Flexible Protein Annotation with Sequence- and Secondary Structure Information

Constantin Bannert

Thesis submitted to the

Faculty of Technology,
Bielefeld University,
Germany

for the degree of Dr. rer. nat.

Supervisor

Prof. Dr. Jens Stoye, Bielefeld University

Referees

Prof. Dr. Jens Stoye, Bielefeld University
Prof. Dr. Ina Koch, MPI for Molecular Genetics, Berlin

November 2008

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706
(Printed on non-aging paper according to DIN-ISO 9706)

Abstract

The number of known gene sequences is still rising at an increasing pace. A standard task in sequence analysis is homology recognition, where a database is searched for homologous sequences. Even though already several software tools for homology recognition exist, there is still room for improvement. One possibility is to use more of the information in the source data. Another possibility is to add new features to support research on specific questions.

The Jumping Alignment algorithm (Jali) is a method for database searching. It works on multiple sequence alignments. In contrast to previous approaches, Jali is able to consider the information in the rows *and* the columns of a multiple alignment. Initial evaluations of Jali showed promising results. We evaluate Jali once more to see if the good results can be reproduced and investigate the reason. We analyze a set of jumping alignments with underlying secondary structure information for cases of "structural significance", where Jali seems to recognize the secondary structure in the alignment. Even though we found a few examples, these cases are rare. In a second experiment, we simulate the evolution of ten protein superfamilies and evaluate Jali on these families. The findings suggest that jumping offers Jali more flexibility in homology recognition, especially in conjunction with suboptimal alignments.

Many alignment-based approaches in homology recognition share certain limitations. They are not well suited to cope with duplications or rearrangements in the input sequences. In addition, their output is usually a list of scores that is not very informative.

We develop a new method "Passta" that circumvents these limitations. The first stage of the protocol (Pass One) serves as filter. Pass One tries to find a set of targets that are related to a given query. This candidate set is then submitted to Pass Two, where the query is annotated with alignments of Secondary Structure Elements (SSEs). In a graph-based approach, we select those alignments that reproduce the query in an optimal way.

Prior to a systematic evaluation of Passta, we train some of its parameters and discover interesting differences in the behavior of the three SSE classes coil, helix, and strand. We also calibrate the rearrangement cost parameter of Pass Two and use our method to find SCOP families with rearrangements or duplications, with nice results.

Finally, we present a comparative evaluation of Passta with Jali and BLAST. Unfortunately the results show that Pass One is not working very well. The main reason is that SSEs are short sequence fragments and alignments with SSEs are often unspecific. We are however able to show the potential of Pass Two: When we add the true positive targets missed by Pass One, Pass Two is able to compete with Jali and BLAST.

Contents

I Introduction **1**

1 Biological background **1**
1.1 The building blocks of biological macromolecules . . . 1
1.2 The central dogma of biology . . . 1
1.3 Mutations drive evolution . . . 2
1.4 Similarity and Homology . . . 2

2 Proteins **3**
2.1 Protein composition and protein chemistry . . . 4
2.2 Protein folding and structure . . . 5
2.2.1 Primary structure . . . 6
2.2.2 Secondary structure . . . 6
2.2.3 Tertiary structure and protein domains . . . 7
2.2.4 Quaternary structure . . . 8

3 Biological Sequence Alignment **8**
3.1 Score functions and substitution matrices . . . 9
3.1.1 The PAM and BLOSUM substitution matrices . . . 10
3.2 Pairwise alignments . . . 11
3.2.1 Global alignment and homogenous gap cost . . . 11
3.2.2 Local alignments with affine gap costs . . . 12
3.3 Co- and suboptimal Alignments . . . 13
3.4 Multiple Alignments . . . 14

4 Database searching by homology recognition **15**
4.1 Homology recognition . . . 15
4.2 Statistics in database searching . . . 17

5 Motivation and goals of this thesis **18**

II Material and Methods **21**

6 Databases **21**
6.1 The Protein Data Bank (PDB) . . . 21
6.2 PDBFinder II . . . 21
6.3 PTGL . . . 23
6.4 SCOP and ASTRAL . . . 23
6.4.1 ASTRAL . . . 24

7 External Algorithms and Methods **25**
7.1 Phase4 . . . 25

7.2 ROSE 25
7.3 BLAST 25
7.4 DSSP 26

III The Jumping Alignment Algorithm 27

8 Introduction 27
8.1 Motivation 27
8.2 The Jumping Alignment (Jali) Algorithm 28

9 Projects 30
9.1 Biological Significance of Jumping Alignments 30
9.1.1 Overview 30
9.1.2 Experimental setup 31
9.1.3 Results and Discussion 32
9.2 Experiments with simulated and natural protein families 34
9.2.1 Overview 34
9.2.2 Experimental setup 34
9.2.3 Results and Discussion 36
9.3 Conclusion 40

IV Passta 43

10 Introduction 43
10.1 Motivation 43
10.2 Method outline and related work 44

11 Integration of data into PasstaDB 48
11.1 Origin of the source data 48
11.2 The structure of PasstaDB 49
11.3 Data integration with PasstaBuild 49
11.3.1 Integration 49
11.3.2 Representative and redundant sequences 53

12 Protein Annotation with PasstaRun 53
12.1 The directed acyclic graph used in PasstaRun 54
12.1.1 Data structure 54
12.1.2 The optimal path 55
12.1.3 Related work 55
12.2 Pass One 55
12.2.1 Pass One targets 56
12.2.2 SSE target set 56
12.2.3 *Valid* alignments 57
12.2.4 Acceptance of the target domains 57

12.3 Pass Two 59
12.3.1 Computing suboptimal alignments 59
12.3.2 Two different edge definitions 60
12.3.3 The optimal path in Pass Two 62

13 Projects 64
13.1 Optional secondary structure profile 64
13.2 Pass One Training / Parameter settings 64
13.2.1 Choice of the Blosum62 threshold 66
13.2.2 SSE-alignments without a minimum length constraint 68
13.2.3 Secondary-structure specific gapcost 69
13.2.4 Choice of the MaxScoreRatio threshold 71
13.2.5 Conclusion 73
13.3 Calibration of Rearrangement and Deletion costs 74
13.3.1 Motivation 74
13.3.2 Model and Definitions 75
13.3.3 Experimental approach and setup 76
13.3.4 Results and Discussion 78
13.3.5 Conclusion 82

V Comparative Evaluation of Jali, Passta, and BLAST 84

14 Experimental Setup 84
14.1 Choice of the evaluation model 84
14.1.1 Passta 86
14.1.2 Jali and BLAST 87
14.2 Evaluation and calibration data 88

15 Results and Discussion 88
15.1 Passta Pass One 89
15.2 Passta Pass Two 89
15.3 Jali and BLAST 93
15.4 Comparative Discussion 94

Appendix 100

Lists of Figures and Tables 103

Acknowledgements 104

References 105

Part I
Introduction

1 Biological background

1.1 The building blocks of biological macromolecules

DNA-, RNA- and protein molecules have one thing in common: They are all linear biological macromolecules, polymers composed of monomeric building blocks. These building blocks are structurally similar within each molecule class, with only minor chemical differences between them. They are almost identical in DNA and RNA: each monomer is composed of a sugar molecule, a phosphate group, and a cyclic base. The DNA bases are adenine, guanine, cytosine, and thymine. In RNA, thymine is replaced by uracil and the sugar component ribose is slightly modified. Both DNA and RNA were discovered in the nucleus of eukaryotic cells, hence they were named *nucleic acids*. Their monomeric building blocks are called *nucleotides*.

The building blocks of proteins are the amino acids, simple molecules containing an amino and a carboxyl group. Proteins comprise of many different amino acids, but only about 20 of them are coded for by the genetic code (more details are given in Section 2.1).

1.2 The central dogma of biology

Every living cell we know today stores its genetic information in one or several double-stranded DNA molecules. All of those molecules that can be passed on to its progeny form the *genome* of a cell. The genome is organized into coding and non-coding segments. The coding segments are called *genes*. They can be defined as single functional units. There are many different types of non-coding segments, discussing them is out of scope in this thesis.

A gene can only express its function if it has been *transcribed* from its DNA state into a single-stranded RNA molecule. During *transcription*, each coding nucleotide in the DNA pairs with a specific RNA nucleotide. Some of the resulting RNA polymers can now be used by the cell, but most are just an intermediate between the DNA blueprint and the final gene product, the protein. Because protein molecules are unlike from the nucleic acids, transferring the information from RNA into protein is not as straightforward as from DNA to RNA, it needs to be *translated*. Basically, each triplet of DNA/RNA nucleotides is encoding one specific amino acid. The encoded amino acid is bound to the previous one until the gene translation is complete. Proteins have many different functions; they are even a vital part of the machinery that is transcribing DNA to RNA, and translating RNA into proteins.

The whole process is also known as the central dogma of biology: In all living cells, the flow of information is from DNA via RNA to proteins (this excludes the viruses, of which some are able to reverse transcribe their RNA genome into DNA).

1.3 Mutations drive evolution

All cells have sophisticated biochemical systems that prevent their genome from taking damage. Ionizing radiation and mutagenic substances are possible sources of such damage. Some organisms are extremely well adapted to hostile environments, with an almost perfect DNA repair system. *Deinococcus radiodurans* is a very well understood example, surviving even near atomic reactors [64, 26, 63]. However, none of the repair systems is perfect, and the same holds for DNA *replication* and *recombination* (in Eukaryotes). On rare occasions, a *mutation* event results in a modification of the genome. “Mutation” is a collective term describing permanent DNA changes. The change of a single nucleotide is called *point mutation*; the removal of one or more nucleotides is called *deletion*. Adding nucleotides is an *insertion*, and a *duplication* is a special case of insertion where an existing DNA segment is copied within the genome. When comparing two sequences for evolutionary events, an insertion in one sequence can also be seen as a deletion in the other one. If the point of view does not matter, this event is often abbreviated *indel*.

The effect of a mutation depends on its location, type, and size. A large deletion in a non-coding DNA segment may be *neutral*, without any consequence. Even mutations in coding segments are not necessarily harmful for the cell. For example, a *silent* point mutation within a gene does not change the encoded amino acid (→ *degenerated code*). Most mutations are disadvantageous for the cell, ranging from unwanted up- or downregulations of gene expression to the lethal loss of essential functions. However, mutations can also be beneficial for the cell. A mutation that is improving the efficiency of an enzyme for example confers a *selective advantage* to the cell or organism. This means, its offspring has better chances to survive, slowly replacing less well adapted members of the population.

Nevertheless, all this does not explain the origin of a new species. If the less well adapted members in a population are just replaced by better adapted ones, a new species will not arise. *Speciation* is always linked to a population of organisms, not to one individual. Within this population, a group of individuals emerges that is able to live in a new way, at the same time acquiring a barrier to genetic exchange with the remaining population (see, e.g. [40]). Usually, this is due to environmental changes that lead to a spatial separation. After the separation of the two populations, both diverge from each other during evolution.

1.4 Similarity and Homology

The phylogenetic tree in Figure 1 shows the main groups of organisms in the three domains of life. Our current understanding is that all life on earth originated from one source. Today, we believe that about 2.5×10^9 years have passed since this first creature lived. Since then, its original genome was subject to countless evolutionary events and has *diverged* far beyond reconstruction. Nevertheless, reconstructing our

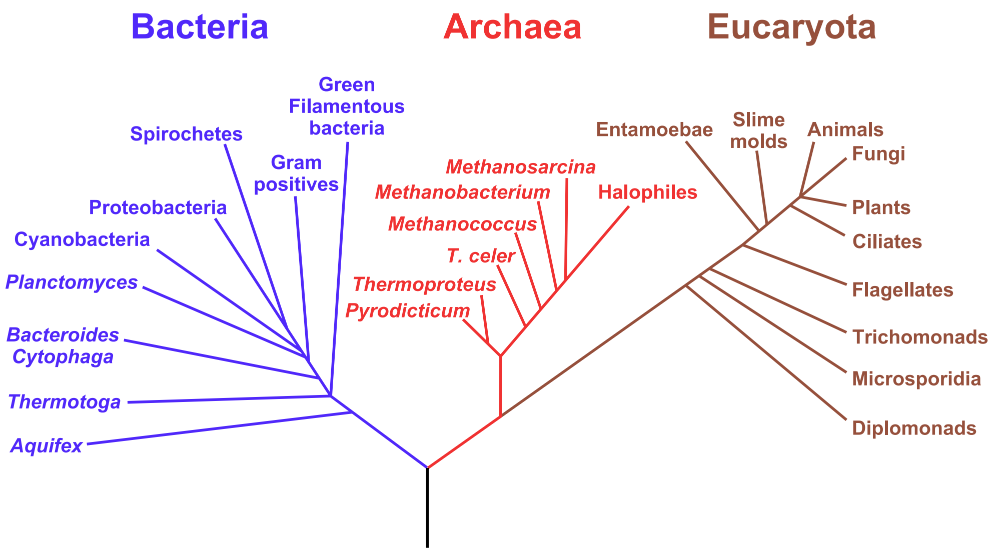


Figure 1: *The "Tree of Life", showing the three major domains of living organisms: The Bacteria, the Archaea, and the Eukaryota. It is based on their ribosomal RNA sequences (16S and 18S).*

biological history has always been a main goal of science. Basically, we assume two organisms or *characters* to be related if there is enough supporting evidence at hand. During the last centuries, this evidence had to be similarity in anatomy and function. Today, we can directly compare and quantify the similarity of selected genes or even whole genomes. If there is sufficient similarity and no contradictory facts present, we assume a *common ancestor*. Then, the two compared characters are said to be *homologous*, even though similarity does not guarantee *homology*. The last common ancestor in the tree of life can often be determined with confidence today.

Beside speciation, *duplication* is also very important in evolution. A gene with an important or even essential function can usually not mutate and aquire a new function, because it has to maintain the old one as well. After a duplication of this gene, however, one of the copies is free to evolve.

The term "homology" does not distinguish between relatedness by duplication and relatedness by speciation. In the former case, two related genes are called *paralogs*, in the latter case, *orthologs*. This is however a simplified explanation, more details can be found in introductury textbooks, or in the nice clarification by Walter Fitch [33].

2 Proteins

Proteins are in quantity and quality very important biomolecules. They constitute most of a cell's dry mass and execute almost all cell functions. Their variety is re-

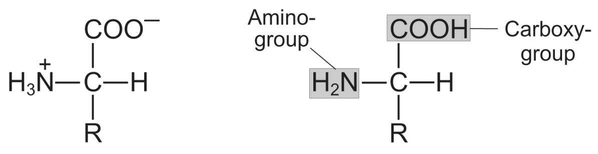


Figure 2: *Amino acid prototypes: In both schemes, "R" refers to the variable portion of the amino acid, the side chain. The left structure shows an amino acid in an aqueous solution at a neutral pH value around 7 ("Zwitterion").*

markable: Embedded in the cell membrane, some form channels and pumps that control the passage of small molecules. Others serve in signal transduction, either as signal transmitters or as sensors with associated functions. Proteins can also be like tiny molecular machines. One prominent example is *myosine*, which is driving muscle motion. Another one is the *enzyme topoisomerase*, responsible for untangling the DNA during replication. Enzymes are a large group of proteins with many different biochemical functions. They *catalyze* chemical reactions, i.e. the reaction takes only a fraction of the time it would take without the catalyst. Another important protein function is structural stabilization of the cell, carried out by elastic fibers or ropes. Besides, many other specialized proteins exist [2].

Such a large repertoire of different functions requires a high degree of structural variability. The following sections provide the basics on protein chemistry and protein structure that are needed to understand the versatility of these fascinating molecules.

2.1 Protein composition and protein chemistry

The building blocks of proteins are the amino acids. Aside from the rarely used amino acid selenocysteine [113] and the recently discovered pyrrolysine [42], 20 "standard" amino acids are commonly used. Every one of them has the same general chemical structure, except proline[1]. Each amino acid has an amino (-NH_2) and a carboxyl (-COOH) group connected to a central carbon atom, the C_α atom, see Figure 2. Many more amino acids are used in proteins, but these are derived from the standard amino acids subsequent to protein synthesis, mostly by *posttranslational modification*.

A specific side group unique to every amino acid determines its physicochemical properties. Based on these properties, the standard amino acids can be loosely grouped into classes. Three classes are commonly accepted: hydrophobic, polar, and charged. Subclassifications based on other properties like size or pH value are possible [18].

During the formation of a protein molecule, the amino acid monomers are joined into longer units. The carboxyl group of one amino acid is joined with the amino

[1]Technically, proline is an *imino* acid. Nevertheless, it is commonly referred to as one of the standard 20 amino acids.

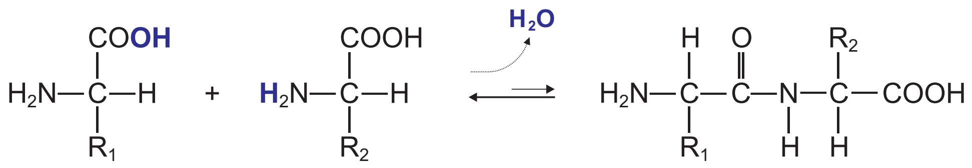


Figure 3: Peptide bond formation from two amino acids. The carboxy group of the first amino acid is joined to the amino group of the second amino acid and a water molecule is released. The bond in the newly formed dipeptide is the "peptide bond".

group of the next in a condensation reaction, releasing a water molecule. The covalent *peptide bond* formed by this reaction is part of the peptide *backbone*, the regular part of the molecule without the side chains, see Figure 3. The first amino acid in the growing polypeptide always retains his NH_2 group; it is referred to as the *N-terminal* end of the polypeptide. The other end of the chain is called *C-terminal*, because the corresponding last amino acid always retains its carboxy group. The peptide bond is a partial double bond that does not allow free rotation about it. Therefore it limits the molecules flexibility. Other constraints like possible collisions between the side chains and the backbone further restrict the steric freedom of the polypeptide.

Short peptides of up to a few dozen amino acids are referred to as *oligopeptides*, longer ones as *polypeptides*. Once an amino acid is incorporated into a peptide, it is referred to as "residue", and the atoms involved in the peptide bond (the regular part of the protein chain) are referred to as peptide "backbone" [18].

2.2 Protein folding and structure

Even though a polypeptide is a sequence of joined amino acids, its *in vivo* structure does not remain linear. Depending on the amino acid sequence and on environmental parameters, it folds into a specific 3D structure. Only proteins with a correct fold can fulfill their biochemical function. Long it was thought that all information required for folding is only contained in the amino acid sequence of the polypeptide. Recent evidence however suggests that it can also be influenced by the codon usage at the DNA level [55]. The folded polypeptide almost always adopts the energetically most stable *conformation*, i.e. the shape with the minimal free energy. The resulting protein conformation is stabilized by several types of non-covalent ("weak") bonds and other forces. The hydrogen bond can be regarded as the most important weak bond in proteins. It may form between an amino or hydroxy group (the donor) and an adjacent electronegative atom (the acceptor, usually nitrogen or oxygen) if the distance and the angle between donor and acceptor is convenient. A single hydrogen bond could not stabilize a proteins conformation, but the total contribution of all hydrogen bonds is significant.

Most proteins are immersed in an aqueous solution. In terms of free energy, the water molecules prefer to be in contact with other hydrophilic molecules. This leads to the *hydrophobic effect*, another important force in protein folding [103]. Here, most hydrophobic side chains associate and form a hydrophobic core, excluding the solvent. Polar atoms and even ions can only be packed into the core if they are neutralized, otherwise they are destabilizing the core.

Protein structure is a complex topic. To allow a concise description, we distinguish four different levels of organization: primary, secondary, tertiary, and quaternary structure. Each level emphasizes a certain view on a protein, while going into detail from primary to quaternary. Each level is shortly described in the following sections, with a focus on secondary structure.

2.2.1 Primary structure

A polypeptide is a large *chain* of amino acids. The unfolded linear sequence is called the *primary structure* of a protein. All other levels of organization (secondary, tertiary, quaternary) are results of protein folding. The folding process is guided by physicochemical constraints, which limit the flexibility of the protein.

2.2.2 Secondary structure

The limitations imposed by the primary structure dictate the possible *secondary structure* or *local structure* of the polypeptide chain. It can be seen as the local 3D conformation of a polypeptide, largely independent of the rest of the protein. One can separate between regular secondary structure patterns and polypeptide segments without regularity. The most dominant secondary structure classes with regular conformation are the α-helix and the β-sheet.

The *α-helix* has the shape of a cylinder. It is the result of hydrogen-bonding within the polypeptide backbone, not involving the side chains of the residues. Since the formation of α-helices is not side-chain specific, many polypeptides are able to build them. Other secondary structure patterns with helical structure exist as well (see Section 7.4), but the α-helix is by far the most prevalent.

In contrast to the α-helix, *β-sheets* are built by two or more adjacent segments of the polypeptide chain, forming a sheet-like, planar structure. The segments are called *β-strands*, and the individual β-strands in a β-sheet are connected by hydrogen bonds. A β-sheet of only two β-strands is also called a *ladder*. One can separate between *parallel* and *antiparallel* β-sheets. This depends on the orientation of the adjacent strands, i.e. N-terminal with N-terminal (antiparallel) or N-terminal with C-terminal (parallel). A schematic picture of an antiparallel β-sheet is shown in Figure 4. The arrow always points from the N- to the C-terminal end of the polypeptide.

Coils or *loops* are segments of the polypeptide chain that do not fold into a regular local conformation. Loops are usually present at the surface of globular proteins, just linking two regular structures. Other segments with irregular structure are called coils. *Random coils* do not even have a stable structure, their secondary structure

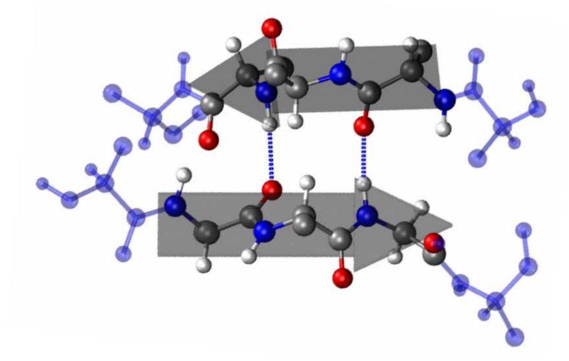


Figure 4: *Antiparallel β-sheet: The arrows show the direction of the sketched polypeptide from N- to C-terminal, the dotted lines represent the hydrogen bonds. Picture by O. Lenz, License: GNU Free Documentation License*

state is not static [18]. Even though this may appear strange, a certain amount of flexibility is an important property in protein structure. Especially enzymes depend on flexible secondary structure elements (SSEs) to fulfill their biochemical function. Therefore the exact length of a SSE is hard to determine, and methods that predict secondary structure states have inherent limitations [47].

2.2.3 Tertiary structure and protein domains

The secondary structure elements can combine in a variety of ways during protein folding. Hydrophobic and neutral elements associate and form the core of the polypeptide, while hydrophilic elements are on the outside and interact with the solvent. The final 3D-structure of a polypeptide is the *tertiary structure*, as defined by its atom coordinates. While the secondary structure is rather a result of local interactions, the tertiary structure depends also on remote interactions: The corresponding amino acids may be far apart in the sequence. Disulfide bonds between two cysteine residues are such an interaction. Under certain circumstances, they form a covalent bond that links two portions of the protein, stabilizing its fold and hence, tertiary structure.

The tertiary structure of a polypeptide sometimes contains more than one *structural domain*. A domain is a compact unit of folding, an association of one to several segments of a polypeptide chain. It is often self-stabilizing and folding independently of the rest of the polypeptide. Aside from being a unit of structure, a protein domain is usually also a unit of function and evolution. Some biochemical functions are generally useful, e.g., DNA- or substrate-binding. Domains realizing these functions were combined and reused in different functional contexts during evolution. Many proteins consist of more than one domain, and some proteins contain the same domains in different arrangements.

2.2.4 Quaternary structure

The level of tertiary structure decribes the structural organization of a single polypeptide chain, a monomer. However, many protein monomers are inactive. The active form of a protein often requires the association of other polypeptides and one or several cofactors. The complete, active protein is designated as the quaternary structure.

In summary, one protein can consist of n loosely associated polypeptide chains, where each chain can contain m structural domains. The parameters n and m usually take rather small values, often just “1”.

3 Biological Sequence Alignment

The biological macromolecules introduced in Section 1.1 are linear polymers, so each of them can be written as a *sequence* of letters, in computer science also known as *string*. The experimental determination of protein sequences is possible since 1950, when Frederick Sanger determined the insulin sequence [88] and Pehr Edman developed the *Edman degradation* (see [75] for an overview). Both procedures are rather complicated and time-demanding, so the number of sequenced proteins remained low. In 1977, two independent groups published less demanding protocols to “read” DNA sequences [67, 89]. Soon the number of known DNA sequences started to rise exponentially, and has continued since then. With the sequencing technology becoming widely available, the interest to analyze and compare biological sequences increased.

Sequence analysis is a sub-category in computational biology that deals with gaining, transferring, and integrating knowledge from the analysis of biological sequences. Sequence analysis is *the* foundation of computational biology, it is needed in such diverse areas as structure prediction, gene finding, and inferring phylogenetic trees. Most of its applications rely on a comparative analysis of two or more related sequences. The easiest way to compare two sequences is the dot matrix method of Gibbs and McIntyre, which is also known as *dot plot* [37]. One sequence corresponds to the horizontal axis of a twodimensional matrix, the other one to its vertical axis. Whenever the same two letters intersect in the matrix, the field is marked with a dot. The dot plot allows to compare one sequence against itself or to compare two sequences against each other. Insertions, deletions, matches and repeats can be easily identified. Dot plots are suitable for manual inspections, showing all possible mappings between two sequences. They do however not provide a measure of similarity or distance between both sequences, because none of the possible mappings is selected and fixed.

The mapping is also called *alignment* and its computation is an important standard task in sequence analysis. An alignment is an arrangement reproducing the divergent evolution of the aligned sequences in the best possible way. If the evolutionary history is correctly reproduced, the alignment is *biologically true*. The alignment in Figure 5 is made of two homologous sequences and could be biologically true.

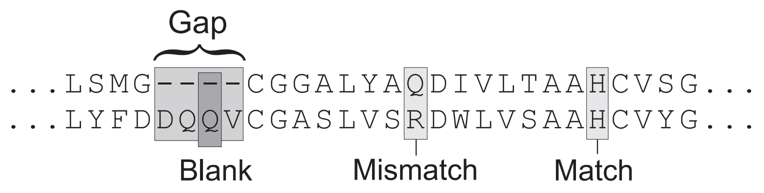

Gap
...LSMG---CGGALYAQDIVLTAAHCVSG...
...LYFDDQQVCGASLVSRDWLVSAAHCVYG...
Blank
Mismatch
Match

Figure 5: *Excerpt of a pairwise alignment illustrating some important technical terms in the alignment context. Both aligned sequences are serine proteases. The upper one is an* enteropeptidase *from cow (Bos taurus, PDB id "1ekb"), the lower one is trypsin from Streptomyces griseus (PDB id "1sgt").*

The matching characters show those positions that were *conserved* during evolution, while mismatched characters correspond to mutations. Characters aligned with the dash symbol "–" are called *blanks*, and each contiguous stretch of blanks is a *gap*, indicating insertions and deletions. An alignment of two sequences is called *pairwise*, while an alignment containing more sequences is called a *multiple* alignment. Multiple alignments can be regarded as generalization of pairwise alignments. Both types have much in common, requiring two or more sequences, a scoring function and an alignment algorithm. Therefore, we first introduce pairwise alignments along with some common concepts. In Section 3.4, we give a short introduction to multiple alignments and some specific details.

3.1 Score functions and substitution matrices

The automatic computation of alignments requires scoring systems that can be easily implemented in a computer program. The most convenient way is to rate matches, mismatches, insertions, and deletions with numerical values. The goal is always to compute the optimal alignment, either the one with the highest *similarity score* or the lowest *distance cost* between the input sequences. Basically, it does not matter whether similarity or distance is measured. It is often possible to transfer one into the other.

The simplest cost function defines the *unit cost*, often used in entry-level bioinformatics courses to explain sequence alignment. Here, the cost for an exact match between two letters is zero, the cost for a mismatch and for an indel is one. An example is shown in Figure 6a). In practice, more sophisticated scoring schemes are used. Especially in the alignment of protein sequences a more fine-grained approach is necessary to consider the physicochemical properties of the amino acids. Glutamic acid (*Glu*) and aspartic acid (*Asp*) are almost identical in size and polarity, being small and negatively charged. They were preferably substituted for each other during evolution, in contrast to amino acids that do not share the same properties. For example, exchanges of *Glu* and tryptophane (*Trp*) are rare evolutionary events, because *Trp* is uncharged and large. *Substitution matrices* capture the observed substitution frequencies in a single likelihood value that also depends on the abundance of

	L	S	M	G	-	-	-	-	C	G	G	A	L	Y	A	Q	D	I	V	L	T	A	A	H
	L	Y	F	D	D	Q	Q	V	C	G	A	S	L	V	S	R	D	W	L	V	S	A	A	H
a)	0	1	2	3	4	5	6	7	7	7	7	8	8	9	10	11	11	12	13	14	15	15	15	15
b)	4	2	2	1	-1	-3	-5	-7	2	8	8	9	13	12	13	14	20	17	18	19	20	24	28	32
c)	4	2	2	1	-6	-7	-8	-9	0	6	6	7	11	10	11	12	18	15	16	17	18	22	26	30

Cumulative alignment cost / score

Figure 6: *A pairwise alignment and the corresponding cost/scores of three different scoring functions: a) unit cost; b) Blosum62 similarity scores with homogenous gapcost, $g(l) = 2l$; c) as b), but affine gap costs with $g(l) = -6 - l$.*

the corresponding amino acid(s). For convenience, the substitution matrix contains logarithms of the computed likelihoods. A substitution matrix P contains a numerical score for every combination of characters from the sequence alphabet that can be easily looked up. A very frequently used family of substitution matrices are the BLOSUM matrices, shortly introduced in Section 3.1.1.

However, substitution matrices do not specify the treatment of gaps. Given a gap of length l, an additional function $g(l)$ specifies the penalty for this gap. The simplest such function are *homogenous* gapcosts: Each position within a gap is penalized by a constant value $c \in \mathbb{N}$, such that the costs for a gap of length l are $g(l) = c \cdot l$. An example alignment with homogenous gapcost is shown in Figure 6b). Clearly, this concept has its limits. If c is set to a low value, many short gaps will be introduced into the alignment. On the other hand, large indels are getting too expensive if c is too high.

This problem can be alleviated by using *affine* gapcosts. Once a gap is opened, a *gap initiation* penalty gi is subtracted. Every blank in the gap induces *gap extension* costs ge. So, the complete gapcost are given by $g(l) = gi + l \cdot ge$. Affine gapcost are an intermediate between homogenous gapcost and *general* gapcost. General gapcost functions are arbitrary, their implementation into the dynamic programming scheme is usually computationally expensive. In practice, affine gapcost are often preferred, because a considerable gain in quality is achieved by only a small increase in computation time.

3.1.1 The PAM and BLOSUM substitution matrices

The first series of substitution matrices were the PAM matrices by Dayhoff and coworkers [28]. They counted mutations in closely related sequences and constructed a matrix under a certain evolutionary model. By matrix multiplication, other matrices were extrapolated from the original, e.g. the PAM250 matrix for comparing evolutionarily divergent sequences. However, this methodology turned out not to work very well, because the changes over large evolutionary time scales differ sometimes from those over short time scales.

The approach used to compute the BLOSUM (BLOck SUbstitution Matrix) [44] series of matrices was more successful. Henikoff and Henikoff used multiple alignments of more divergent protein sequences, clustered by their sequence identity values. They identified the conserved "blocks" in the multiple alignment and computed the matrix by the observed substitutions. For example, the Blosum62 matrix is calculated from multipe alignments of proteins that share 62% sequence identity or more.

3.2 Pairwise alignments

A naïve approach to compute an optimal pairwise alignment is to enumerate all possible alignments, evaluate their quality and choose the best one. However, even for input sequences of moderate length and a small alphabet, this means countless possibilities to choose from. The number of comparisons needed to compare two protein sequences of length 300 is about 10^{88} if insertions and deletions are also considered [107].

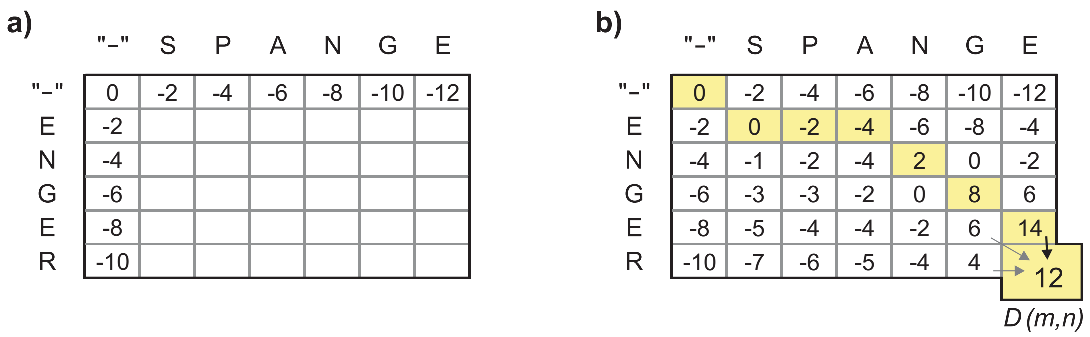
Luckily, a better solution is available. The idea is to process both sequences letter by letter, storing the optimal alignment of any two prefixes. The solution of each iteration depends on the optimal results of previous iterations, but not on *all* previous results simultaneously. In effect, this reduces the original problem of finding the optimal alignment into smaller subproblems of the same structure that are easier to solve. This approach is known as *dynamic programming*. It was coined in the 1950s by the mathematician Richard Bellman [15], and the method has found many applications since then.

3.2.1 Global alignment and homogenous gap cost

Let s and t be two sequences of length m and n that shall be aligned, the scores for matches and mismatches are provided by a substitution matrix P. A *global* pairwise alignment $A(s,t)$ of s and t contains the complete sequences, i.e. $s[1..m]$ is aligned with $t[1..n]$. Pairwise global alignment is also called *Needleman-Wunsch* alignment [74]. In conjunction with homogenous gap cost, it is the easiest way to illustrate the application of dynamic programming to the alignment problem.

The central element is the *edit matrix D*. As in the dot plot, one sequence corresponds to its horizontal axis, the other one to its vertical axis. Every field of D stores the optimal solution for an alignment of $s[1..i]$ and $t[1..j]$, $1 \leq i \leq m$ and $1 \leq j \leq n$, i.e. an alignment of the sequence prefixes. The size of the edit matrix depends on the length of the input sequences, being $(m+1) \times (n+1)$. The +1 is needed for the empty prefixes in the first row and column. They store the scores for alignments that start with a gap:

$$\begin{aligned} &D_{0,0} = 0 \\ &D_{i,0} = i \cdot gapcost \quad \text{for } 1 \leq i \leq m \\ &D_{0,j} = j \cdot gapcost \quad \text{for } 1 \leq j \leq n \end{aligned}$$



a)

	"-"	S	P	A	N	G	E
"-"	0	-2	-4	-6	-8	-10	-12
E	-2						
N	-4						
G	-6						
E	-8						
R	-10						

b)

	"-"	S	P	A	N	G	E
"-"	0	-2	-4	-6	-8	-10	-12
E	-2	0	-2	-4	-6	-8	-4
N	-4	-1	-2	-4	2	0	-2
G	-6	-3	-3	-2	0	8	6
E	-8	-5	-4	-4	-2	6	14
R	-10	-7	-6	-5	-4	4	12

Figure 7: *Global alignment with homogenous gapcost of -2. a) Edit matrix after the initialization of the first row and column. b) The score of every entry outside the first row and column depends on three predecessors. This is best illustrated by computing the last entry of the edit matrix, $D_{m,n}$. The shaded area illustrates the path of the optimal alignment.*

The initialization of the edit matrix is a necessary step before the dynamic programming part of the computation begins. An example with homogenous gapcost is shown in Figure 7a. Starting at $D_{1,\,1}$, the algorithm proceeds until the lower right entry $D_{m,\,n}$ of the matrix is reached. This entry holds the score of the optimal global alignment. Obviously, the three adjacent predecessors $D_{m-1,\,n}$ (horizontal), $D_{m,\,n-1}$ (vertical), and $D_{m-1,\,n-1}$ (diagonal) have to be computed before, see Figure 7b). The optimal alignment score at $D_{m,\,n}$ is the maximum over the three predecessors, plus either the value for aligning $s[m]$ and $t[n]$, or the gapcost. This can be generalized for every (i, j):

$$D_{i,j} \quad = \quad max \begin{cases} D_{i-1,\,j-1} & + \quad P(s_i,\, t_j), \\ D_{i-1,\,j} & + \quad gapcost, \\ D_{i,\,j-1} & + \quad gapcost \end{cases} \tag{1}$$

After computing the entry $D_{m,\,n}$ of the edit matrix, the *score* of the optimal global alignment is known. However, to find the correponding optimal alignment itself, the path that led to the optimal score has to be traced back. Starting at $D_{m,\,n}$, one possibility is to iteratively determine the predecessor for each optimal value in the 3-way maximization, until $D_{0,\,0}$ is reached. Another possibility is to store the back-tracing information in a second matrix while computing the alignment.

The time complexity of this approach is $O(m \cdot n)$, or $O(n^2)$ if $m \simeq n$. The same holds for space complexity if the alignment is the desired result. However, sometimes the mere alignment *score* is sufficient. Then, the space complexity can be reduced to $O(n + m)$.

3.2.2 Local alignments with affine gap costs

The global similarity between two sequences is often low, even though conserved regions with high similarity exist. These conserved regions are usually more interesting than variable regions with a high mutation frequency. If only a few conserved sequence "islands" are present, a global alignment may even fail to reveal those regions of biological interest. The *local* similarities between the sequences remain hidden.

Smith and Waterman [96] recognized that it takes only a few changes to transform the global alignment algorithm into a local alignment algorithm. A local alignment is an alignment of two subsequences of s and t. The local alignment problem is to find and align those subsequences whose score is maximal among the optimal alignments of all possible subsequences of s and t. The *Smith-Waterman* algorithm solves the local alignment problem. It differs in 2 important details from the Needleman-Wunsch algorithm:

1. The backtracing in the Smith-Waterman algorithm starts at the matrix entry with the maximal score, not necessarily at $D_{m,n}$. It proceeds until an entry with a value of 0 is encountered, marking the begin of the local alignment.

2. A fourth case "0" is added to the recursion (1). It guarantees that the edit matrix does not contain negative values anymore, corresponding to the removal of negatively scoring prefix alignments.

While being quite simple with homogenous gapcost, computing the alignment with affine gap cost requires additional effort. The problem is to determine whether a gap is initiated or extended. An approach using only the edit matrix needs to investigate up to n previous positions, leading to exponential time complexity. This behavior can be avoided by using two additional matrices, V and H, as proposed by Gotoh [38]. V stores the optimal scores of all alignments that end with an insertion in s; H stores those that end with a deletion. Obviously, V and H have to be considered during the three-way maximization, effecting additional changes to formula (1). The optimal score in V and H has to be computed prior to the maximization of $D_{i,j}$:

$$\begin{aligned}
V_{i,j} &= max(D_{i-1,j} + gapinit, V_{i-1,j}) + gapext \\
H_{i,j} &= max(D_{i,j-1} + gapinit, H_{i,j-1}) + gapext \\
D_{i,j} &= max(D_{i-1,j-1} + P(s_i, t_j),\ V_{i,j},\ H_{i,j})
\end{aligned}$$

3.3 Co- and suboptimal Alignments

Obtaining a local alignment is an important first step in sequence analysis. It shows how the substrings with the highest similarity relate, and permits to speculate on the homology status of the input sequences. However, if several regions with high similarity exist, usually only one of them is reported in the optimal local alignment. Computing *co-* and *suboptimal* alignments is often completing the picture. A cooptimal alignment has the same score as the optimal one, while the score of a suboptimal alignment is lower.

Finding suboptimal alignments is not straightforward. Mostly, the alignment with the next highest score in the edit matrix will largely overlap the optimal one. A second drawback of this naïve approach is that some suboptimal alignments can not be found in this way, because they are *shadowed* by certain high-scoring segments.

Waterman and Eggert found a nice solution for this problem [108]. They defined a suboptimal alignment as *nonoverlapping* or *non-intersecting* if it has neither matches nor mismatches in common with any other alignment from the same input sequences, i.e. s_i may only be aligned once with t_j in all alignments of s and t. The *Waterman-Eggert* algorithm computes the r best nonoverlapping local alignments between two sequences. First, the optimal local alignment is computed with the Smith-Waterman algorithm. During the backtracing in the edit matrix, all matches or mismatches being part of the optimal alignment are marked as being used. Then, a part of the edit matrix is recomputed. In the maximization of the used fields, matches and mismatches are now forbidden, leading to many changes in the edit matrix.

The time complexity of the Waterman-Eggert algorithm is $O(mn + \sum_{i=1}^{r} l_i^2)$ time, where l_i is the length of the i-th computed alignment. Huang and Miller [48] showed that the quadratic space complexity of the Waterman-Eggert algorithm can be reduced to linear time.

3.4 Multiple Alignments

Many questions in computational biology can not be answered by comparing a pair of sequences. The comparison of multiple sequences delivers much more information. *Multiple alignments* are alignments with $k > 2$ sequences. They can be seen as a generalization of pairwise alignments.

Multiple sequence alignment is a powerful tool in sequence comparison. If the aligned sequences are related, even short conserved motifs can be reliably identified. If k is large, even a statistical analysis of certain evolutionary events is possible. That is one reason why multiple alignments are very important in the reconstruction of phylogenetic trees, and also in database searching (see Section 4).

Several methods can be used to construct multiple alignments. The most accurate method is to compare all sequences *simultaneously*. This requires the construction of a k-dimensional matrix, one dimension for every sequence to be aligned. The time complexity of this computationally challenging approach is exponential, $O(n^k)$, with n being the length of the largest sequence to be aligned. Within a reasonable amount of time, only few sequences can be aligned in this way. If a heuristic is applied, their number can be increased. Two well-known heuristics are the Carillo-Lipman heuristic [22] and the "Divide and Conquer Alignment" algorithm (DCA) by Jens Stoye [100]. The Carillo-Lipman heuristic saves computation time by exploring only those regions of the k-dimensional edit matrix that could actually be part of the optimal multiple alignment. In its default version, it is an *exact* or *runtime* heuristic that guarantees the optimal result, but not a decrease in runtime. DCA is a true heuristic that will not always deliver the optimal solution, but will almost always save computation time. The DCA idea is to recursively *divide* the set of input sequences until the sequence fragments are small enough to be aligned simultaneously. These alignments are then combined in the *conquer* step of the algorithm. The problem is to find a family of score-optimal cut positions. Stoye used an heuristic approach based on pairwise alignments (→ *minimal additional costs*) to find the cut positions and showed that they are often optimal or near-optimal.

Progressive methods are able to build multiple alignments from many sequences within seconds or minutes. This is achieved by aligning not all sequences in parallel. A common protocol is to first compute a "seed" alignment of two or three sequences, then adding a sequence in each subsequent iteration. This is usually guided by a measure of sequence similarity, ensuring that sequences with higher similarity are aligned first. The rather simple *star* alignments align each sequence with a common consensus sequence. *Tree* alignment methods are often more sophisticated. "Clustal" [105] is a widely used example: Before the sequences are actually aligned, it computes a phylogenetic guide tree using the "Neighbor joining" algorithm [86]. In each iteration, the two branches with the least distance are aligned. Advanced techniques like sequence-weighting and position-specific gap penalties are applied to improve the final multiple alignment.

As in pairwise alignments, several score functions can be applied in computing a multiple alignment. Most frequently, the *sum-of-pairs* score function is used. For a given alignment column, each pair of aligned symbols is scored, the scores are added up to the sum-of-pairs score of that column.

4 Database searching by homology recognition

4.1 Homology recognition

The first step in the *functional annotation* of a newly sequenced gene is usually *homology recognition*, the search for related genes. If related genes with known function can be found, the function of the new gene can often be predicted. The easiest approach in homology recognition is to search a sequence database for sequences with high similarity. The new gene is the *query*, the entries in the database are called *targets*. High similarity between the query and a target usually implies a similar 3D-structure of their gene products, a necessary prerequisite for sharing the same biochemical function. A similar 3D-structure does however not imply sequence similarity, because structure is more conserved than sequence. We assume two genes with strong sequence and structural similarity to be related, because it is highly unlikely that the observed similarities arose independently during evolution.

Pairwise and multiple alignment is not only very useful in the manual comparison of a few sequences, but also for homology recognition. However, other approaches exist as well, and many of them do not rely alone on sequence information. Shi et al. [93] define four categories that classify approaches in homology recognition:

1. Methods that do pairwise sequence comparison, usually by computing pairwise alignments. They are able to detect closely related homologs, but often

miss remote homologies.

2. Tools in the second group are also based on sequence comparison. However, they use multiple alignments of related sequences and compute profiles or probabilistic models from them to improve the detection of remote homologs.

3. The third group of methods uses structure- and sequence information.

4. Homology detection in the fourth group relies on structure information only. Methods in this group are usually threading methods.

First group: This group contains well-known traditional methods like *SSEARCH* [78], an implementation of the Smith-Waterman algorithm [96] for database searching, *FASTA* [77], and *BLAST* [5].
Second group: Methods in the second group use information from multiple sequences, usually a multiple alignment. Many methods then transform the multiple alignment into a profile or a statistical model. The easiest kind of profile is the *position-specific scoring matrix* (PSSM). The frequencies of the letters in each alignment column are converted to log-odds scores in a $|\Sigma| \times N$ matrix, where Σ is the alphabet and N is the length of the multiple alignment. When comparing a PSSM to a sequence, indels can not be modeled. *PSI-BLAST* (Position-Specific Iterative BLAST) [6] is using a PSSM during its search for related sequences. After an initial search with a single query, it computes a PSSM of all targets that are assumed to be related to the query. This profile is the input for the next iteration, where the additional information is used to find more related targets that are again added to the profile, and so on. Similar approaches were also developed by other authors, e.g. in [104].
Profiles are a rather simple way of using multiple sequence information, *Hidden Markov Models* (HMMs) and profile HMMs (pHMMs) are more complex. A HMM is a general statistical model describing a system that is producing a sequence of visible *emissions*, e.g. a string. The *emission probabilities* depend on the current *state* of the system, which is invisible (or $\rightarrow$ *hidden*). A transition between the states of the system is possible if the *transition probability* is greater than zero. HMMs can be used to predict the hidden states by the sequence of observed emissions, see the book by Durbin *et al.* [31] for more information.
Profile HMMs are a statistical description of a sequence family. The emission probabilities at each position in the familycorrelate with the the observed letter frequencies. A target sequence in a database search is considered as being emitted by the pHMM. The maximum probability to emit the target sequence is used to decide if the target sequence is related to the pHMM. A large advantage of pHMMs is that indels can also be modeled. *HMMer* [31] and *SAM* [52] are standard software tools to generate pHMMs and work with them. Profile HMMs are also used to model protein domain families in *PFAM* [32] and *SMART* [91, 61].
Group three: Sequence and structure information is often used in complex protocols that combine several methods in one pipeline. One interesting example is *3D-PSSM*

by Kelley and coworkers [53]. The main idea is to combine profiles of multiple sequences with structure information The structure information originates from multiple structure alignments of related SCOP domains. The secondary structure states and solvation potentials of the aligned residues are also known and used to aid in the search process. *FUGUE* [93] uses also structural alignments, from the *HOMSTRAD* database [69]. One of the main differences to *3D-PSSM* is that *FUGUE* automatically selects the alignment algorithm with structure dependent gap penalties to align a target sequence to the structure.
Group four: Methods that rely on structure information as only source of information are basically out of scope in this thesis, because they would require an introduction to *structure comparison*. Nevertheless, we would like to mention four methods [3, 11, 56, 66] based on the representation of secondary structure elements. Only [11, 66] are tailored to database searching, but the other examples could easily be adapted to it, too. These approaches show that secondary structure information can indeed be a valuable source of information.
Many methods relying on structure information require sequential input structures. A recent study by Shih and Hwang [94] investigated permuted alignments by structural comparison where the SSEs were not required to be sequential. Their results indicate that this area is to some degree overlooked.
Intermediate group: Of course, not every method for database searching fits cleanly into one of these four groups. *Treesearch* [82] is such an intermediate approach. Treesearch fits best into the second group, because its input is a sequence family. It constructs a phylogenetic tree of the family, to which each target sequence is temporarily added. The length of the resulting new edge then determines the homology status of the target. The method of Koretke *et al.* [57] is also an intermediate that could be placed in group two. Even though their semiautomatic approach is mostly based on multiple sequence information, predicted secondary structure is used under certain circumstances.

4.2 Statistics in database searching

Searching a large database like Genbank[2] requires millions of comparisons. The probability that a random similarity scores higher than the similarity of two distantly related sequences is increasing with database size [99]. Spang and Vingron [98] express this very nicely from another point of view: “Scores resulting from genuine evolutionary relationship obviously remain constant, while their credibility decreases as the database grows.”

It is therefore important to know if a score obtained in a database search is higher than one would expect from two unrelated sequences. This expectation is expressed in the *e-value*. For each search result, it describes the number of hits expected to see by chance with the same score in a database of known size. A target sequence from the database matching the query with a score s and a corresponding e-value

[2] see `http://www.ncbi.nlm.nih.gov/Genbank/index.html`

of 1 means that in a database of the current size, one match with a score s can be expected by chance. Karlin and Altschul [51, 4] compute the expectation value as:

$$e = K \cdot N \cdot exp(-\lambda \cdot s)$$

where e is the expectation value, K and λ are the "Karlin-Altschul" parameters, and N is the product of the query length and the length of the database (i.e., the summed lengths of all target sequences in the database). The random sequence model of Karlin and Altschul does only approximate the situation in database searching, where we are dealing with many short biological sequences. Therefore the effective lengths of the query sequence ($Qlen_{eff}$) and the target database ($DBlen_{eff}$) have also to be considered. The final equation to compute the e-value in BLAST is:

$$e = Qlen_{eff} \cdot DBlen_{eff} \cdot exp(-\lambda \cdot s + logK)$$

The lower the e-value, the higher is its "significance". Database hits with an e-value of 10^{-10} or less are usually considered to be reliable enough to indicate homology. For example, a threshold of 10^{-30} was used in "SYSTERS" [58].

5 Motivation and goals of this thesis

As of today (November 2008), the complete genomic sequences of more than 700 different species are publicly available[3]. Many of those species were chosen because of their commercial or medical significance. Even though the most interesting species have already been sequenced, enough targets are still worth investigating. This is supported by the ongoing improvements in sequencing technology [65] that allow to obtain relatively cheap high-coverage reads within a short period of time. The current and future sequencing efforts will result in many new genes with unknown structure or function. Sequence analysis will therefore remain an important area in bioinformatics research.

The need to characterize and annotate the enormous amount of genes from the genome sequencing projects led to the development of many useful algorithms. A common characterization approach is homolgy recognition, where a query sequence is compared to one or many already characterized sequences or structures. If significant similarity between those can be found, we assume the existence of a common ancestor, i.e., homology.

Database searching (see Section 4) is the standard method in homology recognition. Early methods were mostly based on pairwise comparisons. Because of the limited amount of information available in pairwise alignment, comparisons of multiple sequences became more and more popular. These methods are usually better at remote homology detection and achieve on overall good results. However, even they can not always detect remote homology between 20% and 35% sequence similarity, in the so-called *twilight zone* [84].

[3] see GOLD database at `http://genomesonline.org/`

Even though the statistical properties of pairwise-, and to some degree also multiple alignments are well understood, it is getting more and more difficult to improve on the existing methods. There are basically three possibilities to allow for further improvements:

1. Increase the amount of available information: The intrinsic sequence information is largely explored by modern approaches in homology recognition. Other sources of information are available that could be used in more sophisticated models, allowing to improve the existing methods. Examples are statistics on codon usage in DNA and models that capture the information associated with catalytic sites in enzymes. Protein structure is another source of information. Existing knowledge about secondary and tertiary structure can be combined with sequence information. Such approaches already exist [57, 93], often implemented in the complex protocols for identifying homologous targets in the CASP[4] experiment. The goal however also requires to investigate the use of information in existing methods. Otherwise it is not clear on how they can be improved.

2. Add more flexibility to existing annotation pipelines. Genes and their products are to some degree modular. Often, several domains are combined in different ways during evolution. Below the domain level, the products of eukaryotic genes can contain different combinations of exons if they are subject to alternative splicing. Most of the existing approaches in homology recognition are unable to detect and work with sequence fragments. The detection of repeated and translocated seuence fragments could also improve the search process.

3. Enhance the presentation of the available knowledge: The value of an annotation approach is not only determined by its detection power, but also by the way it presents the available information. The linear list of targets returned by most methods in database searching answers basically only one question: What is the optimal ranking of the targets from my database? Even though this is the most important answer in this scenario, other questions may be asked, for example: Which combination of targets could explain my query sequence? Which part of my query is most similar to what part of the target? Answering more of such questions may ease manual decisions if the automatic methods do not provide enough evidence for deciding about homology.

The Jumping Alignment algorithm (Jali, see Part III) [97]) can be used to search a sequence database for homologous targets. It aligns a single target sequence to a multiple alignment of a sequence family. Each letter of the target is aligned to only sequence of the multiple alignment, but Jali can "jump" or change to another sequence. Thereby Jali is able to use horizontal *and* vertical information from the input alignment and can be regarded as method with an advanced use of information (see

[4] *Critical assessment of techniques for protein structure prediction*, see
`http://predictioncenter.org/`

topic 1 above). An initial evaluation of the algorithm showed good results. To implement further improvements, we first need to better understand it.

The goals of this thesis are to further investigate the jumping alignment algorithm and to develop a new method for homology recognition. The investigation of the Jali algorithm will focus on its jumping behavior, especially with respect to biological properties of the aligned sequences. A part of the necessary data will be obtained in a modified evaluation scenario with a newer version of the SCOP database and artificial data. We want to ensure that the good results of Jali in the first evaluation can be repeated with a newer version of the SCOP database. The insights won in the investigation of the Jali algorithm will be used to develop a new method for homology recognition. The design of the new method will implement some of the strategies described above: An additional source of information shall be available to the search protocol, aside from the pure sequence data. The method shall be flexible, allowing to consider rearrangements and repeats within reasonable limits. Last not least we would like to present the search results in an enhanced way, aiding the scientist in his decisions rather than asking him to trust a single e-value.

Part II

Material and Methods

6 Databases

6.1 The Protein Data Bank (PDB)

The Protein Data Bank (PDB) was founded in 1971 as an archive of biological macromolecules with known 3D structure [17, 16]. The PDB entries contain the atomic coordinates of the macromolecule and associated information. Most PDB entries are proteins, but other molecules are stored as well, e.g. nucleic acids or complexes of proteins and nucleic acids. The majority of protein structures was solved by X-ray crystallography, followed by NMR spectroscopy and electron cryomicroscopy (also called cryo-electron microscopy).

Until the 1990s, the PDB entries were stored as plain text or "flat" files. Figure 8 shows a few lines of PDB entry "1eft", the procaryotic elongation factor "EF-TU". Each line or record contains 80 characters and starts with a keyword of at most six characters. The keyword specifies the "record type" of stored information, i.e. the PDB identifier `Header` or the `Title` of the entry. Position 7 to 80 contains the data, although sometimes additional characters are used to mark certain subtypes of information. The `SEQRES` records store the complete sequence of the protein chain(s) in the protein. The `ATOM` records contain the atom coordinates of every residue that could be identified in the structure determination experiment. Sometimes not all the residues can be identified, especially at the begin or end of a sequence. Therefore the `ATOM` sequence is not necessarily equal to the `SEQRES` sequence. In the 1990s, the PDB entries were gradually transfered into the modern *Macromolecular Crystallographic Information File (mmCIF)* format (see [110] for an overview). However, the flat files remained available because many legacy software tools use the plain text format.

6.2 PDBFinder II

The PDBfinder database [46] contains a summary of each PDB entry. The database is stored in one text file with an easy-to-interpret format and is regurlarly updated. Besides PDB data, the PDBfinder database also stores data derived from DSSP [50] and HSSP [30], and from the literature. Relevant statistical data (e.g. solvent accessibility) are also included. The PDBfinder II database[5] is a slightly extended version of the PDBfinder database. The main difference is the added secondary structure information (computed with the DSSP method) that is available for each protein chain in PDBFinder II. Table 1 shows an example entry.

[5] available at `ftp://ftp.cmbi.kun.nl/pub/molbio/data/pdbfinder2/`.

```
1        10         20         30         40         50         60         70
HEADER    ELONGATION FACTOR                       24-AUG-93   1EFT
TITLE     THE CRYSTAL STRUCTURE OF ELONGATION FACTOR EF-TU FROM
TITLE    2 THERMUS AQUATICUS IN THE GTP CONFORMATION
JRNL        AUTH   M.KJELDGAARD,P.NISSEN,S.THIRUP,J.NYBORG
JRNL        TITL   THE CRYSTAL STRUCTURE OF ELONGATION FACTOR EF-TU
SEQRES   1 A  405  ALA LYS GLY GLU PHE ILE ARG THR LYS PRO HIS VAL ASN
SEQRES   2 A  405  VAL GLY THR ILE GLY HIS VAL ASP HIS GLY LYS THR THR
ATOM      1  N   ALA A   1      75.082  -7.178  43.255  1.00 27.28           N
ATOM      2  CA  ALA A   1      74.276  -6.678  42.092  1.00 34.44           C
ATOM      3  C   ALA A   1      75.143  -5.790  41.184  1.00 33.12           C
```

Figure 8: *A few selected records of PDB entry "1EFT" illustrating the two sources of possibly different sequence information. Only ten out of all 3690 lines are shown.*

Field and example entry			Description
`ID`	:	`101M`	PDB identifier
`Compound`	:	`myoglobin`	Name of the solved structure
`Source`	:	`(physeter catodon)`	Species or vector
`Exp-Method`	:	`X`	Structure determination method (here: X-Ray crystallography)
`Resolution`	:	`2.07`	Resolution of protein crystal (in Å)
`Chain`	:	`A`	Chain identifier
`Amino-Acids`	:	`154`	Number of amino acid residues in chain
`Sequence`	:	`MVLSEGEWQLVLHVWAKVE...`	PDB "ATOM" sequence
`DSSP`	:	`CCCCHHHHHHHHHHHHHHG...`	DSSP secondary structure states
`Access`	:	`2907693254025116413...`	Solvent accessibility (digits)

Table 1: *PDBFinder II excerpt of PDB entry "101M". The indented rows contain information describing entries in the non-indented rows in more detail. For example, the specified PDB chain "A" contains 154 residues. Most of the data shown here is stored in PasstaDB (see Section 11.2). Using the information on solvent accessibility is an interesting option for future refinements.*

6.3 PTGL

The Protein Topology Graph Library (PTGL) [68] is a library of graphs representing protein topologies. Each graph represents the regular secondary structure elements (helices and β-strands) of a single PDB chain and the spatial contacts between them. The vertices correspond to the SSEs. They are enumerated from the N- to the C-terminus. An edge connects two vertices if the represented SSEs are in contact with each other. The edges are labeled to indicate the relationship between those SSEs: parallel, antiparallel, or mixed. The modeled proteins can be shown as 2D-diagrams or in 3D (with an external structure browser like Rasmol [90]). The PTGL website[6] also allows the search of toplogies and subtopologies and can therefore be used to find proteins with similar (sub)structure.

The topology information is not used in this thesis. We use only the regular SSEs of the decomposed protein chains, and some of the associated position information in our project Passta (see Chapter IV). SSEs with irregular secondary structure, i.e. coils and loops, are not stored in the PTGL.

6.4 SCOP and ASTRAL

The SCOP database (Structural Classification of Proteins) [73, 9, 10] classifies protein domains with known 3D-structure. Protein domains are suitable units of classification, because they can be defined as the working unit of structure, function and evolution. The SCOP classification is a hierarchy with four main levels:

- *Class:* The different folds are grouped into classes according to the main type of secondary structure that occurs within them. For example, folds that contain only α-helices are placed in the class "all-alpha".

- *Common fold:* Protein domains have a common fold if their main secondary structure elements have the same arrangement. Protein domains in the same fold are however not necessarily homologous.

- *Superfamiliy:* Protein domains related at the superfamily level are remotely related. They usually have a low percentage of sequence identity, but high structural and/or functional similarity.

- *Family:* Families contain closely related protein domains usually sharing at least 30% sequence identity. Members of the same family are also very similar in terms of structure and function.

A given protein domain may be found in different species, and in different isoforms (enzymes are known examples). To group these almost identical "protein domain species" together, there is another level below family, called "Protein" in [10]. The focus of this term is on protein function. However, we believe this term may be

[6] `http://sanaga.tfh-berlin.de/~ptgl/ptgl.html`

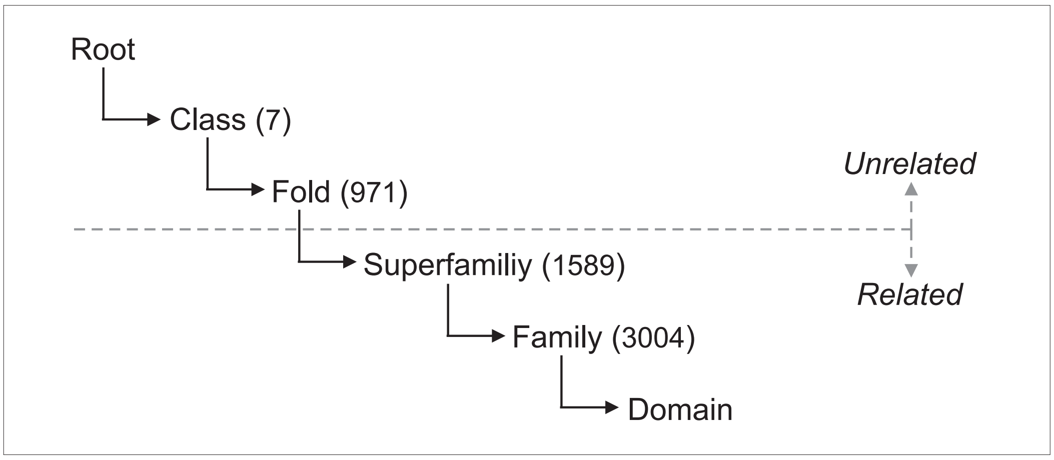


Figure 9: *Hierarchy of the SCOP classification: Protein domains related above the dashed line are assumed to be unrelated. Those below the line share significant structural similarity and are assumed to be related. The brackets contain the number of entries on each hierarchy level in SCOP 1.71.*

misleading, and call this level "Protein/Domain" (we admit this is only slightly less misleading).

The classification process in SCOP is manual. A team of experts first identifies all domains in each PDB (see Section 6.1) entry, and then classifies them into the SCOP hierarchy. The main property used in the classification is structural similarity, but sequence similarity and biochemical function are also important. SCOP is available online[7].

6.4.1 ASTRAL

The SCOP team is only responsible for the classification itself. The ASTRAL consortium [20, 24] provides databases and tools bridging the gap between SCOP the PDB. One example are the "Spaci" and "Aerospaci" scores that the ASTRAL team assigns to each PDB entry. They indicate the stuctural quality of the PDB entry.

The ASTRAL website[8] also provides certain subsets of the SCOP database. These subsets are filtered by sequence similarity or e-values. For example, the "PDB SEQRES 40" subset contains only domains that are less than 40% identical and correspond to the PDB "SEQRES" data.

[7] `http://scop.mrc-lmb.cam.ac.uk/scop/`
[8] `http://astral.stanford.edu/`

7 External Algorithms and Methods

7.1 Phase4

Phase4 [81] is a semi-automatic evaluation environment for sequence database searching. The evaluation protocol is basically split into four distinct phases: "Model Construction" (1), "Execution" (2), "Evaluation" (3), and "Report" (4). During the first phase, the user has to choose a database with known homology relationships and an evaluation model. The evaluation model defines the queries and targets in the database search and divides the corresponding data into test and training sets. For example, if the SCOP database is chosen, all superfamilies can be divided into test and training sets. The members of the test set have to be found by the method(s) under evaluation, while the training sets provide the queries, e.g., a multiple alignment or HMM. The second phase executes the considered method, searching the test sets of the database for related entries. After the execution phase, the evaluation method has to be specified. While the numbers of true and false positives or negatives is known, there are several methods to assess the quality of the search result, e.g. the *minimum false positive count* "minfpcount" or the *receiver operating characteristic* "roc50". Several kinds of plots and tables are available in the fourth phase to report the results of the third phase.

7.2 ROSE

ROSE (Random Model of Sequence Evolution) generates families of random protein or DNA sequences guided by an evolutionary tree. The root of that tree is a common ancestor sequence, which can be generated by Rose or may be provided by the user. The program mutates the root sequence by insertion, deletion and substitution events along the tree edges, until a predefined distance has been reached. During this artificial evolutionary process, the 'true' history is logged and the 'correct' multiple sequence alignment is created simultaneously [101]. ROSE is available online[9].

7.3 BLAST

BLAST (Basic Local Alignment Search Tool) [5] and *FASTA* [77] were the first widely used tools capable of searching a sequence database in a reasonable amount of time. Especially *BLAST* was very successful and became *the* standard in sequence database searching. Since the original publication in 1990, the software has been enhanced several times and distributed in different flavors.

BLAST is well covered in many introductory bioinformatics books (see e.g. [71]). Here, we describe only the central idea and some aspects of `blastp`, the *BLAST* excutable used in the evaluations of *Jali* and *Passta* (see Chapters III and IV, respectively).

[9] `http://bibiserv.techfak.uni-bielefeld.de/rose`

The time savings achieved by *BLAST* are due to a clever heuristic search strategy. First, the query sequence is transformed into a set Q of overlapping k-mers (usually, $k = 3$ in proteins). Given a *word* $q \in Q$ and a substitution-matrix dependent threshold T, *BLAST* determines all sequence fragments of size k that score above T in a gapless alignment with q. Looking up these sequences in the database can be done in linear time.

The k-mers from Q and their counterparts from the target database are the so-called *seed alignment*. Depending on the *BLAST* version, the seed alignments are extended in various ways. At last, the significance of the final alignment is assessed by computing e-values.

7.4 DSSP

The 3D-structure of a protein can be experimentally determined. After the successful experiment, the atom coordinates of the protein are known. However, the secondary structure states are not yet assigned. If the assignment was made on the basis of a visual inspection made by researchers and scientists, secondary structure assignment would be highly subjective. An objective algorithm is therefore a much better choice. One of the first algorithms for secondary structure assignment was the DSSP algorithm [50]. Even though newer and slightly better approaches exist (e.g., [36]) DSSP is still the accepted standard. The correctness of secondary structure assignments is anyway a difficult issue. Proteins are dynamic macromolecules that often require some flexibility to perform their function *in vivo* [47].

DSSP detects certain patterns of hydrogen bonds and geometrical features in the data and assigns one out of eight secondary structure states to each residue. These states are coded for by the characters *T, G, H, I, B, E, S* and the blank character " ". There are two basic patterns recognized by DSSP, *turns T* and *bridges B*. The more complex patterns are modeled by sequences of turns and bridges. The letters *G, H, I* are helical patterns, the α-helix *H* being the most fequent helical pattern. β-strands *E* are extensions of the bridge pattern. The letter *S* is used to indicate *bends* in the protein backbone, all other cases are left blank.

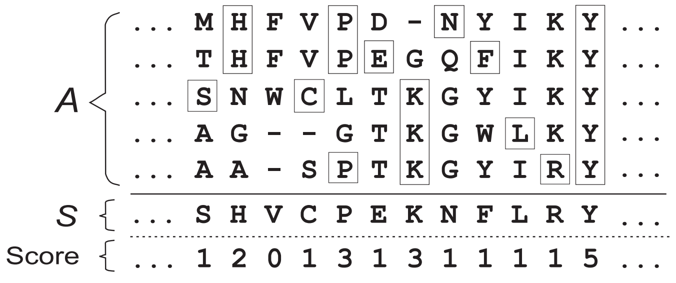
Part III

The Jumping Alignment Algorithm

8 Introduction

8.1 Motivation

A protein *family* is composed of closely related proteins similar in sequence, structure, and function. A multiple alignment of the protein family reveals variable and conserved regions. The larger the family, the more information is contained in the multiple alignment. The additional information allows to identify new family members in a sequence database search more easily. Figure 10 shows a multiple alignment of a query sequence and a hypothetical protein family. The scores at the bottom



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

Figure 10: *Excerpt of a multiple alignment A and an aligned query sequence S. The "Score" row lists the number of occurrences of each letter in the corresponding alignment column. The score does only depend on the vertical information in each column.*

of the figure show that the query and the multiple alignment are globally similar: all except one of the query letters appear at least once in the same alignment column. The observed similarity is however distributed among all sequences of the family. In a database search with a large and variable family, this kind of column-based scoring could lead to the acceptance of unrelated query sequences. The reason is the column-based point of view illustrated in Figure 10. Only the *vertical* information of the multiple alignment is used in the scoring. Figure 11 shows the same alignment, however illustrating the opposite viewpoint. The comparison of the query sequence and the protein family is *row*wise, the highest score indicating the family member with the highest similarity to the query. Only *horizontal* information is used in the scoring. Figure 12 shows that the alignment contains horizontal *and* vertical information. The five sequences subdivide into two subfamilies (shaded bars) with certain conservation patterns (dotted boxes). These patterns are not seen by column-based approaches, and row-based approaches are not flexible enough to switch between them. This lead to the development of the *Jumping alignment algorithm* (Jali) [97]. This intermediate algorithm is able to exploit both sources of information from the

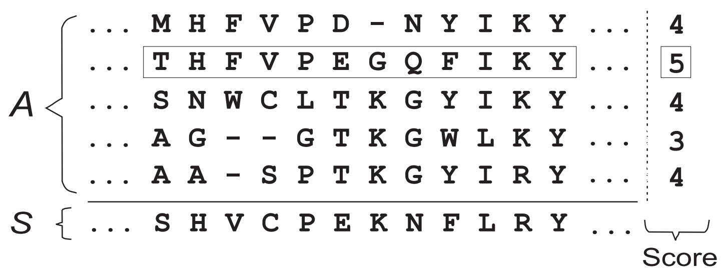

A
... M H F V P D - N Y I K Y ... 4
... T H F V P E G Q F I K Y ... 5
... S N W C L T K G Y I K Y ... 4
... A G - - G T K G W L K Y ... 3
... A A - S P T K G Y I R Y ... 4
S
... S H V C P E K N F L R Y ...
Score

Figure 11: *Alignment between a query sequence S and a multiple alignment A. Only horizontal information is used to compute the score for each row.*

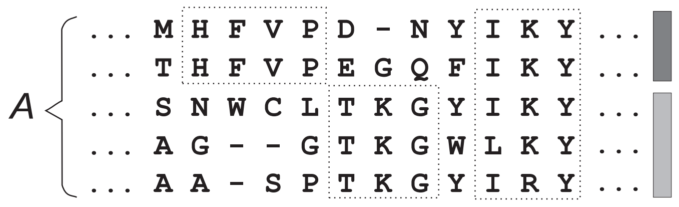

A
... M H F V P D - N Y I K Y ...
... T H F V P E G Q F I K Y ...
... S N W C L T K G Y I K Y ...
... A G - - G T K G W L K Y ...
... A A - S P T K G Y I R Y ...

Figure 12: *The family divides into two subfamilies, indicated by the shaded bars. The three dotted boxes show fully or highly conserved motifs, two of them being specific for one of the two subfamilies.*

multiple alignment. It basically aligns the query to one sequence of the multiple alignment. The algorithm may however change this *reference* sequence and "jump" to another one. The resulting "jumping alignment" shows the score-optimal alignment of the query and the aligned family under the given parameters. The details are described in the following Section 8.2.

The results of an initial evaluation were promising and illustrated the power of the approach. The reasons for the good results were however not obvious. Most probably, the jumping capability is the responsible component, but how exactly does it contribute to homology recognition? A thorough investigation will help us to answer this question and aid in improving the algorithm. Beside experiments on secondary structure recognition by Jali, we also collect quantitative data with families of biological and random origin.

8.2 The Jumping Alignment (Jali) Algorithm

The Jali algorithm [97] is based on the dynamic programming paradigm. It can be seen as an extension of the Smith-Waterman algorithm (see Section 3.2.2), which computes a pairwise local alignment of two sequences. The input of Jali is a query sequence S of length n and multiple alignment A of length m, with $1 \leq i \leq n$ and $1 \leq j \leq m$, respectively. The multiple alignment A consists of K sequences or rows, $1 \leq k \leq K$. The Jali algorithm aligns the query S with A, the result being a *jumping*

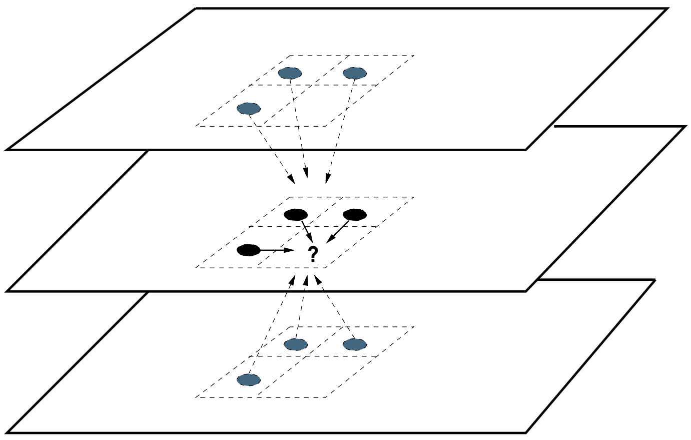


Figure 13: *Each plane symbolizes an edit matrix for the query and one of the K sequences in the multiple alignment. During the computation of a given entry $D_k(i,j)$ ($\rightarrow$ question mark), the Jumping Alignment algorithm considers not only the three predecessors in the same plane k, but also those in the other ones, adding up to $3K$ altogether.*

alignment, a special kind of a local alignment. Here, every character S_i of the query is aligned with a symbol $A_{k,j}$ in a specific row of the multiple alignment. This row may however change from k to k', $k' \neq k$. We call this event a *change in the reference sequence*, or shorter a *jump*. In this case, S_{i+1} is aligned with $A_{k',j'}$, unless a new gap is introduced into the query.

The quality of the jumping alignment is expressed by the *jumping alignment score*. The score function of the standard Smith-Waterman algorithm for pairwise alignments has been generalized for multiple sequences. Instead of the single edit matrix D as in Smith-Waterman, Jali employs K edit matrices $D_1, \ldots, D_K$, one for each alignment row in A. For $1 \leq k \leq K$, the cell $D_k(i,j)$ holds the maximal score of all alignments between the query sequence and the multiple alignment that end with positions S_i and $A_{k,j}$. The computation of $D_k(i,j)$ requires to maximize over $3K$ predecessor cells (see Figure 13). The three predecessors have to be considered in D_k and in the other $K-1$ matrices where $k \neq k'$. The latter cases are penalized by subtracting the *jumpcost* from the alignment score. This decrease is necessary to avoid wild "hopping" between the sequences of A during the computation of the jumping alignment:

$$D_k(i,j) \quad = \quad \mathit{max} \begin{cases} 0 & \\ D_k(i-1,\ j-1) & +\, w(s_i,\ A_{k,j}) \\ D_k(i-1,\ j) & -\, \mathit{gapcost} \\ D_k(i,\ j-1) & -\, \mathit{gapcost} \\ D_{k'}(i-1,\ j-1) & +\, w(s_i,\ A_{k,j}) - \mathit{jumpcost} \\ D_{k'}(i-1,\ j) & -\, \mathit{gapcost} - \mathit{jumpcost} \\ D_{k'}(i,\ j-1) & -\, \mathit{gapcost} - \mathit{jumpcost} \end{cases} \tag{2}$$

Equation (2) illustrates a basic version of the Jali algorithm with a time complexity of $O(nmK^2)$. Since every jump has the same cost, the time complexity can easily be reduced to $O(nmK)$: The maximal alignment scores obtained from the diagonal, vertical, and horizontal predecessors can be precomputed in $O(K)$ time, reducing the time complexity by one order of magnitude. The space usage however remains $O(nmK)$ (see [97] for further details).

The placement of gaps, especially in connection with affine gap costs, poses more complications. One has to distinguish between gaps that already existed in the input alignment and new gaps introduced by the Jali algorithm. Spang *et al.* describe five cases for the placement of a new gap. The gap penalties depend on where the gap is introduced (into S or A) and if it runs in parallel to a character or to an already existing gap. The time complexity of the basic algorithm can be maintained under affine gap costs if the auxiliary matrices V and H are included, as proposed by Gotoh [38]. The idea is shortly explained in Section 3.2.2. To save memory, the Jali implementation includes also the space saving technique of Huang *et al.* [48]. It reduces the space complexity of the algorithm to $O((n+m)K)$, if the alignment as such is not needed.

The initial Jali implementation featured two executables: The first one, `Jali`, computes a jumping alignment from a query sequence and a multiple alignment. The second one, `Jsearch`, is used for database searching. Its input is a multiple alignment and a database of sequences in Fasta [77] format. Early in this project we developed a third executable "`Jscan`" that completed the functionality of the suite. `Jscan` compares a single sequence against a database of multiple alignments in the *ProDom* [92] file format. All executables are available online[10] for different UNIX-based operating systems.

9 Projects

9.1 Biological Significance of Jumping Alignments

9.1.1 Overview

The results of the first evaluation in the original publication [97] were promising. In terms of quality, Jali could compete with HMMer [31], an established HMM-based method in homology recognition. In [12], we confirmed these results with a newer version of the SCOP database. The former results also showed the influence of the jumpcost: In conjunction with the Blosum62 score matrix, the Jali performance was poor at zero jumpcost, while being good at infinite jumpcost. The best results were however obtained at a jumpcost value of 22. The additional degree of freedom introduced by the jumping capability of Jali clearly makes a difference. We wondered whether the good results are due to the increased flexibility of the algorithm, or if the jumps indeed reflect structural properties of the aligned protein domains. For instance, one would expect more jumps within variable segments of the protein family

[10] `http://bibiserv.techfak.uni-bielefeld.de/jali`

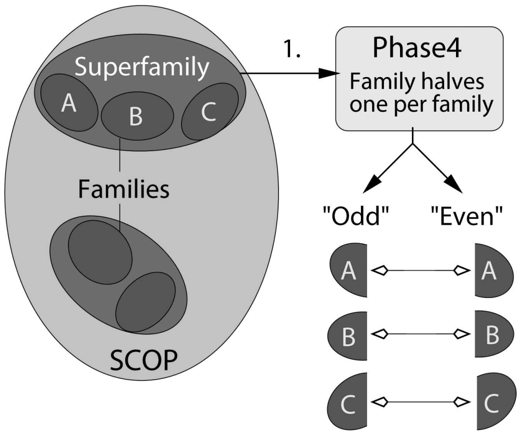


Figure 14: *Phase4 construction of the "family halves one per family" model (fhopf), step 1: Each family of every SCOP superfamily is split into two parts, the "odd" and the "even" set.*

than in conserved segments. The regular secondary structure elements α-helix and β-strand are needed to maintain the structure of the protein. They are therefore more conserved in terms of sequence space and less variable than coils and loops. Coils and loops are nevertheless indispensable parts of the protein, even though being less constrained. We expect to observe more jumps disrupting a loop element than an α-helix or β-strand if Jali can indeed detect structural features within homologous proteins. Therefore we define a Jali jump as *structurally significant*, if it occurs between adjacent secondary structure elements, not disrupting the involved secondary structure segments. If a coil or loop is involved, this strict requirement is slightly loosened. The jump may be within the coil or loop, but it has to be close to its border (one residue), because secondary structure is not absolutely rigid. We would appreciate to find many such cases, because this would suggest a connection to protein structure. In this case, Jali might be further improved by incorporating structure information in the jump strategy.

9.1.2 Experimental setup

We constructed the evaluation environment with *Phase4* (described in Section 7.1). Each family within a given SCOP superfamily was split into an odd and an even half, applying the Phase4 "family halves one per family" (fhopf) model (see Figure 14). The sequences in the odd set are the query sequences. They are pooled into one large set (see Figure 15). The sequences in the even sets are aligned with ClustalW (version 1.81) to serve as input for the Jali algorithm. Jali computes jumping alignments from these multiple alignments and each query sequence in the odd set (see Figure 16).

We applied this procedure to two SCOP subsets of sequences that were obtained from the ASTRAL website (see Section 6.4.1). One contained sequence families with at most 40% sequence identity, the sequences in the second one were up to 70%

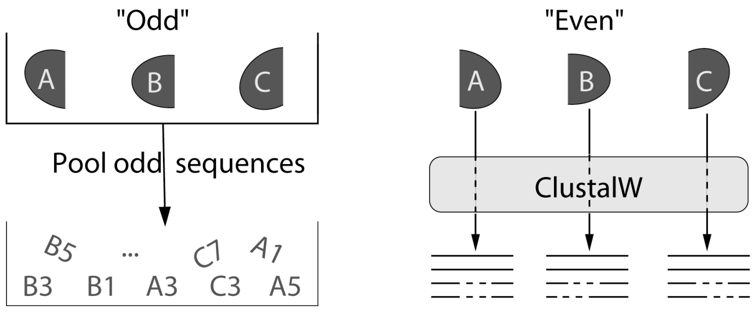


Figure 15: *Phase4 construction of the "fhopf" model (see text), step 2: The odd sequences are pooled. The sequences in the "even" family sets are aligned with ClustalW.*

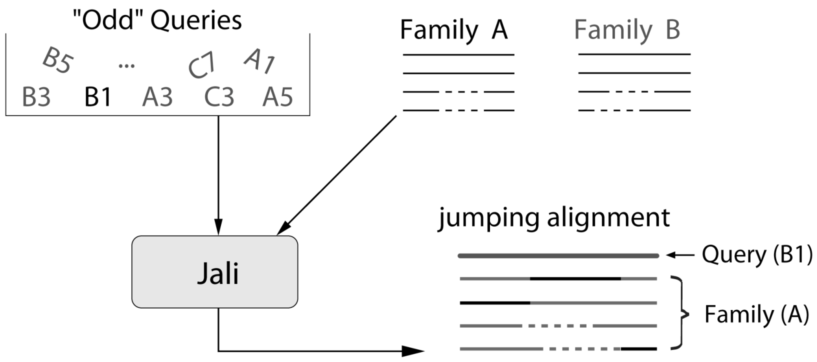


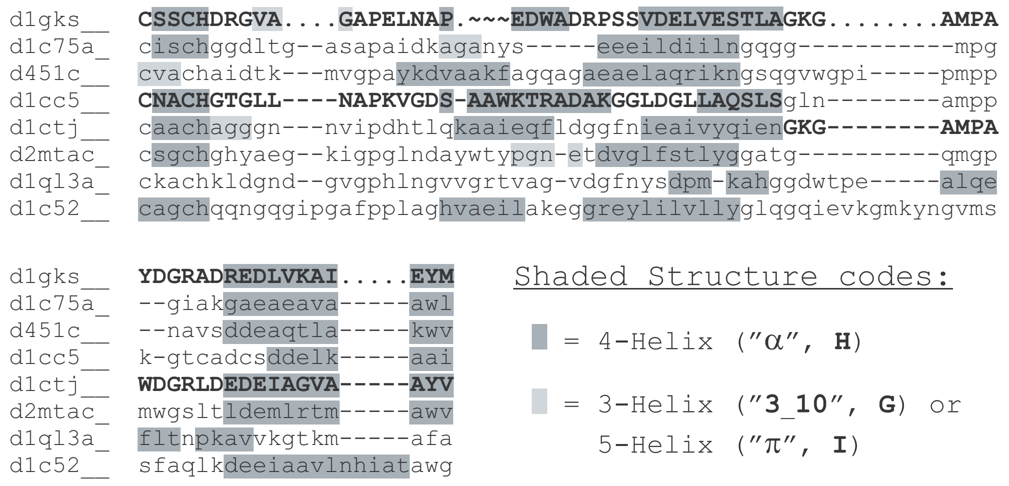
Figure 16: *Phase4 construction of the fhopf model (see text), step 3: A jumping alignment is computed in an all-against-all approach from every (query, multiple alignment) pair.*

identitcal. Each jumping alignment was computed with three different Blosum62 jumpcost values: 14, 18 and 22. After that, the secondary structure information was added in a semiautomatic fashion. It was parsed from the "PDBfinder II" database (see Section 6.2). Then we checked the jumping alignments manually for cases of structural significance.

9.1.3 Results and Discussion

We checked almost 40 different jumping alignments from both SCOP subsets at three different jumpcost values for structural significance. Only one alignment with structural significance was found, from the "astral40" subset (see Figure 17). The structural significance was present at jumpcost values of 18 and 22. At jumpcost 14, the number of jumps increased by one and the jump position changed, losing the structural significance.

In this experimental setup, the Jumping Alignment algorithm does rarely detect the secondary structure underlying the protein multiple alignment. Some of the



```
d1gks__   CSSCHDRGVA....GAPELNAP.~~~EDWADRPSSVDELVESTLAGKG........AMPA
d1c75a_   cischggdltg--asapaidkaganys-----eeeildiilngqgg-----------mpg
d451c__   cvachaidtk---mvgpaykdvaakfagqagaeaelaqrikngsqgvwgpi-----pmpp
d1cc5__   CNACHGTGLL----NAPKVGDS-AAWKTRADAKGGLDGLLAQSLSgln--------ampp
d1ctj__   caachagggn---nvipdhtlqkaaieqfldggfnieaivyqienGKG--------AMPA
d2mtac_   csgchghyaeg--kigpglndaywtypgn-etdvglfstlyggatg----------qmgp
d1ql3a_   ckachkldgnd--gvgphlngvvgrtvag-vdgfnysdpm-kahggdwtpe-----alqe
d1c52__   cagchqqngqgipgafpplaghvaeilakeggreylilvllyglqgqievkgmkyngvms

d1gks__   YDGRADREDLVKAI.....EYM
d1c75a_   --giakgaeaeava-----awl
d451c__   --navsddeaqtla-----kwv
d1cc5__   k-gtcadcsddelk-----aai
d1ctj__   WDGRLDEDEIAGVA-----AYV
d2mtac_   mwgsltldemlrtm-----awv
d1ql3a_   fltnpkavvkgtkm-----afa
d1c52__   sfaqlkdeeiaavlnhiatawg
```

<u>Shaded Structure codes:</u>

▮ = 4-Helix ("α", **H**)

▯ = 3-Helix ("**3_10**", **G**) or 5-Helix ("π", **I**)

Figure 17: *Jumping Alignment of length 82, score 96: Sequence of Cytochrome C551 vs. a multiple alignment of sequences from the same family (monodomain cytochrome C). The secondary structure annotation shows the hypothesized "structural significance".*

reasons lie in the algorithm itself: Jali jumps as soon as the gain in the Jali score exceeds the jumpcost penalty. There are cases where co-optimal alternative jumps are also possible, but the current version of Jali can not compute co-optimal jumping alignments. There may be cases of structural significance we have missed because of that.

The quality of the input alignments is another factor whose influence is difficult to assess. We believe that some of the errors within a multiple alignment can be avoided by Jali due to its jumping capability. The overall alignment quality however matters even for Jali. Especially the multiple alignments from the astral40 subset are far from perfect, because of the low sequence similarity between the sequences. This is easy to see when structure information is also available: The syntenic secondary structure segments are not always found in "blocks". The few cases of structural significance we observed are therefore not surprising. But even at better alignment quality, we would probably not have observed many more cases. What we basically tried to do here was detecting sequence-structure relationships with a sequence-based algorithm. The difficulty of this task is generally accepted.

The relevance of our results is hard to discuss. Statistical considerations were totally left out, because they are very hard to obtain. The null hypothesis for this investigation would have been to assume that the Jali algorithm does not recognize secondary structure. Then we could have compared the number of structurally significant alignments found in this investigation with those in a setup with random sequences, yet with simulated secondary structure. If both numbers were not significantly different, the null hypothesis was confirmed. The difficulties in this scenario are obvious: Generating families of random sequences being similar enough to real families is a problem by itself. Modeling the secondary structure in these random sequences is even more difficult. This would require to understand protein folding, something we are still far away from. Therefore statistical considerations were not possible in this experiment.

9.2 Experiments with simulated and natural protein families

9.2.1 Overview

Our work concentrates on understanding the good results of the Jali algorithm. In the previous project, we manually investigated a few jumping alignments for structurally significant jumps. This resulted in interesting insights, but not in reliable answers. One reason is that the investigated cases deliver not enough evidence to justify assumptions. So the next step is to obtain and compare quantitative data from a different experimental setup. We have set up experiments involving protein domains of biological and artificial origin. They are described in Section 9.2.2. We ran the Jali algorithm in these settings and monitored several parameters. The main objectives were (a) to see the behavior of the jumpcost parameter, (b) to observe significant tendencies, and (c) to investigate interesting jumping alignments, indicated by outliers in the generated data.

Parts of this work have been published at the *German Conference of Bioinformatics* in 2003 [13] and in a technical report [12] that is available online[11]. It supplements this work and describes material, methods, and the evaluation procedure in greater detail.

9.2.2 Experimental setup

Biological setting and evaluation procedure. The experimental setup is similar to the one described in the previous project (see Section 9.1.2). We apply *Phase4* to treat all SCOP superfamilies as shown in Figures 14 to 16: Each family containing at least eight sequences is split into an *odd* and an *even* half. This ensures that the multiple alignments computed from the even family halves contain at least four sequences. The even sets are then multiply aligned with ClustalW (version 1.81). The odd sequences are query sequences, they are pooled in one file. After these preparations, Jali is run with all queries in the odd set against all family alignments in the same superfamily. We record the values for three parameters: The jumping alignment score, the number of jumps, and the length of the jumping alignment.

This experimental setup was varied in several ways. We used two different SCOP subsets, astral40 and astral70. The numbers indicate the maximal percentage of sequence identity between any two sequences in each subset. In some experiments, several jumpcost penalties were applied. In addition, the hierarchical structure of the SCOP database allows to distinguish two kinds of jumping alignments. A jumping alignment where query and multiple alignment come from the same SCOP fam-

[11] `http://www.cebitec.uni-bielefeld.de/~bannert/pubs.html`

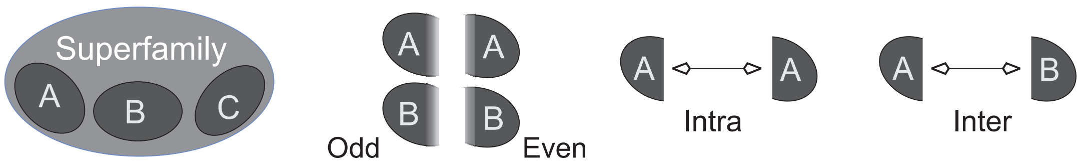


Figure 18: *An example superfamily (left) consists of three families A, B and C that are split into halves (center left). All measurements within one family have an intra relation (center right), between two different families we speak of an inter relation (right).*

ily ("intra") and a jumping alignment where the query is from family A and the multiple alignment is from family B ("inter"), see Figure 18.

If a superfamily consists of only one family, an inter setting does not exist. If at least two families A and B are present and both contain eight or more sequences, the evaluation is straightforward. If a family has less than eight sequences, its odd sequences are nevertheless put into the set of query sequences. A multiple alignment of the even sequences is however not constructed. The idea behind this is to generate more data in the inter setting. However, it would make no sense to include the evaluation results of a superfamily whose family B consists of only one sequence. At least three sequences with an inter relationship had to be present in each superfamily to contribute experimental data.

All runs were made with the VTML160 score matrix [72], gap initiation cost of 12 and gap extension cost of 4, if not otherwise specified.

Artificial setting. The data obtained with the natural sequences in the biological setting is complemented with data from random sequences. The advantage in creating artificial sequence families is that we know their evolutionary history. We used the program ROSE [101] (see Section 7.2) to create ten "superfamilies", every one containing three "families". For each superfamily, we first generated a random sequence S of length 200 as common ancestor. Two more sequences were derived from S, see also Figure 19: The sequence S^p has a length of 180 amino acids. The twenty missing residues are due to two deletions of size ten. Besides being shortened, another ten-residue segment of S was substituted with an equally sized segment of random sequence. Similarly, the longer sequence S_q is 220 amino acids long and it differs from S by two insertions and one substitution. The positions of these simulated evolutionary events were the same for all sequences. We call these three sequences a *seed set*. We derived twenty sequences from each seed sequence in a randomized evolutionary process with ROSE. The average evolutionary distance was 160 PAM, with an insertion/deletion probability of 2%. The ten obtained superfamilies finally contained three families of 20 sequences each, with average lengths of 180, 200, and 220 amino acids. They were treated with *Phase4* as described in the biological setting in (a), resulting in the same final all-against-all Jali runs within each superfamily. We made several experiments with slight modifications: In our

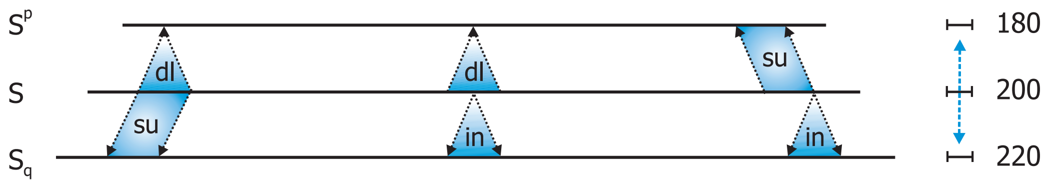


Figure 19: Creation of the protein superfamilies used in the artificial setting: The original sequence S (length 200, center) served as template in the creation of S^p (length 180, top), and S_q (length 220, bottom). S^p differs from S by two deletions (dl) and one substitution (su) with random sequence. The difference between S and S_q is due to two insertions (in) and another substitution.

first experiment, we collected data at jumpcost values of 14, 18, and 22. We call this experiment the *random200* setting. In a second experiment, we generated 90 random sequences of length 200, unrelated to the families in the random200 setting. By Jali runs of these 90 sequences against the ten families of 200 amino acids average length, we obtained data for unrelated sequence-to-family comparisons. Since there is no structure in the 90 sequences mentioned before, we call this experiment *unstructured*.

9.2.3 Results and Discussion

Biological setting. In our biological setting, we compared both SCOP subsets (astral40 and astral70) at jumpcost 18. The values were calculated from all superfamilies with an inter setting (at least two families being present, one of them with eight or more sequence, the other one with at least three sequences). All of those families always have an *intra* setting as well. The results are shown in Figure 20.

The astral70 score measured in the *intra* setting is twice as high as the corresponding astral40 score. The inter scores are roughly equal, but three to six times lower than the *intra* scores. The values measured for the local alignment length also show a good distinction between the *intra* and the inter setting. The difference between the astral40 and astral70 subsets seems to be less important here. The *intra* alignment length is about twice the inter alignment length. Those characteristics also hold for the average number of jumps. The standard deviation for all observed parameters is high, in case of the number of jumps it is about as big as the value itself.

The higher average score and alignment length in the *intra* setting can easily be explained. The SCOP definition for a family states that two sequences in a family share a certain degree of sequence identity. The 40% maximal identity between two family members in the astral40 subset can be detected by our method, the up to 70% identity in the astral70 subset even better. But why are both inter scores almost equal? Would one not expect a higher score for the astral70 subset? Mostly no. It must be kept in mind that astral70 and astral40 are only subsets of the full sequence set within a SCOP family. The only difference involved is the subset size. The dis-

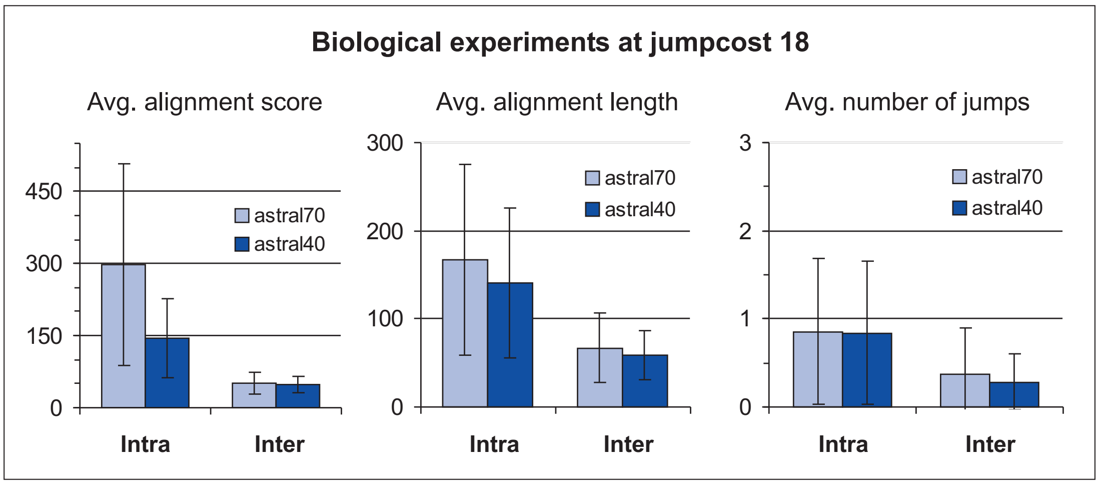


__Figure 20:__ Results of the biological setting using the astral40 and astral70 subsets. Shown are the averages for three jumping alignment parameters: the alignment score (left), the alignment length (center), and the number of jumps (right). The standard deviation is indicated by the error bars.

tance to other families remains the same in sequence space (see Figure 21). The only reason for a higher score in the larger astral70 subset is the extended choice that a sequence has to find similar sequences.

The number of jumps observed in the alignments is partly dependent on score and alignment length, and vice versa. Lower scores and shorter alignments allow for less jumps, which explains the fewer jumps in the inter setting. On the other hand, the score and length difference between the astral70 and astral40 datasets in the *intra* setting do not seem to affect the jumping behavior, or the effect is compensated by other influencing factors. A closer look at the individual jumping alignment results from this evaluation (not shown) explains the high standard deviation. Even within a family, the collected values distribute over a broad range. If different superfamilies are compared, the distribution range changes and/or the values distribute around different arithmetic averages. One reason is that the distances (in sequence space) between the SCOP domains are not uniform. The classification itself can be another reason: Sometimes a sequence with an inter relationship scored unexpectedly high in our Jali runs, whereas an *intra* sequence scored rather low. We investigated one of these cases, the corresponding jumping alignment is shown in Figure 22. The query is a *Trypsin* from *Streptomyces griseus*. In SCOP version 1.55, where we made these experiments, the query is classified into the SCOP family 2.47.1.1 "Prokaryotic proteases". The multiple alignment is from family 2.47.1.2 "Eukaryotic proteases", corresponding to an inter relationship. Figure 22 shows the query to be quite similar to the aligned family, so one may inquire why it was not classified into the same SCOP family. The difference in this classification is the evolutionary origin of the

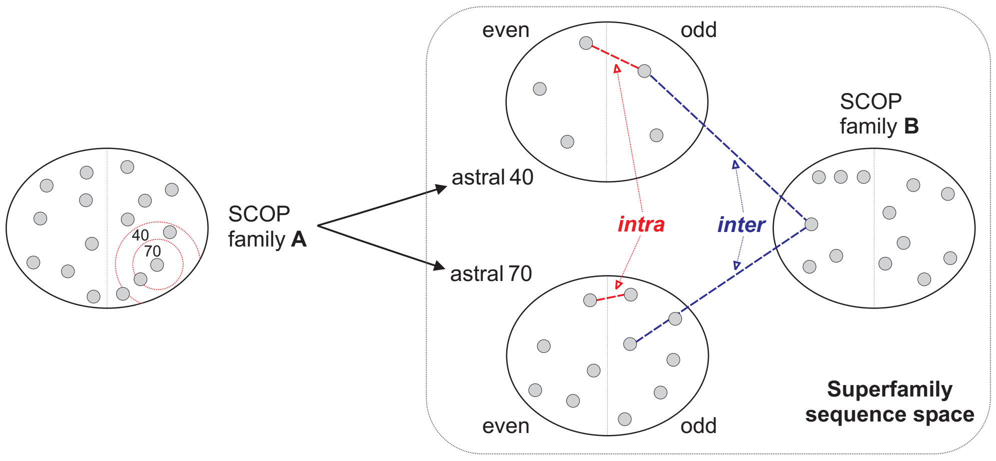


Figure 21: *Illustration showing why the intra distances between the astral40 and astral70 SCOP subsets differ in sequence space. In comparison with the astral70 filter, the astral40 filter has an increased radius (left). This is causing the larger intra distances in the astral40 subset (center, top). The inter distances are much more pronounced (right). The differences imposed by using the two astral subsets are therefore negligible.*

sequence. In spite of the different origin, the sequence similarity and the secondary structure patterns in Figure 22 are remarkably similar. We believe this classification to be at least unfortunate. So far, it has not been revised in SCOP up to version 1.73. As long as classification problems like this occur in biological databases, no database search method will be able to show its full strength in an evaluation.

Artificial setting. Figure 23 shows a comparison of the *intra* and inter random200 settings at three different jumpcost values. The variation of the jumpcost parameter within this range has little effect on the score (Figure 23, left) and the local alignment length (Figure 23, center). However, the average number of jumps is noticeably decreasing with increasing jumpcost (Figure 23, right). Aside from the effect of the jumpcost variation, the *intra* scores are much higher than the inter scores. It seems that the differences in the seed set suffice to account for a drop of about 75% in score. Figure 24 shows the results of the unstructured setting in comparison with the random200 results at jumpcost 18. All values in the unstructured setting are clearly below the random200 inter values. For example, the unstructured average score and alignment length are only about one fourth and one third of the corresponding random200 values, respectively.

A comparison of the artificial and biological setting shows that our random model is partly sufficient to simulate SCOP families. The ratio between the artificial *intra* and inter scores is in the same range as in the SCOP astral70 subset. The absolute scores in the artificial setting are however clearly higher than the biological ones, indicating less sequence similarity within the SCOP superfamilies. The average SCOP

```
d1sgt__    VVGGTRAAQGEFPFMVRLSMG......~~~CGGALYAQDIVLTAAHCVSG.......SGNNTS
            BT EE  TTSSTTEEEETTT         EEEEEEETTEEEE GGGSS        SEE
d1ekbb_    IVGGSDSREGAWPWVVALYFD---dqq--vcgaslvsrdwlvsaahcvyg----rnmepskwk
            BS EE  TTS TTEEEEEET   TEE  EEEEEE SSSEEEE HHHHTT      SSGGGEE
d1buia_    vvggcvahphswpwqvslrtr---fgmh-fcggtlispewvltaahclek-----sprpssyk
            BSSEE  TTS TTEEEEEET   TS E EEEEEESSSSEEEE TTTTSS      S GGGEE
d1azza_    ivggveavpnswphqaalfidd-----MYFCGGSLISPEWILTAAHCMDG--------AGFVD
            BS EE  TTSSTTEEEEEETT     TEEEEEEEEETTEEEE HHHHTT         S EE
d1a7s__    ivggrkarprqfpflasiqnqg-----rhfcggaliharfvmtaascfp----------gvst
            BS EE  TTSSTTEEEEEETT     EEEEEEEEEETTEEEE GGG             SEE

d1sgt__    ITATGGVVDLQSGAAVKVRSTKVLQAPGYNGTG....KDWALIKLAQP....~~INQPTLKIA
            EEEES SBTT TT EEEEEEEEEE TT  SSS       EEEEESS         S   EE
d1ekbb_    avlglhmasnltspqietrlidqivinphynkrr-knndiammhlemk----vnytdyiqpic
           EEES  BTT    TT EEEEEEEEEE TT BTTT TBS  EEEEESS         SS    B
d1buia_    vilgahqevnlephvqeievs-rlfleptr-------KDIALLKLSSP----AVITDKVIPAC
           EEES SBSSS  TT EEEEEE EEEE TT         S EEEEESS        BTTB   B
d1azza_    VVLGAHNIREDEATQVTIQSTDFTVHENYNSFVi--sndiavirlpvp----vtltaaiatvg
           EEES SBSSS  TT EEEEE  EEE TT BTTTT  BS  EEEE SS         SSS   B
d1a7s__    vvlgaydlrrrerqsrqtfsissmsengydpqqn--lndlmllqldre----anltssvtilp
           EEES SSTTS  TTT EEEEEEEEE SS BTTTT  BS  EEEEESS        BTTB

d1sgt__    TTT.....~~AYNQGTFTVAGWGANREGGS......QQRYLLKANVPFVSDAACRSAYG.NEL
           SSS       TTSSSEEEEEESS SSTT          SB EEEEEEEE HHHHHHHG  GG
d1ekbb_    lpe---enqvfppgricsiagwgaliyqggst------advlqeadvpllsnekc-qqqmpeyn
                 TT    TT EEEEEESSBSSTTS B       SB EEEEEEB  HHHH HHH TTS
d1buia_    LPS---pnyvvadrtecfitgwgetqgtfgag-------llkeaqlpvienkvcnryeflNGR
                 TT    SS EEEEEE    SS S  S       B EEEEEEEE HHHHTSTTTTTT
d1azza_    lpst-----dvgvgtvvtptgwglpsdsalg-----isdvLRQVDVPIMSNADCDAVYG----
             SS        TT EEEEEESS SSTT SS     S SB EE  EEEE HHHHHHHS
d1a7s__    lplqn---ATVEAGTRCQVAGWGSQRSGGR------LSRFprfvnvtvtpedqcrpnn-----
              TT       TT EEEEEES  SSTT          SS EEEEEEE  GGGS TTE

d1sgt__    .......VANEEICAGY..PDTGGVDTCQGDSGGPMFRKD.NADEWIQVGIVSWG..YGCARP
                   TTTEEEES   TTT   B  TT TT EEEEE  TTS EEEEEEEEE   SSSS T
d1ekbb_    -------itenmvcagy---eaggvdscqgdsggplmcqe--nnrwllagvtsfg--yqcalp
                    TTEEEE     TT   B  TT TT EEEEEE  TTEEEEEEEEEE   SSSS T
d1buia_    -------VQSTELCAGH---LAGGTDSCQGDSGGPLVCFE--KDKYILQGVTSWG--LGCARP
                    TTEEEE     TT S B  TT TT EEEEEE  TTEEEEEEEEEEE  SSSS T
d1azza_    ------ivtdgnicids----tggkgtcngdsggplnyng------ltygitsfgaaagceag
                    TTEEEE      TTT B  TT TT EEEETT      EEEEEEEEEETT TTS
d1a7s__    -------vctg--vltr----rgg--icngdggtplvceg------lahgvasfslgp-cgrg
                  EEEE   SSS    S B    TT TT EEEETT      EEEEEEEEE  SS TTSS

d1sgt__    GYPGVYTEVSTFASAIASAAR
           T  EEEEEHHHHHHHHHHHHH
d1ekbb_    nrpgvyarvprftewiqsflh
           T  EEEEEGGGTHHHHHTT
d1buia_    NKPGVYVRVSRFVTWIEGVMR
           T  EEEE GGGTHHHHHHHHH
d1azza_    y-pdaftrvtyfldwiq---t
              EEEEESGGGHHHHH   H
d1a7s__    --pdfftrvalfrdwidgvln
              EEEEEGGGGHHHHHHHHH
```

Figure 22: *Jumping alignment showing a questionable classification in SCOP 1.55. The query "d1sgt__" (blue and boldfaced) is quite similar to the multiple alignment. This is supported by the underlying secondary structure. Nevertheless, the query was classified into a different family. The reference sequence is shown in boldfaced, red capital letters.*

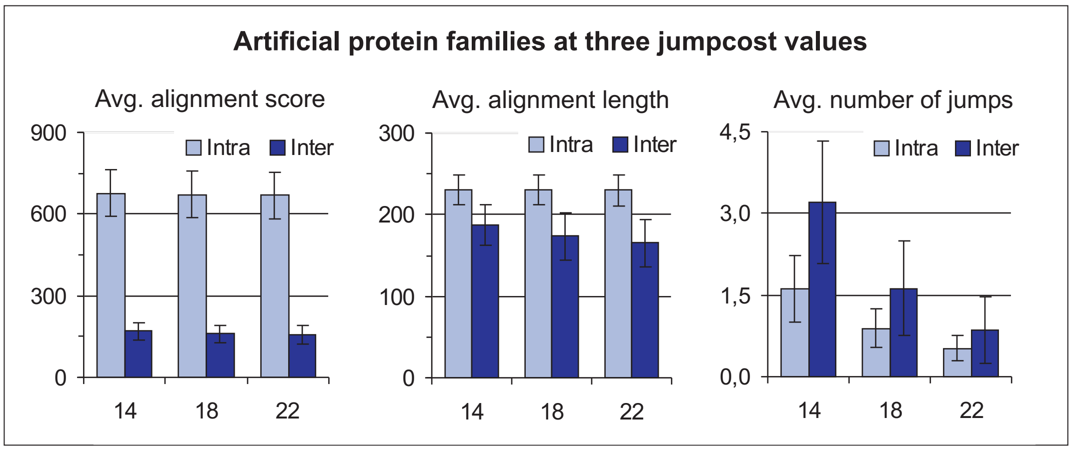


Figure 23: Results of the random200 setting at jumpcosts 14, 18, and 22: Shown are the averages for alignment score (left), the local alignment length (center) and the number of jumps (right) for the inter and intra settings. The error bars in each diagram display the standard deviation.

domain is about ten percent shorter than the average domain in our artificial setup, which justified slightly increased artificial scores. The scores in our experiment can not be explained with this length difference, because they are far too high, at least twice the biological ones. This also shows that especially inter-related SCOP domains are far apart in sequence space. The average score and local alignment length are close to the random similarities observed in the artificial unstructured experiment. The signal intensity seems to be so low that the promising results achieved by Jali in an earlier evaluation are somewhat surprising [97].

9.3 Conclusion

We have made another step in understanding the Jali algorithm as a database search tool. It largely behaved as we expected, we encountered no surprises. Jali is just sensitive enough to separate between remote homologs and unrelated sequences, as shown by the "inter" results in Section 9.2.3. This separation is however not reliable, because the amount of detectable evolutionary information is low. The "unstructured" results show that the inter scores are often close to random similarities.

We wondered about the impact of the input alignments. Section 9.1.3 already showed that certain structural features are sometimes lost in multiple alignments computed with ClustalW. In a similar experiment ("random300") in [12], we used the evolutionarily correct ROSE alignments as input to Jali, to compare the results with those of ClustalW. Even though the random300 setup is slightly different from the random200 setup, the results should be similar in quality. We report them in Figure 25. The *intra* results show that the re-alignment with ClustalW leads to a significant loss of evolutionary information. The corresponding Jali scores are much

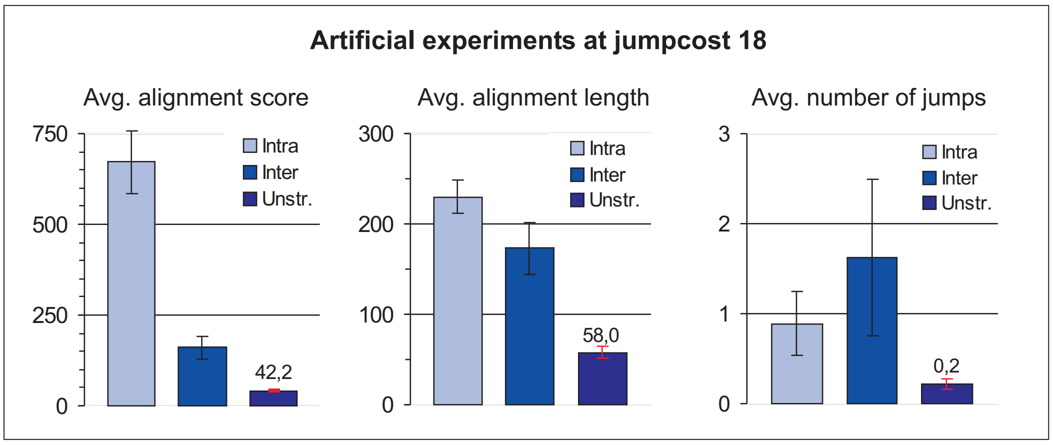


Figure 24: *Comparison of the unstructured (Unstr.) results to those of random200, at jumpcost 18: Shown are the averaged values for alignment score (left), local alignment length (center), and number of jumps (right). Since the unstructured values are so low, exact values are provided. Please note that no intra or inter setting exists for the unstructured setting.*

lower than the scores obtained with the ROSE alignments.
Jali is able to detect the additional sequence similarity in the ROSE alignments and we conclude that the alignment procedure is the limiting factor when the input sequences are closely related. This does not hold in the inter setting with more distantly related sequences. Here, the Jali scores are roughly equal for both alignment sources. The additional evolutionary information in the ROSE alignments can hardly be used. At this low level of sequence similarity, Jali is (under the given parameters) the limiting factor to a more pronounced separation.

The evolutionary model for the artificial protein families featured a large evolutionary distance of 160 PAM and two percent indel probability. The SCOP domains however seem to be even more distantly related. Can the differences only be explained by the number of mutations and indels? Another possibility is that the used mutation model does not cover all evolutionary events that occur in practice. For example, it does not model rearrangements of sequence segments, e.g. circular permutations. It would be nice to use more refined models in the future, and to incorporate them into algorithms for homology recognition.

The overall influence of the jumping behavior is hard to judge. It does often, but not always, correlate with the alignment score and length. Within a certain jumpcost range, it surely offers more flexibility to Jali than algorithms without this capability. Figure 25 shows that if Jali is run on evolutionarily correct input alignments, the number of jumps is lower than with alignments with estimated evolutionary history. The correct alignments allow Jali to use less sequences of the input alignment. The ROSE alignments are probably more consistent than the ClustalW alignments.

That supports our earlier assumption that jumping is useful to cope with suboptimal alignments. Here, it promotes longer alignments with higher scores. Most of the score units won by changing the reference sequence are then however traded in for jumping. In conclusion, jumping offers more flexibility to homology recognition,

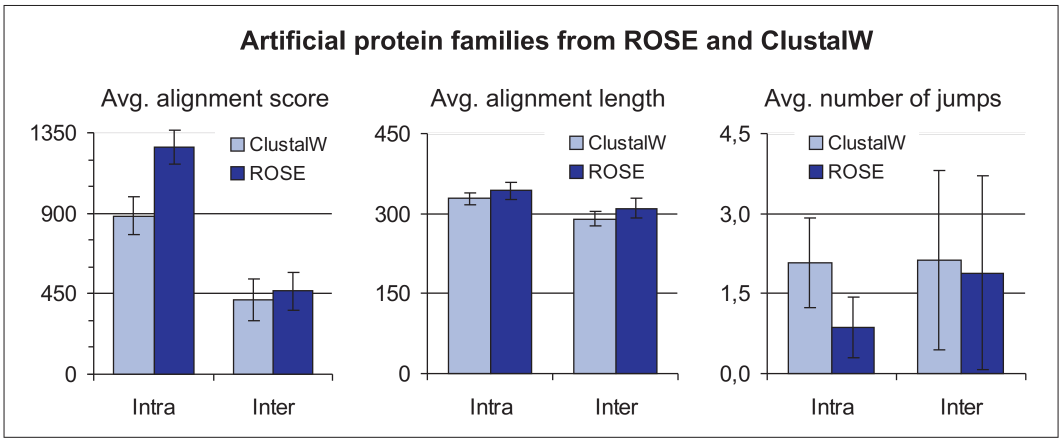


Figure 25: Comparison with multiple alignments from ClustalW and ROSE at jumpcost 18. The results are from a slightly different experimental setup and can only qualitatively compared to the "random200" results. The comparison shows that ClustalW loses a significant amount of information during the alignment of intra-related sequences, because Jali is able to reach much higher alignment scores with the ROSE alignments (left). The average number of jumps (right) supports the importance of the input alignments. With the ROSE alignments, it is lower than with the ClustalW alignments. The evolutionary changes seem to be more accessible with intra-related sequences, Jali can often pick the alignment row with the fewest evolutionary differences. In contrast, the corresponding inter values are almost equal, regardless of the used alignment source.

if the jumpcost are within a reasonable range. Nevertheless, the current concept is not as flexible as it could be. It would be interesting to implement an algorithm that could leave out whole regions of the input alignment that do not fit to the query at all. Then, one had to use certain constraints that allow to separate a few blocks of random similarity from real homology. In Part IV of this thesis we will develop a method that implements some of these ideas.

Part IV

Passta

10 Introduction

10.1 Motivation

Algorithms for local and global alignment are the foundation of many methods that search a sequence database. Both approaches are suitable in the detection of closely related sequences that are separated by only few mutations. The picture is changing with increasing evolutionary divergence. The many additional point mutations in combination with other evolutionary changes often result in poor sequence similarity. Sometimes it is so low that only a few conserved "islands" remain within seemingly unrelated sequences. When one is searching for such remotely related sequences, global alignment is not the method of choice. The reason is that positive contributions to the alignment score are easily evened out through mismatches or gaps. Local alignment does then often achieve better results, because it requires the presence of only one reasonably conserved segment of sequence.

However, local alignments are often short, close to the length of high-scoring alignments reflecting random similarities between unrelated sequences. This situation can be improved if several local alignments contribute to a combined alignment score. A well-known example of such a method is *Gapped BLAST* [6], which can be seen as a prototype of advanced methods in sequence database searching. It produces neither global nor local alignments, its result is usually an intermediate depending on the quality of one to several local hits. Under certain conditions, they are extended and joined into a longer alignment. If two local hits are however too far apart, separated by a region with little sequence similarity, the result will again be a collection of local hits.

Other advanced methods in sequence database searching include probabilistic approaches like HMMs, which were shortly discussed in Section 4 together with other classic approaches for sequence database searching. Most of these approaches share the following characteristics:

1. The aligned sequences have to be collinear to achieve optimal results: Classic alignment approaches do not explicitely model rearrangements of gene segments. Rearrangements are less frequent modifications that happen during evolution. A two-step mechanism is proposed to explain them [79]. A complete or partial duplication of a gene is followed by certain deletions and/or mutation events, finally leading to new start and stop codons. Another mechanism to obtain rearrangements is posttranslational modification, the rearranged protein fragments are cleaved and ligated in reversed order [23]. The result of these events corresponds to the relocation of a gene or a gene segment. Classic methods in database searching can either align the relocated portion or the

stable portion of the gene, but not both simultaneously (see Figure 26). These methods lose much of the available information during the alignment of rearranged sequences.

2. The target sequences in the database can not be split into reusable segments: Rearrangements are not the only challenge that classic methods in sequence analysis face. Intragenic duplications or even repeats cause similar problems. They are probably caused by intragenic duplication and recombination events [8]. In terms of sequence analysis, they can be regarded as re-insertions of already existing sequence segments into the parent sequence. If the target sequences could be split into reusable fragments, those fragments could be several times aligned with the query, conveniently modeling such repeats. Classic methods for database searching have to model these cases as indels, because most of them can not detect the repeat(s) and reuse the corresponding sequence segment.

3. Classic approaches answer only a few questions: The result of a database search is usually displayed as a 1:n list of target scores or e-values that specify the similarity to the query. Often the corresponding alignments are also available. Clusters of closely related sequences in the target list are a common observation, because many sequence databases contain highly similar or even redundant sequences.

   If there is no global similarity between the query and any target, the result is essentially a list of partial hits. Without additional effort, it is neither obvious whether the unaligned parts of the query bear similarities to other targets in the database, nor which combination of targets could "explain" the query in the optimal way. Besides, experience shows that most users of tools in sequence analysis are only interested in a small number of promising candidates.

We propose a method in homology recognition that is able to circumvent these limitations. The concept is outlined in the following section, where we also describe related work. The implementation details are provided in Sections 11 and 12.

10.2 Method outline and related work

The purpose of our method is to allow homology recognition without the limitations described in Section 10.1. Our method is called *Passta*, short for Protein annotation by secondary structure based alignments [14]. A schematic overview is given in Figure 27. The core of Passta is our database *"PasstaDB"*. It stores sequences corresponding to secondary structure elements[12] (SSEs) and associated data from three secondary databases. The origin of the data and the integration procedure are described in Section 11.

[12]This term is often used when the atomic structure of these segments is focused on. If not otherwise specified, we always refer to the *sequence* of these elements.

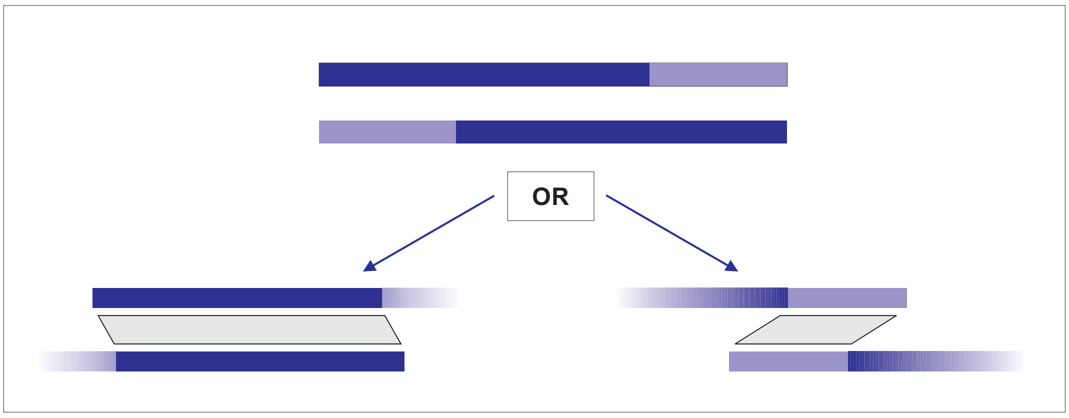


Figure 26: Classic methods in database searching are not well suited to align sequences with rearrangements. They can either align the blue portion (left) or the pink portion (right), but not both. This may affect their search quality.

The annotation of a user-provided query sequence is in two phases. In the first phase, we compute local alignments of each SSE and the query to select a set of promising targets from the database. The resulting *candidate set* is the input of the annotation protocol "Pass Two". Here, we use the Waterman-Eggert algorithm [108] to compute one to several alignments of each candidate SSE with the query. Each *valid* alignment is represented by a vertex in a directed acyclic graph (DAG). These vertices are associated with the alignment score and other information. Any path of vertices corresponds to a set of non-overlapping alignments. Under certain constraints, we compute an *optimal* path with the highest weight, i.e. with the highest sum of alignment scores.

The final result is a HTML file showing the query with the aligned SSEs of the optimal path. Each site in the query can only be aligned with one SSE at a time, but we display all such alignments simultaneously, however ordered by protein- and domain source. The aligned SSEs can originate from arbitrary proteins, thereby showing which part of the query is best explained by what database target. Rearrangements can also be modeled, because the aligned SSEs can be arbitrarily ordered. Using the Waterman-Eggert algorithm provides the co- and suboptimal alignments that we need to model repeats. The structural and functional annotation is provided by links to the SCOP classification, which provides the annotation for the query sequence. The details of the "PasstaRun" protocol are explained in Section 12.

Passta can be seen as a new combination of ideas, with a focus on protein domains and secondary structure in homology recognition. Some of its components were already implemented in other approaches. These approaches are however neither restricted to homology recognition nor to proteins. All of them compare their input molecules in different ways and with different goals. In the following discussion, it does not matter whether this goal is genome alignment or homology recognition. Nevertheless, we describe only approaches that are able to work with fragments

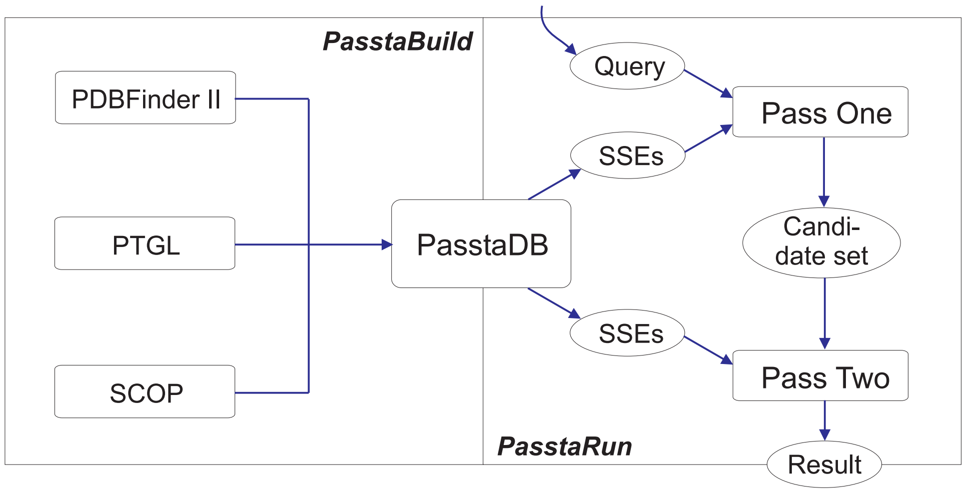


Figure 27: *Overview over Passta: The left side of the figure ("PasstaBuild") shows the integration of the three source databases into PasstaDB (center). The right side illustrates the main steps of "PasstaRun". Details are provided in the text.*

of the input molecules, because that is a prerequisite to model rearrangements and repeats.

Many sequence-based approaches in *genome alignment* can handle sequence fragments. Their purpose is to align two or more genomic sequences, even if the genomic layout is not collinear. One characteristic of these approaches is to divide the experimental protocol into a filtering and an alignment phase, similar to our approach in Passta. In the first phase, long exact or approximate matches of the genomes are seeked, and in the second phase a common layout of the fragments is computed [21, 27, 60]. When this layout is required to be a compatible (i.e. non-overlapping and collinear) sequence of the fragments from the first phase, this procedure is called *chaining* [1]. Sometimes the relationships between the sequences are more complex and a collinear alignment of the given sequences is impossible. This may even be true for closely related genomes [29]. Then rearrangements need to be considered in a more *general* chaining procedure [21, 27].

These methods are customized to work well on genomic scale, but they are usually not suited for the alignment of gene-sized segments. Especially protein-coding genes can cause problems if they are composed of multiple protein domains. Many proteins share a complete set or a subset of domains, often however in different order. A robust way to deal with such cases of rearranged building blocks is to identify each domain independently. This also allows to detect domain duplications and new domain combinations. SMART [91, 61] is an example of such a method. It was initially developed to detect signalling domains in proteins. Newer versions also allow the detection of domains with other functions. SMART uses HMMs based on manually curated multiple alignments of domain families. Another example is PFAM [32], a database of domain families. Each PFAM family is represented by two multiple alignments and two profile HMMs. The PFAM website[13] offers a service for domain identifications by sequence. Two special kinds of rearrangements are domain swaps (AB → BA) and circular permutations (ABC → CAB or ABC → BCA). In comparison with the rather rare swaps [35], circular permutations (CPs) of complete domains are more frequent. Weiner *et al.* [109] developed *RASPODOM*, a special method to detect CPs in a database of known domains or shorter motifs. They use a modified version of the Needleman-Wunsch algorithm [74] to align the sequences in question. Instead of the usual setup, both sequences are duplicated prior to the alignment, resulting in an edit matrix with four quadrants. If the aligned sequences are circularly permuted, the resulting patterns are recognized by *RASPODOM*. Depending on the dataset, the method is not limited to CPs between complete structural domains, it can also detect rearrangements *within* protein domains. Even though such cases are less frequent, intra-domain rearrangements exist as well as intra-domain repeats. They occasionally cause problems during the alignment process. A possible solution was proposed by Sammeth and Heringa in [87]. Their multiple alignment approach first identifies and temporarily removes repetitive regions from the sequences that are aligned. They are later re-inserted after the intervening regions were subjected to a standard alignment procedure. This approach is in principle not restricted to DNA sequences, it could also be applied to proteins. *DIALIGN* and its recently improved reimplementation *DIALIGN-TX* [70, 102] do also construct multiple alignments. Even though working with sequence segments, the approach is not specially suited to deal with repeats and rearrangements, because it is unable to reuse or reorder aligned segments. A great advantage of *DIALIGN* is however that low-similarity segments are not forced to align with the target sequence, they are left unaligned. The method usually produces an intermediate between a local and a global alignment, depending on the input sequences.

Protein SSEs are the next smaller unit below protein domains. They are useful in modeling intra-domain events. For example, some domains contain long insertions of several SSEs or repeats of some motif, others contain circularly permuted SSEs. Several methods with different purpose are based on real or predicted SSEs, either as sequence- or as structural fragments. SEA [112] is an approach for pairwise protein comparison at sequence level. For both proteins, SSEs are predicted with several secondary structure prediction methods and represented in two unweighted graphs. Ye *et al.* then solve a network matching problem: They search for a path in each graph/network, such that the corresponding SSEs in both paths are maximally similar. One main difference between their concept and Passta is that we represent residue-level alignments from many different proteins simultaneously in one graph and search for those that best explain the query, while Ye *et al.* use one graph for each target. Therefore, the SEA approach can not detect similarities to different database hits at the same time. *MAP* [85] is also based on predicted SSEs, its purpose is however fold recognition. *MAP* derives a "map" to represent the secondary structure elements of the input sequence. This map is used to find the most likely fold from a database of domains with known structure. An and Friesner [7] propose another

[13]`http://pfam.sanger.ac.uk/`

method for fold recognition. Their two-staged protocol also relies on predicted SSEs from several prediction servers. In the first stage, the predicted SSE pattern is compared to the known ones from the PDB. Database targets with a different secondary structure pattern are excluded, similarly to *MAP*. In the second stage, promising targets are chosen by means of a residue-based alignment.

Approaches working with SSEs are not restricted to using sequences. On the contrary, most SSE-based methods work with structural SSE data in a 3D context. One example is given by Alesker *et al.* [3] in protein comparison. Their method represents SSEs as vectors and can detect substructures made of these vectors in other proteins. *Rosetta* [83, 95] is another method in that realm. It predicts the structure of a protein sequence *de novo* by inserting short protein fragments with known structure into the growing model. Even though these fragments are often smaller than complete SSEs, secondary structure is here also important in choosing the best fragments.

Some approaches try to use the "best of both worlds", in terms of sequence and structure. These approaches often have several stages and combine their results from multiple sources of data. Some of them can be found in remote homology recognition. For example, *FUGUE* [93] is trying to detect remote homologues by sequence-structure comparison. Among other features, *FUGUE* can choose between two different alignment algorithms, where the gap penalties depend on the secondary structure class of the compared protein structure.

11 Integration of data into PasstaDB

11.1 Origin of the source data

Passta works on a set of SSEs with known secondary structure class. The secondary structure can be reliably determined if the 3D structure of the protein is known, otherwise it has to be predicted from the protein sequence. Even though secondary structure prediction is quite advanced today [45], we prefer to use proteins with known structure. This makes the Protein Data Bank (PDB) our primary source of data. However, parsing our data from the PDB is disadvantageous, because many of the old entries contain mistakes in the data format or in the annotation. It is esasier to retrieve the needed information from secondary databases (see Section 6 for more details). The *PTGL* provides SSEs with helical and strand conformation, but unfortunately no coils or loops. We extract these missing SSEs from the *PDBFinderII* database. The PDBfinderII contains the complete chain sequences and the corresponding secondary structure states, but not split into SSEs. The separation requires knowledge about secondary structure in general, and about the DSSP algorithm. By using both the PTGL and the PDBFinderII database, we get the SSEs of all three secondary structure classes and do not need to separate them ourselves.

The SCOP database was chosen as our source of structural and functional annotation. First, it allows to map the SSEs in our database to SCOP protein domains. This is an important feature if rearrangements are to be modeled. Second, we can compare the Passta evaluation results to that of the Jumping alignment algorithm.

11.2 The structure of PasstaDB

The structure of PasstaDB is shown with some additional information in Figure 28. Aside from the primary key in each table ("PK"), foreign keys and additional indices ("FK", "IX") are also included. The two shaded areas bsaically divide the database into a PDB and a SCOP part. The only table that can not be assigned to either SCOP or the PDB is the MaxScore table. It keeps the exact match score for each SSE, computed with several substitution matrices. The SSE table is the central element of the database, it links the SCOP, PDB and MaxScore tables. Some important fields of the database are explained in Table 2.

11.3 Data integration with PasstaBuild

All of the source databases are available as well-structured text files. Most of the information is also available online. The PTGL database was kindly provided as "dump" file by Patrick May. Accessing the data is easy, but its integration into PasstaDB is not straightforward at all. This section contains a short outline of the integration procedure, along with some problems and the corresponding solutions.

11.3.1 Integration

The PDB entries in each of the source files are not in the same order. Prior to the actual integration, we therefore index the begin of every PDB entry in the PDBfinderII file.

The integration of the data is chain-wise. For each protein chain in the source files, we read all SSEs from the PTGL. As pointed out in Section 6.3, these are only helices and strands. The PTGL however provides other important information as well. One example is the position of the SSEs in their original PDB chain. We use this information to find the coils in the PDBFinderII chains. We try to map the PTGL-SSEs to the PDBFinderII chain sequence, nearby the indicated position. If all SSEs can be mapped to the PDBfinderII chain sequence, the intervening regions are stored as coils. If the mapping does not work out, we exclude the whole chain from the database.

SSE irregularities: Before the SSEs are actually stored in PasstaDB, they are checked for certain irregularities: Some SSEs in the PTGL contain lower letters symbolising cysteine residues that form a disulphide bridge with other cysteines. This important information is not stored in PasstaDB because the search protocol does currently not consider it. The lower letters are replaced by uppercase `C`'s before the SSEs are stored in the database.

As already explained in Section 6.1, a PDB entry contains the protein sequence in twice. The `ATOM` records store the sequence as determined from the atom coordinates

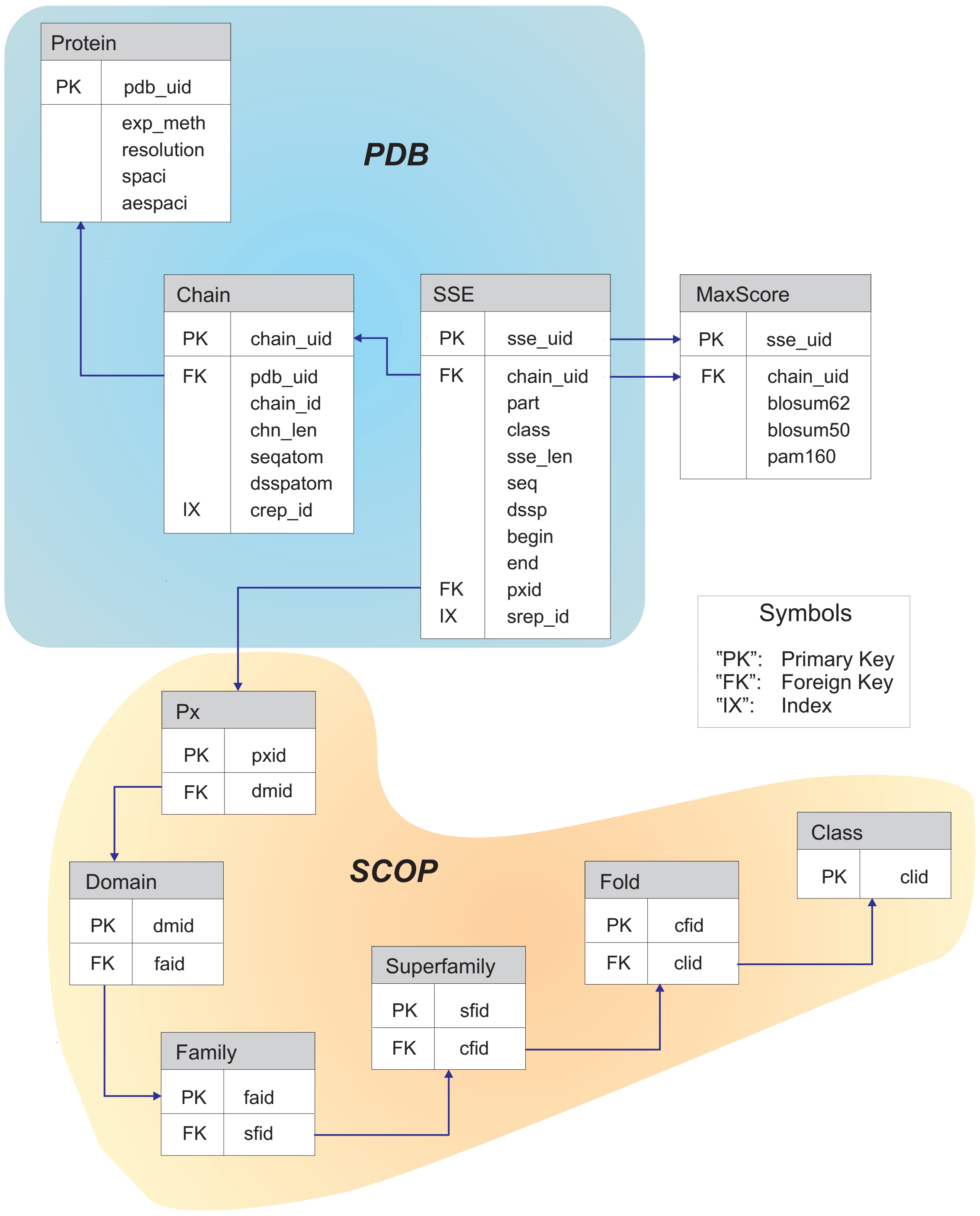


Figure 28: *Schematic view of PasstaDB. The tables* Protein, Chain, *and* SSE *contain the PDB data, parsed from the PTGL and PDBFinderII databases.*

Table	Field	Comment
Protein	“pdb_uid”	*PDB unique identifier*: Holds the unique alphanumerical four-byte ID given to each PDB entry. Its uniqueness makes it also a good primary key (PK).
	“exp_meth”	*Experimental method*: One letter code indicating the method used in structure determination. ‘X’ means crystallography, ‘N’ stands for NMR, and ‘O’ for other methods.
	“resolution”	Resolution of the protein crystal in a crystallographic experiment. Could not be used as measure of quality because this value is not available in NMR experiments.
	“spaci” and “aerospaci”	Value describing the structural quality of a PDB entry. See Section 6.4.1 for more details.
Chain	“chain_uid”	*Chain unique identifier*: Generic ID used as PK for each chain.
	“chain_id”	One byte labeling the chain *within* the PDB entry. Alphabetically ordered in multichain entries, single chain entries are often labeled ‘_’.
	“seqatom”	Protein sequence as determined from the atom coordinates of the protein structure (→ PDB “ATOM” records instead of “SEQRES”). This is necessary because otherwise the sequence of secondary structure states may not fit the protein sequence.
	“dsspatom”	Sequence of DSSP secondary structure states.
	“crep_id”	*Chain representative ID*: If the sequence of this chain is redundant and the structure quality is suboptimal, this field stores the “chain_uid” with the best spaci value.
SSE	“sse_uid”	*SSE unique identifier*: Generic ID used as PK for each SSE.
	“part”	The SSEs in each chain are numbered starting at one. Needed to check if two aligned SSEs are consecutive.
	“class”	Secondary structure class of this SSE: “C”, “H”, or “E” for coil, helix, and strand.
	“seq”	Amino acid squence of this SSE.
	“dssp”	Secondary structure state sequence of this SSE.
	“pxid”	Provides the link to the SCOP domain classification. Most SSEs are assigned to SCOP domains.
	“srep_id”	*SSE representative ID*: If the sequence of this SSE is redundant and the structure quality is not the best, this field stores the “sse_uid” with the best spaci score.
MaxScore	“blosum62”	Contains the Blosum62 exact match score for every SSE in the database.
Px (SCOP)	“dmid”	*Domain ID*: The structure of all SCOP tables is the same (except *class*). The primary key entries are unique, the foreign keys link upward in the classification hierarchy.

Table 2: *Description of selected tables and fields in PasstaDB.*

of the protein structure, whereas the SEQRES records contain the complete sequence. Both sequences may differ. This is important because the ATOM records are the input for the DSSP algorithm that defines the secondary structure. We are bound to use the ATOM sequence, because our approach depends on the exact correspondence with the secondary structure. Some sequences in PasstaDB are therefore incomplete with respect to the SEQRES records. Other problems during the experiment can cause further problems in the sequence data: If the resolution of the protein structure is too low, sometimes not all amino acids can be reliably identified. This results in positions with unknown amino acids or *chain breaks*. Unknown amino acids are marked by an 'X'. If more than twenty percent of a SSE sequence is made up of 'X's, we discard the SSE. If more than twenty percent of SSEs are discarded, the whole chain is excluded from the database. Chain breaks ('$-$') are spots with missing electron density, where the distance between successive Cα atoms indicates a gap in the chain. This gap can span more than one residue. Chain breaks can occur in proteins with multiple chains, when one chain ends close to a different one. This is no error. However, a chain break within one chain indicates insufficient data. In this case, we split the SSE at the chain break into two SSEs with the same secondary structure class.

Precomputed maximum scores: The "MaxScore" table stores the maximal score that each SSE sequence can reach in an exact alignment against itself. This score is computed under several substitution matrices. We use these values to compute a rough estimate on alignment quality ("MaxScoreRatio", see Section 12).

Integration of the SCOP classification: The SCOP classifcation is available in four text files. The filenames have the form dir.X.scop.txt_Y, where "X" is either des, cla, hie, or com. The "Y" is the SCOP version, e.g. 1.69. We parse our data from the des and the cla files. The file format is well defined and allows to reliably integrate the corresponding data into PasstaDB. The des file contains all SCOP unique identifiers[14] together with an abbreviation that shows their level in the SCOP classification hierarchy, e.g. domain (dm) or family (fa). A short domain identifier[15] and the PDB identifier are also included. By crosslinking the common information in the des and the cla file, we can rebuild the SCOP classification tree and transfer this information to PasstaDB.
The cla file also contains the range(s) that a domain spans in its chain. Connecting the SSEs in PasstaDB to the individual domains in SCOP is a prerequisite for sequence annotation in PasstaRun. We identify all SSEs that lie within the given range and assign them to the corresponding SCOP domain. If a SSE is almost in range, with a small part of it slightly exceeding the domain boundaries, it is also assigned to that domain if it exceeds the domain boundaries by up to two positions. Unfortunately, the domain ranges are sometimes inconsistent with the sequence. We tried to repair these chains by mapping the SSE sequences to the domain sequence, but the database contains a small number of cases that could not be repaired.

[14]called "sunids" by the SCOP team
[15]the "SCOP concise classification string (sccs)"

After the integration, PasstaDB contains most of the data available in the source databases. The last version contained 21571 proteins, 44047 chains, and about 1.52 million SSEs, about 90% of the available data.

11.3.2 Representative and redundant sequences

Many chain sequences in PasstaDB are not unique. Including these duplicates in the search protocol could only increase the number of covered PDB entries and the runtime of Passta, but not the annotation quality. We therefore mark *redundant* chains to exclude them from the search protocol. We compute a temporary index on the "seqatom" field of the "Chain" table and check each (distinct) sequence for duplicates. If multiple instances exist, we identify the one with the best structural quality, i.e. spaci score. This is the *representative* chain, and the value in its "crep_id" field is set to zero to indicate that no better instance exists in PasstaDB. The "chain_uid" of the representative chain is stored in the "crep_id" fields of all duplicates. This scheme allows us to search with the representative sequences only, and to retrieve the duplicates only if they are needed. Finding the representative chain from one of the duplicates is also easy. In both cases a simple SQL query is sufficient.

Beside the redundant chains we have already dealt with, many chains with high sequence similarity exist as well. They share many sequence-identical SSEs. In analogy to the procedure described above, we mark also the SSEs in representative chains as representative or redundant: We select all equal sse sequences and compare the spaci scores of their 'parent' proteins. The SSE sequence with the best spaci score is marked as being the representative, i.e., its "srep_id" is set to zero. The "srep_id" fields of the remaining SSEs are set to the "sse_uid" of the representative SSE.

12 Protein Annotation with PasstaRun

PasstaRun is the annotation part of Passta. As explained in Section 10.2 and shown in Figure 27, it has two stages, *Pass One* and *Pass Two*. Both stages rely on alignments of SSE sequences with the query. The alignments are computed with secondary-structure specific gapcost to consider the properties of the three secondary structure classes.

PasstaRun sports two different working modes, "fast" and "default". The working mode affects a set of thresholds that control the behavior of PasstaRun. In the current Pass One protocol, changing the working mode has only a small effect. This is different in Pass Two, where the working mode also affects the annotation quality.

To use a directed acyclic graph in the annotation of the query is one of the central ideas in Passta, and a simplified version of it was also used in the first implementation of Pass One (see the Appendix for a short description). The DAG is not an integral part of Pass One or Pass Two, therefore we formally describe it at the begin of this chapter. This is followed by a short discussion of similar graphs used in genome alignment.

12.1 The directed acyclic graph used in PasstaRun

12.1.1 Data structure

Let R be a query sequence of length n and A be a set of SSE alignments with R. We define a directed acyclic graph $G = (V, E)$ with V being the vertices and E being the edges. The set of vertices V represents the alignments in A and contains two additional vertices *head* and *tail*, such that $V = A \cup \{head, tail\}$. Each alignment $\alpha \in A$ is represented by a 5-tuple (b, e, s, t, p) of information. The elements b, e and s are parameters of α: s is the alignment score, b and e mark the begin and end of the alignment with respect to the query R, where $1 \leq b \leq e \leq n$. The elements t and p are associated with the aligned SSE, with t ("target") identifying the SSE and p being its position in the parent chain ("part" in PasstaDB). For a given alignment $\alpha = (b, e, s, t, p)$, we refer to the individual components of the 5-tuple as

$$\begin{aligned}
&b(\alpha) := b, \\
&e(\alpha) := e, \\
&s(\alpha) := s, \\
&t(\alpha) := t, \text{ and} \\
&p(\alpha) := p.
\end{aligned}$$

The two additional vertices *head* and *tail* represent artificial empty alignments outside the query range, i.e. $e(head) < 1$ and $b(tail) > n$:

$$\begin{aligned}
&head = (0, 0, 0, 0, 0) \\
&tail = (n + 1, n + 1, 0, 0, 0).
\end{aligned}$$

The edge definition determines the number of edges and the possible paths in the graph, and therefore has a strong effect on the search algorithm. The easiest edge definition was to connect every vertex with each other. This fully connected graph would result in a vast number of edges and hence, slow runtime. Our basic edge definition guarantees a small number of edges that connect only non-overlapping alignments: An edge exists between two vertices $u, w \in V$, $u \neq w$, if and only if

1. u and w do not overlap (and u is before w), i.e. $e(u) < b(w)$, and
2. there is no alignment v between u and w, i.e. $\nexists\, v \in V : e(u) < b(v)$ and $e(v) < b(w)$.

A *path* P in G is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that v_i and v_{i+1} are connected by an edge for all $1 \leq i < k$. Any path from *head* to *tail* corresponds to a selection of non-overlapping alignments. Without additional constraints, the *weight* of a path $P = (v_1, v_2, \ldots, v_k)$ is given by $weight(P) := \sum_{i=1}^{k} s(v_i)$.

12.1.2 The optimal path

Our goal is to find a combination of non-overlapping alignments that explain the query sequence in the best possible way. The scores of these alignments indicate the matching quality of the aligned SSE to some degree. Therefore our problem is to find a path with maximum weight in G. On any directed graph with non-negative edge weights, this *single-source shortest path* problem can be solved with the *Dijkstra algorithm* (see e.g. [25]). Our setting allows an even simpler solution, because the known alignment positions allow to topologically sort the vertices before the optimal path is computed. The linear order of the vertices is the reason why G is directed and acyclic. It is therefore unnecessary to maintain a priority queue as suggested for the Dijkstra algorithm. Our problem can easily and efficiently be solved in $\Theta(|V| + |E|)$ time [25].

12.1.3 Related work

Similar graphs and problems are known in genome alignment. Lippert *et al.* [62] present a framework to find a set of reliable anchors for genome alignment. They describe a *chaining graph* with parameters for diagonal shifts between parts of the input genomes, reversals, and translocations. Brudno *et al.* [21] build a *1-monotonic conservation map* of local alignments to allow chaining with rearrangements. Their chaining procedure uses four different gap penalties under an affine gap model, and they emphasize the need to find good values for those and other possible parameters.

12.2 Pass One

Pass One serves as a filter. Its purpose is to retain the homologous candidates, while discarding as many unrelated candidates as possible. The Pass One protocol is similar to that of classic filters in sequence analysis, except that we are dealing with multiple sequence fragments instead of complete sequences. Given a query, the targets from PasstaDB (see Section 11.2) are treated sequentially by the search protocol: every target is associated with a set of SSEs that are aligned with the query. The quality of each SSE alignment is subsequently assessed. *Valid* alignments are kept, while invalid alignments are discarded. Finally, we know the number of valid alignments and their scores for every target. Depending on this information, we decide whether to *accept* a target or not. The Pass One result is a set of accepted candidates for Pass Two. The protocol can be divided into the four steps we describe in this chapter along with some necessary definitions and clarifications.

In contrast to the first implemetation of Pass One (see Appendix), the new protocol does not require suboptimal alignments anymore. This allows us to use the Smith-Waterman algorithm [96] to compute the alignments, instead of the Waterman-Eggert algorithm [108]. We can also omit the backtracing through the edit matrix, because the current Pass One does only depend on the alignment scores. The final Pass One implementation is therefore significantly faster than the first one.

12.2.1 Pass One targets

So far, the Pass One "targets" were not clearly characterized. At first sight, PasstaDB offers "Proteins" and "Chains" as possible targets. Proteins can be easily ruled out, because they are often associations of several gene products, while queries are usually genes corresponding to the "Chain" level. Even though protein chains would therefore make good targets, protein *domains* are the better choice. The reason is the special characteristics that protein domains possess (see Section 2.2.3). Especially their reuse in different genes during evolution makes them a better target than protein chains.

12.2.2 SSE target set

Short SSEs of a few amino acids in length align well with many sequences of different origins. They are therefore useless in discriminating between related and unrelated targets. SSEs of length three or less are even unspecific in their secondary structure conformation. Many of them can be found in either of the three classes [59].

Let S be the set of SSEs associated with a target domain d in PasstaDB. Our problem is to choose a target set S' of S that we can align to the query. The set S' has to contain as few unspecific SSEs as possible. However, every SSE we remove from S reduces the amount of information available to Pass One. This is no problem for large domains with many SSEs, e.g. chain "`A`" of PDB entry "`1ukl`" (93 SSEs), but many sequences in PasstaDB are rather short. More than 3% of the (non-redundant) chains in PasstaDB have less than 50 residues. Many of them are composed of just one long SSE and a few short ones, an example is displayed in Table 3. If we removed all SSEs below length six, the SSE subset S' of this domain would contain just one SSE.

Therefore, we use a variable threshold t on SSE length to select the target SSEs for S'. We initialize t with a value t_0 and add all SSEs with equal or greater length to S'. If $|S'|$ is smaller than a predefined constant, e.g. three, t is lowered by one unit to add shorter SSEs. For example, we build S' for the domain shown in Table 3 and assume $t_0 = 6$. The set S' then contains just one SSE of length six. Hence, we lower t to 5, and add the first SSE "`NEPVS`" to S'. In the next iteration, t is lowered to 4 and the SSE "`CIRN`" is added.

However, we have to ensure that the short SSEs we add are not too unspecific. The amino acid composition of a SSE can serve as a rough measure for being specific. If it is composed of rare amino acids, it is usually more specific than a SSE composed of very common amino acids. Therefore we define a last, substitution-matrix dependent constraint w_{min} to discard SSEs that are entirely composed of common amino acids. We define a minimal *maxScore* $M = t_0 \cdot w_{min}$, where w_{min} is the minimal log-odds score for an exact match in the substitution matrix. In case of using

Part	Length	Sequence	Class	MaxScore
1	5	NEPVS	C	26
2	4	CIRN	H	24
3	2	GG	C	12
4	3	ICQ	E	18
5	17	YRCIGLRHKIGTCGSPF	C	100
6	3	KCC	E	23
7	1	K	C	5

Table 3: *PasstaDB excerpt showing the SSEs of PDB entry "1e4r", chain "A". If a minimum length of six was applied to select the target SSEs, only SSE number 5 would qualify.*

the Blosum62 matrix and $t_0 = 6$, $M = 24$. SSEs below length four are never added, because of their aforementioned secondary-structure unspecifity.

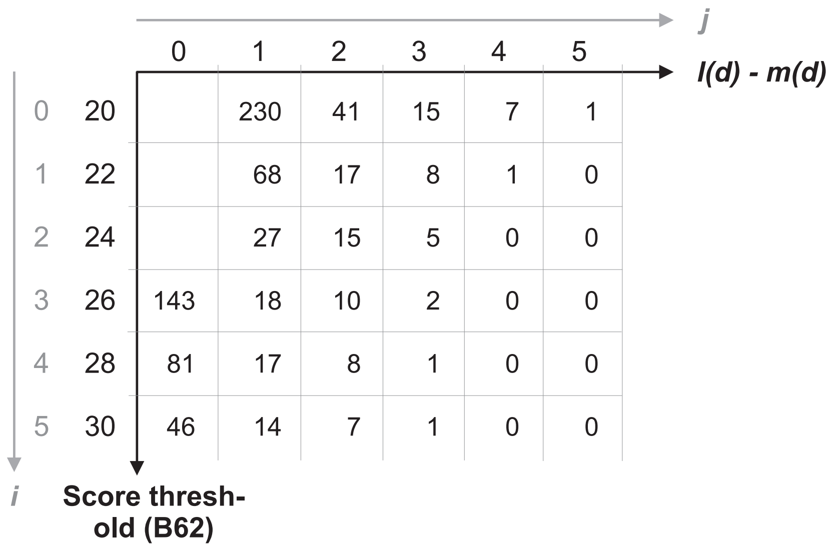
12.2.3 *Valid* alignments

Most alignments computed during Pass One are unrelated to the query. We try to discard them by keeping only *valid* alignments. An alignment is valid if it fulfills two constraints: (a) The alignment score exceeds a substitution-matrix dependent threshold. (b) The ratio *alignment score / maxScore* has to reach and exceed another threshold, the so-called *MaxScoreRatio*. The maxScore is the maximal score that an SSE sequence can reach if it is aligned to itself with a given substitution matrix.

12.2.4 Acceptance of the target domains

For every target domain d, the number of valid alignments $l(d)$ and their associated scores are known. We basically accept d as Pass Two candidate, if $l(d) \geq m(d)$, where $m(d)$ is the number of *needed alignments* for d. We define the number of needed alignments simply as $m(d) := \lceil 0.1 \cdot |S'| \rceil$, i.e., one out of ten SSEs in S' has to produce a valid alignment. For example, if 23 SSEs qualify for being aligned, i.e. $|S'| = 23$, we accept d as Pass Two candidate if at least three of the 23 SSEs lead to valid alignments. In that case, we add d to the *candidate set* C. However, the set of accepted targets is often quite large, and the running time of Pass Two depends on the number of candidates. This relationship is not linear, leading to high time demands especially in the "default" mode. Therefore, we introduce a heuristic that selects a subset of promising candidates C', $C' \subseteq C$, usually restricting their number to less than a constant value k. Our heuristic depends on two ideas: (1) Every valid alignment is increasing d's probability of being related to R. (2) The higher the alignment scores, the more likely is d being related with R.

The heuristic runs on a *candidate matrix* $Cmat$ that contains the number of accepted domains at the given parameters. More precisely, every field in $Cmat$ contains the size of a subset of C depending on the number of valid alignments at a minimal alignment score. The values along the x-axis are $l(d) - m(d)$, i.e. the number



i	Score threshold (B62) \ j: l(d) - m(d)	0	1	2	3	4	5
0	20		230	41	15	7	1
1	22		68	17	8	1	0
2	24		27	15	5	0	0
3	26	143	18	10	2	0	0
4	28	81	17	8	1	0	0
5	30	46	14	7	1	0	0

Figure 29: *Real-world example of a candidate matrix with a desired number of at most 100 candidate domains. Each field shows the number of domains with a certain number of valid alignments (x-axis) at a given minimal alignment score (y-axis). The variables i, j are displayed to ease the understanding of the algorithm that selects a combination of promising candidates from the matrix. Some of the fields at $j = 0$ are deliberately empty: The algorithm does not compute them because it will never use them (see text for more details).*

of valid alignments exceeding $m(d)$. Along the y-axis are alignment score thresholds, larger than the basic threshold explained in Section 12.2.3a). A real-world example is shown in Figure 29. The simplified pseudocode of the algorithm follows:

```
1:  $C' = \emptyset$
2:  for all $j = jmax \dots 0$ do
3:      $i \leftarrow imax$
4:      $C'_{tmp} = Merge(C', Cmat[i, j])$
5:      while $|C'_{tmp}| < maxCand$ do
6:          $C' = C'_{tmp}$
7:          $i = i - 1$
8:          $C'_{tmp} = Merge(C', Cmat(i, j))$
9:      end while
10: end for
11: $C = C'$
```

Basically we start at the rightmost column of the matrix and the bottom row, i.e. the last row that contains values larger zero. The fields are traversed row-wise from bottom to top. At each field the heuristic checks if the specified domains can be added without exceeding the desired threshold. For clarity, we have left out code of minor importance in the above algorithm. For example, if all fields contain values exceeding the threshold, the heuristic picks the smallest one. It never returns an empty candidate set, except if there are no accepted candidates at all.

Basically, one could think of many other ways to assemble the final candidate set or to decide about the acceptance of the target domains. Even though the proposed heuristic may seem somewhat arbitrary, we believe this approach to be feasible and well comprehensible.

12.3 Pass Two

The goal of Pass Two is to annotate the query with the best selection of targets from the Pass One candidate set. Multiple targets can be used for the annotation of the query, although Pass Two tries to keep their number to a minimum. The number of rearrangements and other rare evolutionary events are also minimized according to several parameters that are introduced in this chapter.

12.3.1 Computing suboptimal alignments

For each target t in the Pass One candidate set C, Pass Two retrieves *all* SSEs from PasstaDB, regardless of redundancy status or sequence length. In order to model repeats and duplications, we occasionally need more than one alignment between a SSE and the query R. Therefore we compute these alignments with our own implementation of the Waterman-Eggert [108] algorithm (described in Section 3.3) with affine gapcosts.

Unfortunately, we face the problem to align SSE sequences between length 1 and 143[16]. If we aligned R and small and unspecific SSEs with the Waterman-Eggert algorithm, the number of alignments would explode and lead to huge runtimes. Therefore, we divide the alignment phase into an *align* and an *extend* part, to treat small SSEs with a more suitable protocol.

Align: Let $l(SSE)$ be the length of an SSE sequence. If $l(SSE) = 2$, we compute all exact matches between the SSE and the query R and insert the corresponding alignments into the alignment set A. If $l(SSE) \geq 3$, we use the Waterman-Eggert algorithm to compute the optimal local alignment, possibly followed by a few co- or suboptimal alignments. We continue to compute alignments as long as they are *valid*. An alignment α between SSE and R is valid whenever its *MaxScoreRatio* ratio (see Section 12.2.3) is larger than a predefined constant.

Extend: Each time we insert an α into A in the *align* phase, we consider the adjacent SSEs as well, SSE_L to the left and SSE_R to the right. If SSE_L exists and $l(SSE_L)$ is less or equal to four, we try to match it left of α without allowing insertions or deletions. The position of the new alignment is $b(\alpha_L) = b(\alpha) - l(SSE_L)$ and $e(\alpha_L) = b(\alpha) - 1$. If the score $s(\alpha_L)$ is positive, we include it into A. The procedure for the right neighbor SSE_R is analogous: We try to insert a new alignment α_R at $b(\alpha_R) = e(\alpha) + 1$ and $e(\alpha_R) = e(\alpha) + l(SSE_R)$.
We can however not be sure that α is a global alignment. If it is local, α_L and α_R will rarely fit beside α. In that case we also try to insert alignments of SSE_L and SSE_R beside the borders of the global alignment, as if it existed. Figure 30 shows a real-world example.

12.3.2 Two different edge definitions

The main difference between the "Fast" and the "Default" modes in Pass Two is the edge definition used in the directed acyclic graph $G = (V, E)$. The edge definition in the fast mode is equal to the one introduced in Section 12.1. For convenience, we repeat it here. An edge exists between two vertices $u, w \in V$, $u \neq w$, if and only if:

1. u and w do not overlap (and u is before w), i.e. $e(u) < b(w)$, and
2. there is no alignment v between u and w, i.e. $\nexists v \in V : e(u) < b(v)$ and $e(v) < b(w)$.

This definition reduces the number of edges to a minimum, leading to a decrease in the Pass Two runtimes. While actually every vertex is reachable in a traversal starting at the *head* vertex, there is one unwanted side effect. It can be best explained with query sequences that are not well covered by PasstaDB, i.e. where only a few SSEs are similar to the query. Some areas in G will represent only a few valid alignments then. In that case, it is possible that the optimal path contains artifact vertices that had to be selected because of missing alternative connections.

[16]a coil in PDB entry "1K28", chain "A".

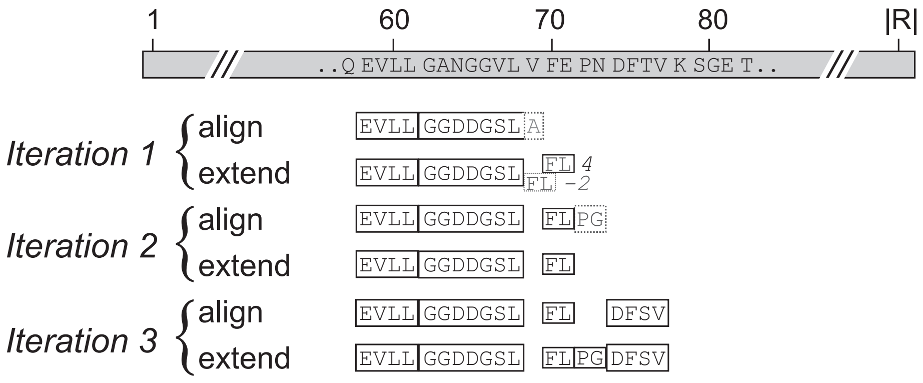


Figure 30: *Real-world example illustrating the align & extend strategy in Pass Two. Aligned SSEs with full height are from an 'align' phase, aligned SSEs with half height are from an 'extend' phase.*
Iteration 1, align: A local alignment of the SSE sequence ***`'GGDDGSLA'`*** *and the query is computed, yielding (****`'GGDDGSL'`****,* ***`'GANGGVL'`****).*
Iteration 1, extend: The SSE left of ***`'GGDDGSLA'`*** *is already aligned, so no extension is performed here. However, the right SSE qualifies for extending, and we match it (a) at the end position of the* ***local*** *alignment and (b) at the position where the* ***global*** *alignment would have ended. Only the latter is accepted here because its score exceeds zero.*
Iteration 2, align: 'PG' is of length two, and there is no exact match with the query, so it is not aligned.
Iteration 2, extend: No valid alignment was produced in the alignment phase, so nothing is to be extended.
Iteration 3, align: The SSE sequence ***`'DFSV'`*** *is aligned to the query.*
Iteration 3, extend: Now it is possible to extend with ***`'PG'`*** *to the left of* ***`'DFSV'`****.*

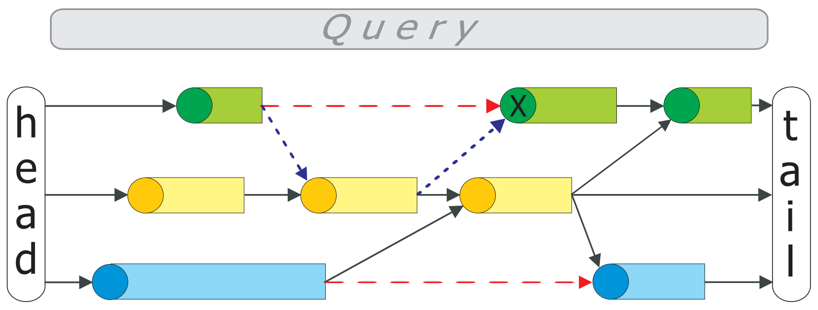


Figure 31: *Example graph illustrating the two edge definitions used in Pass Two. The solid black edges and the dotted blue edges exist in both working modes. In the "Fast" mode of PasstaRun, the two dotted edges are in the optimal path and thus have to be used if the algorithm wants to pick up the green alignment marked with an "X". In that case, it does also pick up the yellow alignment in the center, even though this causes two jumps and is penalized accordingly. The red dashed edges are added in the "Default" mode of PasstaRun and allow to pick up the marked alignment without additional jumps.*

The reason for this problem is that the first edge definition does not consider the origin of the aligned SSEs. Therefore we have modified the edge definition in the default mode. Here, an edge also exists between two vertices $u, w \in V$, if and only if:

1. $c(u) = c(w)$ i.e., both SSEs come from the same chain,
2. $e(u) < b(w)$ (the no-overlap condition), and
3. there is no other alignment v between u and w, where the SSE in v comes from the same chain as the one in u and w: $\nexists v \in V : c(u) = c(v) = c(w)$ and $e(u) < b(v)$ and $e(v) < b(w)$.

An example graph illustrating the effect of the two edge definitions and the problem with the first one is shown in Figure 31.

12.3.3 The optimal path in Pass Two

After the alignment set A is complete, the directed acyclic graph G is built as described in Section 12.1. We compute the optimal path in G, starting with the first vertex *head*. To obtain biologically feasible results, we have added certain constraints to the search algorithm:
A path $P = (v_1, v_2, \dots, v_k)$ in Pass Two can contain *jumps*, *rearrangements*, *deletions*, and *repeats*. Each such condition corresponds to a rare biological event and is penalized accordingly. If two aligned SSEs in P originate from different targets, i.e. $t(v_i) \neq t(v_{i+1})$, we call this a *jump*. Jumps are in spirit similar to the jumps described in the "Jumping Alignments" part of this thesis. If the two aligned SSEs are from

the same target, i.e. $t(v_i) = t(v_{i+1})$, there is no jump. Then they are checked for deletions, repeats, and rearrangements. Those events can be detected if the aligned SSEs are not consecutive, i.e. $p(v_{i+1}) - p(v_i) \neq 1$.
If one or more SSEs are "missing" between the aligned SSEs, that is $p(v_{i+1}) - p(v_i) > 1$, we speak of a *deletion*. If the position of the second aligned SSE is before the first one, i.e. $p(v_{i+1}) - p(v_i) < 0$, we speak of a *rearrangement*. A special case is $p(v_{i+1}) - p(v_i) = 0$, corresponding to a reuse of the same SSE. This event is called a *repeat*. For reasons of simplicity, repeats are currently treated like deletions.

Let $j(P)$ be the number of jumps in a path P, $r(P)$ the number of rearrangements, and $d(P)$ the number of deletions and repeats. We penalize those events by three parameters, *jump cost* (jc), *rearrangement cost* (rc), and *deletion cost* (dc). The weight of a path P is then given by

$$weight(P) \quad = \quad \sum_{i=1}^{k} s(v_i) \; - \; j(P) \times jc \; - \; r(P) \times rc \; - \; d(P) \times dc.$$

If these penalties are chosen well, the annotation of the query with small chance alignments from different chains is unlikely. Finally, the alignments in the optimal path are visualized in a HTML page.

13 Projects

In this chapter, we describe smaller projects associated with Passta. The projects are in chronological order. The two larger ones have their own results and discussion sections, preceded by a section describing preliminary work.

13.1 Optional secondary structure profile

Knowing the secondary structure of a protein is important information that can be helpful in answering a variety of questions in sequence analysis. For example, the presence of about seven α-helices is characteristic for transmembrane proteins []. Secondary structure information can also be the input for methods in fold recognition, a good example is MAP [85].

During Pass One, we usually compute several hundreds and thousands of alignments. The secondary structure class of each aligned SSE is known. This valuable information can be used to generate a simple secondary structure profile (SSP) of the query sequence. For each position in the query, it shows the percentage of valid alignments where this position is in coil-, helix-, and strand conformation. An example is shown in Figure 32.

Implementation: Most of the terms used here were introduced in Section (12.1), where we described the directed acyclic graph. The conditions which qualify an alignment as *valid* are reported in Section 12.2.3. Basically we maintain a two-dimensional array AQ of size $4 \times n$, four fields per query position. All fields in AQ are initialized with zero. For each alignment α in the alignment set A, we increment all fields from $AQ_{c,b(\alpha)}$ to $AQ_{c,e(\alpha)-1}$, $0 \leq c \leq 2$ by one, where c depends on the secondary structure class of α, i.e. coil, helix, or strand. This corresponds to counting the number of alignments with a specific secondary structure conformation covering a given position of the query. Finally we compute the sum of all alignments per query position, i.e. $AQ_{3,k} = \sum_{p=0}^{2} AQ_{p,k}$ for all k, $1 \leq k \leq n$.

In contrast to more sophisticated approaches in secondary structure prediction (a comprehensive overview is given in [18], Chapter 28), the prediction accuracy of this simple profile will usually be lower. The extra information can nevertheless be useful in understanding a Passta result, and to decide whether to believe it or not. Computing the profile is increasing the Pass One runtime somewhat, because knowing the alignment position requires to do the backtracing, which is otherwise omitted. The user can decide about computing the profile by using the `-n` option of Passta. If set, the profile is not computed.

13.2 Pass One Training / Parameter settings

Many kinds of filters were designed for sequence database searching. A filter working with multiple sequence fragments affords even more design decisions than a filter using a single sequence, because each fragment can be seen as a single sequence.

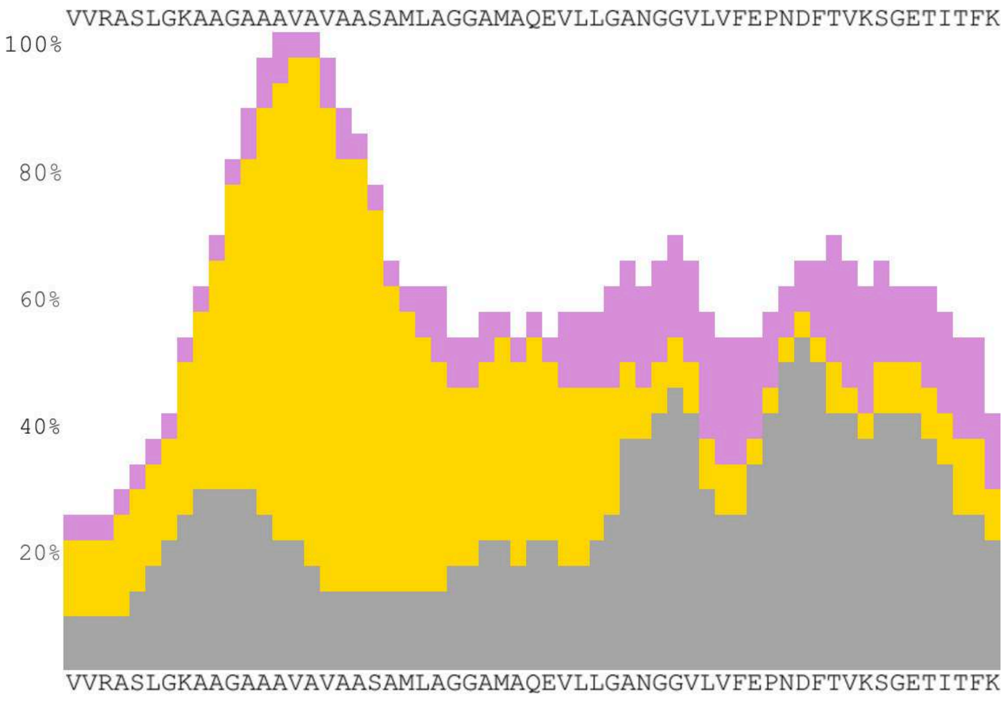


Figure 32: *Excerpt of a secondary structure profile from Pass One. The query sequence is displayed at the top and the bottom of the profile, it is the same plastocyanin from rice that is shown in other examples of this thesis.*
The grey rectangles that abound at the bottom of the profile show the percentage of valid alignments with coil SSEs covering the respective column. Helical SSEs are represented in the middle area by orange rectangles, and the purple ones at the top code for β-strands. The highest number of alignments is set to 100%.
The amino acids around this peak are quite common and can often be found in helical conformation, as the profile also suggests. While helical SSEs prevail in the left half of the sequence, coils and strands seem to be increasingly likely in the right half of it. The first part of this "prediction" can not be verified, because this part of the gene is missing in other plastocyanins with known structure (this one from rice is not in the PDB). The second part definitely shows the correct tendency, because this part of the gene is classified as "mainly beta" (CATH) and "all beta" (SCOP) in other closely related genes.

Even worse, our sequence fragments originate from three different secondary structure classes with different properties. The whole setup has several parameters, too many for their simultaneous optimization in a systematic approach.

Therefore, we made several experiments to answer some design questions and to optimize the Pass One parameters. Each of the Pass One parameters was trained with only a few samples of its parameter space. Even though our experiments are not comprehensive, we present the results in detail, because few data with respect to alignments of secondary structure elements (SSEs) is available in the literature.

13.2.1 Choice of the Blosum62 threshold

Every SSE alignment is required to reach a minimal, substitution-matrix dependent score to be valid. Since we use the Blosum62 substitution matrix, the minimal score is called *Blosum62 threshold*. If it is too high, only alignments with well conserved SSEs will be valid. If it is too low, there will be too many "Unrelated" alignments, compromising the quality of the filter as well as its speed. Our goal is to find a good trade-off between sensitivity and speed.

Preliminary tests had shown that a Blosum62 alignment threshold of less than 18 resulted in too many "Unrelated" alignments. Therefore, we measured the percentage of valid SSE alignments at Blosum62 thresholds of 18, 20, and 22. The cost for gap initiation were 11, those for gap extension were 1, for all three secondary structure classes. As in the evaluation described in the next part of this thesis, we separated between "Close", "Distant", and "Unrelated" homology relationships. These relationships are illustrated in Figure 40 on page 86. The results are shown in Figure 33. It shows the percentage of valid alignments for the three homology relationships and each secondary structure class.

At a Blosum62 threshold of 18, the average sensitivity is good, almost 40% in "Close" to about 25% in "Distant". Unfortunately, the percentage of "Unrelated" valid alignments reaches 20%. Therefore, many "Unrelated" targets are likely to be accepted at threshold 18. At threshold 22, only seven percent of the "Distant" alignments are valid (except in the helix class), hardly more than the five percent "Unrelated" alignments. In combination with the low sensitivity, this weak distinction between "Distant" and "Unrelated" is a problem. Either we sharpen this distinction by improving the Pass One parameters, or we use a lower Blosum62 threshold.

Comparing the results at the different thresholds shows an unwanted effect: A decrease by two units doubles the percentage of "Unrelated" alignments, while this factor is smaller in "Close" and "Distant". This means that in this range of Blosum62 thresholds, a larger threshold worsens the distinction between "Distant" and "Unrelated". Therefore, we can not use a threshold exceeding 22. However, the threshold should exceed 18, otherwise the sensitivity of Pass One will be too low. We decided to use a threshold of 22 in the "Fast" mode of Pass One, because the relatively small number of alignments will improve its runtime. In the "Default" mode, a value of 20 should be a good choice.

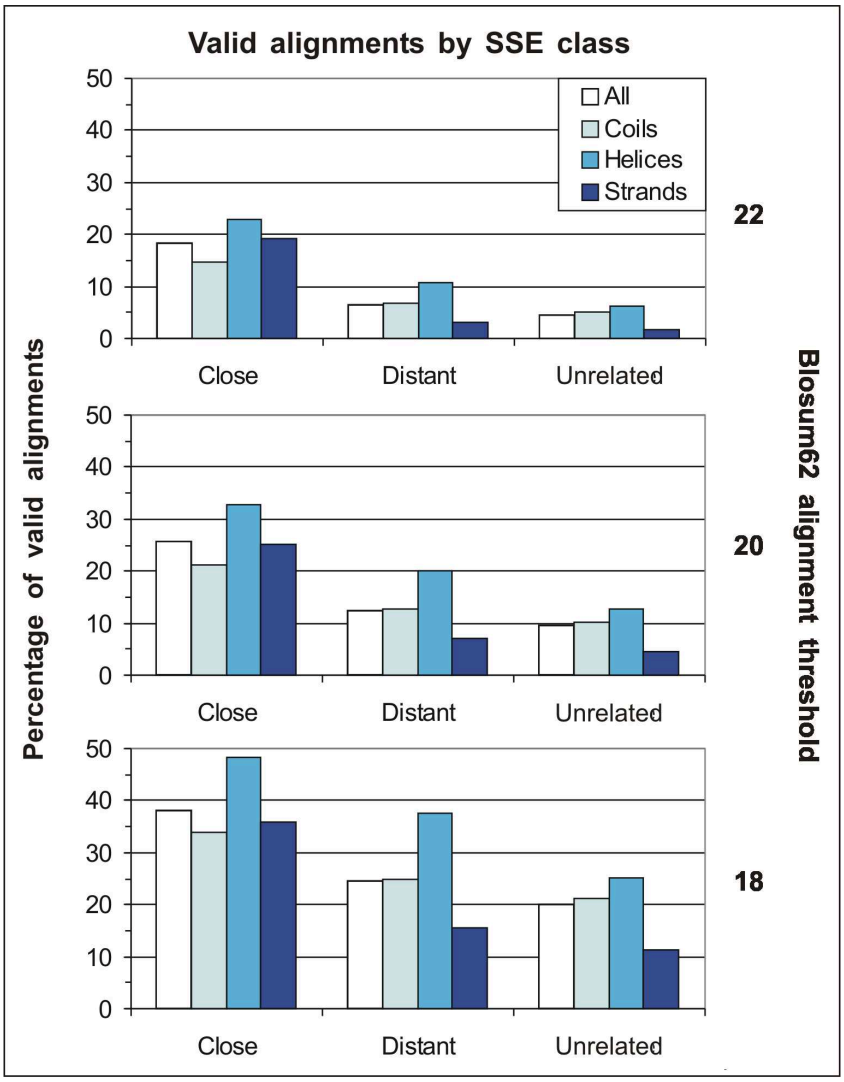


Figure 33: Percentage of valid alignments at Blosum62 thresholds of 18 (top), 20 (center), and 22 (bottom). The results are grouped by their experimental relationship ("Close'", "Distant" and "Unrelated", from left to right). The first colum ("All") summarizes the results of all three secondary structure classes. The next three columns display their data separately in ("Coils", "Helices", and "Strands"). The cost for gap initiation/extension were 11 and 1.

The bias in the results of the three secondary structure classes is interesting, not only with respect to the training of Pass One, but also in conjunction with sequence analysis on proteins. The Blosum62 substitution matrix could to some degree be responsible for this bias, but we believe that the biological properties of the different SSE classes are important as well: Helical SSEs always make up the largest proportion of valid alignments, regardless of the experimental group in question. The reason is that helical SSEs have to satisfy certain biochemical constraints, otherwise they are not stable. Some amino acids are rarely seen in α-helices, because they have a destabilizing effect. In addition, helices are often located at the protein surface, interacting with the water phase (see e.g. [18]). These physicochemical constraints result in a reduced "helical" sequence space that increases the probability for better alignment scores.
Coils are underrepresented in "Close", their occurrence in the other two groups is in balance. This might be explained by the many coils/loops that serve as "linkers" between protein domains, which are in general not well conserved. The mutation frequency of these linkers is probably higher than in the other two SSE classes, which would explain their lower frequency in "Close". This does however not explain their balance in "Distant" and "Unrelated".
Strand SSEs are underrepresented in "Distant" and "Unrelated", while in balance at "Close". We believe that this can be explained with the usual association of several β-strands to a β-sheet. This interdependency will often require the co-evolution of two or more β-strands. This is a rare evolutionary event, but with the power to introduce effective changes into the strand sequences.

13.2.2 SSE-alignments without a minimum length constraint

For each target SSE, Pass One computes a local alignment with the query. Many of these alignments are very short, and the percentage of "Unrelated" alignments among them may be high. We could discard all alignments that are shorter than a predefined minimum length. If and how would Pass One benefit from this constraint? To answer this question, we have measured the length distribution of valid alignments up to a length of 100. The results are shown in Figure 34. It displays the percentage of valid alignments at Blosum62 thresholds of 18, 20, and 22 for up to a length of 30.

There are zero valid alignments at length one, which is easy to explain with the scores in the Blosum62 substitution matrix. The highest entry in the matrix is a score of 11 for a `W-W` match. However, this score is below all of our tested Blosum62 thresholds; 18, 20, and 22. Therefore, every valid alignment has at least a length of two. Figure 34 shows that the percentage of short alignments is increasing with lower Blosum62 thresholds. This is because these thresholds mainly affect the validity of alignments between length 3 and 7. The majority of longer alignments remains unaffected.

The distribution shows that defining a minimum alignment length is not easy. If it was set to a small value like two, less than one percent of alignments would be dis-

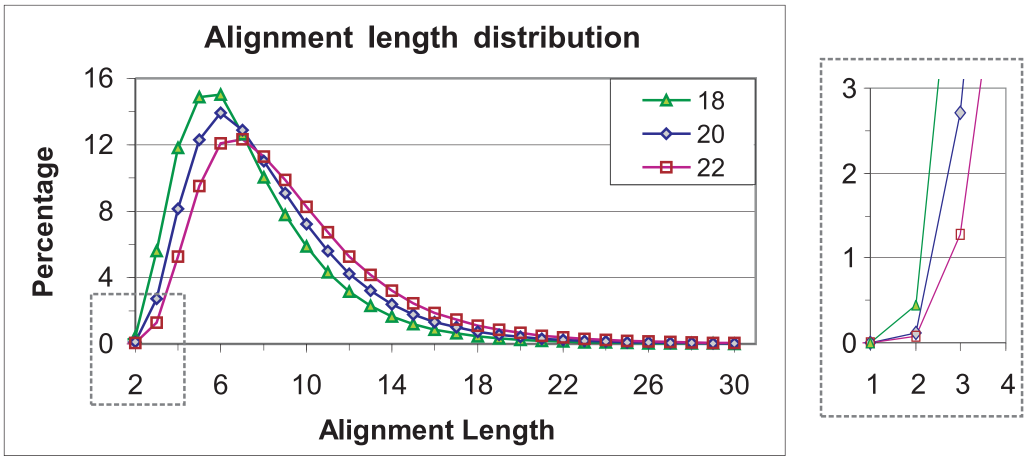


Figure 34: *Alignment length distribution at Blosum62 thresholds of 18, 20, and 22. For clarity, only alignment lengths of up to 30 are shown. The dashed area to the right is a magnified cut-out of the diagram to the left. The displayed data is dicussed in the text. The cost for gap initiation and gap extension were 11 and 1.*

carded, with almost no effect on the Pass One results. Then the constraint will have no effect. At a moderate minimum length of four or five, Pass One would however lose up to a third out of all alignments, depending on the applied Blosum62 threshold. This would mean a substantial loss of sensitivity, something we do not want. Besides, determining the alignment length results in additional computational effort, because the alignment has to be backtraced in the Smith-Waterman edit matrix. This increase in runtime will not necessarily result in improved detection quality, because we either discard too few or too many alignments. To use a probabilistic approach would probably be the better choice. Therefore, we do not employ a length constraint in Pass One.

13.2.3 Secondary-structure specific gapcost

The Blosum62 substitution matrix has been widely used. Several projects investigated the optimal affine gapcost to use with it. Green and Brenner [39] recently found a small range of good values for gap initiation and extension: If the gap extension cost is 1, gap initiation should range between 9 and 13, if the gap extension cost is 2, the gap initiation costs should be 6 to 11. These values were determined in traditional pairwise alignment of complete protein chains or domains. Here, we are dealing with *local* alignments of short sequence fragments, SSEs. *Long* gaps within a SSE alignment are unlikely. Therefore, the cost for gap *extension* should exceed one. We set it to two for all secondary structure classes. This value is probably not optimal, but it is definitely better than 1. We also believe that the gap *initiation* penalties should be class-specific, because of the different biological characteristics

SSE class	Gapcost		Increase in % valid alignments		
	From	To	Close	Distant	Unrelated
Coil		9/2	0,67	0,99	0,84
Helix	11/1	10/2	0,73	1,20	0,46
Strand		8/2	1,64	0,61	0,55
All			1,01	0,86	0,62

Table 4: *Increase in the percentage of valid alignments by changing the affine gapcost in Pass One to a secondary structure specific model. The changes result in a slight advantage for Passta, because the increase in "Unrelated" alignments is in general lower than in the two related groups.*

the secondary structure classes possess. For example, the structure of helical SSEs is very well conserved, gaps should be a less frequent event than in coils. This suggests to increase the cost for gap initiation in alignments with helix SSEs. On the other hand, *decreased* penalties for gap initiation could improve the low sensitivity observed in Section 13.2.1 at Blosum62 thresholds exceeding 20. Besides, secondary-structure specific gapcost may contribute to reducing the observed bias between the secondary structure classes.

After a few tests, we decided to use secondary-structure specific gapcost penalties of 9/2 (coils), 10/2 (helices), and 8/2 (strands). The positive effect of the low gap initiation cost in strands can be seen in Table 4. Here, the percentage of valid alignments is compared between the former gapcost of 11/1 and the new values. The reduced initiation cost for strand SSEs leads to an increase of 1.64% of "Close" alignments, while the percentage of "Unrelated" alignments is only increased by 0.55%. The new gapcost also lead to positive effects in the helix class, while the effects on the coils are rather neutral. Because the coil sensitivity in "Distant" is slightly better, we will use the new gapcost in Pass One.

Another question we wanted to assess is the effect of the changed gapcost on the number of alignments with gaps. Table 5 lists the percentage of alignments with at least one gap at two different gapcost combinations and Blosum62 thresholds of 18, 20, and 22. A comparison among the 11/1 values shows similar results, with fewer gaps at smaller Blosum62 thresholds. This can be explained by the increase in short, mostly gapless alignments at smaller thresholds (see Section 13.2.2). Their increase results in a smaller *relative* percentage of alignments with gaps. However, being around five percent, the values show that the effect of the old gapcost is of minor importance.

The picture is different with the new gapcost of 9/2, 10/2, and 8/2. Here, the percentage of "Close" alignments with gaps has increased to almost five percent, while these values are at about eight percent for "Distant" and "Unrelated" alignments. This may seem much, but the MSR constraint discussed in Section 13.2.4 is discarding many of these alignments, leading to the decreased values of about three percent in the bottom row of Table 5.

Gapcost			Blosum62	**Gapped alignments in %**			
Coil	Helix	Strand	threshold	Close	Distant	Unrelated	MSR
11/1 all			18	2,67	3,05	3,32	-
			20	3,30	4,46	4,78	-
			22	3,54	5,66	6,20	-
9/2	10/2	8/2	20	4,73	7,42	8,09	-
9/2	10/2	8/2	20	2,50	3,81	3,92	0.30

Table 5: *Percentage of alignments with at least one gap. The modified gapcost lead to an increased number of gaps, if no futher constraints are applied (4th row).*

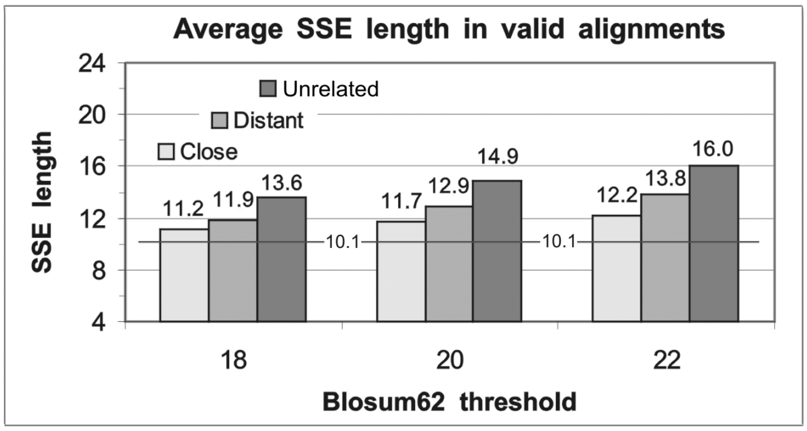


Figure 35: *Average SSE lengths in valid alignments at Blosum62 thresholds of 18, 20, and 22. The costs for gap initiation and gap extension were 11 and 1.*

13.2.4 Choice of the MaxScoreRatio threshold

SSEs are very short in comparison with whole protein chains. Nevertheless, they are also sequences, where all of the rules and problems associated with database searching also apply. The longer the SSE sequence, the more likely is a local similarity to the query, even if the query and the SSE are not related. The distribution of the alignment lengths in Section 13.2.2 in combination with the data in Figure 35 clearly show this effect. Figure 35 displays the length of the average input SSE[17] in *valid* alignments. It is especially high in “Unrelated” SSEs, up to 16, while the length of the average input SSE is 10.1. This means that even in the “Close” setting, longer SSEs are more likely to produce valid alignments. The average alignment length of about 7 (see Figure 34) further implies that up to about one third of the average input SSE is lost. Therefore, the alignment score is often well below the possible maximum, because only a fraction of the SSE sequence is actually used. This is especially true for “Unrelated” alignments, and we try to remove them by applying the

[17]the SSE sequence before the local alignment

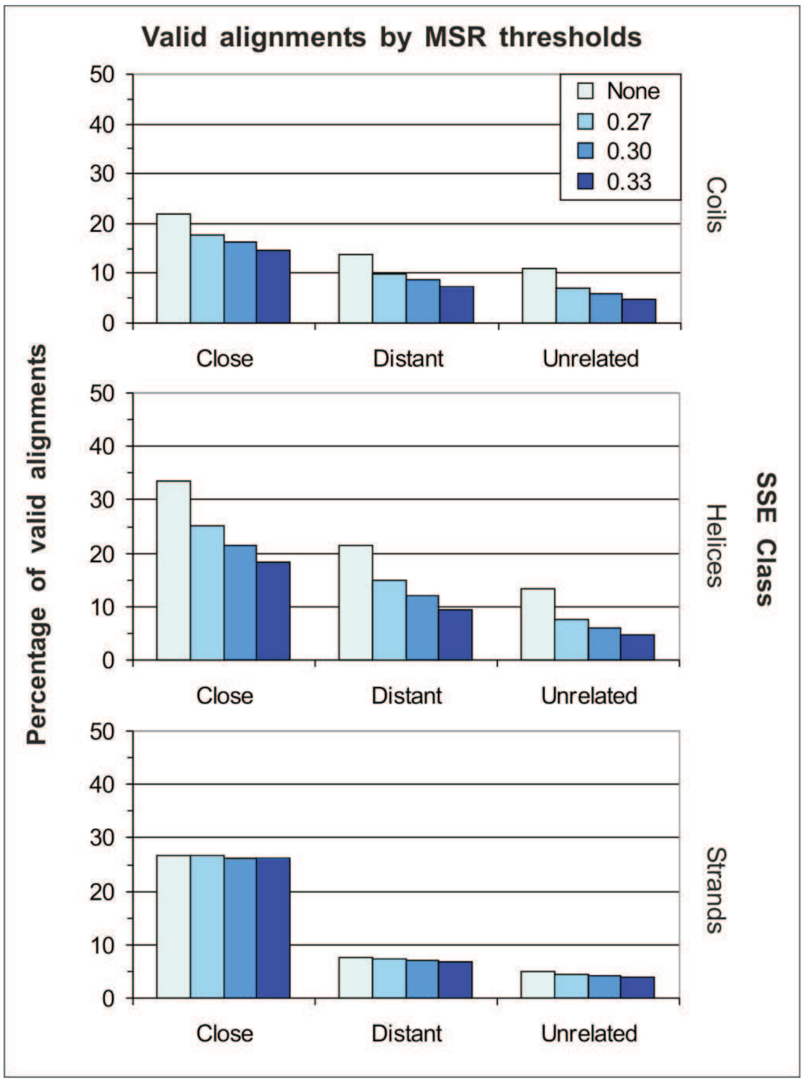


Figure 36: *Percentage of valid alignments at four different MSR thresolds: None, 0.27, 0.30, and 0.33. The results are grouped by SSE class (top: coils; center: helices; bottom: strands) and experimental relationship ("Close", "Distant", and "Unrelated", from left to right). The Blosum62 threshold was 20. Alignments with helical SSEs are much affected by the MSR thresholds, while the percentage of strand alignments remains almost constant.*

MaxScoreRatio constraint (explained in Section 12.2.3).

We investigated a small range of MSR values to select a good threshold for Pass One. Figure 36 shows the percentage of valid SSE alignments at a Blosum62 threshold of 20 without a MSR, and at MSR values of 0.27, 0.30, and 0.33. The costs for gap initiation and extension were the secondary structure specific values chosen in the preceding Section 13.2.3. The obtained results are considerably different in the

SSE-Class	Hom. relationship	% alignments at MSR-Thresh.				
		0.27		**0.30**		**0.33**
	Close	80.7	7.2	73.5	7.6	65.9
Coil	Distant	72.3	9.2	63.1	8.7	54.5
	Unrelated	63.2	10.8	52.4	7.8	44.6
	Close	74.7	10.8	63.9	9.6	54.3
Helix	Distant	69.4	13.6	55.8	11.3	44.5
	Unrelated	57.4	12.4	44.9	10.1	34.9
	Close	99,7	1.7	98.1	0.7	97.4
Strand	Distant	97.0	3.0	94.0	2.6	91.4
	Unrelated	92.3	7.9	84.4	6.4	78.1

Table 6: *The table shows the percentage of alignments that remain valid at MSR-thresholds of 0.27, 0.30, and 0.33. A value of 100% corresponds to the percentage of alignments without the MSR constraint. The columns in between display the difference between the adjacent values. A comparison of the results reveals remarkable differences between the secondary structure classes coil, helix, and strand. It is obvious that there is no common, ideal threshold.*

three SSE classes. The number of strand alignments is hardly affected by the MSR thresholds, the opposite is true for helix alignments. The coil results are somewhere in between. Table 6 allows a more precise overview by listing the MSR-dependent percentages of the remaining valid alignments. A value of 100% corresponds to a setup without the MSR constraint. For convenience, the difference between adjacent columns is also displayed.

Among the different SSE classes, the MSR value for strands can be set near 0.33 or even larger, because only a few percent in "Close" and "Distant" are lost in this range, while about 20% in "Unrelated". However, the MSR value for helices should be less or equal to 0.30, otherwise the separation between "Distant" and "Unrelated" is getting worse. The same holds for coils. A perfect MSR threshold for all secondary structure classes is not available. We choose a value of 0.3, which is a good compromise for all three classes.

13.2.5 Conclusion

We have investigated and set most of the Pass One parameters. The obtained values are probably not the optimal ones that would result from a simultaneous optimization of all parameters. However, even a simultaneous optimization could just lead to a compromise. Our experiments have shown that the characteristics between the individual SSE classes can not be covered with single Blosum62- or MSR thresholds. For example, the sampled MSR thresholds have a strong effect on the number of valid helix alignments, but not on the number of strand alignments.

Even though there will still be many “Unrelated” entries within the Pass One candidate set, we could sharpen the distinction between “Distant” and “Unrelated” by a few percent of valid alignments.

13.3 Calibration of Rearrangement and Deletion costs

13.3.1 Motivation

Most methods for biological sequence alignment produce a linear layout where similar regions of the input sequences are identified with each other. Several dynamic programming algorithms have been developed to address the various flavors of this approach. The main application is database searching, with the goal to identify homologous protein or DNA sequences of a given query.

The quality of a sequence alignment, and hence also the result of a database search, critically depends on the choice of parameters used in the alignment procedure. For traditional collinear, and especially pairwise, local alignments, this relationship and the underlying statistics have been studied extensively and are well understood (see e.g. [4, 41, 106]). The parameters of interest are the score matrix for the substitution of single characters and, for most popular alignment methods, the affine gap cost parameters *gapinit* for opening and *gapext* for extending a gap. Values for both parameters have to be calibrated for each substitution matrix separately, although there is a range in which good values overlap for similar matrices [39]. Calibration was done for the most popular substitution matrices under different test environments [4, 39, 43].

In some more recent applications, however, a collinear alignment of the given sequences is not possible because the relationships between the sequences are more complex. As in Passta, these applications often follow a two-phase strategy. In the first phase a set of local candidate alignments is computed, and in the second phase a collection of good candidates is selected and arranged according to some global optimization criterion. Some approaches in genome alignment are examples of this strategy. We have already discussed them in Section 10.2.

These approaches occasionally introduce new parameters in conjunction with their chaining model. The calibration of the new parameters requires a reliable evaluation setting, usually in a repeated database search context. A testset is split off the database, and the parameters are optimized such that the method recognizes as many true and as few false hits as possible. This requires an annotation of the data that allows to distinguish between ‘true’ and ‘false’ hits. When dealing with protein sequences, often protein structure information is used to form such a “gold standard”. The most prominent databases for such an approach are SCOP [9, 73], which has been used in different evaluation and calibration scenarios [49, 97], and CATH [76].

Here, we present a variation of the existing approach that can be used to calibrate some of the new parameters arising in generalized chaining methods. Passta [14] can be seen as a simple version of such an algorithm. The local alignments it computes are passed to a generalized chaining algorithm that allows for the deletion and rearrangement of segments. The corresponding parameters are *deletion cost* (DC) and *rearrangement cost* (RC). Under the assumption that most protein domains in the same SCOP family are collinear, we measure the minimal rearrangement cost needed to suppress rearrangement events, which we term *collinearity threshold*. Then, we identify SCOP families with high collinearity thresholds and try to find the cause by manually investigating them. Finally, we collect Passta results with validated circular permutations and compare their collinearity thresholds and alignment scores. In conclusion, we discuss the choice of suitable rearrangement costs.

To the best of our knowledge, absolute RC values were never determined before. Biologically justified deletion and rearrangement costs could be useful to estimate similar values for DNA-based approaches, and for comparison of protein structures, when sequence information is also used.

13.3.2 Model and Definitions

In this section we describe the simple model of a generalized alignment procedure that we use to explain and calibrate the new parameters DC and RC. The model is to some degree similar to the one given in [62]. However, we do not consider sequence reversals and assign no distance-dependent penalty for two adjacent matches separated by a gap.

Our procedure is asymmetric with respect to the two given sequences. We consider one sequence as a user-provided *query* and the other sequence as *target* from a given database. This allows us to preprocess the target sequence. Because our application is protein sequence annotation, we preprocess the target by splitting it into secondary structure elements (SSEs), but in theory there are also other possibilities like using the gene or repeat annotation of a genomic sequence.

Our model is derived mainly from Pass Two, the necessary definitions are provided in Sections 12.1 and 12.3. For convenience, we briefly repeat the most important definitions here and highlight necessary changes. The main difference to Pass Two is that we use this model for *one target* only, therefore jumps between different targets are impossible and no jumpcost penalty exists.

In the first phase of the general alignment procedure, each fragment of the target is locally aligned to the query. The particular method is of secondary importance, while multiple and suboptimal hits should be allowed. The result of this first phase is a set A of pairwise alignments with the query, where we assume that the following information is available for each alignment $\alpha \in A$: $b(\alpha)$ and $e(\alpha)$, $1 \leq b(\alpha) \leq e(\alpha) \leq n$, are the begin and end indices of an aligned sequence fragment w.r.t. the query sequence of length n; $p(\alpha) \geq 1$ is the position of the fragment in a consecutive enumeration of the fragments of the target sequence; and $s(\alpha)$ is the alignment score.

The goal of the second phase is to find the best combination of non-overlapping alignments that explain the query. Therefore we define a directed, acyclic *generalized* *alignment graph* $G = (V, E)$ whose set of vertices V consists of the alignments in A plus two additional vertices $head$ and $tail$, such that $V = A \cup \{head, tail\}$. Two vertices $u, v \in V$ are connected by a directed edge $(u, v) \in E$ if and only if $e(u) < b(v)$, that is, the alignment represented by u ends before the alignment represented by v begins.

A *path* in G is a sequence of vertices $P = (v_1, v_2, \ldots, v_k)$, such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$. A path is *complete* if it starts at vertex $head$ and ends at vertex $tail$. The aligned target fragments in a path are not necessarily numbered consecutively w.r.t. their order in the target sequence. Given two adjacent vertices v_i and v_{i+1} in a path:

- if $p(v_{i+1}) - p(v_i) < 0$, we call this a *rearrangement*;
- if $p(v_{i+1}) - p(v_i) > 1$, we call this a *deletion*.

To control the number of rearrangements and deletions in a path, these conditions are penalized by *rearrangement cost* (RC) and *deletion cost* (DC). Let $r(P)$ be the number of rearrangements and $d(P)$ the number of deletions in a path P. The weight of P is then given by

$$weight(P) = \left(\sum_{i=1}^{k} s(v_i)\right) - r(P) \cdot RC - d(P) \cdot DC.$$

Any complete path corresponds to a selection of non-overlapping alignments. The *generalized chaining problem* is to find a highest-scoring complete path in a given generalized alignment graph G.

13.3.3 Experimental approach and setup

While the generalized alignment approach can be and has been applied in several areas of biological sequence comparison, our experiments are applied in protein sequence annotation. A multi-domain protein can easily lose its function if a rearrangement event disrupts the structure of its domains. However, within a correct alignment of two closely related domains, one would expect no rearrangements. The protein domains within a SCOP [9, 73] family are closely related, and they have the same overall 3D-structure (see Section 6.4). Most of them are also collinear at the sequence level. Following our model definition, we expect most of the target fragments to be aligned consecutively. If we observe a rearrangement between two collinear SCOP domains, it should be artificial.

We use this property in our calibration setting. Our queries are SCOP domain sequences with up to 40% identity between each other, which we obtained from the ASTRAL [20, 24] website. The targets are other domains from the same family, either with an '*intra*' or with an '*inter*' relationship. Each target is split into fragments corresponding to DSSP-derived [50] secondary structure elements (SSEs). The setting is outlined in Figure 37. Given the data sets thus prepared, we measure the rearrange-

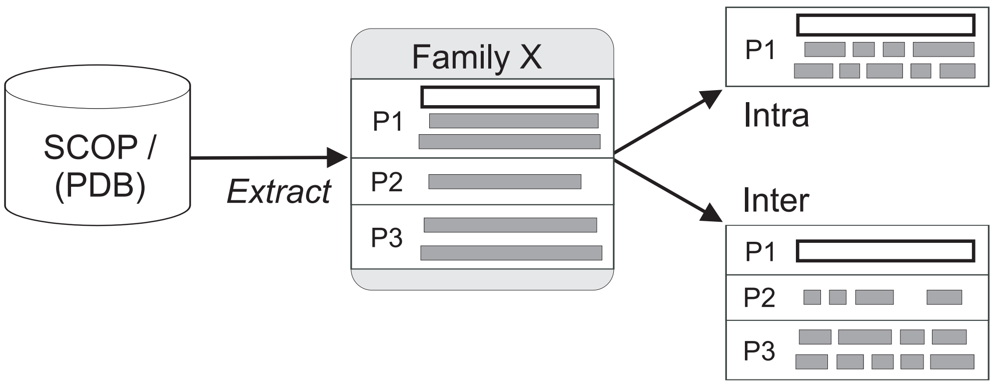


Figure 37: *Given a hypothetical query from "protein/domain" entry P1 of some SCOP family X, the targets are either domain sequences from the same protein/domain (intra), or domains from other protein/domain entries (inter, P2 and P3).*

ment cost (RC) that is necessary to remove all rearrangements from an alignment of the query with the target fragments. For each alignment we record the RC where the number of observed rearrangements drops to zero, the *collinearity threshold*. If there is no rearrangement at all, the result is omitted. In our first experiment, we apply four different deletion cost (DC) values to determine their influence on the collinearity thresholds we obtain.

We compute the alignments and the optimal path, then we plot the resulting family-specific collinearity thresholds for all SCOP families. For each query domain we record the maximum collinearity threshold that it reached (a) with all *intra* and (b) with all *inter* targets. All maximal collinearity thresholds are averaged for (a) and (b) within one SCOP family. These values are far above the average collinearity threshold of each family.

In our second experiment we manually investigate individual query-target combinations with high collinearity thresholds, i.e. that maintain one or more rearrangements even at high RC values, and identify their families. The annotation of each family is looked up in SCOP, to find a reasonable explanation for the high values. We also check whether the results produced by our model algorithm are consistent with the SCOP annotation.

Finally, in our third experiment we study the RC distribution of circularly permuted alignments. We collect the collinearity thresholds of all verified query-target combinations, where either the query *or* the target is circularly permuted. We compute the family-specific average alignment scores and their ratio to the possible maximum score. Again, we separate between *intra* and *inter* data.

We used the following parameters throughout our experimets: An alignment is accepted if a certain length-dependent score percentage is reached. These values are: length 2, 100%; length 3, 50%; length 4, 33%; length 5 and more, 25%. The SCOP and ASTRAL versions were 1.69. The substitution matrix was Blosum62, unless

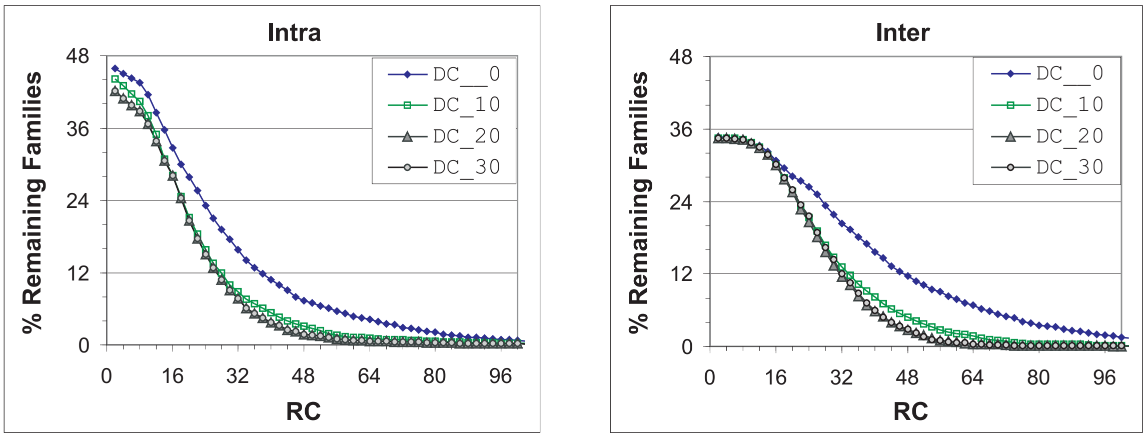


Figure 38: *Influence of the DC on the RC: The y-axis shows the percentage of SCOP families that still have at least one rearrangement at a given RC-value (x-axis). The left part shows the intra results, the right one the inter results.*

otherwise indicated. Penalties for gap initiation and extension were -11 and -1, respectively.

13.3.4 Results and Discussion

Deletion Cost Influence and Selection: We measured the family-specific collinearity thresholds for all SCOP families in the query set with four different DC values: 0, 10, 20, and 30. The RC value was varied between 0 and 256, in steps of two. The results are shown in Figure 38. Almost 50% of the 2829 families in the query set contributed to the *intra* data shown in Figure 38 (left). That means, those families have an *intra* setting and the comparisons lead to collinearity thresholds > 0. After a RC value of 8, the four curves decline rapidly. Most families lose all their rearrangements between RC 8 and RC 40. After RCs of 50 (for DC 20), 56 (for DC 10 and DC 30), and 86 (for DC 0), only one percent of the remaining families contain still at least one rearrangement.

The *inter* curves in Figure 38 have a similar shape. However, only about 35% of the families contributed to the data. Rearrangement costs between 0 and 8 do not reduce the number of remaining families, the values decrease mainly between 10 and about 50, except for DC 0.

We sampled the collinearity thresholds at four different DC values, to find the combination of DC and RC that is most effective in suppressing rearrangements. The results show the importance of the DC parameter. If it is set to 0, the collinearity thresholds are very high. More than 40 families (about 1.5%) need RC values above 100 to lose all their rearrangements in the inter setting. Just increasing the DC value by 10 has a large effect on the measured data. At DC 10, only three families have a collinearity threshold of 100 or more. A further increase of the DC value continues to decrease the collinearity thresholds, but the effect is quite small.

Although not by much, the curve for DC 30 is clearly above that for DC 20, which means that we observe more rearrangements at DC 30 than at DC 20. This behavior can easily be explained with our model: The increase of the deletion cost from 20 to 30 leads to an increased use of rearrangements. Sometimes, one or more deletion-causing fragments are replaced by others that cause a rearrangement. For example, assume an optimal complete path P. When the deletion cost is increased by ΔDC, another, formerly suboptimal, complete path P' will become optimal if and only if

$$weight(P) - weight(P') < (d(P) - d(P'))\Delta DC.$$

The results show that setting the DC value higher than zero is necessary. If it is however too high, the number of rearrangements will increase again. To set the deletion cost to 20 is therefore the best choice among the sampled DC values.

Alignments with high Collinearity Thresholds: We identified 50 families based on the highest collinearity thresholds of single query-target combinations. The top 20 were further investigated, they are shown together with an excerpt of the SCOP annotation in Table 7.

Not all of the rearrangements are due to noise. In fact, few of the high collinearity thresholds seem to be present by chance. Seven of the 20 families contain protein domains with circular permutations. Many of them are well conserved and can be easily identified in our results. However, as always when dealing with real biological data, there are interesting borderline cases. E.g., the methionine aminopeptidase protein in SCOP family 55921 ("Creatinase/aminopeptidase") contains several domains that have an additional inserted domain. And the circular permutation is actually in this insert.

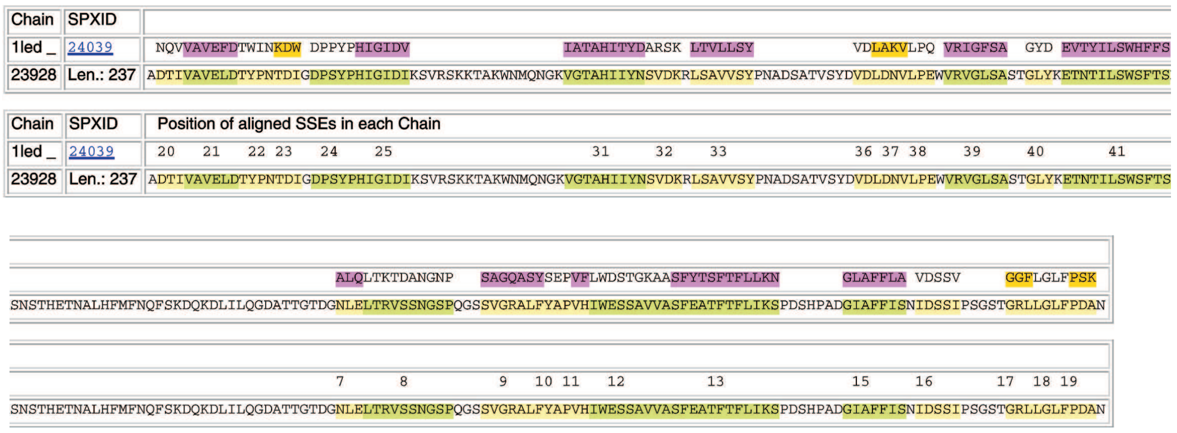
Most of the other families also contain domains whose sequence and/or structural properties explain the detection of highly conserved rearrangements. In a sequence database search, we would like to find those instances. Figure 39 shows an example result with a circular permutation, where the rearrangement is still maintained at RC 160.

Some of the identified families with circularly permuted protein domains are well known in the literature (see [109], and references therein). However, other cases of circular permutations are known in the literature that are not included in Table 7. One example are the "Swaposin" (Family-ID 47866) and the "Saposin B" (ID 81806) families [79]. The reason is that those cases are classified into *different families*, whose circular permutations can not be detected in our calibration setting.

The SCOP annotation was our primary source of information in validating our results. However, its main focus being classification, we were not surprised to see that it is not always complete. For example, when we investigated the high collinearity thresholds of family 50263 ("Single strand DNA-binding domains"), only one query was responsible for the high thresholds, the "ssDNA-binding protein" of *Deinococcus radiodurans*, an extremely radiation-resistant bacterium. The corresponding SCOP entry does not include specific information, however, the N-terminal DNA-binding domain of the protein does indeed occur twice in tandem [111].

Family	RC_A	RC_E	SCOP Annotation	Rem.
49925	>256	70	β-glucanase. Many known CPs.	CP
53057	>256	-	β-carbonic anhydrase. Duplication.	DP
53749	>256	-	Phosphoglycerate kinase. One entry: CP.	CP
100953	>256	-	DsbA-like. One target (99645): CP.	CP
49900	136	228	Legume lectins: several natural CPs.	CP
51323	228	-	Cyanovirin-N. One target (79717): CP.	CP
47502	164	134	Calmodulin. DP: two pairs of EF-hands.	DP
56602	162	74	β-Lactamase. One target (42707): CP.	CP
50877	158	-	Streptavidin. Two targets: CP.	CP
48696	-	154	Cytochrome c3-like. Repeated motif.	DP
52059	-	154	Internalin LRR domain. Variable repeats.	DP
49696	142	108	Crystallin. Duplication: Two domains.	DP
48404	-	138	Ankyrin repeat. Badly conserved.	??
50263	116	-	SSB DNA-binding domain. Duplication.	DP
48372	110	88	Armadillo repeat.	??
81902	-	110	HCP-like Cysteine-rich protein.	DP
48453	-	104	Tetratricopeptide repeat (TPR). DP?	??
89299	100	-	MBT repeat. Tandem repeat.	DP
47875	-	98	Annexin. Possible DP, badly conserved.	??
50979	-	98	WD40-repeat. Badly conserved.	??

Table 7: *The top 20 SCOP families with the highest collinearity thresholds of individual test cases, at deletion cost 20. The collinearity thresholds are displayed separately for test cases with intra (RC_A) and inter (RC_E) relationships. The "SCOP annotation" column contains the family name, and in most cases an additional remark.*
Abbreviations: 'CP': circular permutation. 'DP': (multiple) duplications. '-': results are impossible here, or, all results were without a CP/DP. '??': We could not confidently validate a CP/DP.



Chain	SPXID	
1led _	24039	NQVVAVEFDTWINKDW DPPYPHIGIDV IATAHITYDARSK LTVLLSY VDLAKVLPQ VRIGFSA GYD EVTYILSWHFFS
23928	Len.: 237	ADTIVAVELDTYPNTDIGDPSYPHIGIDIKSVRSKKTAKWNMQNGKVGTAHIIYNSVDKRLSAVVSYPNADSATVSYDVDLDNVLPEWVRVGLSASTGLYKETNTILSWSFTS

Chain	SPXID	Position of aligned SSEs in each Chain
1led _	24039	20 21 22 23 24 25 31 32 33 36 37 38 39 40 41
23928	Len.: 237	ADTIVAVELDTYPNTDIGDPSYPHIGIDIKSVRSKKTAKWNMQNGKVGTAHIIYNSVDKRLSAVVSYPNADSATVSYDVDLDNVLPEWVRVGLSASTGLYKETNTILSWSFTS

ALQLTKTDANGNP SAGQASYSEPVFLWDSTGKAASFYTSFTFLLKN GLAFFLA VDSSV GGFLGLFPSK
SNSTHETNALHFMFNQFSKDQKDLILQGDATTGTDGNLELTRVSSNGSPQGSSVGRALFYAPVHIWESSAVVASFEATFTFLIKSPDSHPADGIAFFISNIDSSIPSGSTGRLLGLFPDAN

7 8 9 10 11 12 13 15 16 17 18 19
SNSTHETNALHFMFNQFSKDQKDLILQGDATTGTDGNLELTRVSSNGSPQGSSVGRALFYAPVHIWESSAVVASFEATFTFLIKSPDSHPADGIAFFISNIDSSIPSGSTGRLLGLFPDAN

Figure 39: *Passta alignment result from SCOP family 49900, "Legume lectins". The query domain (23928) is circularly permuted and has an inter relationship to the target domain (24039). The rearrangement and deletion costs in this example were 160 and 20, respectively. The score is 213 (of at most 1223). The alignment shows one rearrangement between the target fragments 41 and 7, and three deletions (between fragments 25 and 31, 33 and 36, and 13 and 15). The different shades coloring some fragments of the query correspond to the score of the aligned target fragments. Orange target fragments are α-helices, purple ones are β-strands.*

Collinearity Thresholds of Circularly Permuted Alignments: The collinearity thresholds of test cases with validated circular permutations and their associated scores are subsumed in Table 8. Not all families contributed to both settings, and there are only 44 *intra* results, in contrast to 281 cases with an *inter* relationship. Aside from the seven families that were identified in Table 7, we also included data from the G-proteins (ID 52592). Two queries of the G-proteins are circularly permuted (88294 and 107552).

The average *intra* collinearity thresholds range from 62.0 to 230.0, and the average score ratios from 14.2% to 69.7%. Most *inter* values are lower. Here, the average collinearity thresholds are between 34.7 and 133.8, and the average scores are between 5.12% and 13.91% of the possible maximum.

The results show that the determined collinearity thresholds are largely family-specific. If two domains have a low degree of sequence similarity, most target fragments do not align well with the query. The *inter* results show three cases of such families, where the average score ratios reach only five to six percent. The circular permutations in these cases are only supported by a few low-scoring alignments. Here, a low RC value is often sufficient to suppress the rearrangements.

The results of the "β-Crystallin" family (49900) further support this view. It is the only family whose values are similar in both parts of Table 2. Obviously, the sequence similarity within a SCOP family is large enough to reliably find circular permutations, which may not be the case for two different families.

'Intra'

Family	k	RC	Score	MaxScore	Ratio [%]
49925	9	151.6	563.6	1234.7	45.6
53749	12	161.5	908.5	2076.7	43.7
100953	2	189.0	452.0	958.5	47.2
51323	1	230.0	243.0	525.0	46.3
49900	10	121.2	173.1	1222.0	14.2
50877	5	150.8	445.4	639.0	69.7
56602	5	62.0	351.4	1363.4	25.8

'Inter'

Family	k	RC	Score	MaxScore	Ratio [%]
49925	17	38.4	85.4	1418.3	6.02
49900	30	133.8	164.7	1184.4	13.91
52592	227	34.7	65.3	1276.5	5.12
56602	7	41.4	90.1	1569.7	5.74

Table 8: *The table shows the intra and inter results for query-target combinations with validated circular permutations. The RC column contains the average rearrangement cost reached in the experiment. Column k shows the number of results that contributed to the averages for "Score", "MaxScore" and the ratio of Score/MaxScore ("Ratio").*

The properties of the rearrangement-causing event are also important. A large duplication (e.g., of a domain) is generally easy to detect, because many alignments will support it. However, if a rearrangement is caused by a circular permutation of a short fragment, the signal may easily drown in the noise introduced by chance alignments.

13.3.5 Conclusion

The complex structure of biological sequences lead to the development of two-phased alignment strategies able to deal with rearrangements. Those algorithms require the introduction of new parameters, which have to be calibrated to meaningful values.

We introduced a simple algorithm for generalized sequence alignment. Its input is a query sequence and a set of pairwise, local alignments. From those alignments, the algorithm selects the combination of highest weight in a graph-based approach. A path in the graph, i.e., a combination of alignments, may contain deletions and rearrangements. These events are penalized by two parameters, deletion- and rearrangement cost (DC and RC).

We proposed a calibration setting to systematically determine good values for both parameters. In the calibration on SCOP families, we used four different DC values and varied the RC value between 0 and 256. The results show that the model algorithm is able to detect well-conserved rearrangements or duplications. However, if query and target have diverged so far that only very few segments are conserved at all, the method has reached its limits. The limiting factor is mainly the amount of information that pairwise alignments can provide.

The choice of the RC parameter depends on the application scenario. If one needs just to find a few “safe” hits with rearrangements, the RC parameter can be set to a value of 48 or higher in conjunction with the Blosum62 substitution matrix. This value will also be good for an application on multidomain proteins. If one is looking for “new” sequences with rearrangements in a database search scenario, a value of 34 or more may be used. This will remove most of the noise and reveal interesting candidates, which must be investigated further, however.

Our experiments measured the collinearity thresholds necessary to remove rearrangements *within* domains (except for the SCOP multidomain class). What we actually measured is the minimal RC/DC values that are needed to eliminate the background noise introduced by biologically unsignificant alignments of short sequence fragments.

However, these values are probably not very useful in another context, e.g. in genome alignment. Even though the additional noise introduced by the use of longer sequences does not rise linearly with the sequence length [99], the collinearity thresholds should increase here. Besides, a rearrangement of 100,000 base pairs in genome alignment is not due to the same biological process as a circular permutation within a gene. Therefore, we believe new parameters should be defined and calibrated in connection with the underlying biological process. However, this requires model-dependent modifications of the calibration setting to measure corresponding collinearity thresholds.

Part V

Comparative Evaluation of Jali, Passta, and BLAST

Early during the development of a new method, major flaws and problems are easy to detect. This gets increasingly difficult in later stages of the project. Finally, only a systematic evaluation can reveal the strengths and weaknesses of the new method. An evaluation should compare the method under development to at least one other established and well understood method from the same scientific realm.

We compare Passta to BLAST [5] and to the Jumping Alignment algorithm Jali (see Part III). Even though advanced methods for homology recognition were recently developed [54, 80], BLAST is still *the* standard tool used in practice. The BLAST algorithm is a heuristic that conveys a good tradeoff between speed and sensitivity (more details are given in Section 7.3). Due to the well defined statistical background, BLAST is able to use much of the information available in pairwise sequence comparison. Its heuristic approach makes it however less sensitive than approaches exploring the available sequence space completely, e.g. the Smith-Waterman algorithm [96]. Passta does also rely on pairwise sequence comparison, sacrificing some of the available sequence information as well. These analogies make BLAST a good choice for evaluating Passta.

Finally, the Jumping Alignment algorithm is evaluated once more. The latest Jali evaluation was done in 2003 [13]. Since then, the databases have continued their almost exponential growth and it will be interesting to see whether Jali's performance is affected, and how. We compare Jali at three different jumpcost values to Passta, even though a jump in Passta is somewhat different to a jump in Jali. In contrast to all previous evaluations of Jali, we use a modified experimental setup that allows us to compare the three methods in *close* homology detection as well. Most earlier evaluations specialized only on *remote* homology detection.

14 Experimental Setup

14.1 Choice of the evaluation model

Every evaluation requires a reliable standard of truth, i.e. a data set with confirmed homology relationships. If these homologies were inferred by sequence similarity alone, the evaluation would face the "Chicken and Egg" problem described by Brenner *et al.* [19]. The problem occurs when the standard of truth was determined with the same type of information used to evaluate the developed method.

The homology relationships in SCOP (see also Section 6.4) are mainly defined by *structural* similarity. When SCOP is used as standard of truth in sequence analysis, the "Chicken and Egg" problem can not occur. Therefore we use the SCOP classification as standard of truth, as in the previous evaluations.

Most previous evaluations with SCOP concentrated on distant relationships and ignored close homologies within the same SCOP family. We therefore extend the evaluation setup by intra-family close homology detection. This requires a classification level below the "family" nodes of the SCOP tree : the "protein/domain" nodes. They contain groups of closely related protein domains with defined biochemical function. For simplicity, we will refer to this SCOP layer as *domain level*, except where the difference to real SCOP domains, i.e. the leafs of the SCOP tree hierarchy, is of special importance.

In previous evaluation setups, the entries in each family were treated alike, regardless of being from different domain levels or not. The evaluated methods were therefore not characterised on the domain level. This is no problem with single-domain families, but the results with multidomain families may be slightly biased. The reason is that one domain of a multidomain family can be close to all queries in the test set and this domain will always deliver the best result in the evaluation, shadowing the results of the other available domains.
The missing separation may also affect methods working with multiple alignments, e.g. Jali. The multiple alignments used as its input may differ in certain characteristics if they are sometimes composed of one domain, and sometimes of multiple domains. Even though these arguments are of minor importance, we prefer to investigate close *and* remote homology relationships at the domain level.

Given a query sequence in domain A of an arbitrary family with multiple domains, the task in close homology detection is to find the entries of other domains in the same family. This is illustrated in Figure 40. Finding closely related SCOP domain entries is relatively easy, because the sequence similarity within a SCOP family is at least 30%. However, close homology detection is not *always* easy. Some SCOP domains have received modifications at the sequence level, which have however not changed their overall 3D structure by much. These modifications are mainly repeats, insertions, and rearrangements. The affected sequence portions range in size from small motifs to complete domains.

The sequence similarity of remotely related sequences is often lower than 20%, close to that of unrelated sequences [84]. Therefore *remote homology detection* is the supreme discipline for methods in sequence analysis. As shown in Figure 40, close and remote homology detection can be easily combined. Our experimental setup separates three different homology relationships: "Close", "Distant", and "Unrelated".

We compare the three methods by computing *minimal false positive counts* (minFP counts) on their database search results. The result of a database search with a single query sequence (Passta, BLAST) or multiple alignment (Jali) is a list of targets ranked by their scores or e-values. The minFP count determines the number of false positives (FP) that occur before the first true positive (TP) in the target list. At best, the first target in the list is a true positive. The minFP value is then zero. At worst, there is no TP among the ranked targets at all. In that case, the minFP count is set to

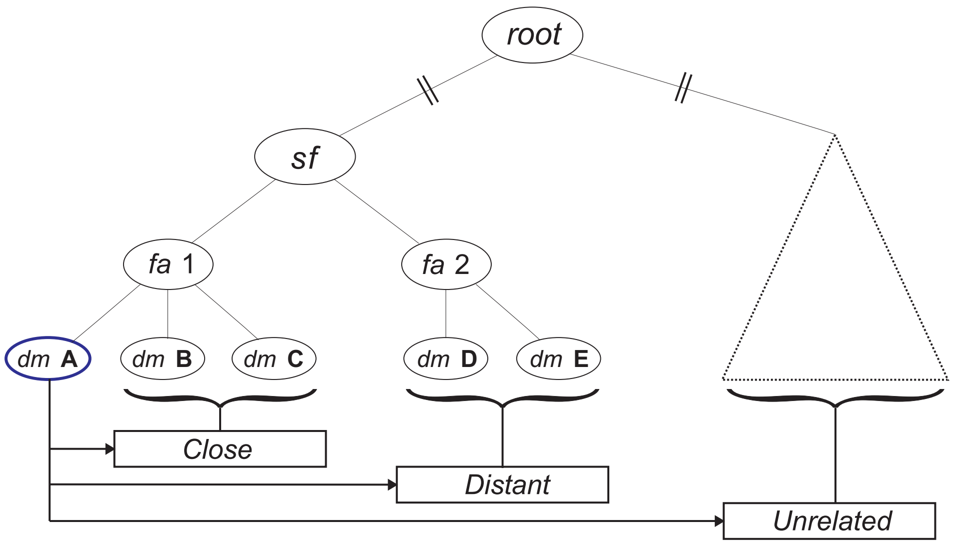


Figure 40: *The three different homology relationships in our experimental setup: Given a query domain (dm A) from an arbitrary SCOP family, the compared methods may find a domain from a) the same family ("Close"); b) the same superfamily, but not the same family ("Distant"); and c) another superfamily ("Unrelated").*

the smallest value exceeding the preferred size of the candidate set, i.e. 101 or 1001. A database search is basically successful if a true positive target can be found. In practice, it is important to know *when* to expect the first TP for a given method. To compare the evaluated methods, we define a *cutoff* value $c \in \mathbb{N}$. A search is considered successful if the minFP count is less or equal to c. In our experiments, we plot the percentage of successful searches for minFP cutoffs up to 100.

We believe that our proposed evaluation setup is well suited to test a new method in database searching. The data we produce will lead to a more refined picture, not only for Jali, but also for Passta and BLAST.

14.1.1 Passta

As described in Section 10.2, Passta has two phases: *Pass One* collects a subset of promising targets then used by *Pass Two* in the annotation of the query. If the candidate set does however not contain any related, i.e. true positive targets, they are obviously not available in Pass Two. Therefore we characterize the strengths and weaknesses of Pass One and Pass Two separately.

Pass One: In the evaluation of Pass One, our primary interest is the percentage of related targets that remain in the candidate set after Pass One has finished. We count the percentage of "Close", "Distant", and "Unrelated" targets before and after we apply the selection algorithm/heuristic introduced in Section 12.2.4. The mentioned selection heuristic tries to restrict the candidate set to a user-provided preferred size.

Parameter	Fast	Default
Gapcost Coil	9 / 2	9 / 2
Gapcost Helix	10 / 2	10 / 2
Gapcost Strand	8 / 2	8 / 2
Blosum62 thresh.	22	20
MSR	0.333	0.30
Needed alignment rate	1/10	1/10

Table 9: *Parameters used in the evaluation of Pass One, as determined in the project Section 13.2. The parameters are explained in Section 12.2.*

We use two different preferred sizes in this evaluation to judge the quality of the heuristic, 100 and 1000. In case it does not work as well as we expect, we also supply all missing true positive targets to the "Single" setup of Pass Two (see below), to allow Passta a second ranking at 100% sensitivity.
The other parameters used in the evaluation of Pass One are those derived from the training in Section 13.2. They are shown in Table 9.

Pass Two: Jali and BLAST consider one target at a time. The optimal result for each target is known and co-optimal hits are visible. By sorting the raw scores (Jali) or e-values (BLAST) of the target hits, a ranking of the targets can be easily produced.
Passta may actually use multiple targets in the annotation of the query. Here, several target sequences contribute to the final score. Decomposing the final score into the partial contributions of each target is problematic. Especially in the "Fast" mode of Passta, the optimal path is influenced by the available edges, and to evenly divide the jumpcost between two targets seems unfair (see Figure 31 on page 62). Besides, only a subset of the candidate set is actually used for annotation. Unused co-optimal targets in the candidate set receive no score and therefore remain hidden. A direct comparison of Pass Two in its native mode with Jali or BLAST is not possible in this way.

Therefore, we evaluate Pass Two in two distinct setups. The setup with the complete candidate set described above is called "Full". A "Full" result is called *successful* if at least *one* of the targets used in the annotation is closely or remotely related to the query.

The second setup is called "Single". Here, Pass Two is iterated over all targets in the candidate set, but only for one target at a time. Each target receives the maximal score that it can reach in Pass Two with the given parameters. The targets in the candidate set can then be ranked by these scores, allowing us to compare the Pass Two results to that of Jali and BLAST.

14.1.2 Jali and BLAST

The setup for the evaluation of **BLAST** is similar to the "Single" setup in Pass Two (see previous paragraph). The BLAST targets are however provided in a file with all SCOP domains that are available in PasstaDB. Targets that are also contained in the query set are removed before the minFP counts are computed. We sample three different combinations of affine gapcost: 9/2, 10/2, and 11/1 cost for gap initiation and extension.

The **Jali** setup requires multiple alignments as input. We construct multiple alignments of all query sets with ClustalW, each alignment containing between three and ten sequences. The alignments are the queries for `jsearch`, the Jali executable for database searching. We use three jumpcost values in the experiments with Jali: 22, 34, and 46. A jumpcost value of 22 was determined as the optimum in the initial evaluation of Jali in [97]. The other two values were chosen because we use identical jumpcost values in the evaluation of Passta (see above). The ClustalW version used was 1.82.

14.2 Evaluation and calibration data

A superfamily has to contain at least two families to have a “Distant” relationship that we can evaluate. A “Close” relationship requires a family with at least two different protein/domain entries. Any family can contribute queries to the evaluation if it has either of these homology relationships. That is however not the only condition: every “Close” or “Distant” test case is required to have at least four query- and three target sequences. One of them was used in training Passta (see Section 13.2), the remaining ones are queries in this evaluation. We use at most ten queries of each test case, even if it contains more than ten sequences. The main reason for this constraint is to limit our results being biased by large domains with many entries. A welcome side effect is the reduced runtime of the evaluation protocol.
All SCOP domains that either have a “Close” or a “Distant” relationship to a given query are putative targets. We say *putative* targets, because not all SCOP domains are represented in PasstaDB. We have to restrict the query and target sets to those domains covered in PasstaDB, even for Jali and BLAST, otherwise a comparison is not possible.

We used the Blosum62 [44] substitution matrix throughout our experiments. To avoid redundant and almost identical sequences, we restricted our experiments to the “P90d” subset of SCOP version 1.69 instead using the complete database. Every two entries in the P90d share at most 90% sequence identity. The subset is available from the ASTRAL website (see Section 6.4.1).

15 Results and Discussion

The following Sections 15.1-15.3 contain experimental results with a short, method-specific discussion. In Section 15.4, we compare and discuss the results of all three methods together.

15.1 Passta Pass One

Table 10 summarizes the Pass One results. The upper set of values was measured before the selection heuristic was applied, we therefore call them “native”. The values in the lower set were measured after the selection heuristic was applied, we call them “filtered”. The filtered values are displayed for both candidate set sizes used, 1000 and 100 preferred candidates. We refer to to them as the “large” and the “small” candidate set.
The three central columns in Table 10 show the percentage of targets accepted by Pass One for each homology group. These percentages are based on the sum of targets in all experiments, not on the average candidate set. That is the reason why the values do not add up to 100%, they can be better compared in this way.

The native results in Table 10 show that about 70% (“Fast” mode) and 80% (“Default” mode) of the “Close” targets are basically accepted by Pass One. The filtered values below show the percentage of “Close” candidates that remained after the selection heuristic was applied. Here, most of the “Close” targets are retained in the large candidate sets (around 70%), and about 28% in the small candidate sets. The results of the “Distant” and “Unrelated” columns can be read in the same way.

The size of the average candidate set is displayed in the rightmost column of Table 10, “Avg. cand.”. The values show the number of accepted targets before and after the selection heuristic was applied. The native “Default” value (2022.9 targets) is twice as large as the native “Fast” one (921.7 targets). This variation is less pronounced among the filtered sets: the average sizes of the large candidate sets are small candidate sets are 84.2 and 89.7, less than ten percent different.

The differences between the “Fast” and the “Default” modes must be attributed to the slight variation of the Pass One parameters (see Table 9). Lowering the Blosum62 threshold to 20 and the MSR threshold to 0.30 in the “Default” mode obviously results in more accepted targets and larger candidate sets. In the upper part of the table, the portion of accepted targets is increased by 10% “Close”, 20% “Distant”, and almost 14% “Unrelated” targets. In the lower part of the table, the differences between “Fast” and “Default” are almost negligible with a candidate set size of 100. It ranges from less than 0.1% in “Unrelated” to about 0.6% in “Close” and “Distant”. The differences in the large candidate sets are more remarkable, ranging between 1.5% in “Unrelated” to 5% in “Distant”.

Unfortunately, the dynamic selection of the Pass Two candidates is rather inefficient. Even though finally less than 1% “Unrelated” domains are accepted by Pass One, their absolute number is too large.

15.2 Passta Pass Two

As explained in Section 14.1.1, a direct comparison of Pass Two with Jali and BLAST is not possible. Therefore, we have split the evaluation into two parts, “Single” and “Full”.

	% accepted targets			
Mode	**Close**	**Distant**	**Unrelated**	**Avg. cand.**
		Native		
Fast	70.14	19.73	11.05	921.7
Default	80.56	40.16	24.75	2022.9
		Filtered		
Fast (1000)	68.31	16.86	7.72	651.7
Default (1000)	71.75	21.86	9.37	787.3
Fast (100)	27.84	3.38	0.91	84.2
Default (100)	28.45	4.04	0.98	89.7

Table 10: *Pass One results: percentage of accepted candidate domains before ("Native") and after ("Filtered") the selection heuristic was applied. We have covered both candidate set sizes of 100 and 1000 in the bottom part of the table.*
Even though the portion of related targets ("Close", "Distant") is always higher than those of the "Unrelated" targets, the separation between "Distant" and "Unrelated" is not reliable.

Single: In the "Single" setup, we compute a score for every target of the candidate set and rank the targets by these scores. If the ranking contains at least one true positive target within the first "cutoff" values (minimal false positive count), the search is considered successful. The percentage of successful searches depending on the given cutoff value is shown in Figure 41. The diagrams on the left side show the results for candidate sets of size 100, the ones on the right side those for the large candidate sets with size 1000. The "Close" results are displayed at the top, the "Distant" results at the bottom of Figure 41. The diagrams to the right show four curves instead of two. Here, the dashed lines ("Extd.") represent the data with 100% sensitivity, where *all* true positive targets were submitted to Pass Two, even if some of them were not present in the Pass One candidate set.
"Close": About two thirds of the test cases are successful at a cutoff of zero, in the small candidate sets as well as in the large ones. With increasing cutoff values, the curve is rising to maxima of about 80% (100 candidates) and 86% (1000 candidates). "Distant": The two "Distant" curves start with slightly less than 20% successful searches. The initial rise of the curves is steep, reaching about 40% at a cutoff near ten. The slope is then decrasing, reaching 60% of successful cases in the "Default" setting at 100 candidates, and slightly less in the "Fast" setting. The large candidate sets of 1000 preferred targets make almost no difference between both modes, both curves reach close to 70%. The "extended" curves in the diagrams to the right reach values close to 90% in the "Distant" setting, and about 95% in "Close".

Full: In the second part of our experimental setup ("Full"), we evaluate Pass Two in its native mode. It annotates the query with a few targets from the candidate set. At least one true positive domain has to be among those targets to consider the

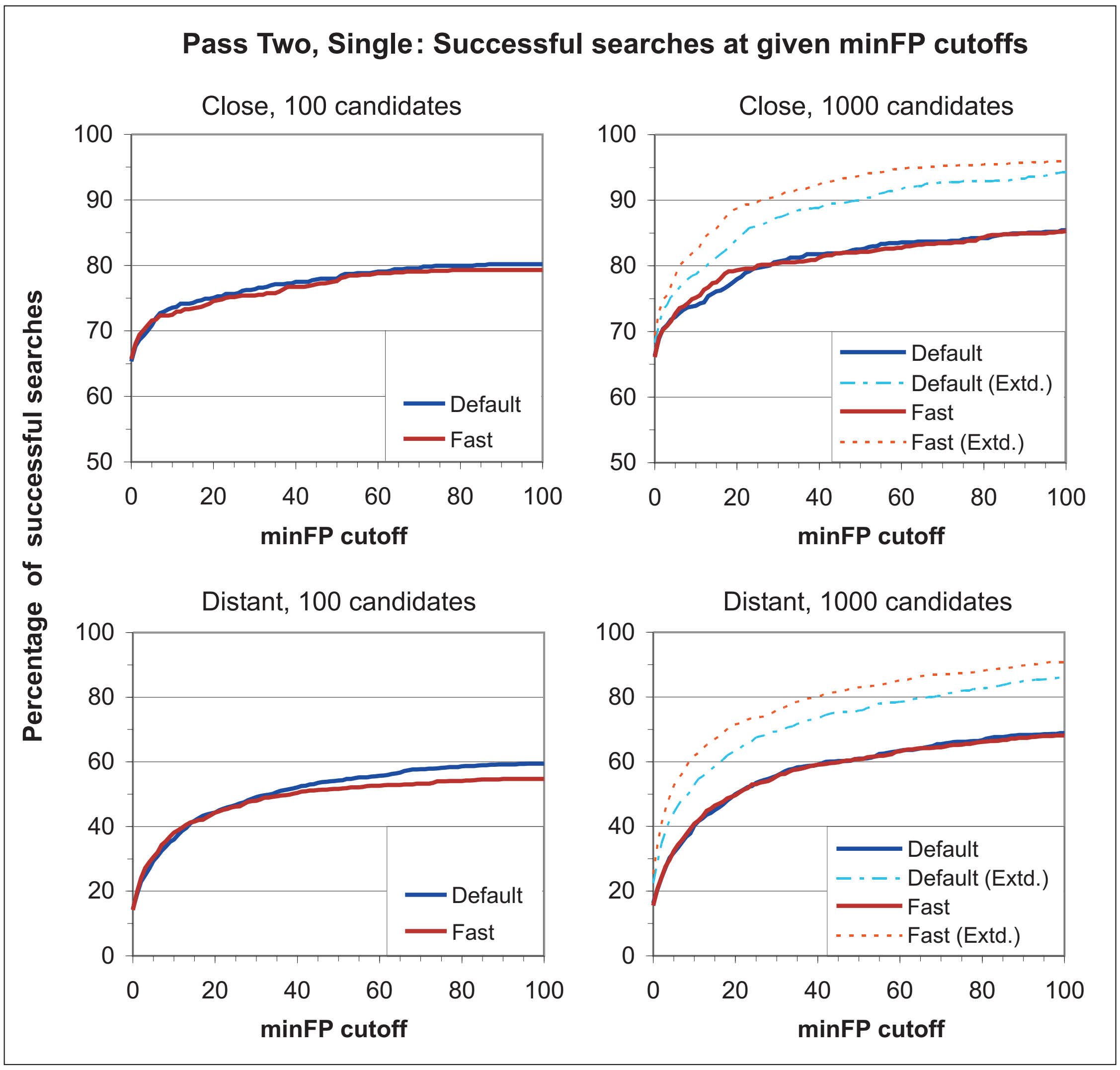


Figure 41: *Pass Two minimum false positive (minFP) counts at candidate set sizes of 100 (left) and 1000 (right). The "Close" results are shown at the top, the "Distant" results are displayed at the bottom of the Figure.*
The dashed lines in the right diagrams show the results of Pass Two at 100% sensitivity. Here, we supplied the true positive targets that were missing in the respective candidate sets. They show the potential quality of Passta if Pass One was improved.

Setting		**% successful searches**			
		Fast		**Default**	
	Jumpcost	**RC 32**	**RC 40**	**RC 32**	**RC 40**
Close only	34	59.83	60.26	58.08	58.08
	40	58.08	57.21	55.90	55.90
	46	56.33	56.33	55.46	55.46
Distant only	34	25.97	26.41	31.60	31.17
	40	23.38	23.38	29.87	29.87
	46	22.08	21.21	27.27	27.27
Both	34	76.53	77.04	75.17	75.00
	40	72.79	72.79	73.64	73.81
	46	70.24	69.05	72.96	72.79

Table 11: *Pass Two "Full" results, first part. A search is considered successful if at least one true positive target is used in the annotation of the query. The percentage of successful searches is shown for all parameter combinations in the experimental setup. The results are discussed in the main text.*

search *successful*. The results are summarized in Tables 11 and 12. Both tables display the data for all combinations of jump- and rearrangement costs in our experimental setup, and also separate the "Fast" from the "Default" mode of Passta.

Table 11 shows the percentage of successful experiments in three categories: "Close only" relates to test cases without "Distant" targets; "Distant only" relates to cases without "Close" targets. "Both" groups of targets were available in the third category. The highest percentage of successful searches was realized in "Both", between 69% and 77%. The values in "Close only" are between 55% and 60%, and those in "Distant only" between 21% and 32%. The differences between rearrangement costs of 32 and 40 are negligible, mostly below 1%, but the differences between the "Default" and the "Fast" mode can be clearly seen. Especially the "Distant only" results improve in the "Default" mode by about 5% more successful searches. Surprisingly, the "Close only" values decrease by about 2%. There is a global tendency of increased jumpcosts leading to less successful searches.

The data in Table 11 is complemented by those in Table 12. It shows the sum of all targets used by Pass Two for the annotation of the query, which is between 1723 and 4367. This clearly explains the decrease of successful searches observed in Table 11. It is indeed due to the increased jumpcost, which lowers the number of Pass Two targets that can be used for annotation.
Besides the number of annotation targets, Table 12 also displays the portion of "Close" and "Distant" targets contained in that number. The portion of "Close" targets is between 23% and 42%, that of the "Distant" targets ranges between 3% and 5%.
The majority of annotated targets is unrelated to the queries and the low portion of "Distant" targets is rather disappointing. On the other hand, the "Distant only" results in Table 11 imply that Pass Two managed to include at least one distant target

Setting		**Annotation targets**					
		RC 32			**RC 40**		
	Jumpcost	**nTarg**	**% Close**	**% Distant**	**nTarg**	**% Close**	**% Distant**
	34	4178	23,91	2,97	4367	23,31	2,95
Fast	40	2654	32,93	3,58	2810	31,85	3,67
	46	1827	42,36	3,94	1827	41,40	3,86
	34	3421	24,64	4,18	3437	24,53	4,10
Default	40	2343	31,46	5,04	2358	31,30	4,96
	46	1723	38,42	5,46	1759	37,58	5,46

Table 12: *Pass Two "Full" results, second part: The table shows the number of target domains used in the annotation process by Pass Two (nTarg, along with the percentage of contained "Close" and "Distant" targets.*

in about a quarter of the test cases, which is rather pleasing. It shows that once Pass Two gets the correct information, it is able to use it. Another positive observation is the increase in the percentages of related targets with rising jumpcosts. It shows that Pass Two rather discards "Unrelated" than related targets.

15.3 Jali and BLAST

The results of the Jumping Alignment algorithm can be easily described and explained. Figure 42 shows the percentage of successful searches at given minimal false positive cutoffs. In "Close" homology detection, the curves start at about 87% successful searches and reach more than 95% at a cutoff of 100. The "Distant" curves start at about 43% and rise quickly to 70% at a cutoff of 20. They finally reach 80% of successful searches.
The effect of the three different jumpcost values is negligible, the three curves are almost congruent. That is in accordance with earlier evaluations of Jali. They showed the minor effect of the jumpcost parameter when working on protein domains. It can be expected that the effect would be much more pronounced with multidomain proteins.

The BLAST results are shown in Figure 43. The "Close" homology curves start at slightly below 85% and reach 95% at a minFP cutoff of 100. That value is however only reached by the curve with the 9/2 cost for gap initiation and extension, the two other curves reach about 2% less successful searches.
The results show why BLAST is such a popular tool: Given the speed of the method and the little resources it requires, the quality of the results is very good - close to the Jali algorithm that can rely on more information. Besides, even though the "Close" results of both methods are impressive, they nevertheless show that close homology detection is not always easy, as 3% to 7% of these relationships could not be found

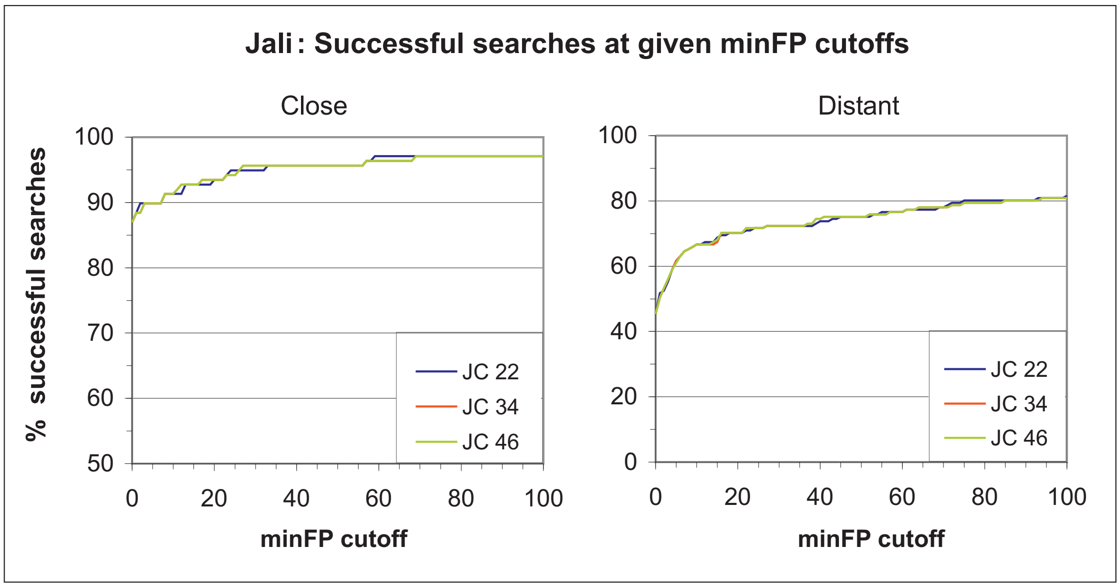


Figure 42: Minimal false positive counts for Jali. Shown is the percentage of successful searches at given minFP cutoffs for three different jumpcost values. The results are very good, in "Close" (left) as well as in "Distant" (right) homology detection. The different jumpcosts have little effect on the results, such that the curve at jumpcost (JC) 34 is mostly covered by the curce at JC 46.

with a cutoff of 100.

15.4 Comparative Discussion

For convenience, we display the most important data from the preceding three Sections in Figure 44. It shows the number of successful searches for minimal false positive counts up to 100. The displayed data is from the "Fast" mode of Passta, obtained with the large candidate sets of size 1000. The Jali and BLAST results are so similar that one curve for each method is sufficient in each diagram.

The left diagram shows that Jali is superior in finding *closely* related targets. At any of the given cutoffs, its curve is above those of BLAST and Passta. The BLAST curve is in between the other two. After a cutoff of about twenty, it is however crossed by the extended Passta curve, where the candidate sets contained all "Close" targets, i.e. at 100% sensitivity.

The diagram on the right side of Figure 44 compares the "Distant" results obtained by the three methods. The order of the three curves is basically the same as in the diagram on the left: The Jali curve is always above the one of BLAST, which itself is clearly above the one of Passta. The rise of Passtas "extended" curve is in this case however very steep, reaching the BLAST curve at a cutoff of about five and the Jali curve at a cutoff of about 15. At cutoff 100, the extended Passta curve is reaching

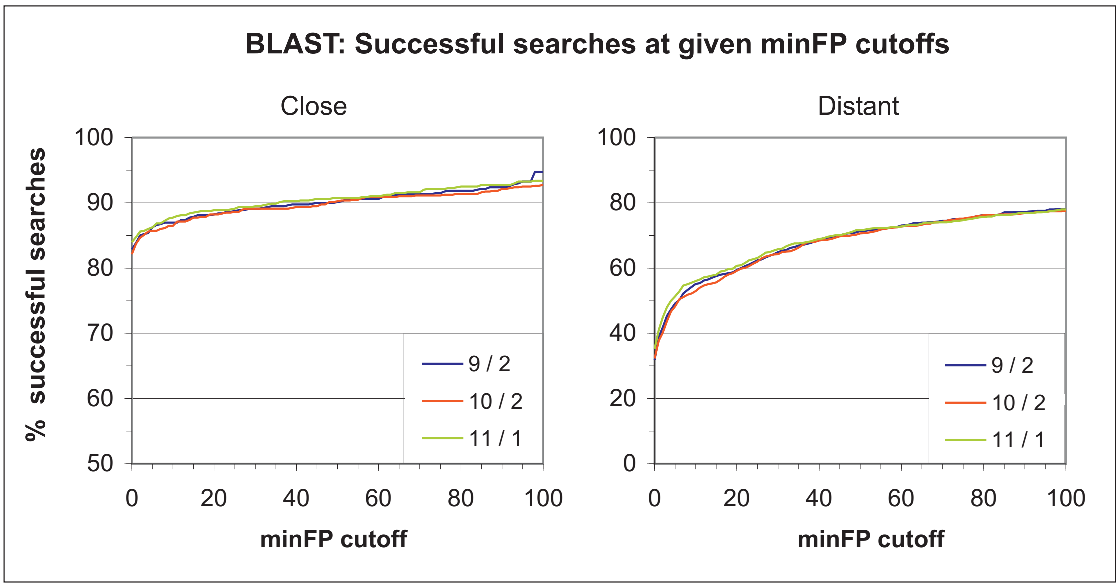


Figure 43: *Minimal false positive counts for BLAST. Shown is the percentage of successful searches at given minFP cutoffs for three different gapcost combinations. The results are good, though not as good as those of Jali.*

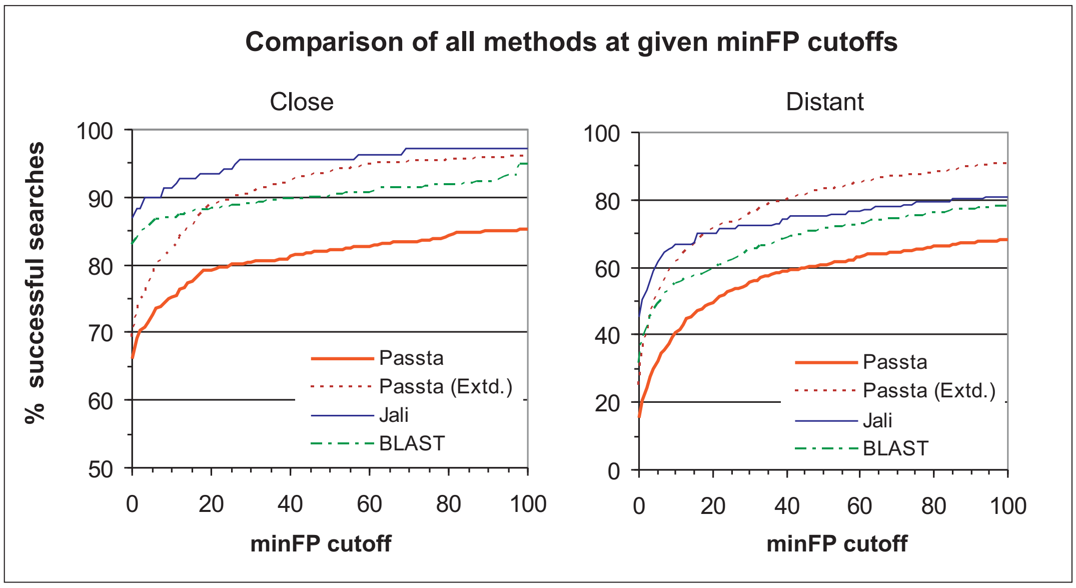


Figure 44: *Minimal false positive counts of all three methods. Shown is the percentage of successful searches at given minFP cutoffs. The "Close" results are on the left, the "Distant results on the right side of the Figure. The results are discussed in the main text.*

90% successful cases.

The results are easy to interpret. Jali reconfirmed that the additional information in the multiple alignments is necessary to improve the quality in sequence database searching, not only in remote homology recognition, but also in the detection of closely related sequences. Regarding Passta, we have to admit that it is unable to compete with BLAST and Jali in its current setup. The main reason is the suboptimal first phase, Pass One. Even though Pass One is not a complete failure, it discards too many of the related targets and too few of the unrelated ones. Nevertheless, the extended results of Pass Two show the potential of the method. If Pass One worked better, Passta should be able to compete with BLAST at least.

Summary and Outlook

More than 30 years have passed since the first protocols for DNA-sequencing were published. The continous improvement of these protocols and their automatization lead to more and more known gene sequences of many different organisms. The need to analyze and compare these sequences fostered the advent of sequence analysis, a suptopic within bioinformatics.
One of the key concepts in sequence analysis are alignments of biological sequences. They present conserved and variable positions of the aligned sequences in an easily accessible way and can answer many scientific questions, especially in the analysis of evolutionarily related or homologous sequences. Alignments are also a key technique in homology recognition. Here, a sequence database is searched for homologous sequences. While most closely related sequences can be found by modern approaches, remotely related sequences are much harder to find.
The Jumping Alignment algorithm (Jali) is a recently proposed method for database searching. Its input is a sequence database and a multiple alignment of related sequences (a sequence family). Jali compares each sequence in the database to the multiple alignment, thereby considering the information in the rows of the multiple alignment. It may however *jump* to another row of the multiple alignment and use the information in the alignment colums as well.
Initial evaluations of Jali showed promising results. In this thesis we asked whether these good results can be reproduced and investigated the reason. We analysed a set of jumping alignments with underlying secondary structure information for cases of "structural significance", where the Jali algorithm seemed to recognize the secondary structure in the alignment. Even though we found a few examples, these cases are rather rare. In a second experiment, we simulated the evolution of ten protein superfamilies and evaluated Jali in close and remote homology recognition on these families. We monitored several parameters of the Jali algorithm during these experiments. The results support our assumption that its jumping ability offers Jali more flexibility in homology recognition, especially in conjunction with suboptimal alignments.
Nevertheless, Jali is not as flexible as it could be. In addition, it shares certain limitations with many traditional, alignment-based approaches in sequence database searching. They have to process the input sequences letter by letter, and they also have to use each letter exactly once. This aggravates finding related sequences with duplications or rearrangements. Besides, the output of these approaches is usually a list of hits sorted by an alignment score or an e-value. This list does however not conveniently show which part of the query aligns best with what target in the database.
We developed a new method that circumvents these limitations, called *Passta*. It tries to "explain" a query sequence with fragments of related sequences in the best possible way. The fragments we use are Secondary Structure Elements (SSEs) from the Protein Data Bank (PDB). Passta uses a two-staged protocol: The first stage (Pass One) serves as a filter. It computes optimal, pairwise alignments of the target SSEs

from the database and the query. These alignments are used to select a set of target domains, the candidate set. This candidate set is submitted to the second stage of Passta, Pass Two. Here, we compute also co- and suboptimal alignments between the SSEs and the query. These alignments are represented in a directed acyclic graph. We compute an optimal path in the graph, which corresponds to a selection of alignments. This path may contain rearrangements, duplications, and deletions of one or more SSEs.
Prior to evaluating Passta, we trained some of its parameters for better performance. When we chose the Blosum62 threshold of Pass One, we discovered that the three SSE classes produce different numbers of valid alignments. We could partly explain these differences with the biological properties of the SSEs. In a second project, we proposed a way to calibrate the rearrangement cost parameter of Pass Two. This parameter is also important in genome alignment. During the calibration, we identified many SCOP families with duplications or circular permutations, a special kind of rearrangement. We could show that Passta is able to detect well-conserved rearrangements and duplications.
Finally, we evaluated Passta together with Jali and BLAST in homology recognition. The evaluation unfortunately revealed that the current design of Pass One has to be improved. It discards too many of the related or too few of the unrelated targets. One of the reasons is that we align short SSEs. Many such alignments are not specific. We were however able to show the potential of Pass Two: When we added the true positive targets missed by Pass One to the candidate set, Pass Two was able to compete with Jali and BLAST. It even outperformed them at minimal false positive cutoffs over 15.

Our experiments answered some, but not all questions related to Passta. Some features worked very well, while others did not meet our expectations. We present a few ideas to alleviate the shortcomings and to improve the strengths of Passta.

One of the main advantages of Passta is its ability to cope with rearrangements and duplications. Since we want to retain this ability, Passta will continue to work with sequence fragments. Using SSEs as fragments is still a good choice, even though Passta can presently only access a part of their biological information content. This information could be used to find optimal paths being more feasible in terms of biology. For example, the hydrophobicity information available in the PDBFinder II database could be stored for each SSE. Then, alignments of hydrophobic SSEs with polar SSEs could receive an extra penalty. We also propose to penalize alignments of α-helices adjacent to alignments of β-strands, if an aligned residue with helical secondary structure conformation is immediately followed by a residue with β-strand conformation. This is *in vivo* because of steric restrictions impossible [34].
Passta uses secondary structure specific gap costs when computing the SSE alignments. Even though some of the structural information is considered in this way, we believe that the signal-to-noise ratio could be improved with a set of modern, secondary structure specific substitution matrices.

Even if we had secondary structure specific substitution matrices, the information content in pairwise alignments is in general rather low. It would be nice to use the information of multiple sequences, as in Jali. We propose to compute *SSE-PSSMs* to improve Pass One. PSSMs[18] are profiles of related sequences that are often used in sequence database searching to facilitate the detection of remotely related sequences, see e.g. [71]. The sequences for our PSSMs would be syntenic SSEs of all protein domains within a SCOP family. A multiple alignment had to be constructed from these SSEs, which is then used to compute a SSE-PSSM. Comparing the PSSM to the query sequence is computationally cheap, and high-scoring random similarities between a PSSM and the query are less likely than with single SSE sequences.
Another option to improve Pass One is to compute e-values for each SSE. Most of the small SSEs will however be unable to produce significant alignments at all. Even though noise would not anymore be a problem for Pass One, it would also lose a large amount of information.

The last feature we suggest is an "interactive mode". This mode could either be realized in a graphical user interface or in dynamic HTML. The purpose of the interactive mode would allow the researcher to further analyze the Pass Two results. Interesting targets could be kept in the candidate set, while uninteresting targets could be dropped, e.g. by selecting or deselecting check boxes. The parameters for jump-, rearrangement-, and deletion cost could be changed, too. Then Pass Two could be repeated to gain new insights on the investigated query.

[18]Position-specific scoring matrices

Appendix

First implementation of Pass One

The domain-based filtering in Pass One (see Section 12.2) was preceded by an implementation that was based on a simplified version of the DAG used in Pass Two (see Sections 12.1 and 12.3). We shortly describe the procedure along with the associated problems and use the DAG definitions provided on page 54.

Given the query R, our first Pass One implementation started by selecting all non-redundant SSEs of length 6 or more from PasstaDB. Then it used the Waterman-Eggert algorithm [108] to compute pairwise, local, non-intersecting alignments of R and each SSE. The Waterman-Eggert algorithm first computes the optimal, i.e. highest scoring alignments, then co- and suboptimal alignments. These alignments were computed until the MaxScoreRatio dropped below a constant value.
As in Pass Two, the alignments were represented as the vertices in a DAG G. Each vertex was associated with a 5-tuple of information, $\alpha = (b, e, s, t, p)$. The edges in G were defined by the basic edge definition provided on page 54.

Then we computed all optimal paths (i.e. with maximal $weight(P)$) in G, however without considering rearrangements, jumps, or even deletions. If there was more than one optimal path, we arbitrarily selected one and extracted all target IDs $\alpha(t)$ of its vertices. These target IDs could have made up the candidate set for Pass Two, but we had used only *non-redundant* SSEs to keep the time demands of the Waterman-Eggert algorithm reasonable. Since redundant SSEs may also exist in other targets in the database, we identified those and collected their target IDs as well. The target IDs in the optimal path and the IDs of corresponding redundant SSEs made up our candidate set for Pass Two.

The first Pass One implementation was designed in an early stage of the Passta project. Basically, it was a simplified byproduct of Pass Two. Even though a few initial tests were promising, certain properties of the graph-based approach are disadvantegous. These properties are enumerated below.

1. *Valid singleton alignments are always in the optimal path:* The optimal path computed during Pass One does always contain all alignments that are not overlapped by other alignments, regardless of the alignment score. Hence, corresponding targets will be in the Pass Two candidate set. If there are redundant entries of the aligned SSE in the database, their target IDs are added to the candidate set, exaggerating the problem.

2. *Long target sequences are preferred candidates:* High-scoring alignments are likely to be in the optimal path. Even a short, local alignment of one long SSE sequence can easily remain in the optimal path. The probability to produce a high scoring alignment is increasing with the length of the target sequence.

3. *Multiple hits are not considered:* A related target may not be in the candidate set, even though several of its SSEs produce valid alignments. If none of its SSEs

aligns optimal with the query at any b, e tuple, overlapping high-scoring alignments may be preferred in the optimal path, even if they come from unrelated targets.

List of Figures

1 Tree of Life 3
2 Molecular structure of amino acids 4
3 Two amino acids forming a peptide bond 5
4 Antiparallel β-sheet 7
5 Pairwise alignment example 9
6 Three score functions used in pairwise alignment 10
7 Edit matrix in global alignment 12
8 Selected records of PDB entry "1EFT" 22
9 Hierarchy in the SCOP classification 24
10 "Vertical information" in a multiple alignment 27
11 "Horizontal information" in a multiple alignment 28
12 Multiple alignment of two subfamilies 28
13 Edit matrix of the Jumping Alignment algorithm 29
14 Phase4 evaluation setup, step 1 31
15 Phase4 evaluation setup, step 2 32
16 Phase4 evaluation setup, step 3 32
17 Jumping alignment with "structural significance" 33
18 *Intra* and *inter* relationships in an example superfamily 35
19 Creation of the artificial superfamilies 36
20 Results of the biological setting 37
21 Distances between the astral40 and astral70 SCOP subsets in sequence space 38
22 Jumping alignment showing a questionable classification in SCOP 1.55 39
23 Results of the "random200" setting at different jumpcosts 40
24 Comparison of the "unstructured" to the "random200" results 41
25 Influence of the alignment quality on the "random200" results 42
26 Classic approaches in database searching are not well suited to align sequences with rearrangements 45
27 Schematic overview over Passta 46
28 Schematic view of the "PasstaDB" database 50
29 Example of a Pass One candidate matrix 58
30 The "align & extend" strategy in Pass Two 61
31 Example graph illustrating the two edge definitions used in Pass Two . 62
32 Excerpt of a secondary structure profile from Passta 65
33 Pass One training: Percentage of valid alignments at three Blosum62 thresholds 67
34 Pass One training: Alignment length distribution 69

35 Pass One training: SSE lengths in valid alignments 71
36 Pass One training: Influence of the MSR threshold 72
37 Calibration of rearrangement and deletion costs: experimental setup . 77
38 Calibration of rearrangement and deletion costs: deletion cost inluence on the rearrangement cost 78
39 Calibration of rearrangement and deletion costs: Passta alignment with a circular permutation 81
40 Comparative Evaluation: the homology relationships in our experimental setup 86
41 Comparative Evaluation: Pass Two minFP counts 91
42 Comparative evaluation: minimal false positive counts for Jali 94
43 Comparative evaluation: minimal false positive counts for BLAST . . . 95
44 Comparative evaluation: minimal false positive counts for all three methods 95

List of Tables

1 Selected records of PDBFinder II entry “101M” 22
2 Description of selected tables and fields in PasstaDB. 51
3 PDB entry “1e4r” split into SSEs, with associated information 57
4 Pass One training: Effect of secondary structure specific gapcosts . . . 70
5 Pass One training: Effect of secondary structure specific gapcosts on the number of alignments with gaps 71
6 Pass One training: Influence of the MSR threshold 73
7 Calibration of rearrangement and deletion costs: twenty SCOP families with high collinearity thresholds 80
8 Calibration of rearrangement and deletion costs: Statistical data for families with circular permutations 82
9 Evaluation parameters of Pass One 87
10 Comparative evaluation: Native and filtered Pass One results 90
11 Comparative evaluation: Pass Two “Full” results, first part 92
12 Comparative evaluation: Pass Two “Full” results, second part 93

Acknowledgements

First off, I would like to thank my supervisor Prof. Dr. Jens Stoye. When I started at the Max-Planck Institute for Molecular Genetics in mid 2001, it was probably some effort to explain the alignment basics to me (I was much more biologist than computer scientist at that time). When I finished with the Jumping alignment project in early 2004, he allowed me to develop Passta. It was clear that sequence database searching with small fragments is not easy, his decision was therefore not self-evident. Besides, I would like to thank him for his support and patience while I prepared this thesis.
Particular thanks go to two members of our Genome Informatics (GI) group at Bielefeld University, Hans-Michael Kaltenbach (Mitch) and Klaus-Bernd Schürmann (Klaus). They were not just colleagues, but friends. I remember many fruitful discussions, coffee breaks, and also the nice "retreat" of the GI group in 2007.
The Genome Informatics group was a great environment to work in. While one sometimes hears about mistrust or envy in other scientific groups, I have never seen any of these problems while being in the group (which was a long time!). That also held for our junior- and more or less associated groups.
External thanks go to Patrick May from the Zuse Institute, Berlin (ZIB). He helped me to understand some of the problems associated with PDB entries. Sergio A. de Carvalho and Mitch did some valuable proofreading for me. When I finished this thesis, Peter Husemann was so kind to offer a place to sleep and a pizza to eat.
Life is not entirely dedicated to science. I remember some nice partys, a Salsa dancing class experience ("Are you a mathematician?") and attending the "Viehtreiben" (a daily fitness class at Bielefeld University) with Sven Rahmann, Veli Mäkinen, Julia Zakotnik, and others. Last not least I would like to thank my mother Isolde Bannert for her support and Nicole Treude for her sheer existence.
I am closing with a piece of personal communication. When I complained about the PDB format for the second or third time, Dr. Elmar Krieger from the PDBFinder II team wrote:

> *"I can safely tell you that I've been working on a PDB parser for seven years now, and it's still not finished [...]."*

References

[1] M. I. Abouelhoda and E. Ohlebusch. Chaining algorithms for multiple genome comparison. *J. Discr. Alg.*, 3:321–341, 2005.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell, 4th Ed.* Garland Science, 2002.

[3] V. Alesker, R. Nussinov, and H. J. Wolfson. Detection of non-topological motifs in protein structures. *Protein Eng.*, 9:1103–1119, 1996.

[4] S. F. Altschul and W. Gish. Local alignment statistics. *Meth. Enzymol.*, 266:460–480, 1996.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[6] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Ac. Res.*, 25:3389–3402, 1997.

[7] Y. An and R. A. Friesner. A novel fold recognition method using composite predicted secondary structures. *Proteins*, 48:352–366, 2002.

[8] M. A. Andrade, C. Perez-Iratxeta, and C. P. Ponting. Protein repeats: Structures, functions, and evolution. *J. Struct. Biol.*, 134:117–131, 2001.

[9] A. Andreeva, D. Howorth, S. E. Brenner, T. J. Hubbard, C. Chothia, and A. G. Murzin. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucl. Ac. Res.*, 32:D226–D229, 2004.

[10] A. Andreeva, D. Howorth, J.-M. Chandonia, S. E. Brenner, Hubbard T. J., C. Chothia, and A. G. Murzin. Data growth and its impact on the SCOP database: new developments. *Nucl. Ac. Res.*, 36:D419–D426, 2007.

[11] Z. Aung and K. L. Tan. Rapid 3D protein structure database searching using information retrieval techniques. *Bioinformatics*, 20:1045–1052, 2004.

[12] C. Bannert. Systematic investigation of jumping alignments. *Technical Report*, 2003-05, 2003.

[13] C. Bannert and J. Stoye. Evaluation of the jumping alignment algorithm with artificial and biological data. In *Proc. German Conference on Bioinformatics (GCB 2003)*, pages 21–25, 2003.

[14] C. Bannert and J. Stoye. Protein annotation by secondary structure based alignments (Passta). In *LNBI*, volume 3695, pages 79–90, 2005.

[15] R. Bellman. On the theory of dynamic programming. *Proc. Natl. Acad. Sci. USA*, 38:716–719, 1952.

[16] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucl. Ac. Res.*, 28:235–242, 2000.

[17] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[18] P. E. Bourne and H. Weissig. *Structural Bioinformatics*. Wiley Liss, 2003.

[19] S. E. Brenner, C. Chothia, and T. J. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. USA*, 95:6073–6078, 1998.

[20] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucl. Ac. Res.*, 28:254–256, 2000.

[21] M. Brudno, M. Chapman, B. Göttgens, S. Batzoglou, and B. Morgenstern. Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, 4:66–77, 2003.

[22] H. Carillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48:1073–1082, 1988.

[23] D. M. Carrington, A. Auffret, and D. E. Hanke. Polypeptide ligation occurs during post-translational modification of concanavalin A. *Nature*, 313:64–67, 1985.

[24] J. M. Chandonia, G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. E. Brenner. The ASTRAL compendium in 2004. *Nucl. Ac. Res.*, 32:D189–D192, 2004.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Ed.* MIT Press / McGraw-Hill, 2001.

[26] M. M. Cox and J. R. Battista. *Deinococcus radiodurans* - the consummate survivor. *Nat. Rev. Microbiol.*, 3:882–892, 2005.

[27] A. C. Darling, B. Mau, F. R. Blattner, and N. T. Perna. Mauve: Multiple alignment of conserved genomic sequence with rearrangements. *Genome Res.*, 14:1394–1403, 2004.

[28] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Science and Structure*, 5:345–352, 1978.

[29] W. Deng, V. Burland, G. Plunkett 3rd, A. Boutin, G. F. Mayhew, P. Liss, N. T. Perna, D. J. Rose, B. Mau, S. Zhou, D. C. Schwartz, J. D. Fetherston, L. E. Lindler, R. R. Brubaker, G. V. Plano, S. C. Straley, K. A. McDonough, M. L.

Nilles, J. S. Matson, F. R. Blattner, and Perry R. D. Genome sequence of *Yersinia pestis* KIM. *J. Bacteriol.*, 184:4601–4611, 2002.

[30] C. Dodge, R. Schneider, and C. Sander. The HSSP database of protein structure-sequence alignments and family profiles. *Nucl. Ac. Res.*, 26:313–315, 1998.

[31] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[32] R. D. Finn, J. Mistry, B. Schuster-Böckler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. Sonnhammer, and A. Bateman. Pfam: clans, web tools and services. *Nucl. Ac. Res.*, 34:D247–D251, 2006.

[33] W. M. Fitch. Homology - a personal view on some of the problems. *Trends Genet.*, 16:227–231, 2000.

[34] N. C. Fitzkee and G. D. Rose. Steric restrictions in protein folding: An α-helix cannot be followed by a contigous β-strand. *Prot. Sci*, 5:633–639, 2004.

[35] A. Fliess, B. Motro, and R. Unger. Swaps in protein sequences. *Proteins*, 48:377–387, 2002.

[36] D. Frishman and P. Argos. Knowledge-based protein secondary structure assignment. *Proteins*, 23:566–579, 1995.

[37] A. J. Gibbs and G. A. McIntyre. The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *Eur. J. Biochem.*, 16:1–11, 1970.

[38] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[39] R. E. Green and S. E. Brenner. Bootstrapping and normalization for enhanced evaluations of pairwise sequence comparison. *Proc. IEEE*, 90:1834–1847, 2002.

[40] A. J. Griffiths, J. H. Miller, D. T. Suzuki, R. C. Lewontin, and W. M. Gelbart. *An Introduction to Genetic Analysis (6th ed.)*. W.H. Freeman and Company, New York, 1996.

[41] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12:312–326, 1994.

[42] B. Hao, W. Gong, T. K. Ferguson, C. M. James, J. A. Krzycki, and M. K. Chan. A new UAG-encoded residue in the structure of a methanogen methyltransferase. *Science*, 296:1462–1466, 2002.

[43] S. Henikoff. Scores for sequence searches and alignments. *Curr. Opi. Struct. Biol.*, 6:353–359, 1996.

[44] S. Henikoff and G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89:10915–10919, 1992.

[45] J. Heringa. Computational methods for protein secondary structure prediction using multiple sequence alignments. *Curr. Protein Pept. Sci.*, 1:273–301, 2000.

[46] R. W. Hooft, C. Sander, and G. Vriend. The PDBFINDER database: A summary of PDB, DSSP and HSSP information with added value. *CABIOS*, 12:525–529, 1996.

[47] J.-T. Huang and M.-T. Wang. Secondary structural wobble: the limits of protein prediction accuracy. *Biochem. Biophys. Res. Commun.*, 294:621–625, 2002.

[48] X. Huang and W. Miller. A time-efficient, linear-space local similarity algorithm. *Adv. Appl. Math.*, 12:337–357, 1991.

[49] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.*, 7:95–114, 2000.

[50] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[51] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci.*, 87:2264–2268, 1990.

[52] K. Karplus, C. Barrett, and R. Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998.

[53] L. A. Kelley. Enhanced genome annotation using structural profiles in the program 3D-PSSM. *J. Mol. Biol.*, 299:499–520, 2000.

[54] W. J. Kent. BLAT – the BLAST-like alignment tool. *Genome Res.*, 12:656–664, 2002.

[55] C. Kimchi-Sarfaty, J. Mi Oh, I.-W. Kim, Z. E. Sauna, A. M. Calcagno, S. V. Ambudkar, and M. M. Gottesman. A "silent" polymorphism in the MDR1 gene changes substrate specifity. *Science*, 315:525–528, 2007.

[56] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *J. Comput. Biol.*, 3:289–306, 1996.

[57] K. K. Koretke, R. B. Russell, R. R. Copley, and A. N. Lupas. Fold recognition using sequence and secondary structure information. *Proteins.*, S3:141–148, 1999.

[58] A. Krause and M. Vingron. A set-theoretic approach to database searching and clustering. *Bioinformatics*, 14:430–438, 1998.

[59] Justina Krawczyk. Analyse von Sekundärstrukturelementen in Bezug auf ihre Konformation (in German). Bachelor thesis, Technische Fakultät, Universität Bielefeld, Germany, September 2004.

[60] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, Shumway M., C. Antonescu, and S. L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biol.*, 5:R 12, 2004.

[61] I. Letunic, R. R. Copley, B. Pils, S. Pinkert, J. Schultz, and P. Bork. SMART 5: Domains in the context of genomes and networks. *Nucl. Ac. Res.*, 34:D257–D260, 2006.

[62] R. A. Lippert, X. Zhao, L. Florea, C. Mobarry, and S. Istrail. Finding anchors for genomic sequence comparison. *Proc. RECOMB'04*, pages 233–241, 2004.

[63] S. T. Lovett. Resurrecting a broken genome. *Nature*, 443:517–519, 2006.

[64] M. T. Madigan, J. M. Martinko, and J. Parker. *Brock Biology of Microorganisms, 8th ed.* Prentice Hall, Inc., 1997.

[65] M. Margulies *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:376–380, 2005.

[66] A. C. Martin. The ups and downs of protein topology: rapid comparison of protein structure. *Protein Eng.*, 13:829–837, 2000.

[67] A. M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proc. Natl. Acad. Sci.*, 74:560–564, 1977.

[68] P. May, S. Barthel, and I. Koch. PTGL - a web-based database application for protein topologies. *Bioinformatics*, 20:3277–3279, 2004.

[69] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Prot. Sci.*, 7:2469–2471, 1998.

[70] B. Morgenstern, A. Dress, and T. Werner. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci.*, 93:12098–12103, 1996.

[71] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis (2nd Ed.)*. Cold Spring Harbor, 2004.

[72] T. Müller, R. Spang, and M. Vingron. Estimating amino acid substitution models: A comparison of dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.*, 19:8–13, 2002.

[73] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

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

[75] H. D. Niall. Automated Edman degradation: the protein sequenator. *Methods Enzymol.*, 27:942–1010, 1973.

[76] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. CATH - a hierarchic classification of protein domain structures. *Structure*, 5:1039–1108, 1997.

[77] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. volume 183, pages 63–98. 1990.

[78] W. R. Pearson. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11:635–650, 1991.

[79] C. P. Ponting and R. B. Russell. Swaposins: Circular permutations within genes encoding saposin homologues. *Trends Biochem. Sci.*, 20:179–180, 1995.

[80] K. R. Rasmussen, J. Stoye, and E. W. Myers. Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.*, 13:296–308, 2006.

[81] M Rehmsmeier. Phase4: Automatic evaluation of database search methods. *Brief. Bioinform.*, 3:342–352, 2002.

[82] M Rehmsmeier and M. Vingron. Phylogenetic information improves homology detection. *Proteins*, 45:360–371, 2001.

[83] C. A. Rohl, C. E. Strauss, K. M. Misura, and D. Baker. Protein structure prediction using Rosetta. *Meth. Enzymol.*, 383:66–93, 2004.

[84] B. Rost. Twilight zone of protein sequence alignments. *Prot. Eng.*, 12:85–94, 1999.

[85] R. B. Russell, R. R. Copley, and G. J. Barton. Protein fold recognition by mapping predicted secondary structures. *J. Mol. Biol.*, 259:349–365, 1996.

[86] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[87] M. Sammeth and J. Heringa. Global multiple alignment with repeats. *Proteins*, 64:263–274, 2006.

[88] F. Sanger. The terminal peptides of insulin. *Biochem. J.*, 45:563–574, 1949.

[89] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci.*, 74:5463–5467, 1977.

[90] R. Sayle and E. Milner-White. Rasmol: biomolecular graphics for all. *Trends Biochem. Sci.*, 20:374–376, 1995.

[91] J. Schultz, F. Milpetz, P. Bork, and C. P. Ponting. SMART, a simple modular architecture research tool: Identification of signalling domains. *Proc. Natl. Acad. Sci.*, 95:5857–5864, 1998.

[92] F. Servant, C. Bru, E. Courcelle, J. Gouzy, D. Peyruc, and D. Kahn. ProDom: automated clustering of homologous domains. *Brief. Bioinform.*, 3:246–251, 2002.

[93] J. Shi, T. L. Blundell, and K. Mizuguchi. FUGUE: Sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.*, 310:243–257, 2001.

[94] E. Shih and M.-J. Hwang. Alternative alignments from comparison of protein structures. *Proteins*, 56:519–527, 2004.

[95] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *J. Mol. Biol.*, 268:209–225, 1997.

[96] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[97] R. Spang, M. Rehmsmeier, and J. Stoye. A novel approach to remote homology detection: Jumping alignments. *J. Comp. Biol.*, 9:747–760, 2002.

[98] R. Spang and M. Vingron. Statistics of large scale sequence searching. *Bioinfomatics*, 14:279–284, 1998.

[99] R. Spang and M. Vingron. Limits of homology detection by pairwise sequence comparison. *Bioinfomatics*, 17:338–342, 2001.

[100] J. Stoye. Multiple sequence alignment with the Divide-and-Conquer method. *Gene*, 211:GC45–GC56, 1998.

[101] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14:157–163, 1998.

[102] A. R. Subramanian, M. Kaufmann, and B. Morgenstern. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecuar Biology*, 3:6–17, 2008.

[103] C. Tanford. The hydrophobic effect and the organization of living matter. *Science*, 200:1012–8, 1978.

[104] W. R. Taylor. Dynamic sequence databank searching with templates and multiple alignment. *J. Mol. Biol.*, 280:375–406, 1998.

[105] J. D. Thompson, D. J. Higgins, and T. J. Gibson. CLUSTALW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Ac. Res.*, 22:4673–4680, 1994.

[106] M. Vingron. Near-optimal alignment. *Curr. Opi. Struct. Biol.*, 6:346–352, 1996.

[107] M. S. Waterman. Mathematical analysis of molecular sequences. *Bull. Math. Biol.*, 51:1–194, 1989.

[108] M. S. Waterman and M. Eggert. A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.*, 197:723–728, 1987.

[109] J. Weiner 3rd, G. Thomas, and E. Bornberg-Bauer. Rapid motif-based prediction of circular permutations in multi-domain proteins. *Bioinformatics*, 21:932–937, 2005.

[110] J. D. Westbrook and P. E. Bourne. STAR/mmCIF: An ontology for macromolecular structure. *Bioinformatics*, 16:159–168, 2000.

[111] G. Witte, C. Urbanke, and U. Curth. Single-stranded DNA-binding protein of *Deinococcus radiodurans*: a biophysical characterization. *Nucl. Ac. Res.*, 33:1662–1670, 2005.

[112] Y. Ye, L. Jaroszewski, W. Li, and A. Godzik. A segment alignment approach to protein comparison. *Bioinformatics*, 19:742–749, 2003.

[113] F. Zinoni, A. Birkmann, T. C. Stadtman, and A. Bock. Nucleotide sequence and expression of the selenocysteine-containing polypeptide of formate dehydrogenase (formate-hydrogen-lyase-linked) from *Escherichia coli*. *Proc. Natl. Acad. Sci.*, 83:4650–4654, 1986.