In order for Cytochrome Oxidase I to carry out its function, the enzyme must have this particular signature: [YWG]-[LIVFYWTA](2)-[VGS]-H-[LNP]-x-V-x(44,47)-H-H. A short explanation of this signature: [ ] means any amino acid between the square bracket should be present in the protein sequence, (2): means 2 times, x: means any of the 20 amino acid residues, and x(44,47): means anything (x) occurring 44,45,46 or 47 times.

4.2.1.2 Fragment chaining algorithm of DIALIGN

DIALIGN uses a space-efficient fragment chaining algorithm in order to obtain an optimal alignment out of a chain of fragments. The algorithm and the objective function are explained in details in [33]. A fragment is a pair of equal-length segments from two distinct sequences. Each fragment has a weight that is based on the probability of its random occurrence. The fragment chaining algorithm, in short, calculates pairwise alignments by using dynamic programming to produce the best chain of fragments out from the set of all possible fragments between two sequences, i.e. the chain of non-overlapping fragments having the highest weight. To obtain this optimal chain, a simple and space efficient fragment chaining algorithm is used. Briefly explaining this algorithm:

Let $S= s_1 \dots s_{L_1}$ and $S'= s'_1 \dots s'_{L_2}$ be a pair of sequences where L_1 and L_2 are the length of S and S' respectively. Designate by F a fragment associated with S and S' and let $W(F)$ be the maximum weight taken over all chains of fragments ending in F such that:

$$W(F) := max \left\{ \sum_{i=1}^{K} w(F_i) : F_1 \ll \dots \ll F_k = F \right\}$$

where the notation $F_1 \ll F_2$ means that the ending position of fragment F_1 is strictly less than the starting position of fragment F_2. Define also $P(F)$ to be the predecessor of F such that if $F_1 \ll \dots \ll F_k = F$ is a chain reaching the maximum in the previous equation, then $P(F) = F_{k-1}$.

The last variable to define is $Pr[i, j]$ which is the last fragment in an optimal chain of prefixes $s_1 \dots s_i$ and $s'_1 \dots s'_j$

$$P(F) = Pr[i-1, j-1]$$

So, the score $Sc[i, j]$ [19] of an optimal alignment of the prefixes $s_1 \dots s_i$ and $s'_1 \dots s'_j$ is calculated recursively from the values $Sc[i, j-1]$, $Sc[i-1, j]$, $Sc[i-l, j-l]$, in addition to the weights ending in (i, j). This is indicated by the following formula

$$Sc[i, j] = max = \begin{cases} Sc[i, j-1], \\ Sc[i-1, j], \\ max\,\{W(F) : F \text{ ending in}(i, j)\} \end{cases}$$

Where:

$$W(F) := Sc[i-1, j-1] + w(F)$$

Depending on where the maximum is reached for $Sc[i, j]$, $Pr[i, j]$ can be calculated according to the following equation:

$$Pr[i,j] = max = \begin{cases} Pr[i, j-1], if Sc[i,j] = Sc[i, j-1] \\ Pr[i-1, j], if Sc[i,j] = Sc[i-1, j] \\ F, \text{ if } Sc[i,j] = \{\text{max } W(F) : F \text{ ending in}(i,j)\} \end{cases}$$

In order to obtain the best chain of fragments between S ans S', $Sc[i, j]$ and $Pr[i, j]$ should be calculated for all positions (i, j) in the comparison matrix $(i, j)_{0<i\leq L_1, 0<j\leq L_2}$. When traversing the matrix, and at any given position (i, j) we obtain all possible fragments starting at position s_i in S and s'_j in S'. *DIALIGN* by default has limited the maximum length of any possible fragment obtained at each position to 40. Moreover, and in order not to consider low scoring fragments, any fragment starting with two amino acids which have a low substitution score is ignored.

Once the whole comparison matrix have been processed, the fragment F_{max} with maximum weight $W(F_{max})$ is found such that $F_{max} = Pr[L_1, L_2]$. By tracing back, the optimal chain of fragments is obtained.

This is how *DIALIGN* calculates an optimal pairwise alignment. On the other hand, obtaining an optimal multiple sequence alignment is computationally unfeasible. Therefore, some heuristics are used by *DIALIGN* to accomplish this task. First, *DIALIGN* finds all optimal pairwise alignments for every possible pair of sequences. Fragments constituting those optimal alignments are all assembled in one list according to their scores. *DIALIGN* then, in a last step, chooses greedily a consistent set of fragments from this list to produce the final multiple sequence alignment.

4.2.2 DIALIGN-PROSITE algorithm

A new option for *DIALIGN* called *DIALIGN-PROSITE* has been developed. Using this option, additional information from *PROSITE* database can be incorporated in the alignment process. Developing this new option required using the fragment-chaining algorithm [33] of *DIALIGN* that was explained above but with one modification: the scoring function of this algorithm has been altered such that fragments having matches to the same segment of a *PROSITE* pattern get higher scores.

The *DIALIGN-PROSITE* algorithm can be defined by the following steps:

1. *Scanning sequences against* PROSITE
2. *Filling the 'additional-scores' matrix*
3. *Obtaining the best chain of fragments for every pair of sequences using* PROSITE *matches*

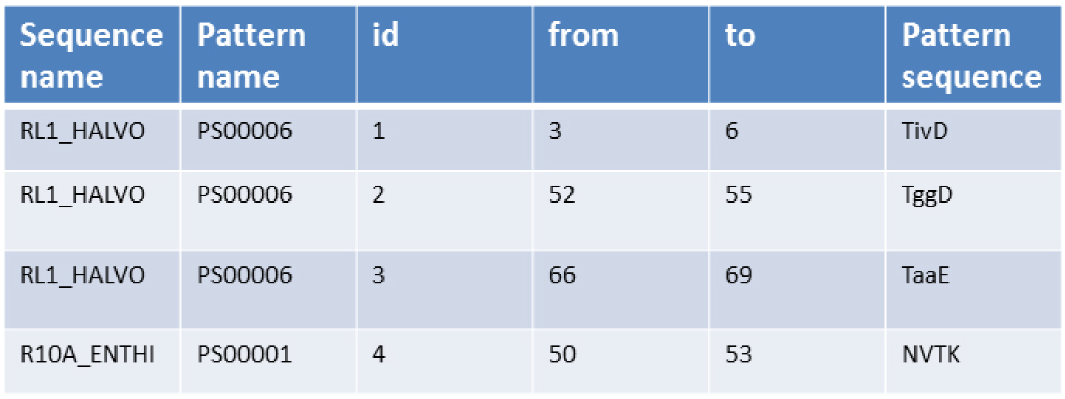
4.2.2.1 Scanning sequences against PROSITE

In the first step of the algorithm, all the input sequences that are intended to be aligned are scanned against *PROSITE* database for possible patterns matches. This is accomplished using a perl script that comes along with the *PROSITE* package. Using this script, every sequence is scanned against the whole *PROSITE* database. For a set of N sequences, the output of the scan of every sequence S_i $(0 < i < N)$ is the following:

- Matched patterns from *PROSITE* in S_i
- The starting position of each matched pattern
- The sequence of amino acids constituting each matched pattern

Figure 4.12 shows an example of a *PROSITE* scan output.

DIALIGN-PROSITE's main goal is to search for common patterns which were matched by two or more sequences and then make use of this information when building the pairwise sequence alignments by giving higher scores to fragments having matches to the same *PROSITE* pattern(s). Following this scenario, patterns which were matched by only one sequence from the set of input sequences are ignored by *DIALIGN-PROSITE* since they are useless. Thus, two matches from two distinct sequences are needed for any pattern in order for it to be considered by *DIALIGN-PROSITE*'s algorithm. Figure 4.13 provides a clarifying example regarding this point.



Sequence name	Pattern name	id	from	to	Pattern sequence
RL1_HALVO	PS00006	1	3	6	TivD
RL1_HALVO	PS00006	2	52	55	TggD
RL1_HALVO	PS00006	3	66	69	TaaE
R10A_ENTHI	PS00001	4	50	53	NVTK

Figure 4.12: The output of scanning four sequences against *PROSITE*. The first column indicates the sequence name. The second column indicates the matched pattern's name. The third column represents the sequence ID. The fourth and fifth columns represent the actual start and end positions of the matched pattern. The last column represents the sequence of the matched pattern. In this example, sequences 1, 2 and 3 were found to have a match to the same pattern PS00006 while sequence 4 had a match to pattern PS00001.

Denote by $P = p_1 \dots p_n$ a set of patterns where two or more sequences had matches to.

Since each sequence might have two or more matches to the same *PROSITE* pattern, denote by $m_{1\{i,n\}} \dots m_{z\{i,n\}}$ the set of matches for sequence S_i to pattern p_n

4.2.2.2 Filling the 'additional-scores' matrix

As it has already been mentioned, the main point of the algorithm is to give higher scores to fragments which have matches to the same *PROSITE* pattern. In order to accomplish this, a matrix called 'additional-scores matrix' is first defined for each pair of sequences. Specific positions in those matrices will be filled in this step and will be used later on during the process of building the pairwise sequence alignments.

Let $M_{i,j}$ be the additional-scores matrix associated with sequences S_i and S_j. In order to fill this matrix, the following steps are performed:

- Suppose sequence S_i had y matches to various *PROSITE* patterns.
- Same for S_j: suppose S_j had z matches to various *PROSITE* patterns.
- For each pair of matches $m_{a\{i,n\}}$ and $m_{b\{j,n\}}$ from S_i and S_j respectively such that:

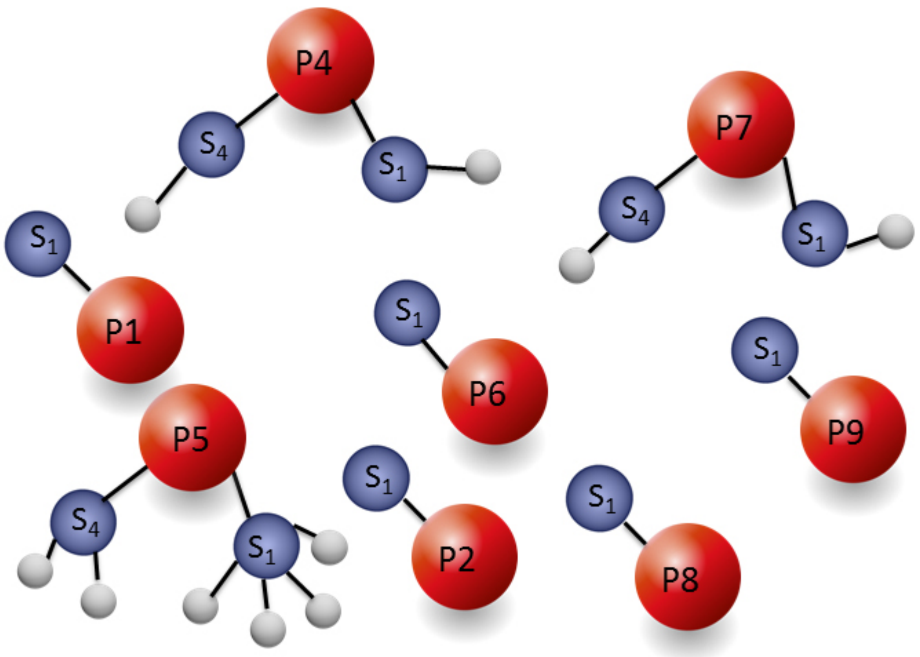


Figure 4.13: In this example, the red balls correspond to patterns that were matched by at least on sequence from the input set of sequences, the blue balls represent the sequences which had matches to certain patterns and the silver balls represent the number of matches a certain sequence had to a certain pattern. For example, Pattern P5 was matched by the sequences S_1 and S_4. S_1 had four matches to P5 and S_4 had 2 matches to P5. Not all of the patterns in this figure are selected for processing by *DIALIGN-PROSITE*, since patterns matched by only one sequence are ignored. In this case, patterns P1, P2, P6, P8 and P9 will be ignored, P4, P5 and P7 will only be considered by *DIALIGN-PROSITE*.

- $0 < a < y$ and $0 < b < z$
- both m_a and m_b are matched to the same pattern P_n (define l_{p_n} to be the length of pattern p_n).
- $m_{a\{i,n\}}$ starts at position u in S_i and $m_{b\{j,n\}}$ starts at position u' in S_j

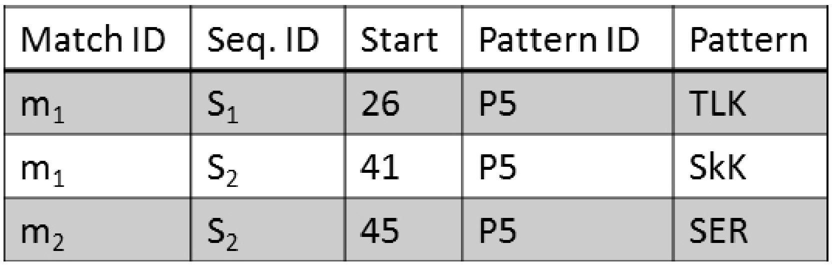
Do the following:

1. In matrix $M_{i,j}$, fill all of the positions between (u, u') and $(u + l_{p_n} - 1, u' + l_{p_n} - 1)$ with blosum scores. For example:
   - position (u, u') should contain the blosum score of the two amino acids from S_i and S_j at positions u in S_i and u' in S_j.
   - position $(u + 1, u' + 1)$ should contain the blosum score of the two amino acids from S_i and S_j at positions $u+1$ in S_i and $u'+1$ in S_j,

and so on.

2. Continue like this until position $(u + l_{p_n} - 1, u' + l_{p_n} - 1)$ is reached.

Figure 4.15 shows an example of how to fill an "additional-scores matrix" associated with a pair of sequences.



Match ID	Seq. ID	Start	Pattern ID	Pattern
m_1	S_1	26	P5	TLK
m_1	S_2	41	P5	SkK
m_2	S_2	45	P5	SER

Figure 4.14: Three patterns matches associated with two sequences are shown in this table. Sequence S_1 has one match to pattern $P5$ at position 26. Sequence S_2 has two matches to pattern $P5$ starting at positions 41 and 45. Information present in this table is used in Figure 4.15 to fill out the additional-scores matrix $M_{i,j}$for the sequences pair (S_1, S_2).

4.2.2.3 Obtaining the best chain of fragments for every pair of sequences using PROSITE matches

For each pair of sequences S_i and S_j, the next step after filling the additional-scores matrix $M_{i,j}$ is to build the best chain of fragments for S_i and S_j.

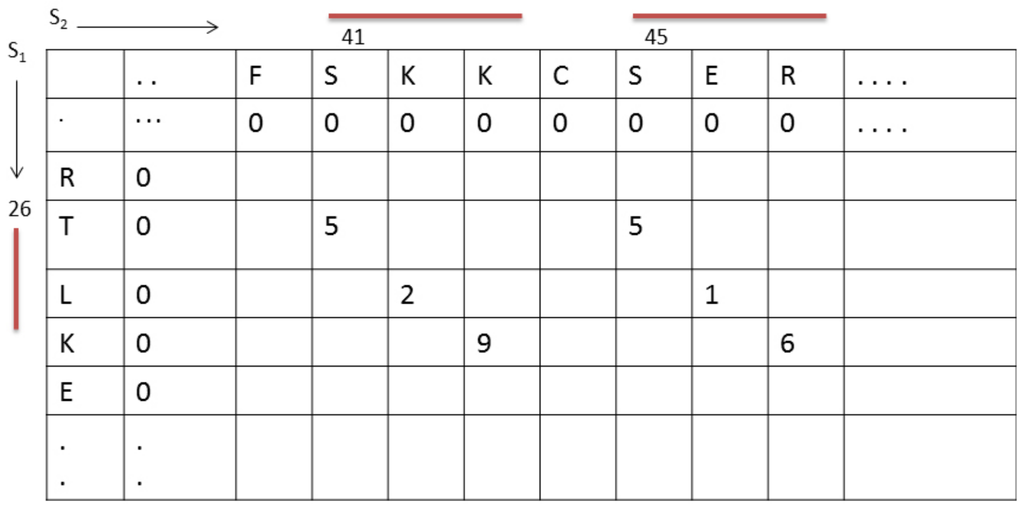
In order to accomplish this, the 'best chain of fragments' algorithm used by *DIALIGN* will be also used by *DIALIGN-PROSITE* with a little modification applied to the scoring function to allow the incorporation of information from the "additional-scores matrix".

The new scoring function is defined as follows:

- Let $F_{i,j}$ be a fragment of length L associated with sequences S_i and S_j
- Denote by t the starting position of $F_{i,j}$ in S_i and by t' the starting position of $F_{i,j}$ in S_j.

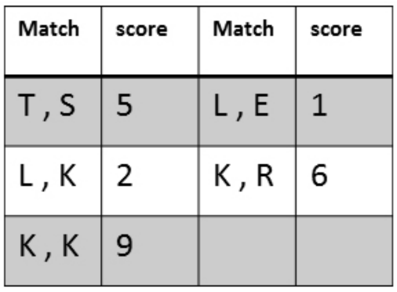
The new weight of $F_{i,j}$ is then defined by :

- $w(F_{i,j}) = w_{old}(F_{i,j}) + w(X)$


S_2
S_1
41
45
26

	..	F	S	K	K	C	S	E	R	....
.	...	0	0	0	0	0	0	0	0	....
R	0									
T	0		5				5			
L	0			2				1		
K	0				9				6	
E	0									
. .	. .									

Figure 4.15: Filling out the "additional-scores matrix" for the sequences pair (S_1, S_2) of Figure 4.14. S_1 is displayed vertically in the first column. Match m_1 of S_1 is represented by a vertical red bar at position 26. S_2 is displayed horizontally in the top row. Matches m_1 and m_2 of S_2 are represented by the two red bars starting at positions 41 and 45 respectively. $M_{i,j}$ will be filled only in the cells where S_1 and S_2 have a match to the same *PROSITE* pattern. This condition holds in this example for the matches m_1 of S_1 and m_1 of S_2. Also, it holds for the matches m_1 of S_1 and m_2 of S_2. As a result, cells $(26, 41)$, $(27, 42)$ and $(28, 43)$ are filled with 5, 2 and 9 respectively for the first matches pair, and cells $(26, 45)$, $(27, 46)$ and $(28, 47)$ are filled with 5, 1 and 6 respectively for the second matches pair. Those numbers correspond to the BLOSSUM substitution scores for the amino acid pairs (T, S), (L, K) and (K, K) (see Figure 4.16 for more information).



Match	score	Match	score
T , S	5	L , E	1
L , K	2	K , R	6
K , K	9		

Figure 4.16: Blosum scores for the amino acid pairs (T, S), (L, K) and (K, K). Those values are used in Figure 4.15 in order to fill the "additional-scores matrix".

where X is the sum of the values in the additional scores matrix $M_{i,j}$ from positions (t, t') to $(t + L - 1, t' + L - 1)$ and $w(X)$ is the weight of the segment $(t, t') \ldots (t + L - 1, t' + L - 1)$ [27] which is based on the probability of its random occurrence.

4.2.2.4 Calculating the final multiple sequence alignment

After obtaining the best chain of fragments for all possible pairs of sequences, the following steps are performed in order to obtain the final multiple sequence alignment:

- For every pair of sequences S_i and S_j, extract all fragments F_1 ... F_n from the best chain of fragments associated with S_i and S_j. Add all the fragments to a list L. in L, the fragments are written using the format of anchor points.
- Run *DIALIGN* using its anchoring option by giving it as an input the list L obtained from the previous step.
- *DIALIGN* will greedily select a consistent set of anchors, add them first to the final multiple sequence alignment, and then align the rest of the sequences.

4.2.3 Requirements

In order to run *DIALIGN-PROSITE*, the following requirements are needed:

- Java 1.3.1 or higher
- *PROSITE* database release 20.105 or higher
- HMMER v3.1b1 or higher
- *DIALIGN* 2.2.1

4.2.4 Documentation of the main functions

The following section speaks about the main functions used in the *DIALIGN-PROSITE* code.

1. $processFasta(StringfileNAme)$

   This function will check if the file which contains the input set of sequences is in FASTA format. The function takes as a parameter the name of the file containing the sequences to be processed (the name includes the path to the file).

2. $extractFasta(StringfileName, LinkedListsequences)$

   This function extracts the sequences from the file of input sequences and place each one in a separate class called Sequence. The function takes as input the following parameters:

   - The name of the file containing the sequences to be processed (the name includes the path to the file).
   - A list where the extracted sequences will be put in.

3. $doCommand(Stringcommand, StringoutputFile, LinkedListsequences)$

   This function scans the the input sequences against the *PROSITE* database and takes as input the following:

   - A string which contains the command that will be called by the *Java* program in order to scan *PROSITE*. The command is:
     "perl ps_scan.pl" +outputFile
     The perl script *ps_scan.pl* scans *PROSITE* and put the output of the scan in the file *outputFile*.
   - The name of the file where the output of the scan will be written in (the name includes the path to the file).
   - A list containing the input sequences. This list is produced by the previous function *extractFasta*.

4. $extractPatterns(LinkedListsequences, StringoutputFile)$

   This function will attach a list of matched patterns for every input sequence. It takes as input the following:

   - The list of input sequences.
   - The name of the file containing the output of the *PROSITE* scan.

5. $fillAdditionalScores(Stringname1, Stringname2, StringseqOne, StringseqTwo, intseqOneId, intseqTwoId)$

   This function fills the "additional-scores matrix" associated with a pair of sequences. It takes as input the following:

   - The name of the first sequence.
   - The name of the second sequence.

- The amino acid residues of the first sequence.
- The amino acid residues of the second sequence.
- The id of the first sequence.
- The id of the second sequence.

6. $processMatrix(Stringname1, Stringname2, StringS1, StringS2, intid1, intid2, Stringsaving)$

   This function fills the dynamic-programming matrix associated with a pair of sequences. The function takes as input the following:

   - The name of the first sequence.
   - The name of the second sequence.
   - The amino acid residues of the first sequence.
   - The amino acid residues of the second sequence.
   - The id of the first sequence.
   - The id of the second sequence.

7. $extractAnchors(StringseqOne, StringseqTwo, intseqOneId, intseqTwoId)$

   This function extracts all possible anchor points from every pair of sequences and writes those anchors in one file.

   The function *extractAnchors* takes as an input four parameters:

   - The amino acid residues of the first sequence.
   - The amino acid residues of the second sequence.
   - The id of the first sequence.
   - The id of the second sequence.

8. $runDialign(intflag, StringfileName)$

   The final step is to run the alignment program *DIALIGN*. The function *runDialign* takes as an input two parameters. The first one is an integer called *flag*. Giving *flag* a value of 2 will order *DIALIGN* to run using its anchoring option. In this way, we can use the anchors file which contains the anchor points extracted from the optimal pairwise alignments produced in the previous steps.

   The second parameter is a String which specifies the name of the file (including its path) that contains the sequences to be aligned.

The output of this step is a file containing the final multiple sequence alignment produced by *DIALIGN*.

4.3 Testing and results

To evaluate our approaches *DIALIGN-PFAM* and *DIALIGN-PROSITE*, we compared them to the standard version of *DIALIGN*, in addition to six other established MSA programs:

- ClustalW 2.1 [8] . Ran with default parameters.
- MAFFT 6.903beta [37] . Ran with the 'linsi' option.
- ProbCons 1.12 [7]. Ran with default parameters.
- MUSCLE 3.8.31 [38]. Ran with default parameters.
- T-Coffee 5.31 [22]. Ran with default parameters.
- COBALT 2.0.1 [14]. Ran with default parameters.

Testing *DIALIGN-PFAM* was carried on two benchmark databases: BAliBASE [34] and SABmark [35]. The results of the tests are displayed in Table 4.1, Table 4.2 and Table 4.3. Tables 4.1 and 4.2 display the SP and TC scores respectively for the testing results on BAliBASE. It is quite obvious that *DIALIGN-PFAM* performed better then the standard version of the program. The winners were Cobalt and ProbCons. The reason behind such scores could most probably be related to the fact that BAliBASE is a database that mainly contains globally related protein sequences, while *DIALIGN* is a local alignment tool.

As an additional observation, while going from RV11 to RV50, i.e. from the twilight zone were sequence similarity is less than 25% to those with similarity that reaches more than 40%, it can be noticed that the scores are improving in general for the different versions of *DIALIGN* and for the other aligners as well. In other words, an increase in accuracy of the alignments produced by the various aligners is reported with increase in sequence identity.

On the other hand, SABmark contains single domain sequences with very low to low and low to intermediate similarity, this explains why the results of *DIALIGN* appears to be almost the best among the other aligners as it can be seen in Table 4.3. The results show that *DIALIGN-PFAM* had the highest 'fm' scores while cobalt had the highest 'fd' scores. Also, the performance of the various versions of

Table 4.1: Performance of different alignment programs on the *BAliBASE* benchmark database. SP scores displayed. Dia+pfam is *DIALIGN-PFAM* using *PFAM* hits as anchor points. Dia+pros is *DIALIGN-PROSITE* using *PROSITE* hits in the fragment chaining algorithm.

Aligner	RV11	RV12	RV20	RV30	RV40	RV50
ClustalW	50.06	86.43	85.16	72.49	78.93	74.25
T-Coffee	57.97	92.49	90.97	79.65	89.19	89.39
COBALT	46.66	79.28	75.0	66.97	83.21	82.41
ProbCons	**66.97**	**94.12**	91.6	84.52	90.33	89.41
Muscle	57.15	91.53	88.91	81.44	86.48	83.51
MAFFT	65.31	93.55	**92.53**	**85.9**	**91.54**	**90.09**
Dialign	50.7	86.6	86.9	74.0	83.3	80.7
Dia+pfam	57.2	87.6	86.8	77.5	86.1	80.7
Dia+pros	50.64	87.8	86.2	74.5	82.15	80.37

Table 4.2: Performance of different alignment programs on the *BAliBASE* benchmark database. TC scores displayed. Notation as in Table 4.1.

Aligner	RV11	RV12	RV20	RV30	RV40	RV50
ClustalW	22.99	80.89	22.16	27.59	39.82	31.15
T-Coffee	31.61	82.1	38.23	36.79	54.81	58.88
COBALT	**61.47**	**90.63**	**88.73**	**79.96**	43.66	45.94
ProbCons	41.96	86.04	41.14	54.72	53.61	57.88
Muscle	32.06	80.89	35.30	41.19	45.31	46.38
MAFFT	43.14	84.31	45.12	58.52	**59.43**	**59.85**
Dialign	26.8	70.0	29.7	31.6	44.5	42.9
Dia+pfam	32.5	71.02	33.6	37.23	47.7	40.3
Dia+pros	26.9	71	32.6	32	42.5	40

DIALIGN performed better on the superfamilies set than on the twilight zone set which is logical, as stated before.

DIALIGN-PROSITE was also tested on BAliBASE and SABmark, the results are shown in Table 4.1, Table 4.2 and Table 4.3. Results on BAliBASE show that this approach had performed better than the standard version of the program in general. Compared to the other aligners, *DIALIGN-PROSITE* was not on the lead, this can be related to the previously mentioned fact that BAliBASE contains globally related sequences while *DIALIGN* is a local alignment tool. On SABmark, excluding *DIALIGN-PFAM* from the results, *DIALIGN-PROSITE* performed better than the standard version of the program and the remaining programs when it comes to the sensitivity scores, while cobalt had the highest specificity scores.

Table 4.3: Performance of the alignment programs on *SABmark*. Notation as in Table 4.1.

Aligner	twi		sup	
	fd	fm	fd	fm
ClustalW	22.46	14.99	50.69	38.03
T-Coffee	23.61	17.9	52.53	41.26
Cobalt	**30.31**	20.64	**58.63**	44.18
ProbCons	28.878	20.8649	56.632	43.5215
Muscle	23.98	16.49	52.756	39.8179
MAFFT	26.432	19.06	55.37	42.487
Dialign	18.707	18.557	46.01	42.354
Dia+pfam	21.89	**24.12**	48.62	**48.48**
Dia+pros	20.86	**20.03**	46.5	**44.4**

4.4 Running time for DIALIGN-PFAM and DIALIGN-PROSITE

In order to check out the running time of *DIALIGN-PFAM* and *DIALIGN-PROSITE* compared to that of *DIALIGN*, we ran those three programs on the two datasets RV11 and RV50 of BAliBASE. Table 4.4 and Table 4.5 show the results.

The time taken by *DIALIGN-PFAM* and *DIALIGN-PROSITE* to scan *PFAM* and *PROSITE* databases respectively is shown separately. It is clear that scanning *PFAM* consumes much less time than scanning *PROSITE*. Also, the time taken by *DIALIGN* to align the sequences in *DIALIGN-PFAM* and *DIALIGN-PROSITE* is shown.

It can be seen that the standard version of the program consumes more time to process the alignments than *DIALGN-PFAM* and *DIALIGN-PROSITE* do, the reason is that *DIALIGN-PFAM* and *DIALIGN-PROSITE* produce a list of anchor points out of the matches to *PFAM* and *PROSITE* databases. Those anchors when given to *DIALIGN* will surely accelerate the alignment process, since after integrating the anchors in the alignment, parts of the alignment will already be built. This reduces the total time taken by *DIALIGN* to produce the final multiple sequence alignment.

The 'Other processing' column of *DIALIGN-PFAM* refers to the time taken to

Table 4.4: Comparison of the runtime of *DIALIGN*, *DIALIGN-PFAM*, and *DIALIGN-PROSITE* on set RV11 of BAliBASE. The Scan column corresponds to the time taken by *DIALIGN-PFAM* and *DIALIGN-PROSITE* to scan *PFAM* and *PROSITE* databases respectively. The Alignment column corresponds to the time taken by *DIALIGN* to produce the aligmnets. The Total column represents the total running time.

Aligner	RV11			
	Scan	Other processing	Alignment	Total
Dialign	-	-	27.896s	27.896s
Dia+pfam	0.014s	77.646s	11.864s	89.524s
Dia+pros	79.22s	69.335s	17.216s	165.771s

Table 4.5: Comparison of the runtime of *DIALIGN*, *DIALIGN-PFAM*, and *DIALIGN-PROSITE* on set RV50 of BAliBASE. Notation as in Table 4.4.

Aligner	RV50			
	Scan	Other processing	Alignment	Total
Dialign	-	-	461.261s	461.261s
Dia+pfam	0.03s	345.208s	142.7s	487.938s
Dia+pros	74.805s	863.553	52.8s	991.158s

check for sequences matches to the same protein domain and extracting anchor points out of them. For *DIALIGN-PROSITE*, this processing time refers to the time taken to check for sequences matches to the same *PROSITE* pattern, building the ”additional-scores matrix”, calculating all possible pairwise alignments (using the algorithm explained in [33] along with some modifications to the fragments scores calculation) and finally extracting anchor points.

4.5 Aligning Alignments with Unaligned Sequences

4.5.1 Motivation

Most of the multiple sequence alignment softwares calculate alignments for a set of sequences from scratch. Yet, researchers often want to calculate a multiple sequence alignment of a sequence set for which a partial alignment is already available. For example, it is possible that a reliable alignment A is already available for a subset of input sequences. In this case, one might want to extend A by adding an additional set of sequences U to it without the need to align the sequences in A and U all together from scratch.

The program *DIALIGN* already contains an anchoring option for this purpose. We have extended the functionalities of this option to allow users to acquire more flexibility when using anchors. A web-based tool and a command line tool have been developed for this. Both accept as an input protein sequence alignments, turn those alignments, or just parts of them (which the user wish to keep fixed in the final multiple sequence alignment), into a set of anchor points and use them to calculate a constrained alignment of the input sequences.

This extended anchoring option is mostly useful when the user already has previous knowledge about the sequences in hand and knows that certain regions are functionally or evolutionary related, and aligning them together would most probably produce an alignment of higher quality.

4.5.2 Webserver

If the user already has one or more multiple protein sequence alignments that must be integrated into one alignment, then it is possible to accomplish this task using our wevserver. There are two cases that should be taken into consideration for each alignment file that will be integrated into the final alignment:

- case 1: keeping the whole alignment fixed as it is in the final multiple sequence alignment.
- case 2: keeping only certain parts of the alignment fixed in the final multiple sequence alignment.

For the second case, the user has to specify the exact positions of those partial alignments (blocks). This will be explained in more details in the next sections.

The user might also wish to align a set of unaligned sequences to an existing alignment/set of alignments; this is also possible. In tis case, the user has to upload those sequences files along with the alignment files in the upload page.

4.5.2.1 Workflow

Four main steps should be done by the user in order to use this webserver efficiently (Figure 4.17):

1. Upload files
2. Specify partial-alignment blocks
3. Submit the blocks
4. View the results

1. Uploading files

The user can submit from one to thirty alignment files and from zero to thirty sequences files (Figure 4.18). The files format should be in Fasta, otherwise an error message is prompted on the screen.

The following conditions should be taken into consideration when uploading the sequences and alignment files in order to avoid error messages:

- Unique sequences names across all uploaded files: The sequences names in all the input files should be distinct since each sequence will be given by the webserver a unique identifier based on its name. In case two or more sequences from distinct files share the same name, the amino acids constituting those sequences should be the same, otherwise an error message is prompted on the screen.

- No special characters: sequences of the input alignment and sequences files should not contain any of the following special characters:'!', '@', '#', '$', '%', 'ˆ', '&', '*', '(', ')', '_', '+', '}', '{', ':', '"', '—', '¡', '¿', '?', '/', '.', ',', ';', '", ' ', ']', '[', '=', '-', '‘', ' '.

- No spaces should exist in the uploaded alignments and sequences files names.

- Unique files names: the uploaded sequences and alignment files should have unique names.

- Avoid uploading empty files.

- Uploaded sequences and alignment files should be in FASTA format.

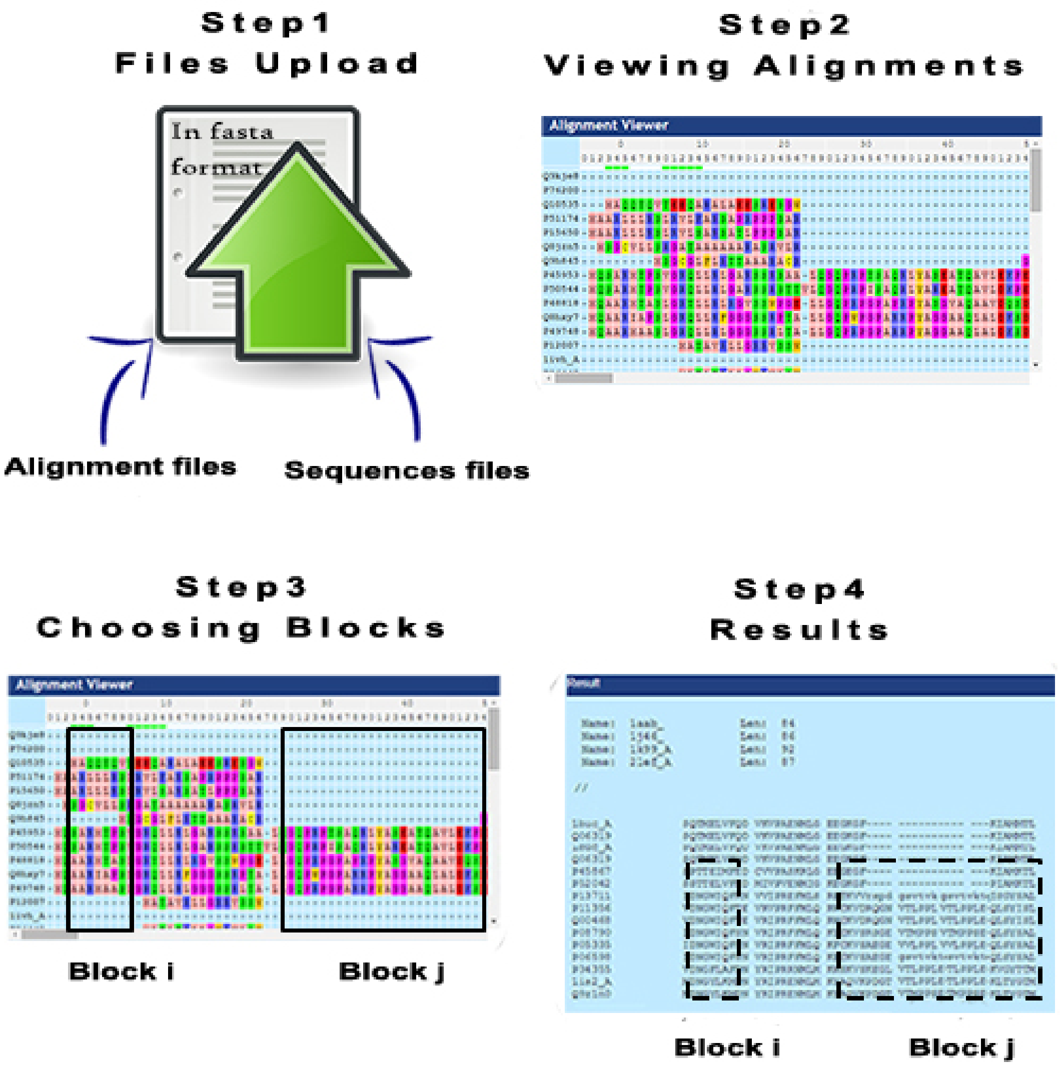


Figure 4.17: Workflow of the webserver

- Avoid empty sequences names.
- At least two sequences should exist in an alignment file, since an alignment file with only one sequence is not actually an alignment.
- All sequences in any uploaded alignment file should have the same length, otherwise, it is not considered an alignment anymore.
- No empty sequences: avoid uploading sequences or alignment files which contain empty sequences, i.e. sequences with defined names but have no amino

acids residues.

2. Specifying partial-alignment blocks start and end positions

After finishing the upload process successfully, graphical representations of the uploaded alignments will be shown to the user (Figure 4.19).

The alignment bases are colored by default with the 'Zappo' coloring scheme where the residues are colored according to their physicochemical properties. The user can choose between eight different coloring schemes: Zappo, Taylor, Clustal, Helix, Strand, TurnX, HPhob and Buried.

A horizontal bar containing numbers lies above every alignment. This bar allows the user to know the respective position of any column in the alignment.

The second step is to specify which blocks in the alignment should be kept fixed in the final multiple sequence alignment. In order to choose a block, the user has to determine the start and end positions of the block by clicking on the associated positions on the horizontal coordinates bar. The positions will be saved automatically in the textarea below the alignment (Figure 4.20). The end position of any block should be greater than or equal to the start position of that block, otherwise, an error message is prompted on the screen

If the user wants to include the whole alignment fixed as it is in the final multiple sequence alignment, he/she can do any of the following two options:

- Do not specify any start/end coordinates. The webserver will then consider the default case and treats the whole alignment as one single block.
- Specify 0 as the start coordinate and the length of the alignment minus one as the end coordinate.

3. Submit the blocks

After processing all the uploaded alignments and choosing the start and end positions of the partial alignment blocks, the webserver processes this information by doing the following steps:

1. Extracting all the possible anchor points from the selected blocks.
2. Inputting the anchors to *DIALIGN* and running it using the anchoring option.

In case the alignment process needs a long time to finish, the user will be prompted with a URL that can be used at any time later to check if the multiple sequence alignment has already been calculated or not. The result file will be kept on the webserver for a period of ten days.

3. View the results

In the final step, the user can view the result alignment.

4.5.3 Command-line version

A command line tool for aligning alignments also exists. It has the same functionality as the webserver, except that the user cannot view the alignments graphically. Therefore, the start and end positions of the partial-alignment blocks should be input to the program via a file.

4.5.3.1 Input and output

The command line tool accepts as an input the following files

- Alignment files
- Sequences files
- Coordinates files. One Coordinate file should exist for every input alignment file

The alignment and sequences files should be in FASTA format. The coordinates files format should respect the following rules:

- Every line in the file should contain two numbers: start and end coordinates of a partial-alignment block. Both numbers should be separated by a space.
- The end coordinate should be larger than or equal to the start coordinate.
- All numbers present in the file should be greater than zero and less than or equal to the alignment length.

The output of running the command line tool is an alignment file in fasta format.

4.5.4 Requirements

In order to run the command-line version for "Aligning alignments with un-aligned sequences", the following requirements are needed:

- Java 1.3.1 or higher
- *DIALIGN* 2.2.1

4.5.5 Documentation of the main functions

The following section speaks about the main functions used in the code of "Aligning alignments with un-aligned sequences".

1. $main(String[]\quad args)$

   This is the main function of the program, it takes as input the following set of parameters:

   - The name of sequences files (the name includes the path to the file). Those files should contain un-aligned sequences in FASTA format. This parameter is not obligatory.
   - The name of one or more alignment files (the name includes the path to the file). The file should be in FASTA format.
   - The name of the coordinates files (the name includes the path to the file). Each alignment file should have one coordinates file. The format of the coordinates file are explained in the program algorithm section.

   The order of the parameters of the main function should be as follows:

   $$usf_1 \ldots usf_n \quad af_1cf_1 \ldots af_mcf_m$$

   usf: un-aligned sequences file.
   n: number of un-aligned sequences files.
   af: alignment file.
   cf: coordinates file.
   m: number of alignment files/coordinates files.

2. $processFasta(String \quad fileName)$

   This function will check if the file which contains the input set of sequences is in FASTA format. This includes all the various checks mentioned earlier including special characters, sequence name duplicates, empty sequences, ... etc. The function takes as a parameter the name of the file containing the sequences to be processed (the name includes the path to the file).

3. $extractFasta(String \quad fileName, LinkedList \quad sequences)$

   This function extracts the sequences from the file of input sequences and put each one in a separate class called Sequence. The function takes as input the following parameters:

   - The name of the file containing the sequences to be processed (the name includes the path to the file).
   - A list where the extracted sequences will be placed in.

4. $extract\_coordinates(String \quad coordinatesFileName, LinkedList \quad coordinates)$

   This function will extract the start and end positions from a certain coordinates file and load them into memory to be used by the next function *extract_new_alignments*. The function takes as input the following:

   - The name of the coordinates file.
   - A list of Coordinate classes, where the start and end positions will be loaded in.

5. $extract\_new\_alignments(LinkedList \quad alignment, LinkedList \quad coordinates, LinkedList \quad newAlignments)$

   This function will be applied on every alignment file. Using the coordinates start and end positions associated with the alignment file, a set of sub-alignments (blocks) will be extracted. The function takes as input the following:

   - A list containing sequences associated with an alignment file.
   - The coordinates associated with the alignment file that is being processed.
   - A list where the newly extracted sub-alignments will be loaded in.

6. $extractAnchorsNew(LinkedList \quad newAlignments, String \quad anchorsFile)$

   This function will be applied to each extracted sub-alignment produced by the previous function *extract_new_alignments*. The main job of this function is to extract anchor points from the sub-alignments and write them in one file. The input parameters of *extractAnchorsNew* are:

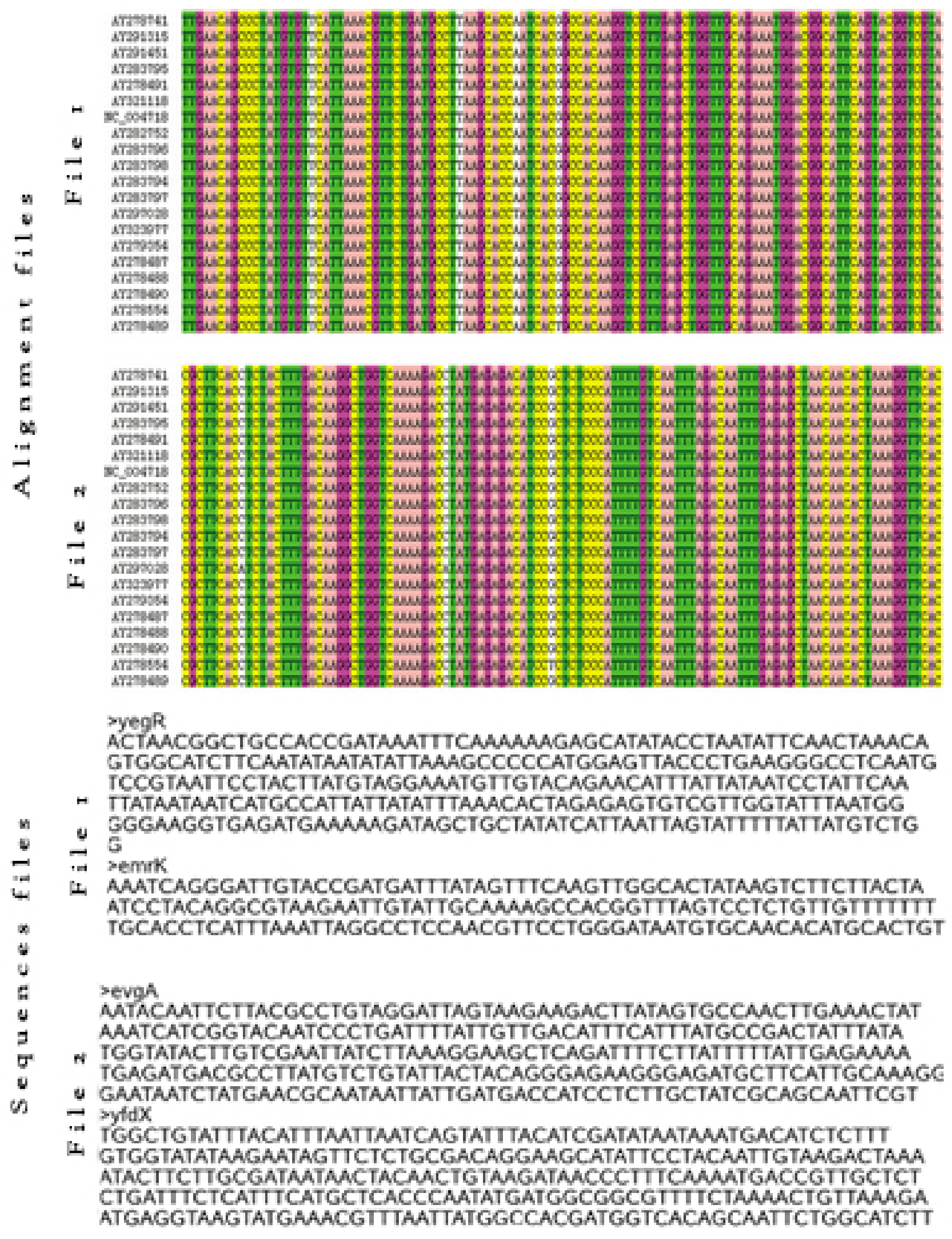
   - A list of sub-alignments which are extracted by the previous function *extract_new_alignments*.
   - The name of the file where the extracted anchor points will be written in.

7. $runDialign(String \quad AllSequences, String \quad resultsFile)$

   The final step is to run the alignment program *DIALIGN*. The function *run-Dialign* takes as an input two parameters. The first one is an integer called *flag*. Giving *flag* a value of 2 will order *DIALIGN* to run using its anchoring option. In this way, we can use the anchors file which contains the anchor points extracted from the specified partial-alignment blocks.

   The second parameter is a String which specifies the name of the file (including its path) that contains the sequences to be aligned.

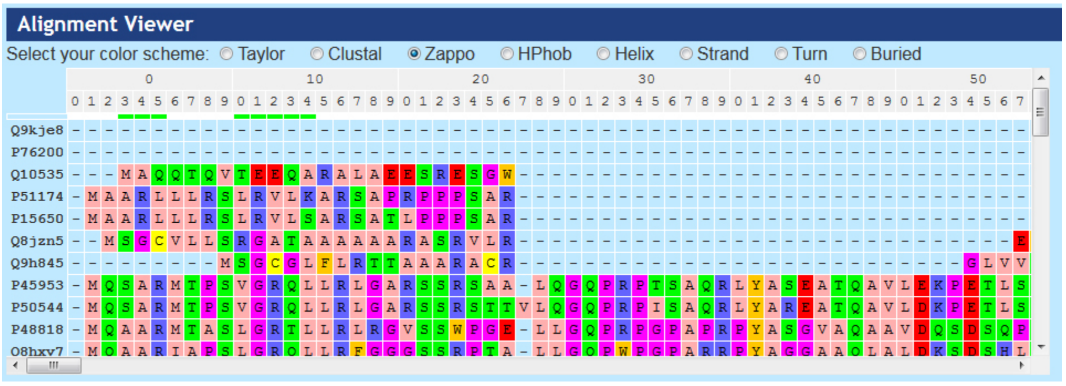
   The output of this step is a file containing the final multiple sequence alignment produced by *DIALIGN*.



```
>yegR
ACTAACGGCTGCCACCGATAAATTTCAAAAAAGAGCATATACCTAATATTCAACTAAACA
GTGGCATCTTCAATATAATATATTAAAGCCCCCATGGAGTTACCCTGAAGGGCCTCAATG
TCCGTAATTCCTACTTATGTAGGAAATGTTGTACAGAACATTTATTATAATCCTATTCAA
TTATAATAATCATGCCATTATTATATTTAAACACTAGAGAGTGTCGTTGGTATTTAATGG
GGGAAGGTGAGATGAAAAAGATAGCTGCTATATCATTAATTAGTATTTTTATTATGTCTG
G
>emrK
AAATCAGGGATTGTACCGATGATTTATAGTTTCAAGTTGGCACTATAAGTCTTCTTACTA
ATCCTACAGGCGTAAGAATTGTATTGCAAAAGCCACGGTTTAGTCCTCTGTTGTTTTTTT
TGCACCTCATTTAAATTAGGCCTCCAACGTTCCTGGGATAATGTGCAACACATGCACTGT
```

```
>evgA
AATACAATTCTTACGCCTGTAGGATTAGTAAGAAGACTTATAGTGCCAACTTGAAACTAT
AAATCATCGGTACAATCCCTGATTTTTATTGTTGACATTTCATTTATGCCGACTATTTATA
TGGTATACTTGTCGAATTATCTTAAAGGAAGCTCAGATTTTCTTATTTTTTATTGAGAAAA
TGAGATGACGCCTTATGTCTGTATTACTACAGGGAGAAGGGAGATGCTTCATTGCAAAGG
GAATAATCTATGAACGCAATAATTATTGATGACCATCCTCTTGCTATCGCAGCAATTCGT
>yfdX
TGGCTGTATTTACATTTAATTAATCAGTATTTACATCGATATAATAAATGACATCTCTTT
GTGGTATATAAGAATAGTTCTCTGCGACAGGAAGCATATTCCTACAATTGTAAGACTAAA
ATACTTCTTGCGATAATAACTACAACTGTAAGATAACCCTTTCAAAATGACCGTTGCTCT
CTGATTTCTCATTTCATGCTCACCCAATATGATGGCGGCGTTTTCTAAAACTGTTAAAGA
ATGAGGTAAGTATGAAACGTTTAATTATGGCCACGATGGTCACAGCAATTCTGGCATCTT
```

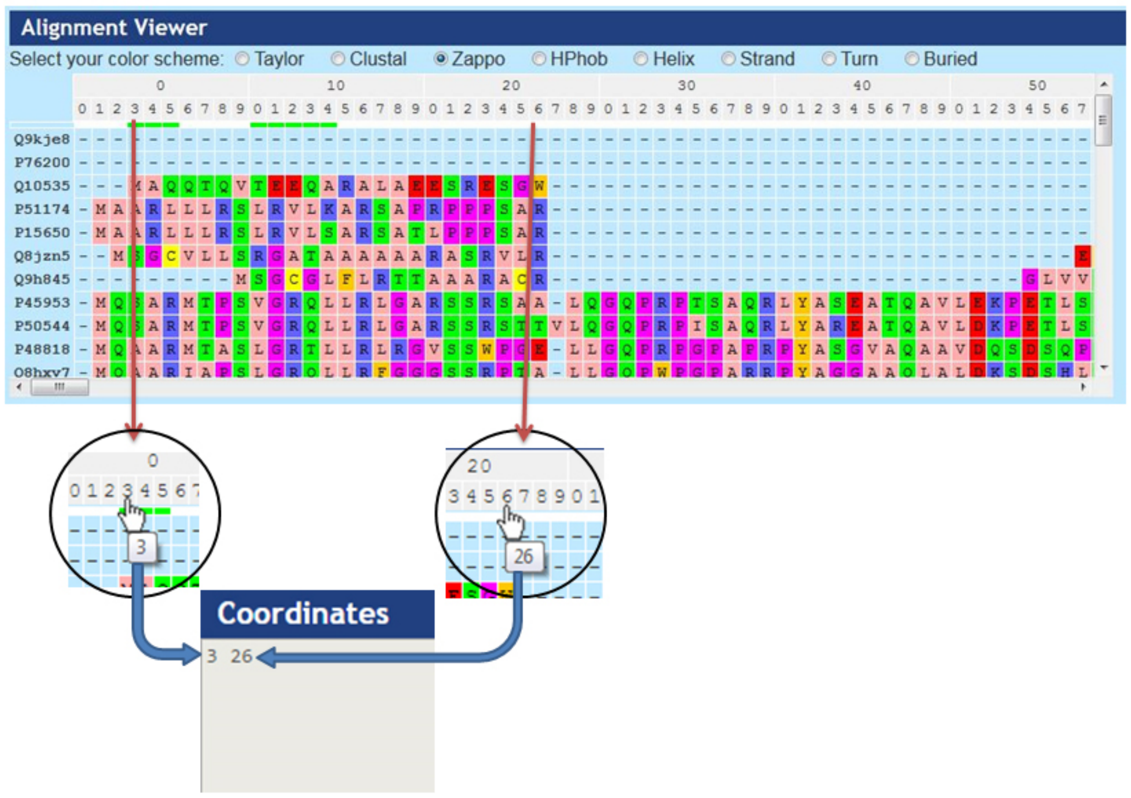
Figure 4.18: Two types of files can be uploaded to the webserver: Alignment files and sequences files. One alignment file at least should be uploaded. The sequences files contain additional sequences which the user wishes to add (align) to the already uploaded alignment files. In this figure, we see as an example 4 uploaded files: 2 alignment files and 2 sequences files.



Select your color scheme: Taylor Clustal [✓ Zappo] HPhob Helix Strand Turn Buried

```
                 0                   10                  20                  30                  40                  50
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
Q9kje8   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
P76200   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Q10535   - - - M A Q Q T Q V T E E Q A R A L A E E S R E S G W - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
P51174   - M A A R L L L R S L R V L K A R S A P R P P P S A R - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
P15650   - M A A R L L L R S L R V L S A R S A T L P P P S A R - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Q8jzn5   - - M S G C V L L S R G A T A A A A A A R A S R V L R - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - E
Q9h845   - - - - - - - - - M S G C G L F L R T T A A A R A C R - - - - - - - - - - - - - - - - - - - - - - - - - - - G L V V
P45953   - M Q S A R M T P S V G R Q L L R L G A R S S R S A A - L Q G Q P R P T S A Q R L Y A S E A T Q A V L E K P E T L S
P50544   - M Q S A R M T P S V G R Q L L R L G A R S S R S T T V L Q G Q P R P I S A Q R L Y A R E A T Q A V L D K P E T L S
P48818   - M Q A A R M T A S L G R T L L R L R G V S S W P G E - L L G Q P R P G P A P R P Y A S G V A Q A A V D Q S D S Q P
Q8hxv7   - M Q A A R I A P S L G R Q L L R F G G G S S R P T A - L L G Q P W P G P A R R P Y A G G A A Q L A L D K S D S H L
```

Figure 4.19: Graphical representation of a multiple sequence alignment. Residues are colored using the Zappo coloring scheme. The user can scroll horizontally and vertically to view the rest of the alignment.



Alignment Viewer

Select your color scheme: Taylor Clustal [✓ Zappo] HPhob Helix Strand Turn Buried

Coordinates

3 26

Figure 4.20: This figure shows how to choose the start/end coordinates of blocks. Suppose the user wants to choose the block which is between the two vertical red arrows, then he/she should click on number 3 for the starting position and number 26 for the end position of the chosen block.

Chapter 5

Conclusion and Future Perspective

5.1 General conclusion

Most methods for multiple protein alignment are based on primary-sequence similarity alone. In this thesis, the main concept that was being investigated is testing the effect of incorporating additional sources of information in the alignment process. The development of all the approaches presented was driven by the hypothesis which states that using external input from the user or from the available data deposited in the public and private databases and integrating them in the alignment process would certainly produce alignments which have a better quality and are biologically more meaningful.

To date, only few MSA programs can include external information in addition to primary sequences in the alignment process. As a base for my approaches, I used *DIALIGN* since this program has an anchoring option which allows the user to specify positions of the input sequences that are to be aligned.

DIALIGN-PFAM and *DIALIGN-PROSITE* have been tested against benchmark databases. The results proved that relying on external sources of information in addition to the input sequences has improved the alignment scores and quality. In principle, it should be possible to use other MSA methods in the same fasion and add a term for external homology information to the commonly used substitution scores.

The *DIALIGN-PFAM* and "Aligning alignments with un-aligned sequences" webservers provide additional functionality for users when dealing with sequence alignments through interactive visualization of various steps in the workflow of both webservers.

5.2 Future Perspective

5.2.1 Development of an interactive webserver for DIALIGN-PROSITE

The idea behind this webserver is to make the user interactively involved in the various processing steps of *DIALIGN-PROSITE*. After uploading the sequences file to the webserver, a *PROSITE* database scan should be performed. Afterwards, the user should get a graphical representation of all the input sequences with highlighted segments that correspond to parts of sequences that match a *PROSITE* pattern. The user has the option to include all those matches for processing in the next step, or discard some. The remaining steps will be done automatically by the webserver. Firstly, the "additional-scores matrix" should be built, then pairwise alignments will be calculated for every pair of sequences, and finally, anchor points will be extracted and input to *DIALIGN* to perform the final multiple sequence alignment.

5.2.2 Improvements for the Anchored-Alignment webserver

There are some additional functionalities that can be added to this webserver in order to make it more effective and flexible to use. After uploading the files (alignment files and un-aligned sequences files) and getting a graphical representation of the alignments, the user must choose start and end positions of the partial alignment blocks he/she wishes to keep fixed in the final multiple sequence alignment. The current scenario is that the partial alignment blocks for a certain alignment must include segments from every sequence in this alignment. As an improvement, the user should be able to switch off some of the sequences from the alignment (for example switch off sequences which share low similarity with the rest of the sequences). In this case, any selected block by the user should not necessarily involve segments from all the sequences of a certain alignment.

Moreover, choosing the partial alignment blocks will be done in a smoother way by selecting the blocks directly from the alignment using the mouse click-and-drag method instead of clicking on the start and end positions on the coordinates bar above the alignment.

5.2.3 Process DNA sequences with DIALIGN-PFAM and DIALIGN-PROSITE

DIALIGN-PFAM and *DIALIGN-PROSITE* accept as input protein sequences only. As a future perspective, a new functionality will be added to those two tools allowing them to accept DNA sequences for alignment. This can be achieved in several ways. One way might be to translate the DNA sequences into the possible protein sequences and then follow the same original workflow of *DIALIGN-PFAM* and *DIALIGN-PROSITE* and take those translated sequences as input.

Moreover, the "Aligning alignments with un-aligned sequences" webserver will also get this additional functionality of allowing the input of DNA sequences.

Bibliography

[1] Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J.Mol.Biol. **48**, 443–453 (1970) (Cited on pages 17 and 20.)

[2] Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. Atlas Protein Seq. Struct. **6**, 345–362 (1978) (Cited on pages 18 and 21.)

[3] Henikoff, S., Henikoff, J.G.: Protein family classification based on searching a database of blocks. Genomics **19**, 97–107 (1994) (Cited on pages 18, 21 and 38.)

[4] Mount DM. Bioinformatics: Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY **2 edition**, (2004) (Cited on pages 17 and 20.)

[5] Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J.Mol.Biol. **147(1)**, 195–197 (1981) (Cited on pages 17 and 20.)

[6] Altshul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. JMB **215**, 403–410 (1990) (Cited on page 24.)

[7] Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S.: ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Research **15**, 330–340 (2005) (Cited on pages 31 and 69.)

[8] Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W:improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research **22**, 4673–4680 (1994) (Cited on pages 30, 33 and 69.)

[9] Pei, J., Grishin, N.V.: PROMALS: towards accurate multiple sequence alignments of distantly related proteins. Bioinformatics **23**, 802–808 (2007) (Cited on pages 30, 31 and 32.)

[10] Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. **25**, 3389–3402 (1997) (Cited on pages 31 and 32.)

[11] Huang, X.Q., Miller, W.: A time-efficient, linear-space local similarity algorithm. Adv. Appl. Math. **12**, 337–357 (1991) (Cited on page 33.)

[12] Wu, C.H. et al.: The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic Acids Res. **34(Database issue)**, D187–91 (2006) (Cited on page 31.)

[13] Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. J Mol Biol **292(2)**, 195–202 (1999) (Cited on page 31.)

[14] Papadopoulos, J.S., Agarwala, R.: COBALT: constraint-based alignment tool for multiple protein sequences. Bioinformatics **23**, 1073–1079 (2007) (Cited on pages 32 and 69.)

[15] Marchler-Bauer, A., Anderson, J.B., Cherukuri, P.F., DeWeese-Scott, C., Geer, L.Y., Gwadz, M., He, S., Hurwitz, D.I., Jackson, J.D., Ke, Z., Lanczycki, C.J., Liebert, C.A., Liu, C., Lu, F., Marchler, G.H., Mullokandov, M., Shoemaker, B.A., Simonyan, V., Song, J.S., Thiessen, P.A., Yamashita, R.A., Yin, J.J., zhang, D., Bryant, S.H.: CDD: A conserved Domain Database for protein classification. Nucleic Acids Res. **33**, D192–D196 (2005) (Cited on page 33.)

[16] Sigrist, C.J.A., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A., Bucher, P.: PROSITE: a documented database using patterns and profiles as motif descriptors. Brief Nioinform **3**, 265–274 (2002) (Cited on pages 1 and 58.)

[17] Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J.D., Higgins, D.G.: Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular Systems Biology **7**, 539 (2011) (Cited on page 35.)

[18] Eddy, S.R.: Profile hidden markov models. Bioinformatics **14**, 755–763 (1998) (Cited on page 11.)

[19] Morgenstern, B., Dress, A., Werner, T.: Multiple DNA and protein sequence alignment based on segment-to-segment comparison. Proc. Natl. Acad. Sci USA **93**, 12098–12103 (1996) (Cited on pages 38 and 59.)

[20] Morgenstern, B., Werner, N, Prohaska, S.J., Schneider, R.S.I., Subramanian, A.R., Stadler, P.F., Weyer-Menkhoff, J.: Multiple sequence alignment with

user-defined constraints at GOBICS. Bioinformatics **21**, 1271–1273 (2005) (Cited on page 40.)

[21] Morgenstern, B., Prohaska, S.J., Pöehler, D., Stadler, P.F.: Multiple sequence alignment with user-defined anchor points. Algorithms for Molecular Biology **1**, 6 (2006) (Cited on page 40.)

[22] Notredame, C., Higgins, D.: Heringa, J.: T-Coffee: a novel algorithm for multiple sequence alignment. J. Mol. Biol. **302**, 205–217 (2000) (Cited on pages 33 and 69.)

[23] Göttgens, B., Barton, L.M., Gilbert, J.G.R., Bench, A.J., Sanchez, M.J., Bahn, S., Mistry, S., Grafham, D., McMurray, A., Vaudin, M., Amaya, E., Bentley, D.R., Green, A.R.: Analysis of vertebrates SCL loci identifies conserved enhancers. Nature Biotechnology **18**, 181–186 (2000) (Cited on page 39.)

[24] Stanke, M., Schöffmann, O., Morgenstern, B., Waack, S.: Gene prediction in eukaryotes with a Generalized Hidden Markov Model that uses hints from external sources. BMC Bioinformatics **7:62**, (2006) (Cited on page 39.)

[25] Stanke, M., Tzvertkova, A., Morgenstern, B.: AUGUSTUS at EGASP: using EST, protein and genomic alignments for improved gene prediction in the human genome. Genome Biology **7**, (2006) (Cited on page 39.)

[26] Morgenstern, B.: DIALIGN 2: improvement of the segment-to-segmentapproach to multiple sequence alignment. Bioinformatics **15**, 211–218 (1999) (Cited on page 39.)

[27] Morgenstern, B., Atchley, W.R., Hahn, K, Dress, A: segment-based scores for pairwise and multiple sequence alignments. Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology 115–121 (1998) (Cited on pages 38, 40 and 66.)

[28] Subramanian, A.R., Kaufmann, M., Morgenstern, B.: DIALIGN-TX: greedy and progressive approaches for the segment-based multiple sequence alignment. Algorithms for Molecular Biology **3**, 6 (2008) (Cited on page 42.)

[29] Ait, L.A., Corel, E., Morgenstern, B.: Using protein-domain information for multiple sequence alignment. In Preceedings of the IEEE 12th Int. Conf. on Bioinformatics and BioEngineering (BIBE 12) 163–168 (2012) (Cited on pages 2 and 46.)

[30] Al-Ait, L., Yamak, Z., Morgenstern, B.: DIALIGN at GOBICS- multiple sequence alignment using various sources of external information. Nucleic Acids Research **41**, W3–W7 (2013) (Cited on pages 3, 51 and 57.)

[31] Feng, D., Doolittle, R.F.: Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol **60**, 351–360 (1987) (Cited on page 27.)

[32] Chakrabarti, S., Bhardwaj, N., Anand, P.A., Sowdhamini, R.: Improvement of alignment accuracy utilyzing sequentially conserved motifs. BMC Bioinformatics **5**, 167 (2004) (Cited on page 29.)

[33] Morgenstern, B.: A simple and space-efficient fragment-chaining algorithm for alignment of DNA and protein sequences. Applied Mathematics letter **15**, 11–16 (2002) (Cited on pages 39, 59, 61 and 72.)

[34] Thompson, J.D., Koehl, P., Ripp, R., Poch, O.: BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. Proteins: Structure, Function and Bioinformatics **61**, 127–136 (2005) (Cited on pages 35 and 69.)

[35] Walle, I.V., Lasters, I., Wyns, L.: SABmark- a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics **21**, 1267–1268 (2005) (Cited on pages 35, 37 and 69.)

[36] Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: Scop:a structural classification of protein database for the investigation of sequences and structures. J.Mol.Biol. **247**, 536-40 (1995) (Cited on page 37.)

[37] Katoh, K., Misawa, K., Kuma, K., Miyata, T.: MAFFT: a novel method for rapid multiple sequence alignment based on fsr fourier transform. Nuc. Acids Research **30**, 3059–3066 (2002) (Cited on page 69.)

[38] Edgar, R.: MUSCLE: multiple sequence alignment with high score accuracy and high throughput. Nuc. Acids Research **32**, 1792–1797 (2004) (Cited on pages 29 and 69.)

[39] Sauder, J., Arthur, J., Dunbrack, J.R.L.: Large-scale comparison of protein sequence alignment algorithms with structure alignments. Proteins **40**, 536–40 (1995) (Cited on page 37.)

[40] Phillips, A., Janies, D., Wheeler, W.: Multiple sequence alignments in phylogenetic analysis. Mol. Phylogenet. Evol. **16**, 317–330 (2000) (Cited on page 16.)

[41] Castillo-Davis, C.I., Kondrashov, F.A, Hartl, D.L., Kulathinal, R.J.: The functional genomic distribution of protein divergence in two animal phyla: Coevolution, genomic conflict, and constraint. Genome Res **14**, 802–811 (2004) (Cited on page 16.)

[42] Rost, B., Sander, C.: Combining evolutionary information and neural networks to predict protein secondary structure. Proteins **19**, 55–77 (1994) (Cited on page 16.)

[43] Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. **292**, 195–202 (1999) (Cited on page 16.)

[44] Sonnhammer, E.L.L., Eddy, S.R., Birney, E., Bateman, A., Durbin, R.: Pfam: Multiple sequence alignments and HMM-profiles of protein domains. Nucleic Acids Res. **26**, 320–322 (1998) (Cited on page 16.)

[45] Finn, R.D., Clements, J., Eddy, S.R.: HMMER web server: interactive sequence similarity searching. Nucleic Acids Res. Web Server Issue **39**, W29–W37 (2011) (Cited on page 44.)

[46] Johnson, J.M., Church, G.M.: Alignment and structure prediction of divergent protein families: Periplasmic and outer membrane proteins of bacterial efflux pumps. J. Mol. Biol. **287**, 695–715 (1999) (Cited on page 16.)

[47] Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Moxon, M.M., Sonnhammer, E.L., Studholme, D.J. ey al.: The Pfam protein families database. Nucleic Acids Res **32**, D138–D141 (2004) lili (Cited on pages 1, 11, 16 and 43.)

[48] Jaroszewski, L., Li, W., Godzik, A.: In search for more accurate alignments in the twilight zone. Protein Sci. **11**, 1702–1713 (2002) (Cited on page 16.)

[49] Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987) (Cited on page 30.)

[50] Bashford, D., Chothia, C., Lesk, A.M.: Determinants of a Protein Fold Unique Features of the Globin Amino Acid Sequences . J. Mol. Biol **196**, 199–216 (1987) (Cited on page 30.)

[51] Thompson, J.D., Plewniak, F., Thierry, J., Poch, O.: DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. Nucleic Acids Research **1**, 2919–26 (2000) (Cited on page 33.)

[52] Brudno, M., Chapman, M., Göttgens, B., Batzoglou, S., Morgenstern, B.: Fast and sensitive multiple alignment of large genomic sequences. BMC Bioinformatics **4**, 66 (2003) (Cited on page 39.)

[53] Blackshields, G., Sievers, F., Shi, W., Wilm, A., Higgins, D.H.: PSequence emBedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol **5**, 21 (2010) (Cited on page 35.)

[54] Söding, J.: Protein homology detection by HMMâHMM comparison. Bioinformatics **21**, 951–960 (2005) (Cited on page 35.)