Exploring Early Stages of Protein Evolution

DISSERTATION

zur Erlangung des Doktorgrades der Naturwissenschaften (Dr. rer. nat.) der Fakultät für Biologie und Vorklinische Medizin der Universität Regensburg

vorgelegt von

Leonhard Josef Heizinger

aus Landshut

Juni 2020

Das Promotionsgesuch wurde eingereicht am:
12.06.2020

Die Arbeit wurde angeleitet von:
Prof. Dr. Rainer Merkl

Unterschrift:

Leonhard Josef Heizinger

Abstract

A comparison of protein backbones makes clear that the number of contemporary protein architectures is limited: Not more than approximately 1400 different folds exist, each specifying the architecture of a protein domain. In agreement with the frequent reuse observed in nature, many folds can accommodate different functions and large proteins are composed of specific domain combinations. These findings make clear that gene duplication and fusion are fundamental forces in evolution. In contrast to the evolution of homologs that share the same fold but possess substantially different functions, the origin of the architecture of individual domains is unclear. If reuse was also the driving force of domain evolution, it should be possible to identify a set of ancestral fragments of sub-domain size that are shared between domains having significantly different architectures. For a fully automated detection of putative ancestral fragments, the algorithm FragStatt was developed. It assesses proteins pairwise in order to find shared fragments. To identify fragments that are homologous, FragStatt compares sequences and to reach maximum sensitivity, HMM searches are cascaded. Subsequently, the program determines and scores the similarity of the fragments' 3D structures. A comprehensive large-scale comparison of proteins from the CATH database yielded 12,533 partially overlapping and structurally similar motifs that clustered to 134 unique motifs. These motifs are concentrated on not more than 18 % of the CATH topologies and at most one topology exists that contains more than one motif. These findings suggest that the reuse of sub-domain sized fragments was not the dominating force of protein evolution.

Contents

List of Figures

List of Tables

Abbreviations

CATH	Protein Structure Classification database
CoMo	Set of putative ancient motifs identified with FragStatt
CoMoClust	Clustered variant of CoMo
DFS	Depth first search
f	Statistical absolute frequency
rf	Statistical relative frequency
FASTA	Text-based sequence file format
Fuzzle	Database of putative ancient motifs (Ferruz et al., 2020)
Gya	Billion years ago
HMM	Profile hidden Markov model
HRR	HHsuite result file extension
MSA	Multiple sequence alignment
PDB	Protein Data Bank
PKL	Format for binary storage of Python objects
Pfam	Protein families database
ProVoc	Set of putative ancient motifs (Alva et al., 2015b)
RMSD	Root-mean-square deviation
SCOP	Structural Classification of Proteins database

Chapter 1

Introduction

1.1 The Origin of Life

Proteins are the main actors in all known life forms. In a functioning cell, a broad range of different biochemical tasks need to be fulfilled, facilitated by a vast variety of different and highly specialized proteins: For example enzymes are optimized to catalyze biochemical reactions with high specificity, structural proteins maintain the shape of a cell, and signaling proteins transduce external stimuli into the cell. The origin of the complex molecular protein machinery which makes up large parts of an organism goes back billions of years, to an era when the first proteins evolved.

Life as we know it on Earth is assumed to have emerged over 3.6 billion years ago (Doolittle, 2000; Glansdorff et al., 2008; Betts et al., 2018). The existence of a simple primordial form of life named *last universal common ancestor* (LUCA) in this era is generally accepted (Darwin, 1859; Woese, 1998; Theobald, 2010) (compare Figure 1.1). Interestingly, the reconstruction of ancestral enzymes and the modeling of "minimal genomes" indicate that in its basics LUCA resembled modern-day organisms and already possessed a relatively complex enzyme-facilitated metabolism (Koonin, 2003; Weiss et al., 2016). This leads to the assumption that LUCA must have already had a repertoire of efficient enzymes and enzyme complexes similar to those we can observe in all organisms today. Consequently, this suggests that the evolutionary origin of many proteins goes back within the pre-LUCA era. The emergence of life in this era is still

unresolved. However, it is a widely accepted hypothesis that RNA played an important role in the earliest stages of abiogenesis and that RNAs could have been the earliest precursors of life.

The so-called RNA world hypothesis (Gilbert, 1986) founds on the unique properties of RNA which make it a promising candidate for the earliest precursor of life: It can store information similar to DNA and it can be enzymatically active in the form of a ribozyme. Furthermore there is increasing evidence suggesting the ability of RNA to self-replicate (Zaher and Unrau, 2007; Horning and Joyce, 2016). However, over time the enzymatic functions performed by RNA in the RNA world must have gradually been transferred to much more effective and specialized proteins which led to the first protein and DNA based life as we know it today. Little is known about this important stage in the emergence of life in which supposedly the first folded proteins evolved.

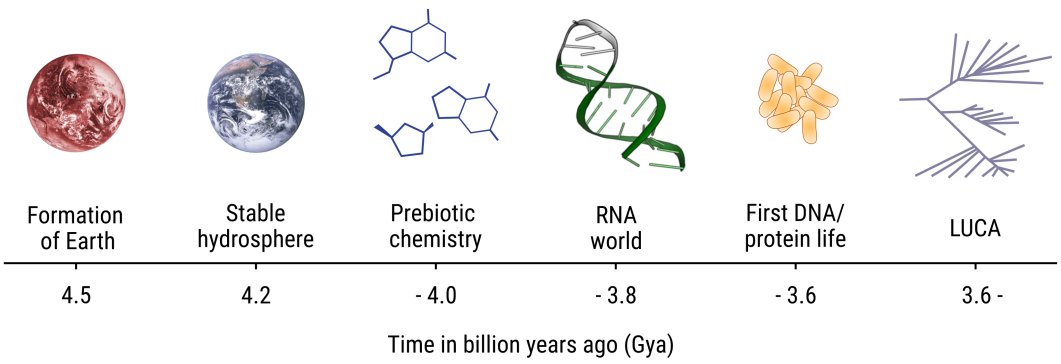


Figure 1.1: Timeline of the early history of life on Earth.
The earth formed at approximately 4.5 Gya and some 300 million years later a stable hydrosphere developed. The fundamental chemical components for the emergence of life were supposedly present at 4 Gya. The earliest RNA based precursors of life are thought to have emerged at approximately 3.8 Gya. It is assumed that the first DNA and protein based life and finally LUCA emerged in a relatively short time span of only a few hundred million years. Note that this timeline is based on rough estimates. New insights and evidence might change these estimates drastically. Adapted by kind permission from Springer Nature Customer Service: The antiquity of RNA-based evolution, Joyce (2002)

Over 50 years ago, before the idea of the RNA world emerged, the pioneering bioinformatican Margret Dayhoff together with here colleague Richard Eck were the first ones to propose the idea that proteins are built from small reoccurring motifs. Dayhoff and Eck deduced their hypothesis from the occurrence of repeating sequence motifs in the few protein sequences which were known at that time (Eck and Dayhoff, 1966a,b). With the rise of the widely accepted

RNA world hypothesis Dayhoff and Eck's observations led to the proposal that the first folded proteins arose by repetition, fusion, recombination, and accretion from a set of ancestral protein fragments which emerged in the RNA world (Fetrow and Godzik, 1998; Söding and Lupas, 2003; Alva et al., 2015b).

In the 1960s Dayhoff and Eck based their hypothesis on internal symmetries they discovered in the sequence of ferredoxin proteins. Since then in the fields of biochemistry, biophysics, and bioinformatics many revolutionary discoveries have been made and game-changing new methods were developed. In this manner, next generation sequencing techniques have led to an enormous amount of known protein sequences, improvements in crystallography made it possible that more than 100,000 protein structures are known, and profile based sequence analysis algorithms allow the identification of faint sequence similarities facilitated by widely available and affordable high performance computing clusters. Today, it is known that proteins can be classified by their basic architecture, called *folds*. Currently approximately 1400 different folds are known (Chothia, 1992; CATH Team, 2020; SCOP Team, 2020). For many of the known folds an internal symmetry can be observed (compare Figure 1.2). For example beta propeller like proteins consist of a symmetrical radial arrangement of multiple so-called blades. A similary striking symmetry can be observed for $(\beta\alpha)_8$ barrel-like proteins.

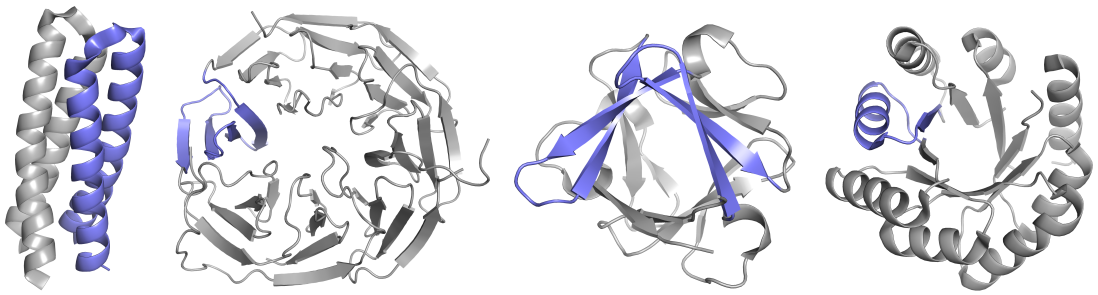


Figure 1.2: Example of symmetrical protein folds.
The four folds updown-bundle (PDB-ID `1F4M`), beta propeller (PDB-ID `3Q7M`), beta trefoil (PDB-ID `1BFG`) and $(\beta\alpha)_8$ barrel (PDB-ID `5BVL`) each show a symmetrical architecture: The folds are composed of a repetition of the same motif (highlighted in blue).

Interestingly, increasing evidence suggest that utilization of reoccurring motifs can not only be detected within a fold but also between distinct folds (Andrade et al., 2001; Höcker et al., 2002; Remmert et al., 2010; Farías-Rico et al., 2014). For the field of protein evolution this is an

interesting finding as in general different folds are considered to be evolutionary unrelated, in a classical sense, i.e. it is assumed that they do not share a common ancestor. Common motifs found in distinct folds could represent remnants from the pre-LUCA era when the first folded proteins emerged. These remote similarities between folds could deliver important insights into the earliest stages of protein evolution and the emergence of life on earth.

The detection and revelation of such unknown and ancient relationships are the subject of this work. In the following sections the required theoretical basics of structural biology and protein evolution will be covered. Subsequently a systematic study aimed at the detection of ancestral protein relationships will be presented.

1.2 Homology

In the most general definition, homology "designates a relationship of common descent between any entities" (Koonin, 2005). Historically the idea of common *homologous* structures in nature is old and was already familiar to ancient Greek philosophers like Aristotle (Hall, 2012). Naturally, in biology the concept of homology plays a fundamental role. Homology of living organisms can be observed on different levels, for example a morphologic homology of bone structures among vertebrates. In this work however, the term homology will be strictly used in the context of protein evolution, i.e. homology on a molecular level.

The following shall introduce the concept of homology. In a genome, proteins are encoded in genes. A hypothetical evolutionary trajectory of a gene "gene A" can be seen in Figure 1.3. Gene duplication is one of the central mechanism which facilitates *divergent evolution* (Koonin, 2005), which is the separate and independent evolution of two genes which both descend from a common ancestor gene. Say a gene A, which is essential for the organism, gets duplicated resulting in the copy gene B. The duplication eases the selective pressure for one duplicate and makes it possible for one copy of the gene to diverge through genetic drift, while the other copy keeps its original function. Subsequently a possible speciation event, i.e. an evolutionary split into distinct species Species 1 and Species 2, results in four different variants of the initial gene A: gene A_1, gene B_1, gene A_2 and gene B_2. These genes (or in the context of this work the

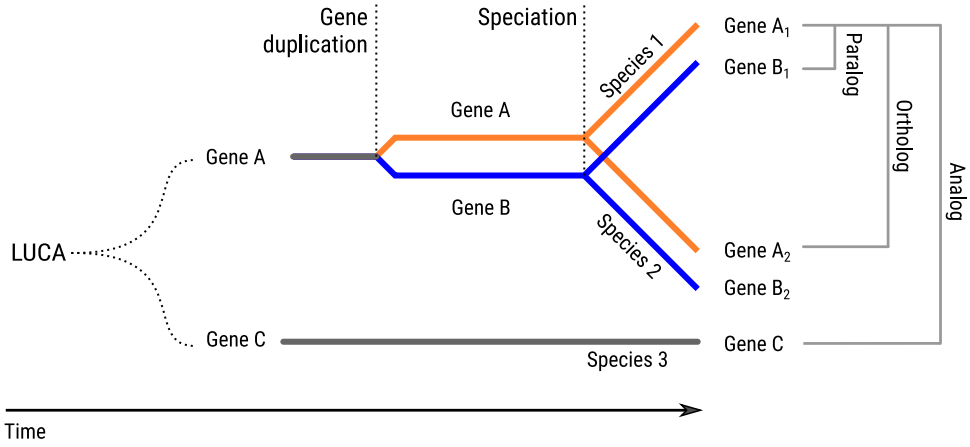


Figure 1.3: Example of the emergence of homologous genes.
Through a gene duplication event a copy of gene A called gene B emerges. Gene A and gene B evolve divergently as selective pressure on one of the copies is absent. Through a speciation event both copies are passed on to two distinct species. This results in four homologous genes gene A_1, gene B_1, gene A_2 and gene B_2. Homologous genes in a species are called paralog. Homologous genes from different species are called ortholog. A third evolutionary unrelated but similar gene C is called analog. Figure based on Shafee 2018 and itself licensed alike under CC BY-SA 4.0

encoded proteins) are called *homologous*. Homologous genes within a species, i.e. gene A_1 and gene B_1 are called *paralog*. Homologous genes shared between species, i.e. gene A_1 and gene A_2 are called *ortholog*. A third gene C which might be similar to the other genes by chance but has no common evolutionary origin is called *analog*. The similarity of gene C and gene A is a result of *convergent evolution*.

To verify a common descent of proteins it is necessary to identify homologous relationships between them. The homology of proteins can be assessed by considering and comparing the three basic properties of proteins: sequence, structure and function.

1.3 Protein Sequence, Structure, and Function

Each protein-encoding gene defines a specific sequence of amino acids: During protein synthesis, these amino acids are linked together by peptide bonds and form a long chain. The length of protein chains, i.e. the number of amino acid residues of the sequence, varies widely: While the median length of proteins of eukaryotes was estimated to approximately 360 amino acid residues (Brocchieri and Karlin, 2005), the human muscle protein titin, with a length of approximately

30,000 amino acid residues (Opitz et al., 2003) surpasses the median value, as one of the largest known proteins, by two magnitudes of order. On the other hand proteins can be quite small, e.g. the ribosomal protein rpmJ of *Escherichia coli* has a length of only 38 amino acid residues (Fu et al., 2019).

Sequences of proteins can be deduced by identifying protein coding regions in sequenced genomes. Sequences of known and also hypothetical proteins are collected in databases. The rise of modern high-throughput sequencing technologies has led to a vast amount of protein sequence data. An example is the UniProt database (UniProt Consortium, 2019) which as of April 2020 includes over 216 million non-redundant protein sequences (UniProt Consortium, 2020).

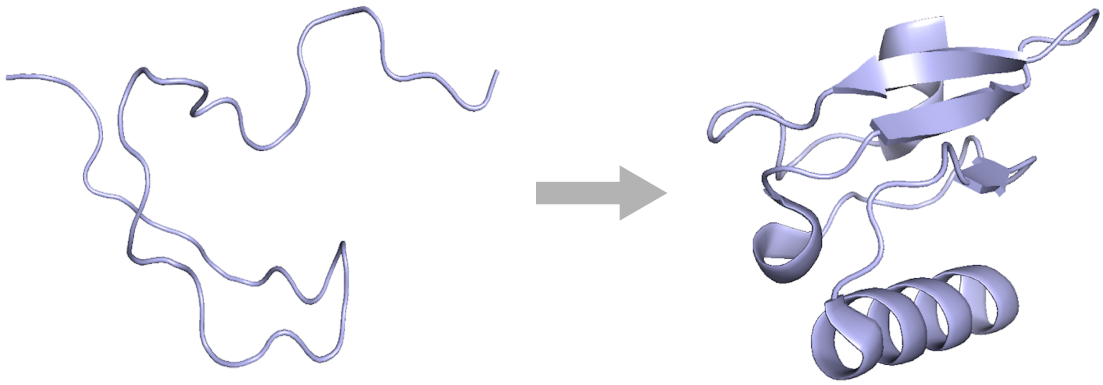


Figure 1.4: Protein folding.
An unfolded unordered peptide chain folds into a compact, stable, and well-defined three-dimensional protein structure. The illustration shows the ribosomal protein rpsS from the 30S subunit of the ribosome from *E. coli* (PDB-ID 6ORE, Fu et al. 2019).

The sequence of a protein is a unique and simple way to describe it, however it is the three-dimensional structure of a protein which determines its function. The three-dimensional shape a protein adopts is a fundamental property as it defines its biochemical characteristics and abilities, i.e. its biological role. The process of a protein adopting its native three-dimensional conformation is called *protein folding*. Figure 1.4 illustrates this concept: The unordered amino acid chain of the ribosomal protein rpsS from the 30S subunit of the ribosome from *E. coli* acquires a complex three-dimensional structure. Only this highly specific and unique spatial conformation allows the protein to play its dedicated role in the large ribosome complex by

binding the ribosomal RNA at a specific position.

While sequencing DNA and deriving the sequence of a protein from its corresponding coding DNA is a relatively simple and highly automatable process, the case is different for determining the spatial shape of a protein. In many cases it is a challenging task to solve a proteins three-dimensional structure using techniques like X-ray crystallography, NMR spectroscopy, or electron microscopy. If the structure determination is successful, proteins with a known structure are deposited in a central database called protein data bank PDB (Berman et al., 2002). The number of structures deposited in the PDB is constantly increasing and in March 2020 it contained over 161,000 entries. The challenging nature of protein structure determination becomes evident when comparing this number to the afore mentioned number of 216 million sequences in the UniProt database: Only for a small fraction of known proteins a three-dimensional structure is available.

The just introduced properties of a protein – sequence, structure, and function – are crucially dependent on each other. It is a fundamental goal in structural biology and life science in general to understand how a sequence determines a proteins structure, which again defines the functional characteristics of the protein. Only with a profound understanding of the relationships between sequence, structure, and function it is possible to completely elucidate the workings of proteins on a mechanistic level.

In the last decades computational biology has made considerable progress in this area. The *protein folding problem*, i.e. the *ab initio* prediction of a native protein structure from sequence data only, can be solved in many cases at a high level of accuracy, even for proteins with a so far unknown fold (Abriata et al., 2019). This indicates that the relationship of sequence and structure becomes increasingly better understood as it can be adequately described by supplying the algorithms with sophisticated force fields and scoring functions.

However, research is still far from understanding the architecture of proteins at a really detailed level. This becomes best evident when taking a look at the discipline of computational protein design. Undoubtedly protein design has delivered many successful and important results (Kuhlman et al., 2003; Looger et al., 2003; Huang et al., 2016). However, in their basic concept, protein design algorithms pursue a guided trial-and-error strategy with a limited rational

component. Today, it is not possible to accurately predict the overall mechanistic effect of a mutation introduced in a protein. Algorithmically it is only possible to score mutations regarding their effect on biophysical energy terms like the Lennard-Jones potential or Coulomb potential and thereby approximate whether a certain mutation is energetically favorable or not. However, in evolution, the minimization of the energy state of a protein is not a selection criteria per se, but favorable functional properties, possibly mediated through energetical stabilization are. This makes clear why current computational protein design algorithms can only provide limited insights to advance the field of protein evolution. Opposed to the relatively new methods of computational protein design the classical rational protein design is based on the knowledge of a specialist researcher which sometimes spent many years studying a specific protein. Also here, while delivering many successful and fundamental findings, the rational approach is often driven by a trial-and-error strategy. The example of protein design illustrates that the relationship of function to sequence and structure is yet to be understood in detail.

To gain further insight in the mechanics of proteins and the relationship of sequence, structure and function it is vital to uncover their evolutionary origin. For this it is of interest to study the process of protein evolution, i.e. to elucidate how functional proteins formed. To study the origin of proteins it is necessary to understand the basic construction of proteins. Thus, in the next section at first the general architecture of proteins will be treated. Subsequently the methodical foundations and definitions which will be required in later chapters will be introduced.

1.3.1 Hierarchy of protein architecture

The structure of proteins can be organized into four different hierarchic levels: primary structure, secondary structure, tertiary structure and quaternary structure. In this order the levels describe the spatial assembly of a protein structure beginning at the polypeptide chain itself (primary structure) on the one end and global arrangements (quaternary structure) on the other end (compare Figure 1.5).

To begin with, the primary structure describes the unique polypeptide chain, defined by the sequence of the protein. The primary structure consists of amino acid residues connected by peptide bonds. The series of peptide bonds defines the *backbone* of the protein. The different

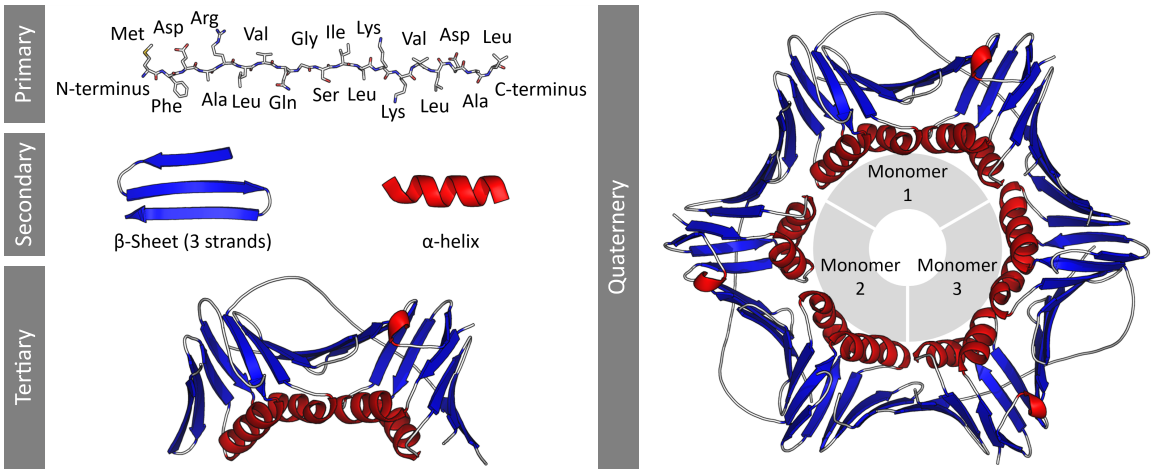


Figure 1.5: Hierarchy of protein architecture.
The figure shows the human DNA-clamp protein PCNA broken down into the different levels of protein architecture. The structural composition of a protein can be divided into four different hierarchic levels: primary, secondary, tertiary and quaternary structure. The primary structure consists of a sequence of amino acids which form a polypeptide chain. The next level is called secondary structure and describes the smallest spacial units the protein chain forms when folding, called secondary structure motifs: α-helices, β-sheets, and loops. The tertiary structure is constituted of a series of secondary structure motifs and forms protein domains. Finally, the highest level of protein architecture is the quaternary structure and describes the oligomerization of individual protein chains. The illustration shows a homotrimer consisting of three identical monomers. (Figure based on Shafee 2016 licensed under CC BY 4.0, PDB-ID `1AXC`, Gulbis et al. 1996)

residues are characterized by their side-chains and the side-chain conformation. It is remarkable, that the amino acid sequence completely and uniquely defines a protein. Higher order structure and consequently function of the protein are predefined by the primary structure of the protein.

Through the formation of hydrogen bonds between its residues the primary structure adapts energetically favorable simple local substructures. These structures are called *secondary structure motifs* and as a total they define the secondary structure of the protein. The most common secondary structure motifs are *α-helices* and *β-sheets*. Unordered regions which connect these motifs are called *loops*. Multiple adjacent secondary structure motifs which form a cohesive unit are called *supersecondary structure* (Levitt and Chothia, 1976). Common supersecondary structure motifs are for example helix-turn-helix or β-hairpin motifs. The supersecondary structure can be interpreted as an intermediate level of organization between the secondary and tertiary level.

Ascending in the hierarchic organization, on the tertiary structure level, the secondary or su-

persecondary structure motifs are packed together and form a complex structure, mainly driven by optimizing compactness, solubility and the ability to form binding and active sites (Kessel and Ben-Tal, 2010). While in wide regards secondary structure motifs are very similar amongst all proteins, on the tertiary level a vast space of possible three-dimensional conformations opens up, which facilitates the enormous functional diversity of proteins.

The last and highest level of protein organization is the quaternary structure. It describes the oligomerization of multiple folded protein chains, called subunits, into larger complexes. Protein complexes can consist of two or more subunits, i.e. dimers, trimers, tetramers, etc. Even large complexes with 20 or more subunits are known; e.g. viral capsids (Božič et al., 2013).

The hierarchic architecture of proteins reflects their intrinsically modular nature. Especially on the secondary structure level it is obvious that reuse of basic motifs, in this case *α-helices* and *β-sheets*, is a general concept in proteins. But also, and even more importantly for this work, on the tertiary structure level the reuse of certain conformations can be observed in the form of *protein domains*. For example the tertiary structure shown in Figure 1.5 shows the symmetrical reuse of a conformation consisting of eight *β-sheets* and two *α-helices*.

1.3.2 The protein domain as a folding unit

The classification of protein architecture into the four levels primary, secondary, tertiary, and quaternary is purely focused on their structural composition. When taking the functional aspect into account the question arises how the smallest functional units of proteins can be described.

At the tertiary structure level the fundamental unit is called *protein domain*. It is defined as a region of the polypeptide chain that can fold autonomously and in many cases has a distinct and unique function (Branden and Tooze, 2012). The tertiary structure of proteins may consist of a single domain or in other cases of multiple domains. A protein can either consist of multiple similar or identical domains or it can be composed of different domains. For the pyruvate kinase isoform PykA, exemplary shown as a multi-domain protein in Figure 1.6, the latter is the case.

In *Pseudomonas aeruginosa* PykA is the main "pacemaker" of the Entner-Doudoroff pathway for glycolysis. It is allosterically regulated, meaning the enzymatic activity of PykA is modulated

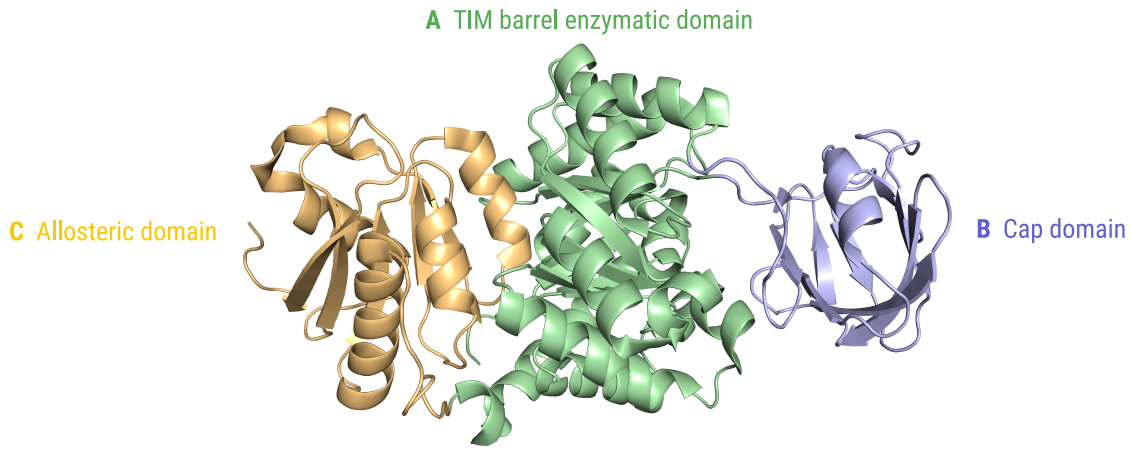


Figure 1.6: Protein domains of a pyruvate kinase.
The pyruvate kinase isoform PykA from *P. aeruginosa* is an example for the modularity of protein chains. It consists of three domains A, B and C. Though occurring in a single protein chain each domain constitutes an individual unit with a distinct folding type. Also functionally the domains fulfill different tasks. Domain A in interaction with the cap domain B is the enzymatically active domain. Domain C is the receiver for an allosteric signal which modulates the enzyme activity (PDB-ID `6QXL`, Abdelhamid et al. 2019).

by an external stimulus, in this case the additional binding of ligands other than its substrate. Interestingly, the allosteric stimulus is sensed in a dedicated allosteric domain which mediates the signal to the enzymatically active site of the protein. The enzymatically active part of the protein again consists of two distinct domains (Abdelhamid et al., 2019). While the exact enzymatic mechanism of PykA goes beyond the scope of this introductory example, the case of PykA serves well to illustrates the functional modularity of proteins, which is based on the combination of multiple dedicated domains.

Besides constituting the basic biologically functional unit, domains can also be seen as the basic evolutionary unit (Kessel and Ben-Tal, 2010): The widely accepted model (compare Figure 1.3) suggests that through gene duplication a gene, coding for a certain domain, gets copied and through divergent evolution a modified homologous version of the domain arises (Ohno, 2013). In this manner, new functions can evolve on the basis of domain duplication events. As as consequence this concept also suggests that structurally similar domains can fulfill different functions as the general structure of proteins is known to be more conserved than sequence (Illergård et al., 2009). In fact naturally occurring domains can be grouped into classes which show a similar structural architecture and in many cases those domains facilitate a large number

of different biochemical functions.

1.3.3 Protein folds

The duplication and divergent evolution of protein domains is thought to be one of the most important mechanisms in protein evolution (Chothia and Gough, 2009; Russell, 2001). It is a natural consequence that certain sets of proteins can be grouped together as they share a similar basic architecture, which supposedly goes back to a hypothetical common ancestral prototypical domain. Such a general and reoccurring type of architecture which in many cases can be observed amongst a broad phylogenetic spectrum of organisms is called a protein *fold*. In Figure 1.7 three different folds can be seen and proteins possessing these architectures. Proteins with similar architectures can be found in all three superkingdoms of life (CATH Team, 2020). This suggests that the origin of some folds must go back at least to the era of LUCA, i.e. to a moment in time before life started to diverge into its three superkingdoms.

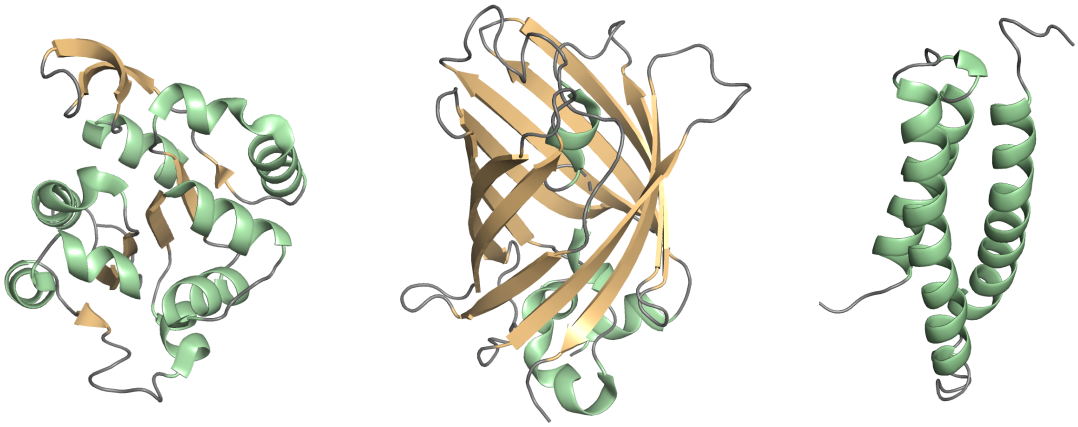


Figure 1.7: Example of protein folds.
The Rossman fold (PDB-ID `1G5Q`) is composed of six parallel beta strands connected by alpha helices. A beta-barrel (PDB-ID `6FWW`) mainly consists of beta strands arranged in an twisted toroidal structure. The up-down bundle fold (PDB-ID `1VCS`) consists of parallel respectively antiparallel alpha helices.

It is easy to see that the folds in Figure 1.7 have a distinct overall architecture and no obvious similarities. This is expected as they don't have a common evolutionary origin. However, as already mentioned, for some folds local similarities have been detected suggesting unidentified relationships between supposedly unrelated folds.

The total number of different folds in nature is estimated to most probably not exceed 10,000 (Koonin et al., 2002). Today, there are approximately 1,400 known folds (SCOP Team, 2020; CATH Team, 2020). The concept of the protein fold and the classification of reoccurring protein architectures plays an import role in this work and will be used extensively. The classification of protein structures will be treated in a following part. Before however, it is necessary to introduce the required tools to measure similarity between proteins.

1.4 Measuring Similarity of Proteins

In the previous sections the general concept of homology was introduced which is fundamental in studying the subject of protein evolution. To make homology methodically accessible for computational analyses it is crucial to establish the ability of deciding on homology. This is achieved by introducing different measures of similarity between proteins, which can be used to identify homologs. In general, homology can be inferred when two proteins share more similarity than one would expect by chance (Pearson, 2013). This means that a similarity measure alone is not sufficient to assess a potential homologous relationship. It is always necessary to consider the background possibility of two proteins being similar by mere chance. In general similarity measures between proteins can be divided into sequence-based and structure-based approaches. It is noteworthy that similarity measures based on the function of proteins do also exist (Pesquita, 2017; Weichenberger et al., 2017). But as such metrics won't be used in this work they will not be discussed any further. The methods used in this work are heavily based on sequence and structure comparisons, hence in this section the basic methods of quantifying sequence and structure similarity of proteins will be introduced.

1.4.1 Sequence similarity

The amino acid sequence of a protein is a simple representation, which completely defines the protein and encodes all of its properties. From a computational perspective it is easy to handle and store, as it can be represented as a simple array of characters, i.e. a string. This makes the sequence a suitable representation to algorithmically assess the similarity of proteins.

To compare sequences, an optimal alignment of the amino acid residues of the sequences has to be found. This is done by introducing gaps as necessary, while maximizing the number of similar or identical amino acid residues aligned to each other (matches). The number of similar amino acid residues aligned to each other (mismatches) should be minimal at the same time. In general this can be done for two sequences (pairwise sequence alignment) or for multiple sequences (multiple sequence alignment). Further one can distinguish between local and global alignments. Local alignments identify optimal alignments of one or more subregions, like single domains to be found in both sequences. Global alignments on the other hand deliver a optimal alignment of the complete sequences, as e.g. needed for phylogenetic analyses. For this work only the local variant of alignments is of importance. Unless otherwise stated, in the following parts of the work the term alignment will always refer to the local variant.

1.4.1.1 Pairwise sequence alignments

The pairwise sequence alignment is one of the most basic methods in sequences analysis. Sequences can be aligned to each other by performing the edit operations insertion, deletion and substitution. An optimal alignment is characterized by a minimal number of applied edit operations while maximizing the number of correctly aligned "matching" positions at the same time.

```
TrpA Ecoli      8 KEGAFVPFVTLGD--PGIEQSLKIIDTLIEAGADALELGIPFSDPLADGP 55
                  |:..|:|::..||  .|:|...:.|..|.:||:.|:|:|:|||||:||||
TrpA Lacla      4 KKNNFIPYIMAGDHEKGLEGLKETIQLLEQAGSSAIEIGVPFSDPVADGP 53
```

Alignment 1.1: Pairwise alignment of TrpA subsequences
An alignment of the N-terminal part of the tryptophan synthase alpha subchain (TrpA) is shown. The homologous sequences are from the organisms *E. coli* (`Ecoli`) and *L. lactis* (`Lacla`).

As an example in Alignment 1.1 the pairwise alignment of the N-terminal part of the tryptophan synthase alpha subchain can be seen. The homologous sequences are from the organisms *E. coli* and *Lactococcus lactis*. An optimal pairwise sequence alignment of two sequences as shown can be found by algorithmically solving an optimization problem that determines the Levenshtein distance (Levenshtein, 1966).

In the early 1980s the authors Smith and Waterman introduced their local alignment algorithm called Smith-Waterman algorithm (Smith et al., 1981), which is until today, in various modified variants, a standard algorithm to compute pairwise sequence alignments. An adapted version of this algorithm was also used in this work.

To align two sequences a and b with length m and n in the Smith-Waterman algorithm the following recursively defined *scoring matrix* S is calculated:

$$S(i,j) = \max \left\{ \begin{array}{cc} 0 & \\ S(i-1,j-1) + \ w(a_i, b_j) & \text{Match or Mismatch} \\ S(i-1,j) + \ w(a_i, -) & \text{Deletion} \\ S(i,j-1) + \ w(-, b_j) & \text{Insertion} \end{array} \right\} \tag{1.1}$$

The function w is called a scoring function which scores match, mismatch, deletion and insertion. S is calculated for $1 \leq i \leq m$ and $1 \leq j \leq n$, with $m = \texttt{length}(a)$ und $n = \texttt{length}(b)$.

The fully populated matrix S describes local alignments of a and b. These alignments can be deduced by identifying *tracebacks* in the matrix, i.e. tracing back the recursive origin of the maximum score values in the matrix. Each traceback in the matrix defines a local alignment.

Pairwise alignments are a simple tool to study the similarity of two sequences. However, in many cases not only two but multiple, often homologous, sequences, for example different TrpA sequences from various organisms, need to be compared which makes the alignment of multiple sequences necessary.

1.4.1.2 Multiple sequence alignments

A multiple sequence alignment (MSA) is an extension of the pairwise sequence alignment which allows to align more than two sequences. With multiple sequences, finding the optimal alignment can become computationally very expensive which makes it necessary to fall back to fast heuristic algorithms. As a consequence, for the creation of MSA, there's a variety of different algorithms available such as MAFFT (Katoh et al., 2002), MUSCLE (Edgar, 2004) or T-Coffee (Notredame et al., 2000), with each of them implementing an individual heuristic approach. A

detailed methodical consideration of these algorithms, however, would go beyond the scope of this introduction. Therefore in the following only the basic composition and the properties of an MSA will be introduced using the example of a TrpA MSA.

In the MSA shown in Figure 1.8 multiple TrpA sequence from different species are aligned. The MSA consists of *rows* (sequences) and *columns* (sequence positions). Also the sequences from *E. coli* and *L. lactis* used in the example of a pairwise alignment from above can be found again. A closer look at the alignment of the sequences from these two species makes it apparent that the optimal pairwise alignment from above is not exactly reproduced in the MSA. The deviation from optimal pairwise alignments naturally results from globally optimizing the alignment of all sequences to each other in the MSA.

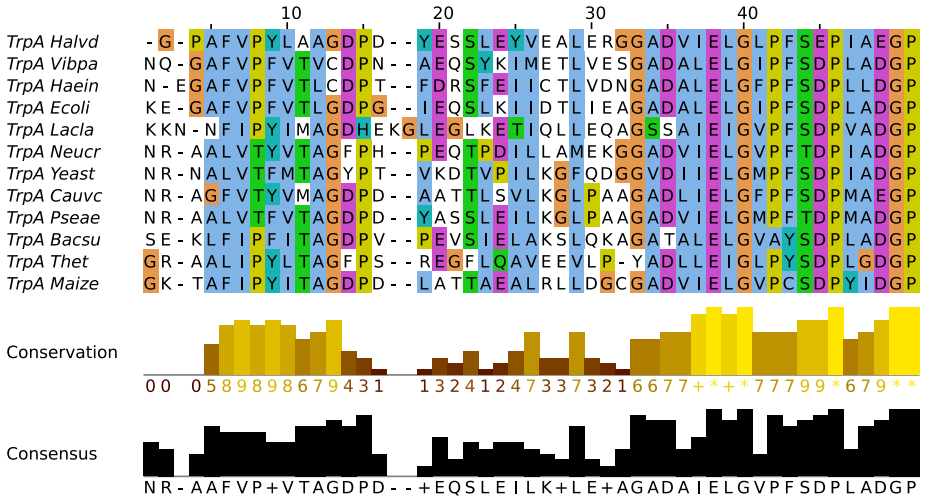


Figure 1.8: Multiple sequence alignment of TrpA sequences.
The MSA shows the N-terminal part of selected SEED sequences from the Pfam family Trp_syntA (PF00290). A position dependent variation of the conservation can be observed in the MSA. The MSA includes sequences from various species, e.g. *Haloferax volcanii* (`Halvd`), *Zea mays* (`Maize`), *Saccharomyces cerevisiae* (`Yeast`), etc. The figure was created with Jalview (Waterhouse et al., 2009).

The sequences in the MSA result from millions or even billions of years of divergent evolution. This becomes especially apparent in variable regions which acquired various mutations amongst the different species (see conservation and consensus graphs in Figure 1.8). However, also highly conserved regions can be observed in the MSA. This indicates that the amino acid composition in these regions is critical for the correct functioning of the protein. As a result the amino acid composition of each column in the MSA varies. This property of MSAs can be used to generate

a position-dependent sequence profile from the MSA. Position-specific sequence profiles have proven to be very useful when iteratively extending an MSA with additional sequences. With the help of a profile this can be done by checking whether a potential new sequence matches the profile. In the next section this concept will be explained in detail.

1.4.1.3 HMMs

Profile hidden markov models, commonly just called HMMs, are a widely used method of utilizing sequence profiles for the identification of homologous sequences. An HMM models a set of aligned sequences as a stochastic process by describing it as a Markov chain. The initial calculation of an HMM is based on an MSA. A HMM generalizes the information of an MSA and describes it as a sequence profile, which results in a representation of the potential sequence space of similar sequences.

In Figure 1.9a, a classical three-state topology of an HMM with five matching positions is depicted. In general the HMM consists of the three different states *match* state, *insert* state and *deletion* state. These states can transit into each other. However, not all transitions are possible and the topology of the HMM dictates which state transitions are valid. Possible state transitions are indicated with arrows. Each state transition has a certain probability called *transition possibility*. The topology of the HMM is based on the number of match states which can be deduced from the initial MSA. The shown MSA has five matching columns. A matching column is defined as column featuring less gap characters than a predefined threshold requires, i.e. less than 50 % gaps. Thus in this example the fourth column of the MSA is not considered a match column. The number of match states is defined by the number of matching columns in the MSA. A match state can emit an amino acid residue. Each match state has unique *emission probabilities* which define the probability of a certain amino acid residue being emitted by a certain match state. These emission probabilities can be deduced from the initial MSA by simply evaluating the column-wise amino acid abundances. Additionally, natural background probabilities are taken into account to make emissions and transition possible which are not observed in the initial MSA. The transition probabilities can be estimated by analyzing adjacent columns of the MSA (i.e the probability of inserts, matches, deletions in the corresponding next column).

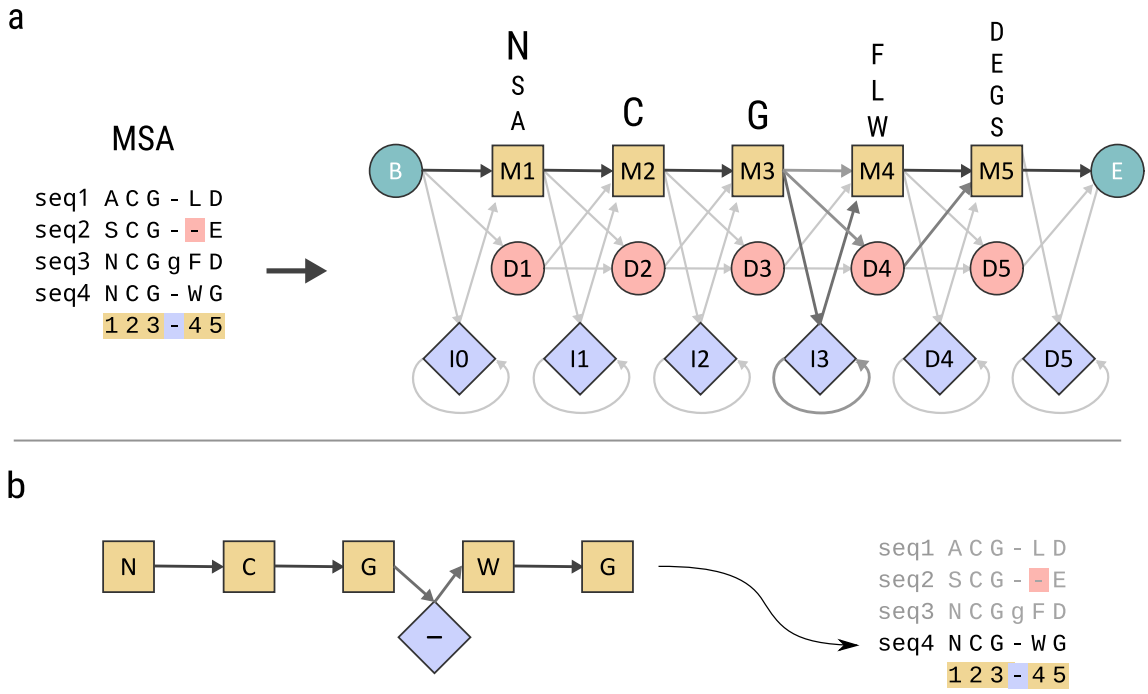


Figure 1.9: Topology of an HMM.
a A three-state HMM consists of match states, insert states and deletion states. The shown HMM features five match states which correspond to five matching columns of the MSA. The possible transitions from one state to another are represented by arrows. The probabilities of a certain transition is indicated by the opacity of the arrow. The emission probabilities of the match states are indicated by different sized amino acid characters above the match state symbols. Begin (B) and end (E) states ensure a formally correct defined HMM and allow a initial insert state I0. **b** A sequence can be described as a path through the HMM. The total probability of the path can be calculated from the individual emission and transition probabilities. Figure based on EMBL-EBI 2020 and itself licensed alike under CC BY-SA 4.0

Figure 1.9a illustrates how an MSA can be used to define the match states of an HMM. In Figure 1.9b it is shown how a certain sequence can be described as series of state transitions and emissions, i.e. as a path through the HMM. As a series of state transitions and emissions, a path can be also interpreted as a series of probabilities. In this manner, the emission of a sequence from the HMM can be assessed with a certain probability. This property can be used to score an arbitrary new sequence (query) based on the given HMM, which makes it possible to evaluate how well a new sequence matches a given sequence profile.

In the case of larger and more complex HMMs, for a given sequence often multiple paths can be found which emit the query. In this case the most likely path through the HMM is of interest. This optimization problem can be solved in a similar manner as the previously introduced Smith-Waterman algorithm finds optimal pairwise alignments. The so-called *Viterbi algorithm*

also calculates a recursively defined matrix and identifies the most likely path, called *Viterbi path* by using back tracking.

An HMM can be iteratively refined by adding matching sequences and rebuilding it. In this manner, highly sensitive profiles for the detection of even remotely homologous sequences can be created.

There are different software packages offering homology detection based on HMMs, most notably HMMER (Eddy, 2020) and HHSuite (Steinegger et al., 2019a). In this work the HHblits tool from the HHSuite package was utilized.

1.4.2 Structure similarity

As seen in the previous part, the pairwise comparison of proteins based on sequences is a relatively clear and straightforward process. Although there are many different algorithms available, historically there is a consensus about the basic methods of sequence alignments which "has led to trusted and generally accepted procedures" (Kolodny et al., 2013). The case is different for the comparison of proteins based on structures. It has proven to be a difficult task to find an optimal superposition of complex three-dimensional protein structures and there is no agreed-upon standard method for this problem (Kolodny et al., 2013).

It could be shown that structure alignments with an optimal score can be found, but the runtime of such an algorithm scales proportional to the eighth power with increasing sequence length (Kolodny and Linial, 2004). As a consequence, there's a variety of algorithms which make heuristic approaches such as DALI (Holm and Sander, 1995), CE (Shindyalov and Bourne, 1998) or TM-align (Zhang and Skolnick, 2005). These algorithms are fast, but do not necessary find the optimal alignment due to their heuristic nature. For this work, TM-align was chosen, which was in comparison to other methods shown to improve accuracy while being several times faster (Zhang and Skolnick, 2005). All structure alignment algorithms have in common that they need to define a metric between two three-dimensional protein structures which they assess and optimize. These are either called structural similarity or structural distance measures, depending on whether the value gets higher with increasing distance or increasing similarity of the protein

structures. In general, one has to differentiate between a measure of structural similarity or distance per se and algorithms minimizing or maximizing these measures to find a good or optimal structural alignment between two protein structures. For example, TM-align uses its own similarity measure called TM-score and aims to maximize it. However, before treating TM-align in more detail, at first the most basic and common three-dimensional distance measure for protein structures called RMSD will be introduced.

1.4.2.1 RMSD

The most common way in structural biology to specify the three-dimensional distance between two protein structures is the so-called root mean square deviation (RMSD). The RMSD is based on the distance between pairs of corresponding atoms and can be defined as a simple formula:

$$\mathrm{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2} \tag{1.2}$$

where N is the number of pairs of atoms and d_i is the euclidean distance between the i-th atom pair. Depending on which atoms to include in the calculation, the RMSD can vary quite much for the same two input structures. The all-atom RMSD which includes all side-chain atoms of the protein is generally larger then an RMSD resulting from a subset of the atoms. It is most common to use either the heavy backbone atoms of the protein structures or only C_α atoms. As it sums up distance values, the RMSD has the dimension of length which is usually given in Ångström (Å, 10^{-10} m).

The RMSD has a substantial drawback when it comes to the comparison of RMSD values resulting from independent pairs of structures: The RMSD is length dependent. For example the significance of an RMSD value of 2 Å for a protein pair with the length of 50 residues is fundamentally different to the significance of the same RMSD value resulting from the comparison of two proteins with a length of 500 residues (Carugo and Pongor, 2001). For this work it was very important to choose a similarity measure which makes it possible to compare the results from independent structural alignments with big size differences. For this reason, instead of using the RMSD it was chosen to use the TM-score and the according structural alignment algorithm

TM-align.

1.4.2.2 TM-score and TM-align

The TM-score is a variant of the Levitt-Gerstein (LG) score which was introduced to overcome several shortfalls of the RMSD, for example the dominance of outliers, i.e. pairs featuring a large distance (Levitt and Gerstein, 1998). However, the LG-score and other derived scores like MaxSub (Siew et al., 2000) or the GDT_TS score (Zemla, 2003) still show a length dependence (Zhang and Skolnick, 2004). As an improved LG-score derivative the TM-score was defined as follows (Zhang and Skolnick, 2004):

$$\text{TM-score} = \max \left[\frac{1}{L_N} \sum_{i}^{L_T} \frac{1}{1 + \left(\frac{d_i}{d_0(L_N)} \right)^2} \right] \tag{1.3}$$

At this place it is important to note that the TM-score evaluates local alignments: It differentiates between the length of the native ("to be aligned") structure L_N and the length L_T of the actual alignment to the target structure (also called template structure). For a global alignment L_N would be equal to L_T. d_i is the distance between the i-th pair of aligned residues. The max operator indicates that the maximum TM-score of all possible spatial superpositions is returned, i.e. the optimal superposition is used for the calculation of the TM-score. The value d_0 is used as normalization. The redefinition of this value is the main adjustment which differentiates the TM-score from other approaches like MaxSub or GDT_TS. The authors Zhang and Skolnick achieved a length independence by approximating the value of d_0 dependent of L_N:

$$d_0(L_N) = 1.24 \sqrt[3]{L_N - 15} - 1.8 \tag{1.4}$$

The formula is based on the statistical analysis of a comprehensive set of PDB structures. In this manner, the authors could approximate d_0 as "the average distance of corresponding residue pairs of random related proteins" (Zhang and Skolnick, 2004) for a given length L_N, which makes the TM-score a length independent structural similarity score for proteins.

TM-align is an algorithm which is designed to identify close to optimal structural alignments

based on the maximization of the TM-score. After making multiple initial alignment steps (e.g. secondary structure alignment) it then applies an heuristic optimization algorithm which iteratively refines the alignment using a TM-score based rotation matrix similar to the RMSD based rotation matrix used in the Kabsch algorithm (Kabsch, 1978), which is one of the earliest methods for the superposition of protein structures.

1.5 Classification of Proteins

As the previous sections showed, proteins can be described at three different levels: based on the sequence of the polypeptide chain, based on their three-dimensional structure, and based on their function. The potential space of manifestations on these three levels will be called *sequence space*, *structure space* and *function space* in the following. It is a crucial understanding that these three levels are not independent of each other: A certain amino acid sequence induces a three-dimensional structure and the spatial conformation facilitates the functional capabilities of a protein. In structural biology, it is one of the main goals to understand the relationship between these levels (Kessel and Ben-Tal, 2010). The classification or in other words grouping of similar and thus presumably homologous proteins, is an essential requirement for studying protein evolution as it is laying the data basis for computationally driven analyses of evolutionary relationships between proteins.

In a previous section the protein domain was introduced as a basic evolutionary unit. The goal of protein classification is to group domains into homologous *protein families* and establish higher order hierarchic relationships between the identified families. From a technical and methodical perspective it has proven to be a challenging task to consider all three levels, sequence, structure, and function at once. Thus, over time, different classification methods for proteins with their primary focus on either sequence, structure or function were developed. In this work different classification schemes, mainly on the sequence and structure level, were used, which will be introduced in the following.

1.5.1 Sequence-based classification

Sequence-based classification schemes make use of the sequence similarity of proteins. The already introduced concept of HMMs has proven to be an ideal method for the iterative compilation and extension of protein families. By using HMMs it is possible to create individual sequence profiles for each protein family. As a consequence, with an increasing number of correctly identified sequences, the profile becomes more representative and sensitive, which makes it possible that even sequences with a weak homologous signal can be correctly classified. One of the first databases which offered a HMM-based classification of protein sequences was Pfam (Orengo et al., 2014; El-Gebali et al., 2018). Another widely used resource is the InterPro database (Mitchell et al., 2014), which follows the approach of collecting data from various sources and aggregates it to increase its coverage (Orengo et al., 2014). InterPro also includes data from Pfam. For this work, it was decided to use the Pfam database because of the following two reasons: Firstly, Pfam offers a manually curated high quality subsets of the database which is less prone to misclassifications compared to highly automated approaches like InterPro and, secondly, pre-built HMM databases are available for Pfam which will play an important role in the algorithm presented in the next chapter. Thus, in this section the concept behind the Pfam database will be introduced.

The Pfam database is a collection of sequences of protein domains which are grouped into homologous families. There are two variants of the database Pfam-A and Pfam-B. Pfam-A is the high quality portion based on a high amount of manual curation, while Pfam-B is based on automated sequence annotation and is trimmed to maintain a maximum coverage (Orengo et al., 2014). In this work the Pfam-A database was used.

In Figure 1.10, the workflow behind a Pfam-A entry is shown. Each entry is based on a so-called SEED alignment which is a manually created small set of sequences which was chosen by experts and represents the family well (Orengo et al., 2014). Subsequently from this set of sequences an HMM is built. Using the SEED HMM profile as a basis, the UniProt database is searched for additional sequences which match the profile. In this way the so-called FULL alignment is built.

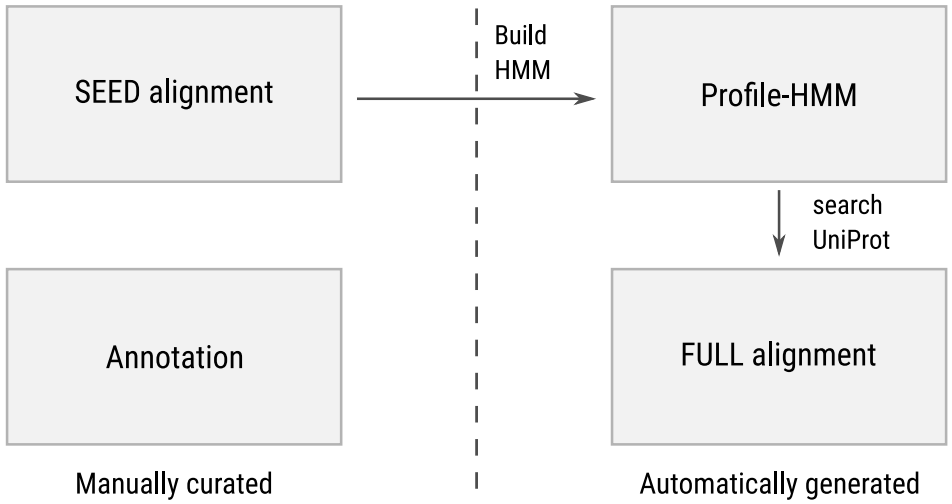


Figure 1.10: Pfam-A workflow.
Each entry of the Pfam-A database is based on a manually created SEED alignment which is curated by experts. A HMM is built and used to automatically search the UniProt database for additional sequences which match the profile. This results in the FULL alignment of the entry. Figure adapted from Orengo et al. 2014 by kind permission of John Wiley and Sons.

The current release Pfam 32.0 contains 17929 entries (Pfam Team, 2020). If available, except from sequence alignments, Pfam offers additional information about the families like functional annotations, phylogenetic distribution or known interactions with other families.

1.5.2 Structure-based classification

In the previous sections it became clear that the principle architecture of proteins, i.e. primary, secondary, tertiary and quaternary structure can be described hierarchically and that the enormous functional variety of proteins is facilitated by the tertiary structure diversity. Naturally, the question arises whether this structure space can be classified and organized in a similar way as possible for the sequence space.

Historically, with increasingly more protein structures available, it has become apparent that the structure space in general, although emerged through billions of years of evolution, is readily comprehensible and comprised of reoccurring fold types (Levitt and Chothia, 1976). As a consequence hierarchically organized structure-based classification systems for protein domains were proposed and established. In this section the systematic classification of proteins according to their structure will be treated.

The two most important and popular protein databases that classify protein domains according to their structure are SCOP (Murzin et al., 1995) and CATH (Orengo et al., 2002). At this point it should be mentioned that neither SCOP nor CATH are purely structure-based. On lower levels of their hierarchy, both databases use sequence information in the form of homologous sequence families for the classification. In this work however, mainly the higher levels, which are purely based on structure data, are of importance. In the following the focus will lie on the CATH database as it was mainly used in this work. As for the comparison to other studies also the SCOP database was utilized, the differences between CATH and SCOP will be treated in the last part of this section.

Figure 1.11 shows the hierarchical classification scheme of CATH. On the highest and most coarse level of the CATH database, called the class level, there are four groups: *Mainly Alpha*, *Mainly Beta*, *Alpha and Beta* and *Few Secondary Structure*. Only the first three are shown in the Figure as a big majority of classified domains falls into these categories, leaving the fourth class for irregular proteins. When descending in the hierarchy of the classification scheme, the levels *architecture*, *topology* (fold) and homologous *superfamily* follow. Currently, in April 2020, the CATH database knows 41 different architectures, 1,391 topologies, and 6,119 superfamilies (CATH Team, 2020).

At the class level CATH differentiates only by the raw secondary structure content and groups domains in those with mainly alpha, mainly beta, and both alpha and beta secondary elements. The relatively sparsely populated *Few Secondary Structures* category collects irregular protein domains. At the architecture level the orientation of secondary structure elements comes into play. This level describes the overall shape of the protein (Kessel and Ben-Tal, 2010). However, the connectivity of the secondary structure elements is ignored. This changes at the next lower level, the topology category which is also referred to as fold. At this level, proteins which share a common fold are grouped together, meaning that the orientation of the secondary structure elements is similar as well as their connectivity. At the next lower level, the level of superfamilies, one can assume a common evolutionary origin of the collected domains. Superfamilies describe a certain manifestation of the fold which in many cases goes along with a certain functional repertoire of the proteins. Finally, the lowest level of the CATH database are sequence

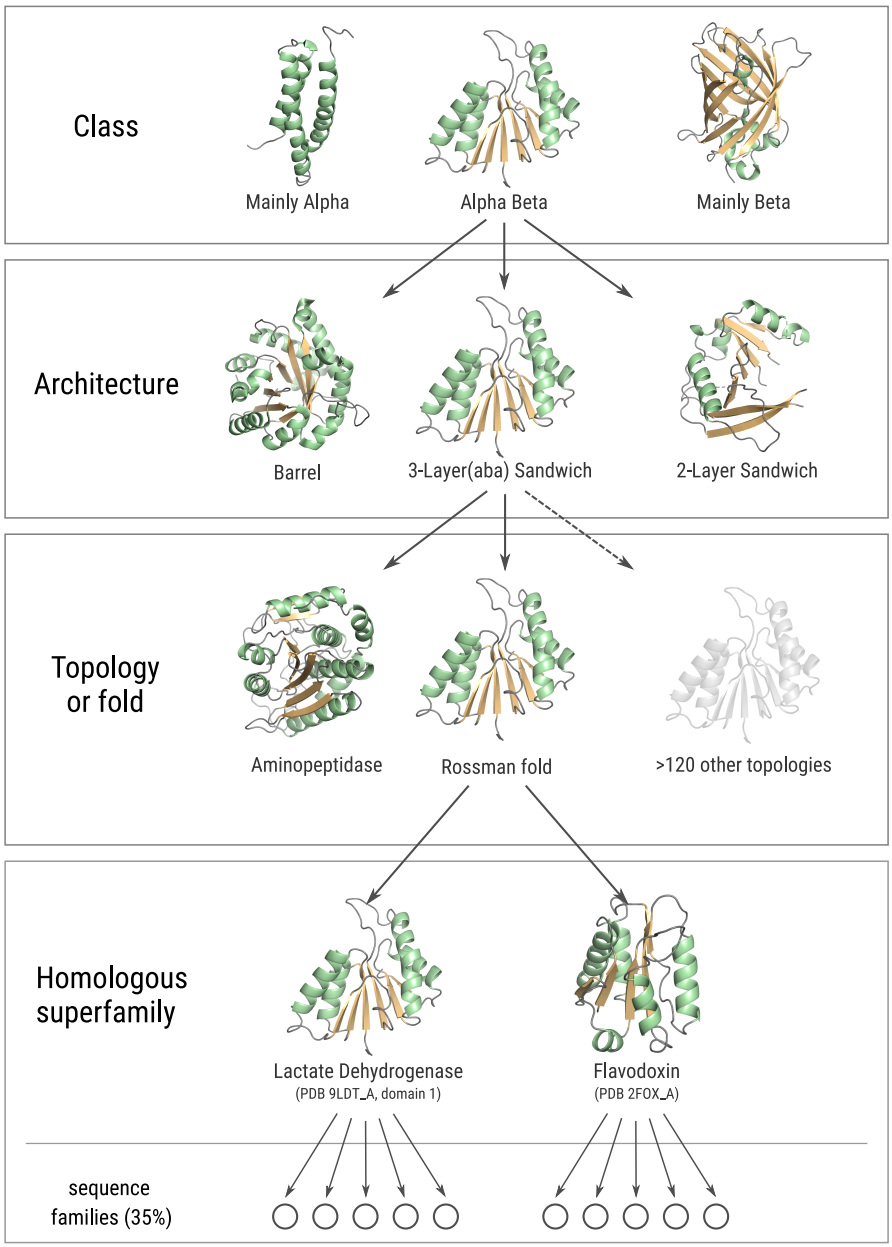


Figure 1.11: Hierarchy of the CATH database.
The CATH database classifies protein domains in the hierarchical categories *class*, *architecture*, *topology*, and *superfamily*. At the class level the secondary structure content is evaluated grouping the domains into those with *Mainly Alpha*, *Mainly Beta*, *Alpha and Beta* and *Few Secondary Structure* (not shown). The next lower level architecture groups structures with similar orientation and arrangement of secondary structure. At the topology (fold) level the connectivity of the secondary structure elements is added to the classification. At the level of homologous superfamilies, an evolutionary relationship between the members of a superfamily can be assumed. The lowest level includes sequence information and groups domains into sequence families at a sequence identity of 35 %. Figure adapted from Orengo et al. 2014 by kind permission of John Wiley and Sons.

families collecting domains with a minimum sequence identity of 35 %. At this level a common evolutionary origin is almost certain.

The just described scheme and its notation can be illustrated by looking at domain 1 of a lactate dehydrogenase (PDB-ID `9LDT_A`, domain 1) as an example. Instead of full names the classification can be written using integer numbers separated by dots. The full CATH classification of the domain is `3.40.50.720`. This indicates that the protein belongs to class `3` (Alpha Beta), architecture `40` (3-Layer(aba) sandwich), topology `50` (Rossmann fold) and superfamily `720` (NAD(P)-binding Rossmann-like Domain). Due to its compactness the numerical notation of CATH classifications will be used from this point on.

From a conceptional perspective the scheme of the SCOP database is very similar to the one of the CATH database: The SCOP database has the categories *class*, *fold*, *superfamily*, and *family*. In this manner, the most obvious difference is that CATH has the additional architecture level. The other differences between SCOP and CATH lie in the process of populating the different levels. To understand this, one has to consider the general difficulties of protein structure classification: At the beginning of this section it was said that the structure space can be classified in a hierarchical manner, which is what CATH and SCOP are trying to achieve. However, for an unambiguous and perfect classification of all proteins the structure space would have to be discrete. As evolution is a continuous and still ongoing process, which began billions of years ago, this is not the case: The structure space has a continuous nature (Shindyalov and Bourne, 2000; Kolodny et al., 2006; Skolnick et al., 2009). Thus, both CATH and SCOP face the challenge to discretize a continuous structure space. This discrepancy makes it necessary to base the classification systems on models using certain assumptions and thresholds. In these regards SCOP and CATH differ as different design decisions were made. An example for this is the domain boundary assignment which requires a, to some degree arbitrary, decisions whether to include linker regions in the domain or not (Orengo et al., 2014). Another example that illustrates the problem of structural classification is the fact that terms like fold and architecture lack a formal and universally agreed upon definition (Orengo et al., 2014). As a consequence it is difficult to rank SCOP and CATH regarding their quality, as there is no generally accepted benchmark framework available to evaluate structural protein classification.

However, despite the difficulties of structural classification, studies comparing CATH and SCOP showed that for approximately 70 % to 80 % of superfamilies a mapping from CATH to SCOP i.e. an agreement of CATH and SCOP can be found (Csaba et al., 2009; Orengo et al., 2014).

There are three reasons why in this work the CATH database was chosen. Firstly, due to an automated assignment process the CATH database tends to contain more protein domains than SCOP (~276,000 in SCOP vs. ~434,000 in CATH in May 2020) . Secondly, it offers the additional level *Architecture* which was used in this work. Thirdly, other studies similar to this work (Alva et al., 2015b; Farías-Rico et al., 2014) used the SCOP database. Using the CATH database discriminates this work from these studies and makes it possible to evaluate the effect of the choice of the classification scheme.

1.6 Aim and Scope of this Work

The previous sections introduced the necessary concepts to study the origin of protein folds. As explained in the very beginning of this chapter the aim of the work is to apply sensitive sequence and structure comparison algorithms to search for previously unknown relationship between protein folds, which can deliver new insights into the earliest stages of protein evolution.

The work presented in this thesis can be divided into four parts: The first objective was the design and implementation of an algorithm for the automated detection of putative ancestral protein motifs. Testing of the algorithm and a comparison to similar approaches represent the second part of the work. Subsequently, the third part was a large-scale application of the algorithm based on a comprehensive set of protein structures. Finally, the fourth part was the analysis of the results from the large-scale application.

The developed algorithm represents an alternative approach to other studies (Farías-Rico et al., 2014; Alva et al., 2015b; Ferruz et al., 2020) , which are methodically similar to each other. The design of the algorithm aims at high sensitivity and an automated execution to minimize the need for manual intervention. The goal of the large-scale scan for protein modules was to identify as many putative ancient relationships between proteins as possible. The final analyses

of the scan results were used as a data basis to assess the hypothesis of a modular origin of protein folds.

It was found that, despite making differing methodical design choices, most of the motifs reported in previous studies could be detected as well. The large-scale scan gives information on the abundance of putative ancestral motifs among a representative set of proteins. The results suggest that these putative ancestral remnants are rarely detectable in modern-day proteins. Further it was found that the detected motifs are not evenly distributed among all folds which questions their universality.

Chapter 2

Materials and Methods

2.1 Databases

In this work the following databases were used.

- CATH (Orengo et al., 2002), version 4.2
- SCOPe (Fox et al., 2014), version 2.06
- Protein Data Bank PDB (Berman et al., 2002), snapshot May 2019
- Pfam-A (El-Gebali et al., 2018) version 32.0
- UniProt (UniProt Consortium, 2019), release 2020_01

2.2 FragStatt

The algorithm FragStatt for the detection of putative ancestral protein motifs consists of the four modules GraphCreator, Pathfinder, Pathanalyzer and SWiFD. In this section their design, implementation, and function will be explained in detail. A practical description and specific applications of FragStatt will be presented in Chapter 3. Generally the motif detection algorithm requires two HHblits cascade results, as described in section 2.4.1, as an input. The two input datasets have a tree structure, with the root representing a protein from the PDB. All parts of the algorithm were implemented in Python 2.7.

2.2.1 GraphCreator

As its name implies, the first component of FragStatt called GraphCreator creates a graph based on two HHblits cascades (compare Figure 2.1) with the two proteins P_1 and P_2 as roots. The nodes in such a graph represent HMMs and the edges represent HHblits hits. HHblits hits are local alignments between HMMs. GraphCreator loads two previously generated edge lists in pickle format and checks them for common nodes. If no common nodes are detected this means, that the individual tree graphs can not be connected and thus no paths between the two root sequences exist. When this should be the case, an error message informs the user and the program terminates.

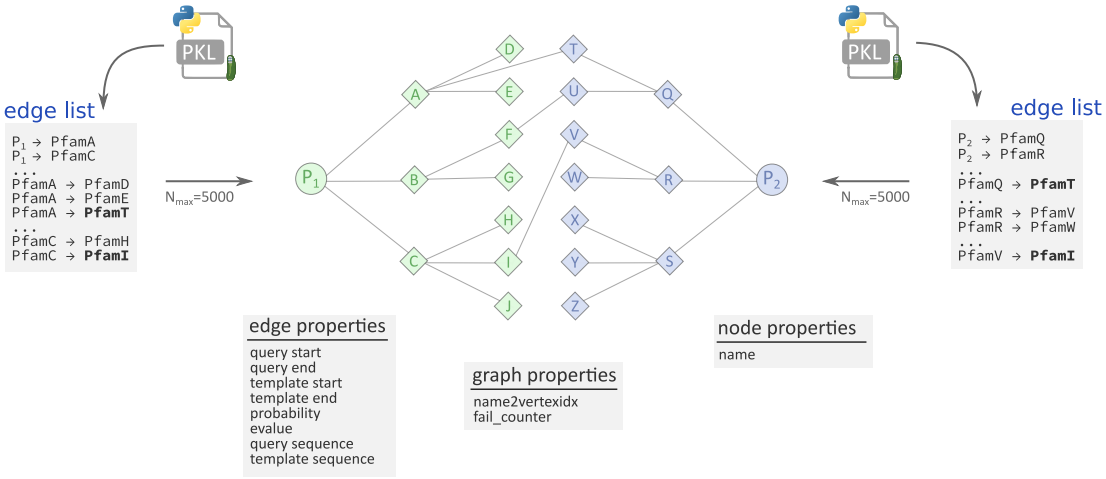


Figure 2.1: GraphCreator.
The first component of FragStatt takes two HHblits cascades in the form of edge lists as an input. At first it checks whether the two subgraphs graphs can be connected, i.e. if the edge lists share common nodes. If this is the case (e.g. PfamT and PfamI in the illustration) a combined graph is built using the graph-tool library. To limit the data amount and required computational resources, for further analysis the number of edges is limited to 5000 for each subgraph. If this number is exceeded a subsample of all edges is considered. Each edge holds the information of the corresponding HHblits hit in the form of edge properties. As by default nodes in graph-tool only have internal IDs, it is necessary to store the real name of a node (e.g. `PF000001`, `1ABC_A`) as additional node property. For this purpose the bidict `name2vertexidx` serves as a bidirectional lookup table stored as a graph property. Additionally, to keep track of errors while generating the graph (e.g. parsing errors) a variable `fail_counter` is stored.

For the positive case of intersecting HHblits cascades, GraphCreator continues to build a graph by combining both edge lists of the tree-structured subgraphs utilizing the graph-tool (Peixoto, 2014) library. Before creating the graph, the number of edges available for both subgraphs is

checked. If either edge list exceeds a number of 5000 edges it is evenly reduced to a 5000 edges subsample. This ensures a similar maximum size of the resulting graph, independent of the input data fed into GraphCreator, which is important for estimating the required computational resources for a large-scale use of FragStatt.

The combined graph has a topology like it is shown in Figure 2.1: The graph spreads from two proteins P_1 and P_2 and eventually reaches the same point in the sequence space from both sides. To keep all data compact and consistent and to provide easy accessibility, the information on the underlying HHblits hits were stored as edge properties as supported by graph-tool. Each edge was assigned the informations extracted from the HHblits results files, as described in Section 2.4.1 and listed in Figure 2.1.

The graph-tool library automatically assigns internal IDs to the nodes and edges and all of its analysis methods, e.g. finding paths between two nodes, refer two these internal IDs. Thus, to interpret the results of such an analysis it is necessary to map the internal node IDs back to the initial real names of nodes (e.g. `PF000001`, `1ABC_A`). For this purpose the real name of each node is stored as a node property. For fast conversion between the real name and the internal ID a bidirectional dictionary `name2vertexidx` is created when initially building the graph and stored as a global graph property. As Python does not support bidirectional dictionaries by default the bidict library (Bronson, 2019) was used for this purpose. Unlike common dictionaries a bidirectional dictionary does not only support looking up the value for a key but also looking up the key for a value. This requires, as it is the case here, a bijective mapping, e.i. an unambiguous mapping both from name to ID and from ID to name.

In the case of errors during parsing of the edge lists, an error counter `fail_counter` keeps track of the number of encountered errors. This was implemented in view of a large-scale use of FragStatt and a very high number of inputs which might be corrupted in some cases. Thus GraphCreator was designed in such a way that non-fatal errors, occurring during the generation of the graph, are ignored and counted using `fail_counter`. After the run the user can then decide, whether a given error rate (e.g. 0.1 % of edges couldn't be parsed) is acceptable.

As result the component GraphCreator returns a graph-tool `Graph` object which embeds all necessary information for further analysis in the form of edge, node, and graph properties. This

object is the basis for the next step of FragStatt.

2.2.2 Pathfinder

The component Pathfinder was designed to identify paths connecting the root P_1 and P_2 (compare Figure 2.2). Based on the graph-tool `Graph` created by GraphCreator in the previous step a depth-first search (DFS) is carried out. The maximum search depth of the DFS is set to 4 by default, but can be changed by the user.

As HHblits hits are local alignments, it can be the case that for a query-target pair multiple hits are detected. In terms of graph topology, this means that nodes can be connected by parallel edges as shown in Figure 2.2. In some cases, this property of the graph leads to vast number of paths connecting P_1 and P_2. Pathfinder may identify hundreds of thousands or even millions of possible paths. To keep the number of paths which have to be evaluated in the following steps manageable, the maximum number of paths is limited to 10000. If this number is exceeded, the set of paths is reduced by randomly sampling 10000 paths.

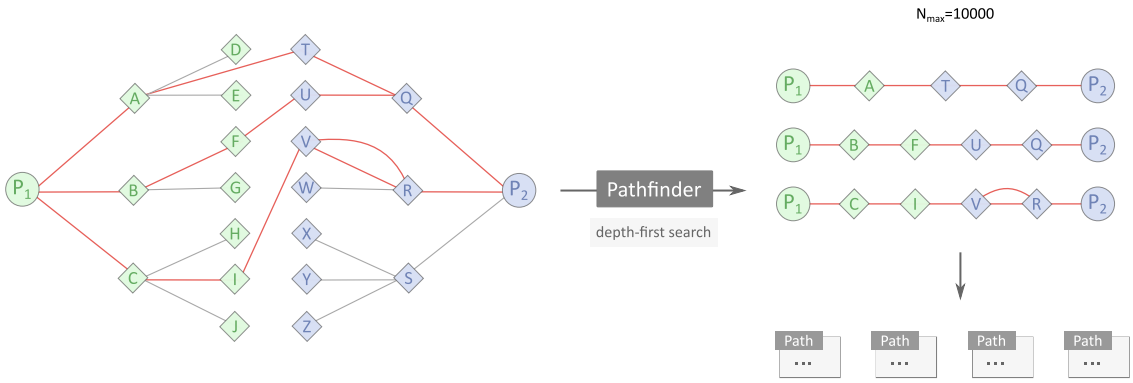


Figure 2.2: Pathfinder.
The second component of FragStatt uses a depth-first search as implemented in graph-tool and identifies paths connecting the root sequences P_1 and P_2. The identified paths are extracted and collected using a dedicated `Path` class. The illustration also shows a case of multiple edges between the same two nodes (V and R), which can occur if more than one local alignment was detected by HHblits for a pair of profiles. Pathfinder resolves these cases and splits them into single paths with exactly one connection for each pair of nodes. Consequently four individual paths are detected in the exemplary case.

Each detected path is saved in the form of a `Path` object (compare Figure 2.3). The path is defined by a sequence of graph-tool edges which is stored in the `edges` attribute of the `Path`

object. Additionally a reference to the graph is kept as an attribute of the class. For later use the class has an attribute `msa`, which is a BioPython (Cock et al., 2009) `MSA` object.

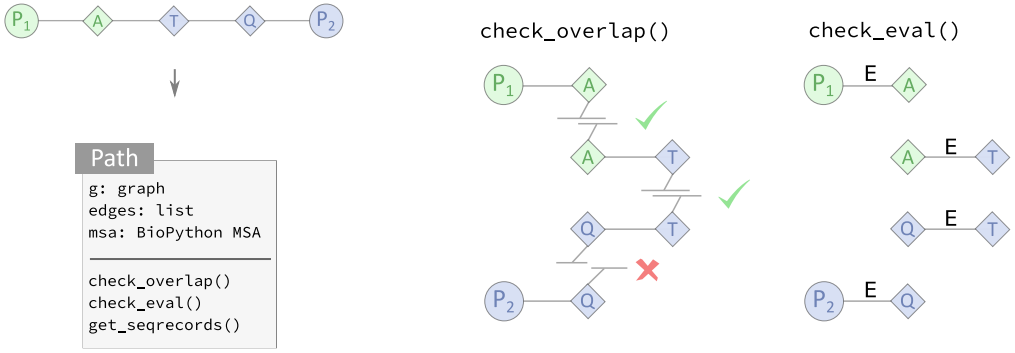


Figure 2.3: Path object.
Each detected path is stored using a `Path` class. It holds three attributes: `g`, a reference to the graph, `edges`, an ordered list of edges defining the path, and `msa`, a BioPython `MSA` object which will be used by Pathanalyzer in the following parts of FragStatt. The `Path` class has three methods. A path consists of individual local HHblits alignments which must not necessary overlap. For that reason the method `check_overlap()` can be used to check if the alignments that determine the path continuously overlap. The method `check_eval()` can be used to check whether all HHblits hits of the path suffice a given E-value threshold. `get_seqrecords()` returns all sequences of the path as BioPython `SeqRecord` object.

Each iteration or hop in a HHblits cascade is an individual local alignment. A path in the HHblits cascade is a series of query-target relationships. For a path to be valid, it must be assured that every hit region in the cascade overlaps with the query region of the next iteration as illustrated in Figure 2.3. For this purpose the method `check_overlap()` as sketched out in Pseudocode 2.1 was used.

```
def intersect_len(s1, e1, s2, e2):
    return(min(e1, e2) - max(s1, s2) + 1)

def check_overlap(path, minlen):
    for e1, e2 in pairwise(path.edges):  # e.g. (P1-A, A-T), (A-T, T-Q), etc.
        ilen = intersect_len(e1.template_start, e1.template_end,
                             e2.query_start, e2.query_end)
        if ilen < minlen:
            return False
        else:
            return True
```

Pseudocode 2.1: Checking overlaps of HHblits cascade paths. The helper method `intersect_len()` returns the overlapping length of two intervals $(s1, e1)$ and $(s2, e2)$. `intersect_len()` checks a path for a consistent overlapping of alignment regions by looping over it's edges in a pairwise manner and applying `intersect_len()`. The minimum required length of the overlap can be defined by setting the value for `minlen` accordingly.

The method `check_eval()` checks all edges of a path regarding the HHblits E-value and returns `True` if they suffice a given threshold. The method `get_seqrecords()` returns BioPython `SeqRecord`s of the alignments and is used for generating an MSA as described in the next section. Pathfinder discards paths which do not overlap. A sufficient overlap of adjacent HHblits alignments in a path is a necessary requirement for the next component of FragStatt called Pathanalyzer.

2.2.3 Pathanalyzer

As a first step the component Pathanalyzer converts all paths as identified by Pathfinder to MSAs, implemented in the sub-component Aligner, by joining the individual HHblits pairwise alignments. Subsequently the MSAs are analyzed and a match matrix is calculated, which is the basis for the last module of FragStatt.

In this section the residues of the sequences of a path will be referenced according to the following formal definitions. The residues of P_1 and P_2 are defined as

$$Res(P_1) = \{p_1, p_2, \ldots, p_n\} \tag{2.1}$$

and

$$Res(P_2) = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_m\}\,, \tag{2.2}$$

with n and m being the length of P_1 and P_2. The residues of intermediate nodes will be referenced in a similar manner by the according lower-case letter, again indexed with the residue position e.g.

$$Res(A) = \{a_1, a_2, \ldots, a_k\} \tag{2.3}$$

and

$$Res(T) = \{t_1, t_2, \ldots, t_l\}\,, \tag{2.4}$$

with k and l being the length of A and T.

A pairwise alignment of two sequences α, β will be called $Ali(\alpha, \beta)$ in the following. The resulting aligned sequences with introduced gaps will be named $Ali_\alpha(\beta)$ and $Ali_\beta(\alpha)$, while $Ali_\beta(\alpha)$ refers to the sequence α aligned to β and $Ali_\alpha(\beta)$ refers to the sequence β aligned to α:

$$Ali(\alpha, \beta) = (Ali_\beta(\alpha), Ali_\alpha(\beta)) \tag{2.5}$$

A path consists of a sequence of HHblits alignments. For the exemplary case in Figure 2.3 in the previous section the path can be written as

$$Path(P_1, P_2) = (Ali(P_1, A), Ali(A, T), Ali(T, Q), Ali(Q, P_2)) \tag{2.6}$$

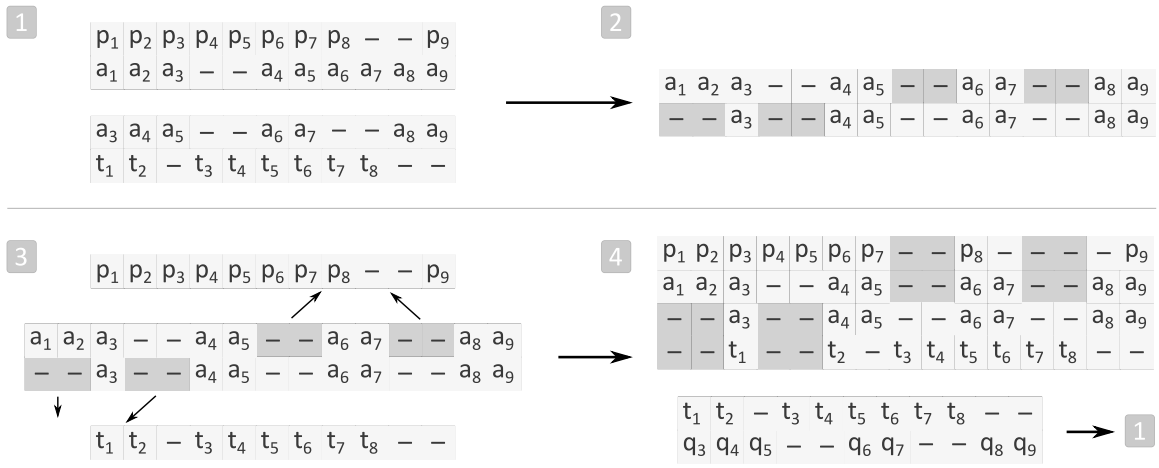


Figure 2.4: Function of Aligner.
The sub-component Aligner joins multiple pairwise alignments of an HHblits cascade path into an MSA. Step 1: The starting point are two individual pairwise HHblits alignments $Ali(P_1, A)$ and $Ali(A, T)$. The sequence A is part of both alignments but can have different gaps and alignment starting/ending positions. Step 2: Gaps of both initial aligned sequences $Ali_{P_1}(A)$ and $Ali_T(A)$ are combined and added to the respective other sequences (highlighted in dark grey). Step 3: The gaps added to $Ali_{P_1}(A)$ and $Ali_T(A)$ are transferred "up" or "down" to $Ali_A(P_1)$ and $Ali_A(T)$ (highlighted in dark grey). Step 4: The "regapped" sequences of $Ali_A(P_1)$, $Ali_{P_1}(A)$, $Ali_T(A)$ and $Ali_A(T)$ can be joined to an MSA. As indicated, this procedure is repeated until all pairwise alignments of the path are merged into an single MSA.

Naturally, because of the iterative nature of the HHblits cascade, consecutive alignments always share an identical hit and query sequence, e.g. A for $Ali(P_1, A)$ and $Ali(A, T)$ in the path from above (Equation 2.6). $Ali(P_1, A)$ and $Ali(A, T)$ are individual alignments; this means, A

can acquire different gaps for both cases (compare Figure 2.4, Step 1). The task of Aligner is to join all pairwise alignments which constitute a path and convert them into an MSA. This is done by successively adding the gaps from the pairwise alignment to the other alignments. For the case of $Path(P_1, P_2)$, the first step is to join $Ali(P_1, A)$ and $Ali(A, T)$. Gaps from $Ali(P_1, A)$ are added to $Ali(A, T)$ and vice versa (compare Figure 2.4, Step 2 and 3). This way $MSA(P_1, A, T)$ is generated (compare Figure 2.4, Step 4). Analogously the next pairwise alignment $Ali(T, Q)$ can be added to $MSA(P_1, A, T)$ resulting in $MSA(P_1, A, T, Q)$. Adding the last pairwise alignment $Ali(Q, P_2)$ yields the final result $MSA(P_1, A, T, Q, P_2)$. The Figure 2.5 illustrates Aligners function again using concrete sequences as examples.

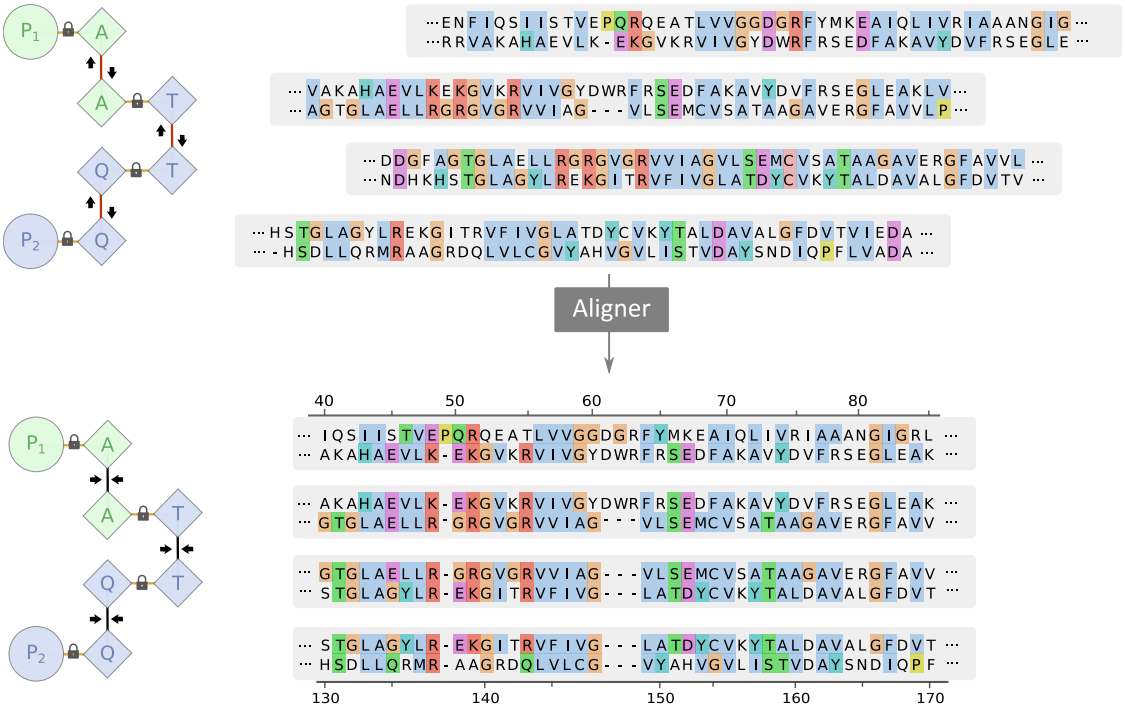


Figure 2.5: Example case for Aligner.
The path $Path(P_1, P_2)$ as defined in Equation 2.6 consists of four HHblits alignments. To join the alignments into an MSA, it is necessary to add gaps to all the sequences as described previously and illustrated in Figure 2.4. Aligner preserves the matching residues of the individual pairwise alignments and only adds gaps as necessary to create a valid MSA.

At this point, for each path a corresponding MSA was created and stored as a BioPython `MSA` object in the `msa` attribute of the previously introduced `Path` object (compare Figure 2.3). The second task of Pathanalyzer is to analyze the created MSAs.

The goal of Pathanalyzer is to calculate the match matrix H defined as

$$H(i,j) = \texttt{match\_count}(p_i, \tilde{p}_j). \tag{2.7}$$

The method `match_count()` evaluates how often the residue on position i of P_1 (i.e. p_i) was aligned to the residue on position j of P_2 (i.e. $\tilde{p}_j$) by processing all MSAs. In Figure 2.6 the procedure is shown for a single path, respectively MSA. Each column of the MSA gets checked for whether it is a match column, i.e. if no gaps occur in the column. The according element of H gets incremented for the positive case of a match column.

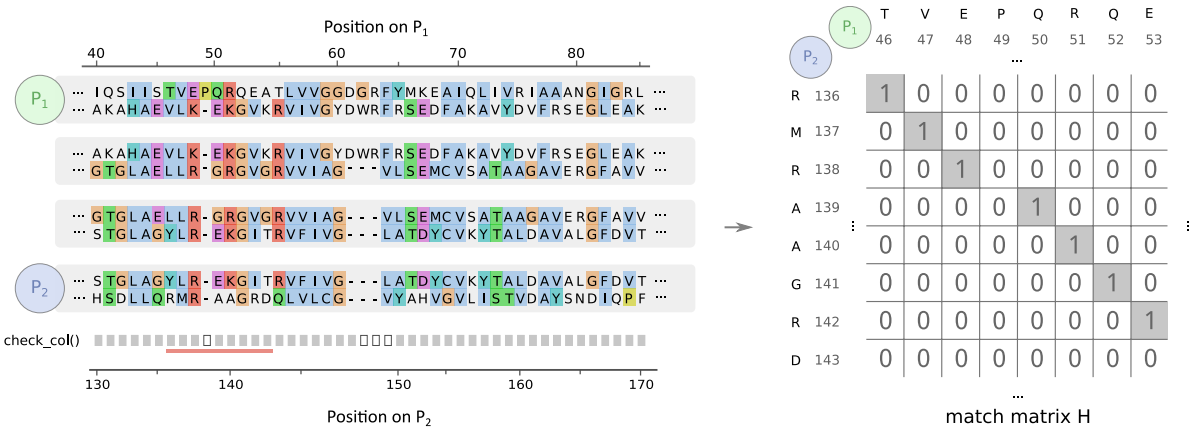


Figure 2.6: MSA match matrix.
To populate H, all path MSAs are processed identically in a column-wise manner. The approach is shown for a single path. Each column of the MSA is checked by a method `check_col()`, which returns *True* if the column does not contain any gaps and is a match column therefore (indicated by gray or white boxes). For the positive case of a match column, the corresponding position indices of the root sequences P_1 and P_2 are identified and the according element of the matrix H gets incremented. For simplicity the matrix is only shown for P_1 residue $46-53$ and P_2 residue $136-143$ (highlighted by the red bar in the bottom MSA position axis). The illustration only shows the first path being processed. However, all other paths are handled analogously and the counter values of H will increase accordingly.

All paths identified by Pathfinder are processed in this manner. However, in an actual implementation, instead of `match_count()` a method like `match_count_all()` as sketched out in Pseudocode 2.2 is used, which processes all paths and MSAs at once. The result of Pathfinder is the populated matrix H of dimension $n \times m$ according to the sequence length of P_1 and P_2 (see Equation 2.1 and Equation 2.2).

```
def check_col(col):
    if "-" not in col:
        return True

def match_count_all():
    M = zero_matrix(m, n)
    for p in paths:
        for col in p.msa:
            if check_col(col):
                M[col.pos_p1, col.pos_p2] += 1
    return M
```

Pseudocode 2.2: Calculating match count of MSAs. The function `match_count_all()` initializes a zero matrix of the dimension $n \times m$ (sequence length of P_1 and P_2) and iterates all paths and all columns of a path MSA. If a column is a match column according to `check_col()`, the corresponding element of matrix H is incremented. As the MSAs are comprised of local alignments it is necessary to map the MSA position back to the actual sequence positions of P_1 and P_2. This is indicated by the expression `col.pos_p1` and `col.pos_p2`.

2.2.4 SWiFD

The previously defined matrix H holds an integer value for all residue pairs $P_1 \times P_2$, which can be interpreted as an alignment score and gives information about how often a certain residue pair was aligned to each other. When visualizing the matrix H as dot plot it becomes apparent that a meaningful alignment will result in a continuous line. By means of a projection of the line to the position axis of P_1 and P_2, a mapping $P_1 \leftrightarrow P_2$ can be deduced (compare Figure 2.7). The purpose of the last component of FragStatt is to automate this approach by processing

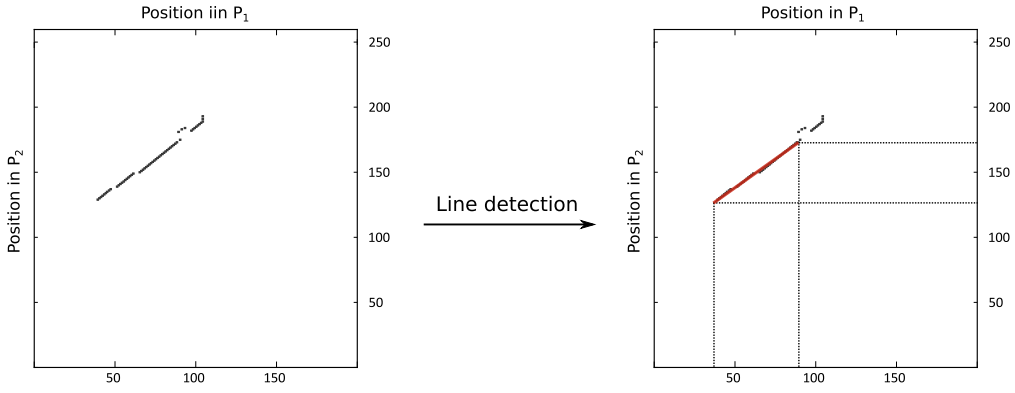


Figure 2.7: Example dotplot of a match matrix.
A match matrix H can be visualized as a dotplot. If a majority of the paths and their associated alignments target the same position on P_1 and P_2 it manifests as a "line" in the plot. By evaluating the dotplot and projecting the line to the corresponding positions on P_1 and P_2 one can deduced a potential common fragment. For this, it is necessary to detect aligned regions automatically, in other word to perform a line detection in the dotplot.

the matrix and detecting continuous sections of aligned regions in the matrix H. A modified version of the Smith-Waterman algorithm (Smith et al., 1981) called **S**mith **W**aterman **i**nspired **F**ragment **D**etection (SWiFD) was implemented to solve this problem. In the following this approach will be explained in detail.

Analogous to the classic Smith-Waterman algorithm the matrix S can be defined as follows.

$$S(i,j) = \max \begin{Bmatrix} 0 & \\ S(i-1,j-1) + u_H(i,j) & \text{match or mismatch} \\ S(i-1,j) + u_H(i,-) & \text{deletion} \\ S(i,j-1) + u_H(-,j) & \text{insertion} \end{Bmatrix} \tag{2.8}$$

The matrix S is linked to the previously introduced matrix H by using the following scoring function u_H:

$$u_H(x,y) = \begin{cases} s_{Match} & \text{if } H(x,y) > t \text{ (match)} \\ s_{Mismatch} & \text{if } H(x,y) < t \text{ (mismatch)} \\ s_{Gap} & \text{if } x = - \text{ or } y = - \text{ (gap),} \end{cases} \tag{2.9}$$

whereas s_{Match} is the match score, $s_{Mismatch}$ is the mismatch score and s_{Gap} is the gap penalty. Whether a position is considered matching (respectively mismatching) can be adjusted with setting the threshold t accordingly. To obtain the highest sensitivity, t is set to 0 by default.

In the following, the operating principle of SWiFD will be demonstrated using a small 8×8 match matrix based on the example from Figure 2.6. For the purpose of the demonstration the following two simplifications will be made:

1. The considered sequences of P_1 and P_2 are limited to positions $46 - 53$ and $136 - 143$.

2. Only one path (the path shown in Figure 2.6) was detected and evaluated, thus resulting in a binary match matrix H^*, i.e. only containing values 0 or 1.

Making these two assumptions results in the matrix shown in Figure 2.8, which formally can be

defined as:

$$H^*(i,j) = H(i,j)_{i\in\{46,\ldots,53\},j\in\{136,\ldots,143\}} \tag{2.10}$$

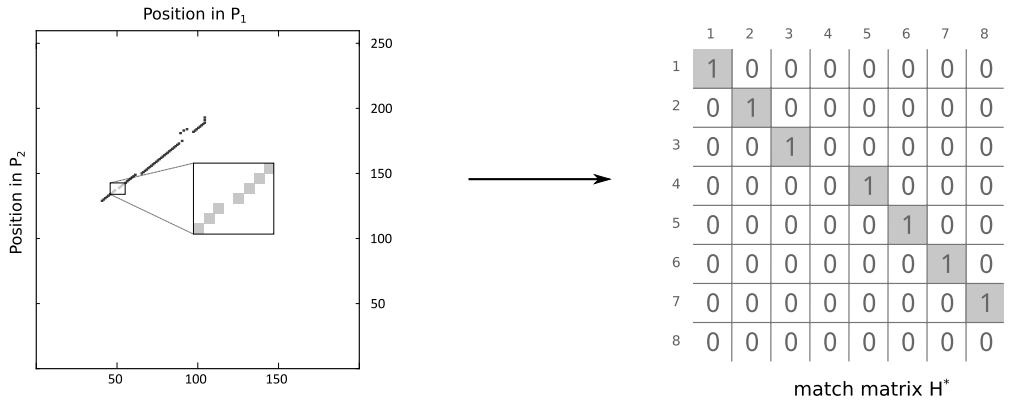


Figure 2.8: Defining a submatrix.
On the left side the dotplot of a FragStatt run can be seen. The path shown in Figure 2.5 and Figure 2.6 is also based on the same example data. For simplicity regarding a demonstration of SWiFD a 8×8 submatrix of H called H^* will be extracted. This can be interpreted as if P_1 and P_2 had a sequence length of 8 amino acid residues. The representation of the "line" in the matrix appears flipped as the indices (position values) in the matrix representation increase from top to bottom.

The Smith-Waterman matrix S can be calculated based on Equation 2.8 and the specific scoring function

$$u_{H^*}(x,y) = \begin{cases} +2 & \text{if } H^*(x,y) > t \text{ (match)} \\ -5 & \text{if } H^*(x,y) < t \text{ (mismatch)} \\ -5 & \text{if } x = - \text{ or } y = - \text{ (gap).} \end{cases} \tag{2.11}$$

The values for s_{Match}, $s_{Mismatch}$ and s_{Gap} were chosen to be convenient for the purpose of this demonstration but other than that in this case they are arbitrary and have no practical meaning. In Figure 2.10 the result of the described method can be seen. Matching positions are connected and the score increases. Gaps can be bridged at the cost of score reduction.

The length of the detected tracebacks can be controlled by setting $s_{Mismatch}$ and s_{Gap} accordingly. A gap penalty value of -6 for example would result in two tracebacks as the score in S would reset to 0 after the initial $2, 4, 6$ sequence when hitting the first gap. The traceback defines the aligned regions and a mapping $P_1 \leftrightarrow P_2$ can be deduced (compare Figure 2.10). To

check for structural similarity TM-align (Zhang and Skolnick, 2005) is used.

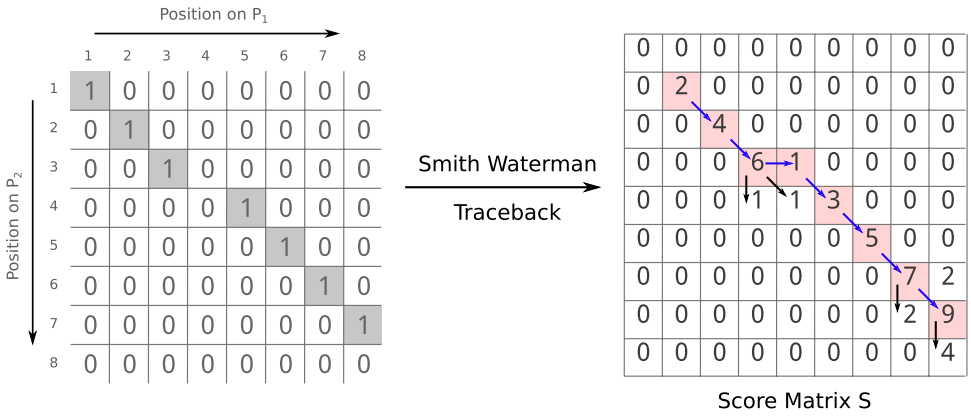


Figure 2.9: Calculating the Smith-Waterman matrix.
The Smith-Waterman matrix S is calculated based on H^* and the scoring function u_{H^*}. The predecessor of each element is stored during the filling of the matrix, indicated by arrows. A traceback is carried out starting from the highest score value by tracing back the predecessor elements (red boxes and blue arrows).

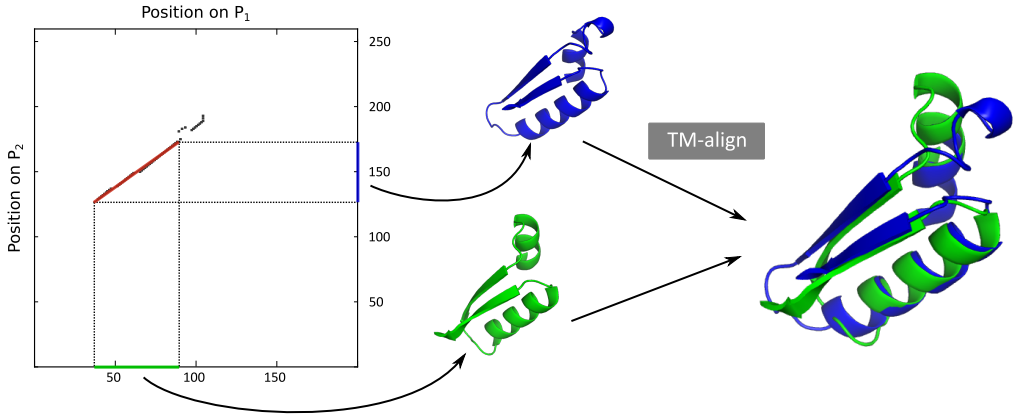


Figure 2.10: Result of SWiFD.
The traceback detected by SWiFD is shown in red. The projection of the traceback on each axis specifies a mapping $P_1 \leftrightarrow P_2$ defining two subfragments of P_1 and P_2. The two fragments can be superimposed by TM-align.

2.3 Testing and Benchmarking FragStatt

2.3.1 Detecting a previously reported ancestral barrel fragment

The FragStatt run for `1WA3_A` and `4JGI_A` was performed using the default parameters. All detected fragments and their positions were extracted from the generated CSV file and are

listed in Table 3.2. The dotplot output was prepared and colored in a meaningful way. Axes were limited to the relevant regions. Visualizations were generated using PyMol. To deduce the Pfam distribution, the FragStatt output file paths.txt was parsed using a Python script. The Pfam↔SCOP mapping was generated by linking the mappings pdb2pfam and pdb2scop, which are based on the files pdb_pfam_mapping.txt (European Bioinformatics Institute, 2020) and dir.des.scope.2.06-stable.txt (Fox et al., 2020) provided by Pfam and SCOP. The histogram was plotted using the Python libraries pandas (McKinney, 2010) and matplotlib (Hunter, 2007).

2.3.2 Reevaluation of a set of putative ancestral motifs

The file elife-09410-fig3-data1-v2.docx (Alva et al. 2015a, excerpted in Figure 3.10), which is provided as an additional online resource, was parsed and converted to a CSV file containing all meta data of the reported fragments: PDB ID, SCOP ID, start position, end position. This resulted in a list of PDB protein chains C_i for each motif $i \in (1, 40)$. For each motif all combinations of PDB protein chains were generated in the following manner:

$$L_{DiffFold,i} = \{\{a, b\} \mid a, b \in C_i,\ a \neq b,\ SCOPFold(a) \neq SCOPFold(b)\} \tag{2.12}$$

Instead of the CATH annotation which is the main structural classification system used in the this work, in this case the SCOP fold classification was utilized, as the work of Alva et al. is based on it, i. e. each of the pairs in $L_{DiffFold,i}$ feature a different SCOP fold. After compiling this list, all fragments were downloaded and saved as PDB files using PyMol. For each motif all combinations $L_{DiffFold,i}$ were submitted to a FragStatt run with the following parameters:

`-search_depth 4`

`-search_mode all`

`-max_depth 3`

`-min_overlap 5`

Subsequently, the results were evaluated for each motif: It was checked whether for a given PDB protein chain the fragment reported by Alva et al. was detected. A fragment was considered to be correctly detected, if the region reported by FragStatt covered at least 75 % of the region

defined by Alva et al. Correctly identified fragments were counted on a PDB protein chain level (Table 3.3). A motif was considered to be detected, when at least one manifestation in a PDB protein chain was detected, which led to the overall detection rate of 85 %.

2.4 Large-scale Scan for Ancestral Protein Motifs

2.4.1 Data acquisition and preparation

The Pfam-A database (El-Gebali et al., 2018) version 32.0 and the Protein Data Bank (PDB) (Berman et al., 2002), snapshot May 2019, served as the fundamental data basis for the analysis. For the all vs. all HHblits (Remmert et al., 2012) runs, the pre-built Pfam-A database was retrieved from the HHsuite web server (Steinegger et al., 2019b). The Pfam queries were extracted in A3M format from the HHblits database using the script extract_a3m.py. For the PDB queries, the file pdb_seqres.txt, containing the sequences of all PDB entries separated in single chains, was downloaded from the PDB FTP-Server (Berman et al., 2019). The file was split into single FASTA files. In this manner, a total of 463,366 queries, 17,929 Pfam MSAs and 445,437 PDB sequences, were generated. Each one of the queries was submitted to a HHblits search using the Pfam database (compare Figure 2.11, Step 1). Except for disabled secondary structure scoring (-ssm 0), the standard parameters for HHblits were used. The 463,366 runs were divided into 34 batches and computations were trivially parallelized on the shell-level using xargs. All computations were carried out on the high performance computing cluster Athene at University of Regensburg.

The first step of the data preparation resulted in 463,366 HRR HHblits result files. Subsequently, in the next step (compare Figure 2.11, Step 2) these files were parsed using the hh_reader.py (Meier, 2017) script provided by HHsuite. For each hit the relevant informations, namely query ID, template ID, query start, query end, template start, template end, probability, E-value, query sequence, and template sequence were extracted. Hits were taken into account up to an E-value of 1. If more than 500 hits were present for a single query, a reduced subsample of 500 hits was generated.

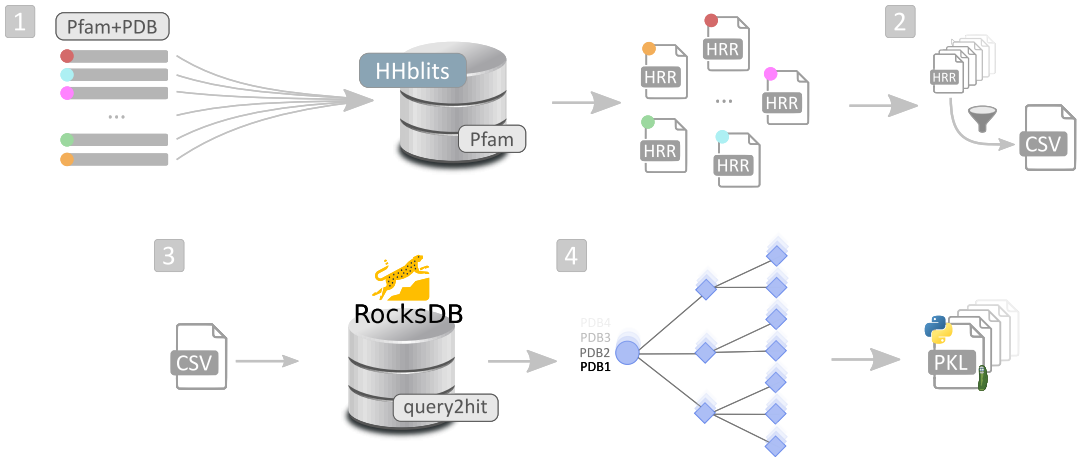


Figure 2.11: Data preparation.
Step 1: All Pfam HMMs and PDB sequences were used as queries for a HHblits search in the Pfam database. The first step resulted in an HHblits HRR result file for each query. Step 2: The result files were parsed and relevant information is extracted. The number of extracted hits per query was reduced to a maximum of 500. The remaining hits were collected in a CSV file. Step 3: To enable quick access to all hits, for a given query a key-value database using RocksDB was built. Step 4: By traversing the query2hit database starting at a certain sequence, a tree based on the cascading of pairwise HHblits alignments can be generated. This was done for each PDB entry in the dataset. The generated trees were stored as edge lists in the binary Python pickle format.

The extracted hits were collected in an intermediate CSV file, which was then converted into a key-value database query2hit (compare Figure 2.11, Step 3) using RocksDB (Facebook, 2020) and the Python bindings pyrocksdb (Hofmockel, 2015). Each line (HHblits hit) in the CSV file was translated to a key-value pair: The i-th hit of a query was assigned the key "`ID_I`", e.g. `1ABC_A_3` for the PDB query `1ABC`, chain `A`, hit number 3. The value holds the aforementioned information on the hit.

In the final step of data preparation, the query2hit database was used to create an HHblits cascade starting from a certain query (compare Figure 2.11, Step 4). For this, the database was traversed in an recursive manner. For example, starting at the query `1ABC_A`, the hit `1ABC_A_0` would lead to the next query in the cascade (e.g. `PF000001`) which again would have hits to other Pfam entries (e.g. `PF000002`), and so on. In this manner, a tree starting from a root sequence can be created. Nodes represent sequences or respectively Pfam profiles and edges represent the connecting HHblits hits. As this recursive approach can produced huge graphs if not restricted by a reasonable stop criteria, the following parameters were introduced: Maximum number of children per node: 500, maximum depth of traversal: 3 hops. Also, an option to limit

the E-value range was implemented. However, this parameter was kept at the maximum range, meaning an upper threshold of 1 and a lower threshold of 0 for the E-value.

...

query	Template	qs	qe	ts	te	prob	evalue	q_seq	t_seq
1GPW_A	PF00121.18	195	233	197	235	89.26	0.013	TLPIIASGGA...	DLIIQYGGSV...
1GPW_A	PF00215.24	197	239	172	214	93.44	0.0014	PIIASGGAGK...	AGGDNLGQ-Q...
1GPW_A	PF00218.21	31	239	66	248	99.02	1.9e-14	DPVELGKFYS...	QPGEIARVYS...
1GPW_A	PF00290.20	30	229	19	227	98.58	6.2e-12	GDPVELGKFY...	ESSLEYVEAL...
1GPW_A	PF00478.25	154	235	273	364	98.33	7.9e-11	RDWVVEVEKR...	YDGAMDLINL...
1GPW_A	PF00697.22	182	241	127	183	89.54	0.012	YDTEMIRFVR...	LNLKAIPHIS...
1GPW_A	PF00701.22	154	247	25	129	59.01	0.56	RDWVVEVEKR...	EKLVEHHIEA...
1GPW_A	PF00724.20	155	241	230	321	94.34	0.00058	DWVVEVEKRG...	GFAKWMKEQG...
1GPW_A	PF00724.20	31	118	227	319	97.66	1.4e-08	DPVELGKFYS...	DHIGFAKWMK...
1GPW_A	PF00834.19	62	228	43	196	98.58	6.4e-12	MLELVEKVAE...	GPVVLENVTQ...

...

Table 2.1: Excerpt from the hitlist CSV file. The table shows exemplary data from the CSV file generated in the second step of data preparation (compare Figure 2.11, Step 2). The excerpt contains hits of the PDB query `1GPW_A` to various Pfam families. For better readability the column headers for query start (qs), query end (qe), template start (ts), template end (te) were abbreviated and as well the sequences were truncated.

Trees were calculated for each of the 445,437 PDB queries from the initial dataset. For later use in FragStatt, each tree was stored as a simple edge list, packaged in a Python pickle binary file.

2.4.2 All-vs-all based on CATH

The all vs. all dataset of PDB entries was compiled based on the CATH database version 4.2 (Orengo et al., 2002). A list of all CATH domains filtered to a maximum sequence identity of 35 % was obtained by downloading the file cath-domain-list-S35-v4_0.txt from the CATH webserver. Based on this file, 12,623 single-domain PDB chains were extracted, resulting in the list L_{SDC}. A list of all possible combinations (pairs) of these PDB chains was generated with the constraint that the CATH topology differs:

$$L_{DiffTopo} = \{\{a, b\} \mid a, b \in L_{SDC},\ a \neq b,\ Topo(a) \neq Topo(b)\} \tag{2.13}$$

The list of unordered PDB chain combinations $L_{DiffTopo}$ contained 77,847,546 pairs covering 867 CATH topologies.

Based on the list $L_{DiffTopo}$, for each pair a FragStatt run was carried out. The runs were split into batches and calculations were distributed on the High Performance Computing Cluster Athene at University of Regensburg. The computational workload cumulated to approximately 11 000 CPUh. The following parameters of FragStatt were used:

```
-search_depth 4
-search_mode all
-max_depth 3
-min_overlap 5
-max_num_paths 400
```

Of the total of 77, 847, 546 analyzed pairs 936, 792 (1.2 %) gave positive results, i.e. paths from P_1 to P_2 were found.

2.5 Evaluation of Motifs

2.5.1 Filtering of the hits

All raw hits delivered by FragStatt in CSV format were parsed and filtered in the following manner: The hits were reduced to those comparing two PDB protein chains which belong to different CATH architectures. All hits coming from the CATH topology 1.20.5 "Single alpha-helices" were filtered out. All hits of combinations of sandwich architectures (CATH 3.30, 3.40, 3.50, 3.55, 3.60, 2.60) were dismissed. The same was done for all propeller architectures (CATH 2.105, 2.110, 2.115, 2.120, 2.130, 2.140). The combination CATH 3.90.180 and 3.40.50 was also filtered out because of a putative homologous background. Hits with an alpha-helix content of more then 90 % were also removed. The maximal length difference of the two fragments specifying one hit was set to 5 residues. The lower length threshold was set to 15 and the upper threshold to 60 amino acids. The minimum TM-score requirement was set to 0.55 The remaining hits were collected and again stored in CSV format.

2.5.2 Hit count statistics

The number of hits were counted on the architecture and topology level of CATH. This was done for the unfiltered and filtered dataset (see Section 2.5.1). Raw counts were also translated to relative frequencies. Additionally, the abundance and distribution of all CATH entries and of all single-domain chain CATH entries were calculated. For this the file cath-domain-list-S35.txt (all CATH entries up to 35 % sequence identity) was parsed and entries were counted. Again, for comparability the relative frequencies were computed.

2.5.3 Generation of CATH classification networks

For the generation of the networks shown in Figure 3.15 and Figure S1 the software Cytoscape (Shannon et al., 2003) was used. The architecture and topology combinations and their associated counts were loaded from a previously generated CSV file and imported as edge list. The yFiles Circular Layout was applied and the edge thickness was scaled according to the abundance of the topology respectively architecture pair.

2.5.4 Evaluation of length distribution

For each fragment F the length was calculated by evaluating the end position e and the start position s in reference to its protein chain of origin. Length values were plotted as an histogram as shown in Figure 3.16. To visualize the abundance of different CATH architectures in the histogram, the length distribution was calculated for each CATH architectures pair individually and plotted as a stacked bar plot using matplotlib and scipy.

2.5.5 Generation of random fragments

For each pair of fragments detected by FragStatt an artificial random pair was generated. The following will explain the procedure for a given pair of fragments $\{F_1, F_2\}$, whereas F_1 is the manifestation of the common motif in the PDB chain P_1 and F_2 is the manifestation of the common motif in the PDB chain P_2. F_1 and F_2 are defined each by a starting positions s_1 and

s_2 and end positions e_1 and e_2 which refer to a position on P_1 respectively P_2. F_1 and F_2 are defined as

$$F_1 = P_1[s_1, e_1] \tag{2.14}$$

and

$$F_2 = P_2[s_2, e_2]. \tag{2.15}$$

For each detected pair $\{F_1, F_2\}$ a random pair $\{\tilde{F}_1, \tilde{F}_2\}$ was generated in the following way: At first two randomly sampled PDB protein chains $\tilde{P}_1$ and $\tilde{P}_2$ were obtained. For this, all PDB protein chains for which FragStatt detected common fragments served as the sampling seed. Next, the lengths l_1 and l_2 of the original fragments were calculated.

Subsequently two new tuples of random end and start positions with length l_1 and length l_2 were generated: $(\tilde{s}_1, \tilde{e}_1)$ and $(\tilde{s}_2, \tilde{e}_2)$. It was ensured that the newly sampled start and end position were within the maximal length of $\tilde{P}_1$ and $\tilde{P}_1$. In this way, the artificial pair of fragments $\{\tilde{F}_1, \tilde{F}_2\}$ can be fully defined by

$$\tilde{F}_1 = \tilde{P}_1[\tilde{s}_1, \tilde{e}_1] \tag{2.16}$$

and

$$\tilde{F}_2 = \tilde{P}_2[\tilde{s}_2, \tilde{e}_2]. \tag{2.17}$$

The described random sampling procedure was carried out for each detected pair of fragments. If the randomly sampled regions were partly not resolved in the sampled PDB protein chain, the sampling was repeated until a fully resolved fragment could by generated. To prevent infinite loops the maximum number of retries was limited to 10.

The fragments were loaded in PyMol and the TM-score was calculated. The fragments were saved as PDB files and the pairwise TM-scores were collected in a CSV file.

2.5.6 Evaluation of TM-score distribution

The filtered hits (filtering described in Section 2.5.1) and their according TM-scores were collected. The same was done for the randomly generated fragment pairs (Section 2.5.5). The TM-score distributions of the two datasets were plotted as a histogram as shown in Figure 3.17. Both a t-test and a Wilcoxon signed-rank test were carried out for the two samples. The t-test (Student, 1908) as well as the Wilcoxon signed-rank test (Wilcoxon, 1992) gave highly significant results of a p-value $< 1E - 99$. The scipy package (Virtanen et al., 2020) was used for the statistical testing.

2.6 Clustering

2.6.1 Additional redundancy removal

CD-HIT (Li and Godzik, 2006) was used to further reduce redundancy in the hits on the sequence level. For this a FASTA, file containing all fragments and their corresponding sequences was generated. The FASTA file was fed into CD-HIT using default parameters. In this manner, the individual fragments were clustered independently of the links made by FragStatt into clusters of 90 % sequence identity. The cluster centers and the cluster members were stored in the CD-HIT CLSTR format.

2.6.2 All vs. all TM-align

For all candidate fragments, an all vs. all distance matrix based on a pairwise TM-score as delivered by TM-align was calculated. If n fragments $F_1, F_2, \ldots, F_n$ are defined as the cluster centers of the previously explained redundancy removal, a similarity matrix $A = (a_{ij})$ can be defined as follows:

$$a_{ij} = t_{ij} = TMScore(F_i, F_j) \tag{2.18}$$

$$A = \begin{bmatrix} 1 & t_{12} & t_{13} & \dots & t_{1n} \\ t_{21} & 1 & t_{23} & \dots & t_{2n} \\ t_{31} & t_{32} & 1 & \dots & t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \dots & 1 \end{bmatrix} \tag{2.19}$$

The distance matrix $B = (b_{ij})$ can be derived from A:

$$b_{ij} = 1 - a_{ij} \tag{2.20}$$

$$B = \begin{bmatrix} 0 & 1 - t_{12} & 1 - t_{13} & \dots & 1 - t_{1n} \\ 1 - t_{21} & 0 & 1 - t_{23} & \dots & 1 - t_{2n} \\ 1 - t_{31} & 1 - t_{32} & 0 & \dots & 1 - t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 - t_{n1} & 1 - t_{n2} & 1 - t_{n3} & \dots & 0 \end{bmatrix} \tag{2.21}$$

A modified distance matrix $B^* = (b^*_{ij})$ adjusted for a TM-score threshold of 0.55 was defined:

$$b^*_{ij} = 1 \text{ if } b_{ij} > 0.45 \text{ otherwise } b_{ij} \tag{2.22}$$

The matrix B^* was used as the input for a `DBSCAN` clustering. The parameter $minPts$, which defines the minimum number of neighbors a point must have to be considered as core point was set to 2. The parameter ϵ defines the neighborhood radius and was chosen by optimizing the number of clusters and the average TM-score of all pairs within each cluster.

2.7 Search for Multi-Motif Proteins

The set of pairwise common motifs CoMo was parsed and for each protein all detected motifs were collected. Subsequently, for a protein P with n detected pairwise motifs M_i the set of all

fragments was defined:

$$Frags(P) = \{F_i = P(M_i) \mid i \in [1, n]\} \tag{2.23}$$

A fragment F_i is further defined by its start an end position in reference to P

$$F_i = P[s_i, e_i] \tag{2.24}$$

To assess the number of unique regions, i.e. positionally distinct groups of fragments F_i, the all vs. all Hausdorff distance was calculated based on the start and end position s_i and e_i of the fragments. For two intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$ the one dimensional Hausdorff distance can be defined as

$$d_{Hausdorff}(A, B) = max(|a_1 - b_1|, |a_2 - b_2|). \tag{2.25}$$

as stated by Chavent (2004).

In this manner, the distance between two fragments was defined as

$$d_{Hausdorff}(F_i, F_j) = ||F_i - F_j|| = max(|s_i - s_j|, |e_i - e_j|). \tag{2.26}$$

A distance matrix $H = (h_{ij})$ was calculated according to

$$h_{ij} = ||F_i - F_j||. \tag{2.27}$$

The matrix H was used as the input for a `DBSCAN` clustering which was performed for each of the protein in the CoMo set. The parameter $minPts$, which defines the minimum number of neighbors a point must have to be considered as core point was set to 2. The parameter ϵ defines the neighborhood radius and was set to 10.

For each protein the result of the clustering was evaluated and those proteins with a minimum of two clusters were collected resulting in a list of multi-motif protein chains. A result file

containing the cluster members and according CATH classification was compiled. To ease visual inspection, for each protein chain a PyMol session was created showing the superposition of the fragments colored according to their cluster membership.

Chapter 3

Results

Separated by billions of years of divergent evolution, one has to assume that the similarity of the proposed ancestral motifs is vanishingly low and barely detectable. The development of an algorithm to detect such faint homologous signals with highest sensitivity lays the foundation for an comprehensive scan for the proposed ancestral remnants. The design, implementation and application of such a software tool was the central aim of this thesis. By applying such an algorithm to a representative set of proteins, it is possible to define a collection of putative ancestral motifs. This set serves as a data basis for further statistical analysis which delivers insights in the distribution and abundance of putative motifs.

In the following the terms motif and fragment will be used extensively, it is therefore important to precisely define them. A super-secondary structure arrangement which can be found in at least two proteins will be called a *motif*. The manifestation of a certain motif in a protein will be called a *fragment*. The fragments defining a common motif must have a detectable sequence and structure similarity.

For a protein chain with known sequence and structure called P the manifestation F of the motif M will be referenced to as

$$F = P(M) = P[e, s] \tag{3.1}$$

whereas e and s are the start and end position of the fragment F in the protein chain P.

3.1 FragStatt: An Algorithm to Detect Putative Ancestral Protein Motifs

FragStatt (an acronym for **frag**ment in**stan**ti**a**tion) is a dedicated software aimed at the detection of common motifs shared by proteins, which are non-homologous in a classical sense. The algorithm is trimmed to detect relatively short local sequence similarities, i.e. common fragments in otherwise highly dissimilar proteins. It furthermore requires a significant structure similarity of the detected fragments. Based on the findings from other studies (Alva et al., 2015b; Farías-Rico et al., 2014) work, the size of the stated motifs seems to have its lower boundary in the realm of super-secondary-structure elements (e.g. helix-turn-helix motifs or beta-hairpin motifs) at approximately 15 residues. As the postulated individual motifs are not considered to define a protein fold on their own, the upper boundary can be located in the sub-domain range at approximately 60 residues.

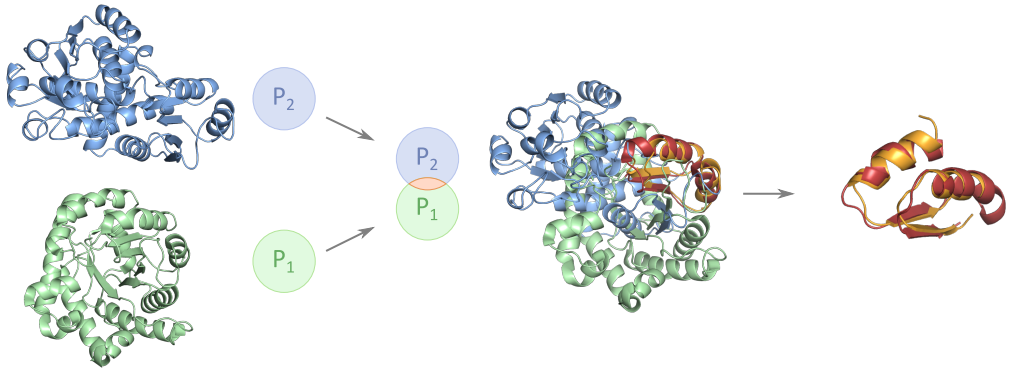


Figure 3.1: Basic principle of motif detection.
Two proteins P_1 and P_2 are analyzed and scanned for local sequence and structure similarities. While globally the proteins feature different folds and thus no overall similarity, locally a common motif, in this case a $(\alpha\beta)_2$ element, can be identified. In this manner, the pairwise comparison of two proteins, aiming at the identification of local common regions, is the basic approach of FragStatt.

The basic principle of FragStatt is depicted in Figure 3.1. The algorithm implements a pairwise comparison approach: Two proteins, referred to as P_1 and P_2, which feature a distinct fold and thus are not considered to be homologous, are scanned for local sequence similarities. For some cases such an analysis yields one or more common shared elements of the previously mentioned size of approximately 15 to 60 residues. For these positive cases FragStatt returns a mapping $P_1 \leftrightarrow P_2$ which defines the fragments F_1 and F_2 constituting a motif M. For instance the two

proteins share a common motif at residue position $1-50$ and $50-100$, respectively: $F_1 = P_1(M) = P_1[1,50]$ matches $F_2 = P_2(M) = P_2[50,100]$. The detected fragments in P_1 and P_2 are scored and filtered regarding their three-dimensional similarity and can be visualized in PyMol (Schrödinger, LLC, 2019).

From a technical perspective FragStatt is based on the comparison of *profile Hidden Markov Models* (HMMs). HMMs are currently the state-of-the-art method for the detection of remote homology, and have been widely used to study ancient protein evolution and unveil relationships between protein folds that might date back to the beginning of life on earth (Coles et al., 2005; Alva et al., 2010). Moreover, studies similar to this work and also aimed at the identification of ancestral protein motifs, have utilized HMMs as well (Alva et al., 2015b; Farías-Rico et al., 2014).

As mentioned in the introduction, there are several different software packages available offering HMM based homology detection. In this work the tool HHblits was used to generate the data basis for FragStatt. In general, HHblits can be seen as an analog method to classical homology detection algorithms like the well-known BLAST (Altschul et al., 1997) software. However, while BLAST serves very well for the detection of highly or moderately similar proteins, the HMM based HHblits is optimized for the discovery of remote homologs, which share only little sequence similarity (Park et al., 1998). Searching with the highest possible sensitivity is the key-feature which makes HHblits the appropriate choice to uncover unknown ancestral relationships between proteins.

As stated above, this work is not the first study aimed at identification of ancestral protein motifs. The main feature of FragStatt which differentiates it from the approaches made in other studies (Alva et al., 2015b; Farías-Rico et al., 2014) is the cascading of HMM based homology detection runs. This practice is aimed at maximizing the sensitivity of the homology detection at the expense of an increased false positive rate. The following shall briefly explain this key difference:
A classical HMM based homology detection tries to find direct hits for a given query P to a database HMM called target T:

$$P \rightarrow T$$

The HMM T is part of a pre-generated database of HMMs, e.g. the Pfam database.

It is known that in this manner, algorithms like HHblits can detect three times as many remote homologs as pairwise methods like BLAST (Park et al., 1998). However, the sensitivity can be increased even further by cascading HMM based search runs (Kaushik et al., 2016). The cascading variant of the HMM homology detection does not make a query-target connection directly but uses intermediate HMMs I_i:

$$P \rightarrow I_1 \rightarrow I_2 \rightarrow \cdots \rightarrow I_n \rightarrow T$$

In this way the search space can be extended and very weak homologous signals can be traced. The implementation of this approach based on HHblits is called *HHblits cascade* which is a central part of FragStatt. While discussing the advantages and drawbacks of an HHblits cascade will be part of Chapter 4, the practical aspects of the algorithm, i.e. the working principle and the application of FragStatt, will be presented in the following.

At this point, is important to mention that with a HHblits cascade as defined above only targets within the HHblits database can be reached, i.e. T and all I_i are part of a pre-built HHblits database. The query P however, is not part of the HHblits database. In the case of FragStatt two queries P_1 and P_2 shall be connected. This can not be achieved with a single HHblits cascade. Instead, two individual HHblits cascades are calculated, one for P_1 and one for P_2. Subsequently the individual HHblits cascades are connected. In this manner, FragStatt takes two "root proteins" P_1 and P_2 with known 3D structure as input. Based on the sequence of the proteins, the two HHblits cascades are calculated which can then be linked together (compare Figure 3.2). By identifying the intersection of the HHblits cascades, FragStatt can deduce local similarities, i.e. a common motif between the root proteins. While the generation and analysis of the HHblits cascade is solely based on sequence data, the final evaluation of the detected motifs requires a protein structure to be available, as it is based on scoring the structural similarity.

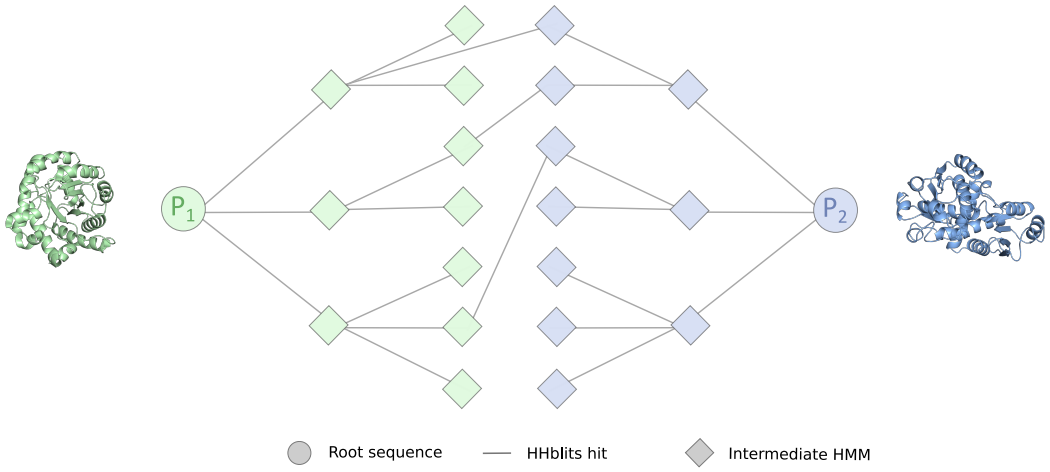


Figure 3.2: Linking two HHblits cascades.
Based on the sequence of two root proteins P_1 and P_2 FragStatt calculates two HHblits cascades and links them by identifying the intersections. The intersections represent a path from P_1 over multiple intermediate HMMs to P_2.

3.2 Working Principle and Implementation of FragStatt

The data basis for FragStatt are two HHblits cascades, each starting from one of the root proteins P_1 and P_2. In the next section it will be shown that by connecting and traversing HHblits cascades FragStatt can find alignments between P_1 and P_2. Before that, the HHblits cascade has to be introduced formally and explained in general for a protein P.

Figure 3.3 illustrates the concept of an HHblits cascade: The cascade starts with the sequence of P submitted as the first HHblits query. Subsequently, each hit of the run is fed into HHblits again, leading to new hits. In this iterative manner a tree rooted at the starting sequence is spanned. The nodes of the graph are HMMs, and the edges indicate HHblits hits, which in turn each define a local sequence alignment. FragStatt computes two HHblits cascades based on the root sequences of P_1 and P_2. With increasing depth HHblits cascades discover a exponentially increasing number of nodes. As computational resources are limited this property is problematic. There are different parameters of the HHblits cascade which can be adjusted to keep its final size manageable .

The graph resulting from an HHblits cascade depends on multiple parameters of FragStatt that have to be chosen by the user: For each hit (i.e. alignment) of HHblits an E-value is reported,

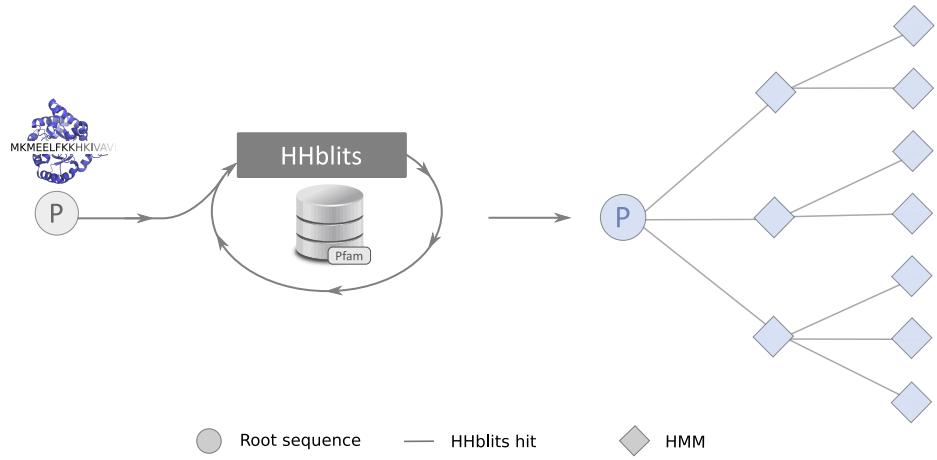


Figure 3.3: HHblits cascade.
The sequence of the root protein P serves as the starting point for an HHblits cascade. The sequence of P is submitted to the initial HHblits run, which delivers hits to the database (in this case Pfam). These hits are again submitted to another round of HHblits. By repeating this process a defined number of times a tree graph with P as the root can be generated.

Table 3.1: Parameters of the HHblits cascade.

Parameter	Description	Default
`eval_upper`	Upper threshold of hit E-value	1
`eval_lower`	Lower threshold of hit E-value	0
`max_children`	Maximum number of children per node	500
`max_depth`	Maximum depth of the cascade	3

which signals the significance of the alignment. A lower and upper threshold of the E-value can be defined to decide whether a hit should be included in the cascade or not. Each HHblits run delivers a certain number of hits to the database. To keep the growth of the cascade per iteration in a reasonable extent, the maximum number of children per node needs to be limited, which represents another parameter. Finally the number of iterations (depth of the cascade) has to be defined. The mentioned parameters and their default values are listed in Table 3.1. A maximum depth of 3 and a maximum number of 500 children per node turned out to be a suitable choice to keep the size of the cascades within a reasonable range. The standard values for the E-value thresholds of HHblits were set high in order to reach highest sensitivity (at the expense of an increased false positive rate). The idea behind this approach is that the final component SWiFD will filter out noise in the form of "true" non-significant hits . This concept will be discussed in more detail in Chapter 4.

HHblits cascades can be precomputed for every PDB entry and stored, as described in Section 2.4.1. For the generation of an HHblits cascade an HHblits database needs to be chosen and in this work the Pfam database (El-Gebali et al., 2018) was used. Technically any HHsuite database could be used, however, as discussed later in Chapter 4, choosing a suitable database is crucial for this method to succeed.

Now that the principle of an HHblits cascade was demonstrated, in the next part, the components of FragStatt will be introduced. Figure 3.4 shows the four core components of FragStatt: GraphCreator, Pathfinder, Pathanalyzer, and SWiFD, which can be seen as a pipeline and are executed in succession.

The obligatory input for FragStatt are two PDB IDs, including a chain identifier (e.g. `1XYZ_A`), which define the root proteins P_1 and P_2. The first component, GraphCreator (compare Figure 3.4, top-left panel) generates the HHblits cascades based on the two root sequences and subsequently creates a combined graph, from the two individual trees. In the actual implementation of FragStatt the HHblits cascades for all PDB protein chains are already pre-computed.

The next element in the pipeline, called Pathfinder (compare Figure 3.4, top-right panel), takes the combined graph and uses a *depth-first search* (DFS) approach to identify paths connecting P_1 and P_2. All detected path are passed to Pathanalyzer (compare Figure 3.4, bottom-left panel). A path defines a series of local alignments connecting the sequences of the root proteins through multiple HMMs. At this point the individual pairwise alignments are independent of each other and the hits must not necessarily overlap. Consequently, the first task of Pathanalyzer is to check whether the hits of the alignments comprising the path are continuously overlapping. If the local alignments of a path fulfill this requirement, they are merged to an MSA. After Pathanalyzer has finished its tasks, for each path their is a valid MSA available, which always includes the sequences of the root proteins P_1 and P_2 (mediated by consensus sequences of Pfam HMMs). The set of path-specific MSAs is the starting point for the final component of FragStatt SWiFD (compare Figure 3.4, bottom-right panel). Each matching column in an MSA defines a residue-wise mapping $P_1 \leftrightarrow P_2$. By analyzing all columns of all MSAs, a match count can be deduced for each residue pair $(p_i, \tilde{p}_j)$ with $p_i \in P_1$, $\tilde{p}_j \in P_2$. The match count indicates how often a pair $(p_i, \tilde{p}_j)$ was aligned to each other. The match count of all residue pairs can be stored in

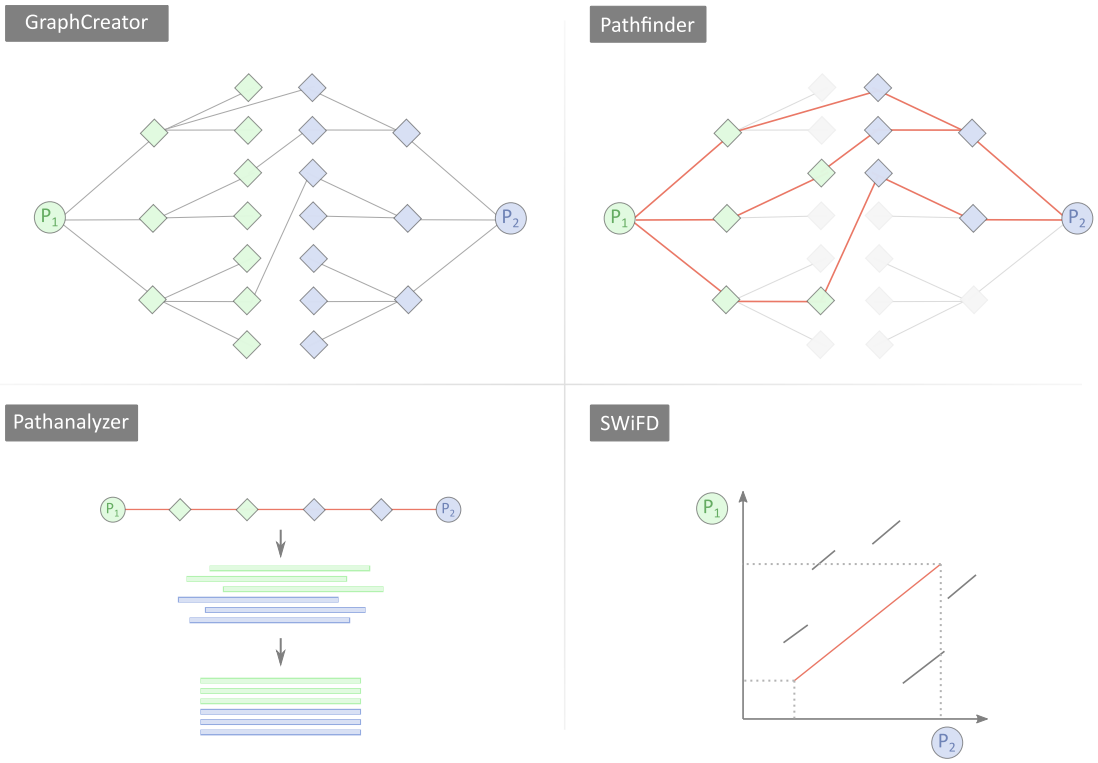


Figure 3.4: Components of FragStatt.
GraphCreator takes as an input two HHblits cascades, referring to the proteins P_1 and P_2, and builds a combined graph. It processes generated HHblits cascades and builds a combined graph. Pathfinder searches for paths connecting P_1 and P_2. In the example, Pathfinder detects three paths. Each of these paths is evaluated by Pathanalyzer. Each path represents a series of pairwise alignments referring to the individual HHblits hits constituting the path. The pairwise alignments are combined by Pathanalyzer which translates all pairwise alignments to a path-specific MSA. SWiFD analyzes these MSAs regarding matching columns and condensates all path-specific MSAs into a single match matrix. Based on the match matrix, a modified version of the Smith-Waterman alignment algorithm detects continuous mappings $P_1 \leftrightarrow P_2$ (red line) while omitting short artifacts (gray lines).

a matrix $H = (h_{ij})$ and the creation of such a matrix is the first task of SWiFD. The second task is to deduce continuous mappings $P_1 \leftrightarrow P_2$ from the matrix H. For this purpose, H is regarded a score matrix as analogously used in the classical Smith-Waterman local alignment algorithm. By using the Smith-Waterman approach, tracebacks can be deduced from H and one or multiple, say k, mappings $P_1 \leftrightarrow P_2$ can be derived, which in turn define the motifs M_k. Using the start and end positions of these mappings the fragments $F_{k1} = P_1(M_k)$ and $F_{k2} = P_2(M_k)$ can be extracted from the protein structures. Finally, the detected motifs are ranked based on their structural similarity as assessed by TM-align.

3.3 Testing and Benchmarking FragStatt

In this section FragStatt will be compared to two other studies which employ HMM based homology detection for the identification of putative ancestral protein motifs. In this manner, the sensitivity of FragStatt can be assessed.

3.3.1 Assessing FragStatt in identifying an ancestral barrel fragment

In their study "Evolutionary relationship of two ancient protein superfolds" (Farías-Rico et al., 2014) the authors detected a homology between the $(\beta\alpha)_8$ barrel and the flavodoxin-like fold. Their approach to identify ancestral relationships was also based on the comparison of HMMs. However, the intent of this study was not to perform a comprehensive search for protein motifs, but to specifically identify a relationship between the two mentioned folds. The authors used similar methods like FragStatt does, but in a much less automated manner, and without the intentional cascading of HMM searches.

The authors gathered all HMMs from the SCOP database (Murzin et al., 1995) which represent $(\beta\alpha)_8$ barrel (SCOP fold `C.1`) and flavodoxin-like proteins (SCOP fold `C.23`) and used them as queries to search the SCOP database using HHsearch (Söding, 2005), a predecessor of HHblits. In this manner, the authors could identify a $(\beta\alpha)_2$ motif that occurs in flavodoxin-like, as well as in $(\beta\alpha)_8$ barrel proteins. Interestingly, the authors identified an intermediate protein family called TM0182 which links the SCOP folds `c.1` and `c.23`: `c.1` $\rightarrow$ TM0182 $\rightarrow$ `c.23`. This observation is a first hint of the higher sensitivity of a cascading approach by utilizing intermediate sequence profiles for the detection of protein motifs. Reproducing the above findings is a first test for the correct functioning of FragStatt. In the following it will be shown, that FragStatt detects this motif in a fully automatic manner.

To begin with, PDB chains `1WA3_A` ($(\beta\alpha)_8$ barrel) and `4JGI_A` (flavodoxin-like) were submitted to a FragStatt run, i.e. P_1 = `1WA3_A` and P_2 = `4JGI_A`. Figure 3.5 shows the dotplot which represents the match matrix H (Equation 2.6) of FragStatt and the detected tracebacks. Interestingly, the algorithm identifies not only one but four tracebacks. This reflects the internal four-fold symmetry of $(\beta\alpha)_8$ barrels consisting of four $(\beta\alpha)_2$ elements. The same motif from

the flavodoxin-like protein `4JGI_A`, positioned approximately at residues $130 - 190$, is mapped to each quarter of the $(\beta\alpha)_8$ barrel `1WA3_A`. By means of a structural alignment the authors Farías-Rico et al. deduced the following mapping (numbers indicate the residue positions on the proteins): `1WA3_A`[47,91] ↔ `4JGI_A`[153,198]

The purely sequence based approach of FragStatt detected a larger region which almost fully covers this motif: `1WA3_A`[25,90] ↔ `4JGI_A`[124,192]

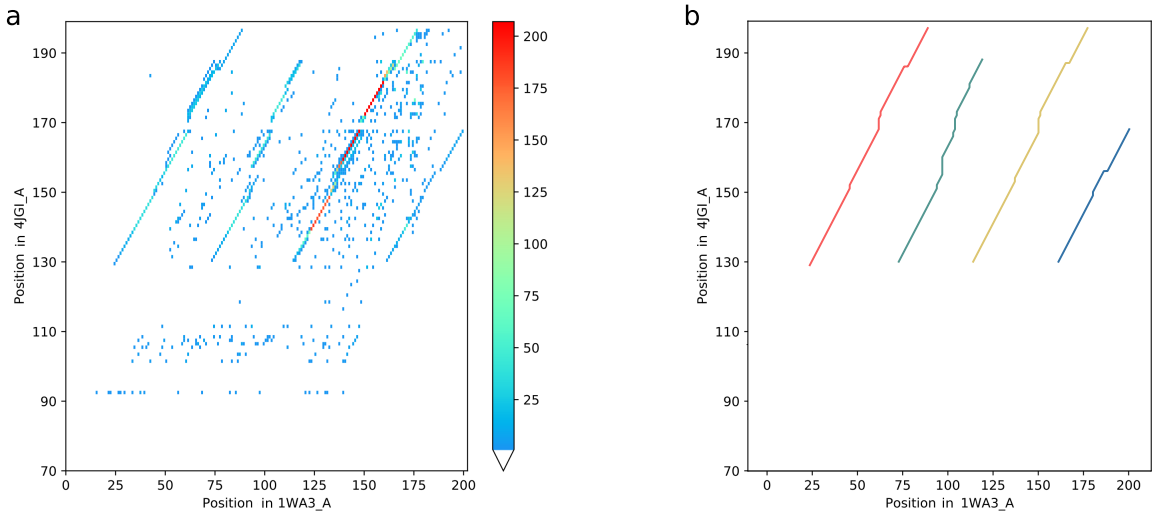


Figure 3.5: FragStatt plots for `1WA3_A` and `4JGI_A`.
a The dotplot shows the regions in the two proteins which were frequently aligned to each other in the HHblits cascade. The frequency of a match is encoded by color (see scale). Four continuously aligned regions are clearly visible, whereas one region is enriched regarding the match frequency. **b** The traceback algorithm detected the four continuous regions correctly and bridged gaps. Altogether, SWiFD filters the noise in the dotplot and identifies the correct mappings.

Table 3.2 lists the position of the four detected motifs and the structural similarity of the fragments assessed by TM-align . The motifs fully cover the $(\beta\alpha)_8$ barrel `1WA3_A` and partly overlap, which can also be seen in Figure 3.6. In the flavodoxin-like protein the motifs occurs only once at the position 124-192. The C-terminal part of the $(\beta\alpha)_8$ barrel is aligned to a shorter version of the motif. The dotplot shows that all motifs consist of higher conserved regions (helices and sheets) connected by less conserved regions (loops). Depending on the parameters for SWiFD, it will either "bridge" these regions of lower alignment quality or the alignment will terminate at these positions resulting in more and shorter sub-fragments. By using the default parameters of SWiFD the individual segments were linked together which leads to the mentioned four continuous tracebacks. The pairwise superpositions based on the four mappings all show a

TM-score > 0.5 as reported by TM-align. On the domain level, a TM-score > 0.5 indicates the same fold, i.e. a significant structure similarity (Xu and Zhang, 2010).

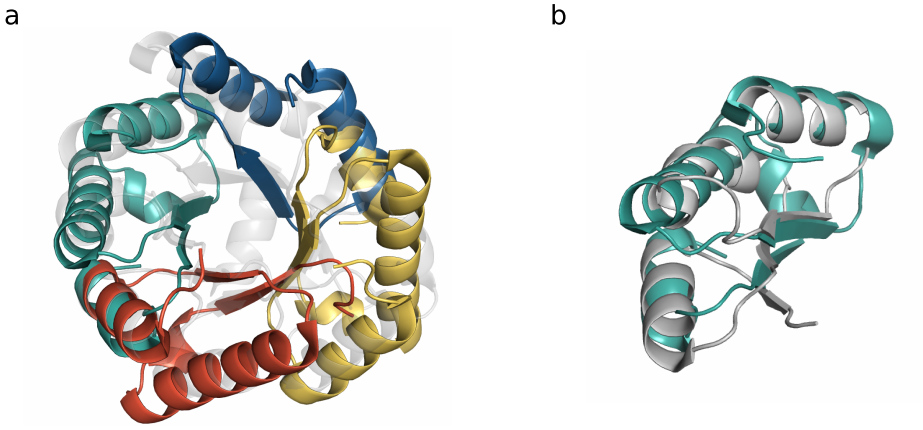


Figure 3.6: Superposition of common motifs in `1WA3_A` and `4JGI_A`.
a The same fragment from `4JGI_A` can be superimposed four times to `1WA3_A` covering the complete $(\beta\alpha)_8$ barrel fold. The structure of `1WA3_A` is shown in transparent gray in the background. The four fragments from `4JGI_A` are shown in different colors according to the coloring of the tracebacks in Figure 3.5. **b** The motifs detected by FragStatt almost fully incorporates the motif defined by Farías-Rico et al. The fragment from `4JGI_A` is shown in green, the one from `1WA3_A` in gray.

Table 3.2: Common motifs detected for `1WA3_A` and `4JGI_A`.
The list shows the start and end positions of the detected fragments in the root proteins. For all four motifs the corresponding fragments show a TM-score > 0.5.

Motif	Start `1WA3_A`	End `1WA3_A`	Start `4JGI_A`	End `4JGI_A`	TM-score
1	25	90	124	192	0.59
2	73	129	124	190	0.60
3	114	178	124	192	0.55
4	162	177	125	140	0.63

The authors Farías-Rico et al. specifically mention that the detected motif occurs in the SCOP superfamilies `c.23.6` (Cobalamin binding domain), `c.1.2` (Ribulose-phoshate binding barrel), `c.1.5` (Inosine monophosphate dehydrogenase) , and `c.1.10` (Aldolase). FragStatt can not report SCOP annotations directly as its intermediate sequence profiles are based on Pfam. Consequently, to compare with these findings, at first the intermediate Pfam families in the paths identified by FragStatt were evaluated, and in a further step a mapping of Pfam to SCOP was utilized. As the Pfam database is aimed at the functional classification of proteins, a analysis of the intermediate Pfam families gives insight in the functional diversity among the proteins

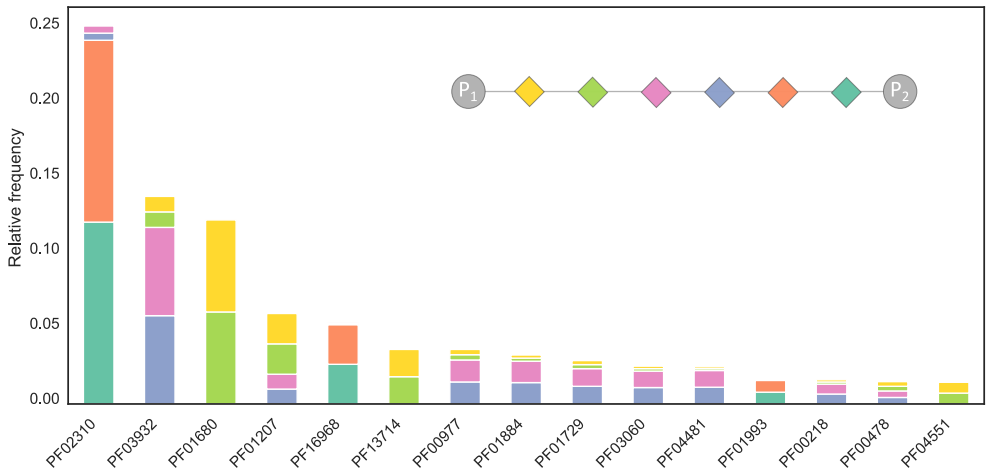


Figure 3.7: Pfam distribution for the FragStatt run `1WA3_A` vs. `4JGI_A`.
The plot shows the relative frequency of the 15 most abundant Pfam families occurring in all paths connecting `1WA3_A` (P_1) and `4JGI_A` (P_2). These relative frequencies summarize for each family the occurrence in the hops within all paths from P_1 to P_2. These occurrences and the hops are coded with the same color. For example, most dominant is `PF02310`, which occurs preferentially in the entry node of `4JGI_A` (dark green) and the subsequent internal node (orange).

utilizing the motif. In the following it will be differentiated between *entry nodes* and *intermediate nodes*. A entry node refers to Pfam family which results from the first hops (starting from either root sequence) in a path detected by FragStatt. As multiple paths are detected it is also possible that multiple entry nodes occur. In a path, starting from the first root sequences, a entry node is followed by multiple intermediate nodes. Eventually the entry node of the second root sequence is reached. To evaluate all detected paths at once, the Pfam distribution for a given distance (in hops) from the root sequences can be assessed. The result of this analysis can be seen in Figure 3.7.

The most prominent entry nodes for `1WA3_A` are `PF01680` (SOR/SNZ family, SCOP `c.1.2`), `PF01207` (dihydrouridine synthase, Dus), and `PF13714` (phosphoenolpyruvate phosphomutase, SCOP `c.1.12`). For `4JGI_A` the most common entry nodes are `PF02310` (B12 binding domain, SCOP `c.23.6`) and `PF16968` (pilus assembly protein TadZ N-terminal).
The most common intermediate families are `PF03932` (CutC family SCOP `c.1.30`), `PF00977` (histidine biosynthesis protein), `PF01884` (PcrB family), `PF01729` (quinolinate phosphoribosyl transferase, C-terminal domain, SCOP `d.41.2`), `PF03060` (nitronate monooxygenase), and `PF04481` (unknown function, DUF561).

By mapping the Pfam families to SCOP families as described in the Section 2.3.1 it could be shown that the SCOP families reported by Farías-Rico et al. are also detected by FragStatt: `c.1.2` and `c.23.6` can be mapped to the previously listed entry Pfam families `PF01680` and `PF02310`. The corresponding Pfam families for the SCOP families `c.1.5` and `c.1.10` were also found in paths detected by FragStatt however they are not in the top-15 most abundant ones as shown in Figure 3.7.

In conclusion, the computational part of the work of Farías-Rico et al. could be replicated by using FragStatt. This indicates that the algorithm is able to detect common motifs shared by proteins belonging to different protein folds in a fully automated manner.

3.3.2 Assessing FragStatt in detecting putative ancient motifs

The authors of "A vocabulary of ancient peptides at the origin of folded proteins" (Alva et al., 2015b) were not interested in a specific pair of protein folds but wanted to identify a comprehensive "vocabulary" of motifs which, as hypothesized, should be occurring amongst a wide range of different protein folds. This approach was also based on HMM comparisons by means of HHsearch. The outcome was a set of 40 putative ancestral protein motifs which will be named ProVoc (**Pro**tein **Voc**abulary) in the following.

Alva et al. used HMMs based on MSAs which were created for each SCOP domain by means of PSI-BLAST (Altschul and Koonin, 1998). To begin with, the SCOP database was filtered to 30 % sequence identity, to exclude homologous domains and reduce redundancy. The resulting HMMs were then submitted to an all vs. all comparison using HHsearch. As the SCOP database classifies proteins by structure, each entry in the database can always be linked to a corresponding structure in the PDB. Thus, each hit resulting from one of these comparisons can be directly translated to a fragment whose structure is known. Consequently, clustering algorithms could be used to identify common motifs occurring in different folds. By a manual inspection of the clustering result, the authors curated a list of common motifs which led to the set ProVoc consisting of 40 primordial motifs. FragStatt also makes pairwise comparisons, but is not restricted to the analysis of proteins whose structure is known. As Figure 3.8 shows, intermediate nodes specified by means of Pfam entries can interlink two proteins P_1 and P_2.

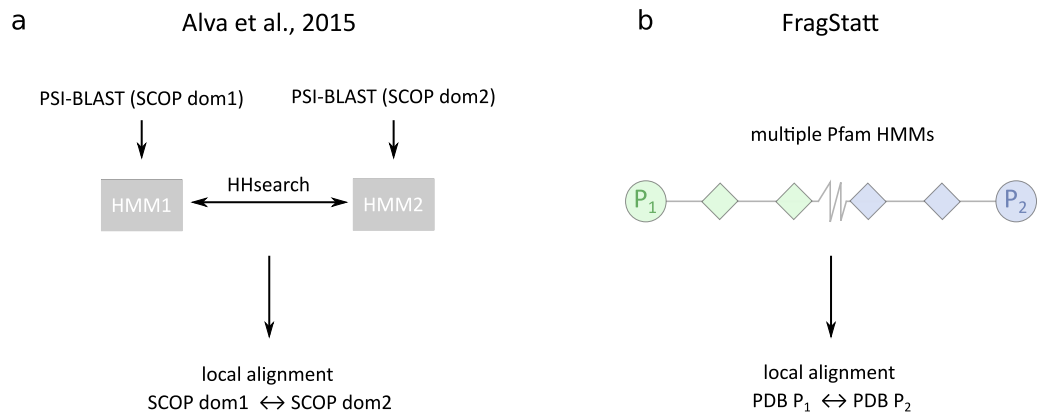


Figure 3.8: Comparison of the approach of Alva et al. and FragStatt.
a Alva et al. performed a PSI-BLAST run for each SCOP domain and created an HMM based on the resulting MSA. This was done for all SCOP domains. Subsequently the HMMs were compared in a pairwise manner using HHsearch, which resulted in a set of local alignments of SCOP domains. **b** Instead of comparing two HMMs, FragStatt extends this concept by utilizing cascaded HMMs to deduce local alignments of two PDB structures.

Figure 3.9 shows the motifs of the ProVoc set. Each of the motifs was detected in several PDB structures, which belong to different SCOP folds and superfamilies. In Figure 3.10, the PDB protein chains featuring motif 1 are listed with their SCOP classification and sequence. For this motif, a manifestation was detected in 20 PDB structures, which belong to 14 SCOP folds and 20 SCOP superfamilies. This finding indicates that these motifs occur in non-homologous and thus evolutionary unrelated proteins.

Although FragStatt uses a different approach and data basis, it should be able to detect the majority of the ProVoc motifs. To verify this, FragStatt runs based on the PDB entries contained in the ProVoc set were carried out and the concordance between the results and the ProVoc set was utilized to determine the program's sensitivity. For each of the 40 motifs from the ProVoc set all pairwise PDB entry combinations belonging to different SCOP folds were analyzed by FragStatt. The example of motif 1 of the ProVoc shall clarify the procedure: The PDB identifiers reported for motif 1 in the ProVoc set resulted in 169 combinations. The number of combinations results from collecting all pairs of PDB structures which belong to different folds (compare Equation 2.12). This has to be done as the authors include multiple manifestations of the motifs from the same fold in their listings (e.g. seven manifestations of motif 1 belong to SCOP fold `A.4`, compare Figure 3.10). In the case of proteins belonging to the same fold FragStatt

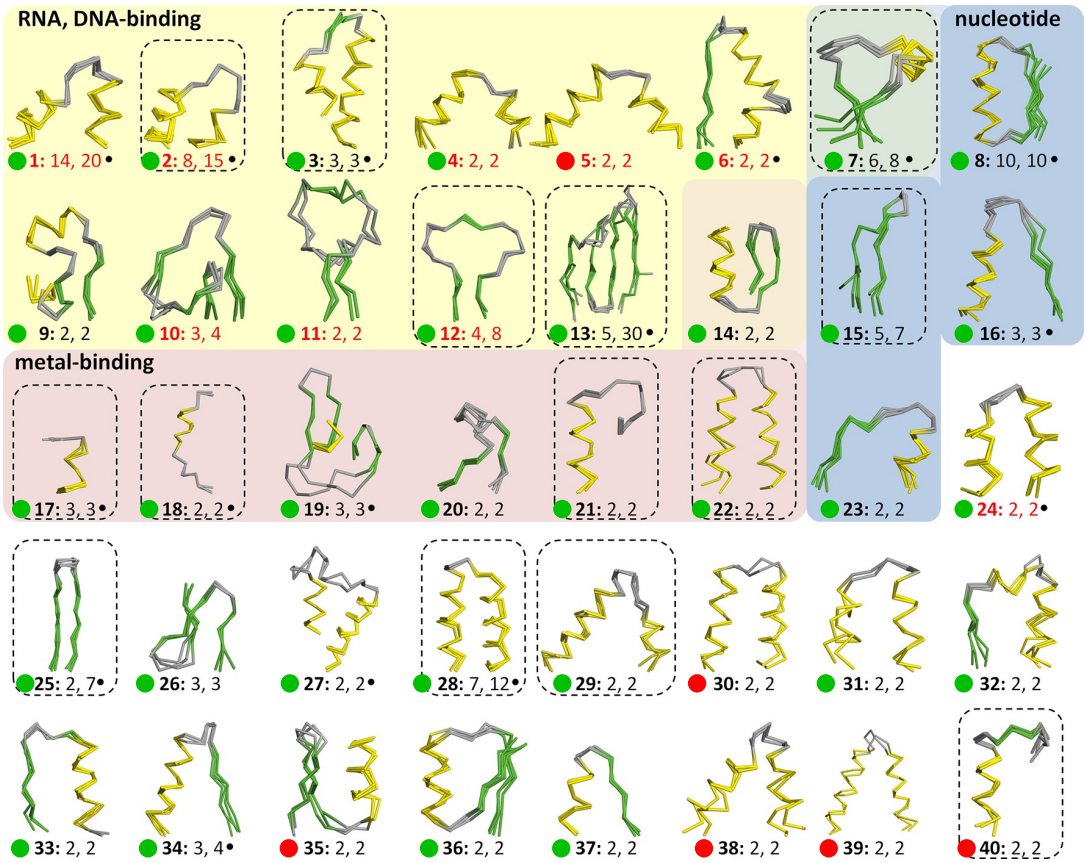
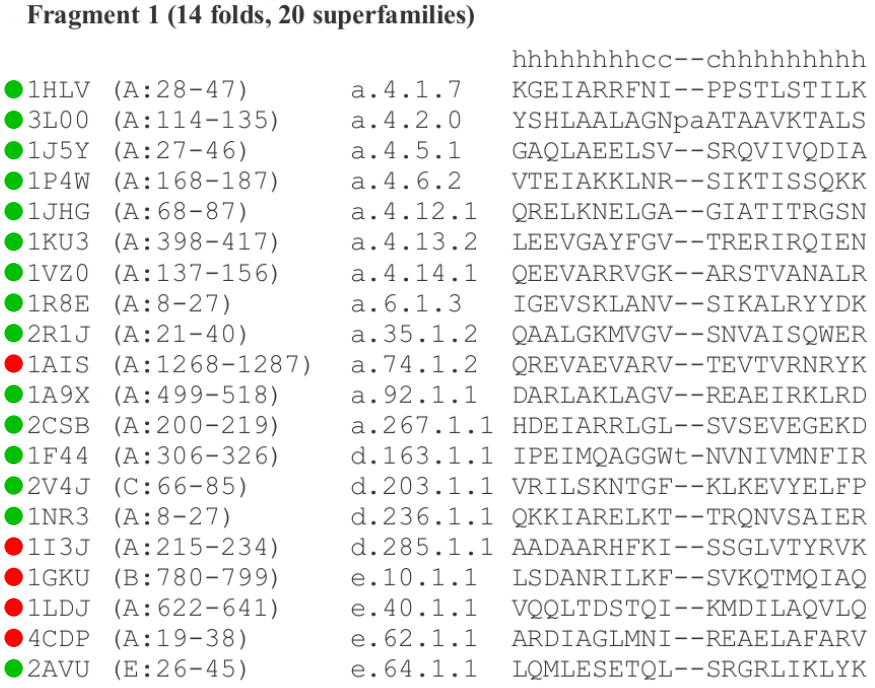


Figure 3.9: Putative ancient motifs identified by Alva et al..
The composition of the ProVoc set consisting of ancient motifs. The motifs are shown in ribbon representation. The given numbers for each motif indicate the count of SCOP folds and superfamilies in which a motif occurs (e.g. 14 folds and 20 superfamilies for motif 1). Motifs which occur repetitively in a domain are boxed with a dotted line. Motifs detected by FragStatt are marked with a green dot or - if not detected - with a red dot. The figure was adapted from Alva et al. 2015b and is originally licensed under CC BY 4.0

would be given two globally similar structures as an input. The identification of a local common motif is not possible for such a pair of proteins as they would be reported as globally similar by FragStatt. In this manner, for motif 1 of the ProVoc set FragStatt detected 15 of the 20 manifestations (75 %).

The results presented in Table 3.3 allow for a comparison of the two approaches. Overall, FragStatt detected for 34 of the 40 ProVoc motifs (i.e. 85 %) two or more manifestations. In total, Alva et al. detected the reported motifs in 239 PDB structures. FragStatt was able to find the motifs in 145 of those proteins (60.67 %). It is unclear, why the six motifs 5, 30, 35, 38, 39, and 40 were not detected by FragStatt. The final compilation of the ProVoc set was

conducted manually and the authors did not report a detailed protocol. Thus, an exhaustive grid search of the program parameters would be required to identify the missing motifs. The re-analysis of the ProVoc set aimed at the assessment of FragStatt's sensitivity and not at the reproduction of published results. Thus, the computationally expensive parameter optimization was not carried out. In summary, the large overlap of detected motifs confirms that FragStatt is capable of finding shared fragments in a fully automated manner, which was not feasible so far.



Fragment 1 (14 folds, 20 superfamilies)

```
                                    hhhhhhhhcc--chhhhhhhhh
●1HLV (A:28-47)          a.4.1.7    KGEIARRFNI--PPSTLSTILK
●3L00 (A:114-135)        a.4.2.0    YSHLAALAGNpaATAAVKTALS
●1J5Y (A:27-46)          a.4.5.1    GAQLAEELSV--SRQVIVQDIA
●1P4W (A:168-187)        a.4.6.2    VTEIAKKLNR--SIKTISSQKK
●1JHG (A:68-87)          a.4.12.1   QRELKNELGA--GIATITRGSN
●1KU3 (A:398-417)        a.4.13.2   LEEVGAYFGV--TRERIRQIEN
●1VZ0 (A:137-156)        a.4.14.1   QEEVARRVGK--ARSTVANALR
●1R8E (A:8-27)           a.6.1.3    IGEVSKLANV--SIKALRYYDK
●2R1J (A:21-40)          a.35.1.2   QAALGKMVGV--SNVAISQWER
●1AIS (A:1268-1287)      a.74.1.2   QREVAEVARV--TEVTVRNRYK
●1A9X (A:499-518)        a.92.1.1   DARLAKLAGV--REAEIRKLRD
●2CSB (A:200-219)        a.267.1.1  HDEIARRLGL--SVSEVEGEKD
●1F44 (A:306-326)        d.163.1.1  IPEIMQAGGWt-NVNIVMNFIR
●2V4J (C:66-85)          d.203.1.1  VRILSKNTGF--KLKEVYELFP
●1NR3 (A:8-27)           d.236.1.1  QKKIARELKT--TRQNVSAIER
●1I3J (A:215-234)        d.285.1.1  AADAARHFKI--SSGLVTYRVK
●1GKU (B:780-799)        e.10.1.1   LSDANRILKF--SVKQTMQIAQ
●1LDJ (A:622-641)        e.40.1.1   VQQLTDSTQI--KMDILAQVLQ
●4CDP (A:19-38)          e.62.1.1   ARDIAGLMNI--REAELAFARV
●2AVU (E:26-45)          e.64.1.1   LQMLESETQL--SRGRLIKLYK
```

Figure 3.10: Sequences of motif 1 from Alva et al..
Each motif is defined by a set of regions in PDB structures. The list contains the following informations: PDB ID, chain and position, SCOP identifier, sequence alignment. Manifestations of the motif which were detected by FragStatt are labeled with a green dot, those not detected are labeled with a red dot.

Table 3.3: Correspondence of motifs detected by FragStatt and Alva et. al..
No. is the number of the motif, # PDBs gives the number of PDB chains the motif was detected in. Fraction gives the proportion of these counts. The last four columns indicate the number of different SCOP folds and superfamilies a motif was found in. The motifs which were not detected by FragStatt are highlighted in red.

No.	# PDBs Alva et al.	# PDBs FragStatt	Fraction (%)	# Folds Alva et al.	# Sfams Alva et al.	# Folds FragStatt	# Sfams FragStatt
1	20	15	75	14	20	9	15
2	15	12	80	8	15	7	12
3	11	8	72	4	4	3	3
4	5	3	60	2	2	2	2
5	6	0	0	2	2	0	0
6	9	6	66	2	2	2	2
7	8	5	62	6	8	4	5
8	12	10	83	10	10	8	8
9	4	1	25	2	2	1	1
10	9	8	88	3	4	3	4
11	2	1	50	2	2	1	1
12	10	9	90	4	8	3	7
13	5	4	80	5	5	4	4
14	4	1	25	2	2	1	1
15	8	2	25	5	7	2	2
16	10	8	80	3	3	2	2
17	8	4	50	3	3	2	2
18	7	3	42	2	2	2	2
19	4	3	75	3	3	2	2
20	4	1	25	2	2	1	1
21	3	1	33	2	2	1	1
22	2	2	100	2	2	2	2
23	7	2	28	2	2	1	1
24	5	4	80	2	2	2	2
25	7	3	42	2	7	1	3
26	7	2	28	3	3	2	2
27	2	2	100	2	2	2	2
28	12	10	83	7	12	7	10
29	3	1	33	2	2	1	1
30	2	0	0	2	2	0	0
31	4	1	25	2	2	1	1
32	7	2	28	2	2	1	1
33	5	4	80	2	2	2	2
34	8	3	37	3	4	2	2
35	3	0	0	2	2	0	0
36	6	2	33	2	2	1	1
37	6	2	33	2	2	2	2
38	3	0	0	2	2	0	0
39	3	0	0	2	2	0	0
40	3	0	0	2	2	0	0

3.4 A Large-Scale Scan for Ancestral Protein Motifs

After having shown that FragStatt can reproduce literature data and that it is a highly sensitive approach to identify shared motifs, the next goal was a comprehensive pair-wise comparison based on a representative set of all known protein structures.

3.4.1 Data basis and basic strategy

By the nature of its design FragStatt can only inspect two protein chains at once. Thus a comprehensive scan for ancestral protein motifs comes down to a pairwise all vs. all comparison of protein chains taken from the PDB, as illustrated in Figure 3.11. Naturally all vs. all computational problems scale with $\mathcal{O}(n^2)$, i.e. the required computational resources grow quadratically with the input size n. A $\mathcal{O}(n^2)$ scaling algorithm quickly becomes unsolvable in a reasonable time for for large n, thus it is necessary to keep n small. This method approach of scanning for ancestral protein motifs is a classical screening approach whose success depends on a sufficient coverage rate, i.e. a high number n.

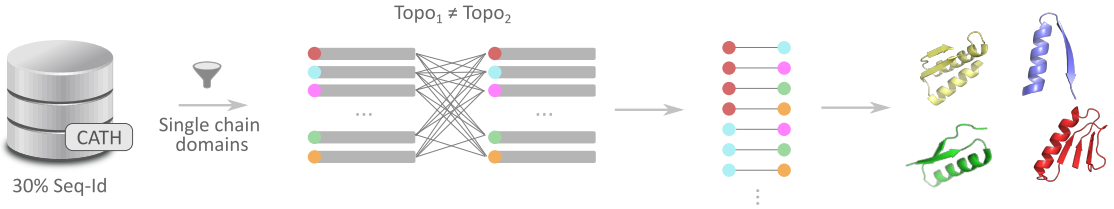


Figure 3.11: All vs. all search for protein ancestral protein motifs.
The CATH database filtered to 30 % sequence identity was used as the data basis for the all vs. all scan. All domains comprising a complete PDB protein chain were extracted and gathered. The all vs all pairs were generated based on the requirement that the CATH topology of the domains differ. All pairs were fed into FragStatt which resulted in a set of protein motifs.

If n different protein chains have to be compared pairwise, the total number of comparisons is

$$\#Comparisons = \frac{n^2 - n}{2}. \tag{3.2}$$

Equation 3.2 results from a simple combinatorial consideration and treats comparisons $A \leftrightarrow B$ and $B \leftrightarrow A$ as equal and does not count self-comparisons, i.e. $A \leftrightarrow A$.

As it was the aim to survey the "protein universe", i.e. all folds observed in Nature, a diligent selection of the protein chains included in the analysis had to be carried out. Currently, in March 2020, the PDB contains over 161,000 entries composed of over 485,000 protein chains. According to Equation 3.2 using the full PDB as a data basis would result in over 117 billion comparisons, which cannot be executed in a realistic period of time. However, it is not necessary to analyze each protein chain, because of the redundancy of the PDB. Both individual proteins but also folds are often represented by more than one PDB entry. Therefore it is possible to reduce the number of protein chains without loosing protein folds.

The resulting data basis should represent the structure and sequence diversity of all known proteins. To remove redundancy while preserving the diversity in a sequence dataset, the most common practice is to filter it to a certain maximal sequence identity percentage. Additionally, databases which classify proteins by their structure can be used to ensure an adequate coverage of the structure space. These two methods can be combined. Thus, the CATH database, filtered to a sequence identity of 30 %, was used as a structurally highly diverse data basis with low sequence redundancy. For simplicity, only single-domain PDB protein chains were taken into account, which brings the advantage that whole chains can be attributed to a certain CATH classification and chains do not have to be split into single domains. This constraint simplifies the generation of the data basis as well as the evaluation of the results. In this manner, over 12,000 single-domain and CATH classified PDB protein chains were gathered. Based on this set all combinations of protein chains which belong to different CATH topologies were generated, which finally resulted in a set of approximately 77 million pairs of protein chains, further referred to as CATH_RF.

3.4.2 Defining a set of candidate hits

The CATH_RF set of protein chain pairs served as the starting point for the all vs. all comparison based on FragStatt. As expected, most of the runs did not find a common motif: Among the total of 77 million analyzed pairs around 900,000 (1.2 %) hits (AllHits) were detected occurring in different CATH topologies. By analyzing these visually, it became evident that there is a great number of topology combinations which are related and do not satisfy the initially assumed

requirement of being evolutionary independent, leading to many false positive hits. This means that the initially made assumption that all CATH topologies are evolutionary independent does not strictly hold for all cases. The CATH_RF set contains 867 CATH topologies, which leads to almost 375,000 possible combinations of topologies. Checking all of these combinations regarding a possible homology is a task which can't be done manually with reasonable effort. It is also difficult to automate this task as often the results of database queries (e.g. Pfam, InterPro, SCOP, etc.) have to be analyzed to check whether a topology pair is strictly non-homologous.

As on the topology level of CATH, the requirement of evolutionary independence can not be guaranteed, it was decided to take a more conservative approach by filtering the hits to those which feature different CATH architectures. The architecture level represents a more coarse classification than the topology level and a homology between CATH architectures is unlikely. Still, even on this level the discussed problem occurs. However, as the number of CATH architectures is much smaller than the number of topologies it is possible to manually filter these combinations. An illustrative example that highlights the problem is the comparison of beta propeller proteins shown in Figure 3.12.

From an architectural point of view the two proteins shown in Figure 3.12 share a common scheme of composition: The annular repetition of the same motif comprised of four consecutive beta strands, which is called a blade. Beta propeller proteins exist in many different sizes, i.e number of repetitions of blades: e.g. 3, 4, 5, 6, 7, and 8 propellers. It is assumed that the different variants of the beta propeller arose divergently and evolved by amplification and diversification (Kopec and Lupas, 2013; Chaudhuri et al., 2008). The origin of the beta propeller architecture can supposedly be dated back to the earliest era of protein evolution (Caetano-Anolles et al., 2009), which makes it an interesting candidate to study. The beta propeller blade may represent an ancestral motif, however for the purpose of this study it is not desired to detect relationships between different types of beta propellers as they supposedly evolved after the formation of a prototypical beta propeller blade motif. Instead the goal is to potentially detect the beta propeller motif in other distinct folds. For this reason comparisons between beta propellers (potential false positives) should be filtered out, while comparisons of beta propellers to other architectures (potential true positives) should be kept. Unfortunately, in the CATH database

the beta propellers are not subsumed under one architecture, but depending on the number of blades, the proteins are grouped into different CATH architectures. Thus, even if the pairwise comparison is restricted to proteins with different architectures, a comparison of beta propeller proteins that possess a differing number of blades cannot be avoided. Thus, it is necessary to define a "black list" containing combinations of architectures to be removed from the analysis.

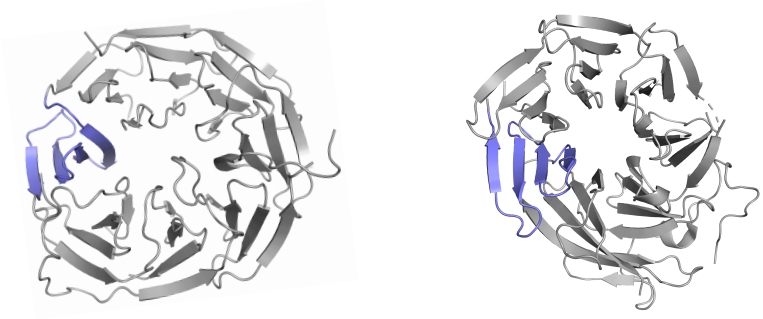


Figure 3.12: Seven (PDB-ID `3Q7M_A`) and eight (PDB-ID `1ERJ_A`) bladed beta propellers. Both proteins share the same overall architecture of a beta propeller and only the number of repetitions of the propeller motifs (blades) distinguishes them from an architectural point of view. However, both proteins are classified into different CATH architectures: `2.140` (8 propeller) and `2.130` (7 propeller)

Based on these insights, a comprehensive comparison of proteins belonging to different architectures was performed. Prior to the in-depth analysis of the candidate hits, several filters were applied. First, as discussed, homologous architecture pairs were eliminated. Additionally, all candidate pairs were removed whose pairwise alignment contained more than five gaps. Moreover, the length of the motifs (mean of the length of the fragments) had to lie between 15 and 60 residues and a minimal TM-score of 0.55 was required.

In this manner, the set CoMo of pairwise **co**mmon **mo**tif relationships was compiled that consisted of 12,533 entries. The CoMo set includes 2870 unique PDB protein chains which belong to 26 CATH architectures and 245 CATH topologies. The CoMo set comprises 1.4 % of the 900,000 initial hits (AllHits) and 0.016 % of the total of 77 million made comparisons (CATH_RF). This means that approximately every 6,000th comparison yielded a candidate hit. This low rate strongly suggests that motifs are rarely shared between proteins possessing different CATH architectures.

3.4.3 CoMo motifs are spread unevenly amongst CATH

The number of members a CATH architectures has, i.e. the number of protein domains associated with it, varies immensely. On the one hand there are relatively few highly populated architectures like the Alpha-Beta barrel (CATH 3.20), which includes over 16,000 domains, and on the other hand there are many sparsely populated architectures like for example the 4 propeller (CATH 2.110) which is comprised of less than 60 domains. By evaluating these counts one can get an idea of how frequent a certain architecture occurs in nature. However, when making such considerations based on the number of solved protein structures, one has to keep in mind that the PDB itself must not necessarily depict a representative snapshot of the natural protein structure space. For example proteins or protein folds which are relatively simple to express, purify and structurally solve in the laboratory are overrepresented in the PDB, whereas other proteins like membrane proteins, known to be challenging from a bioanalytical perspective, are underrepresented (Alexandrov and Fischer, 1996). Nevertheless, with sufficient certainty it can be said that some protein architectures like the alpha-beta barrel have a prominent role in utilization and functional diversity among all domains of life. For example in five out of seven classes of the EC nomenclature (Webb, 1992) the $(\beta\alpha)_8$ fold can be observed making it one of the most utilized protein folds in nature (Nagano et al., 1999).

As stated, the affiliation to different CATH classes, architectures and topologies can be evaluated for all known protein structures. Such a statistic was created for the initial hits (AllHits) and the CoMo set. A statistic was also determined for the full set of analysed single domain protein chains from CATH (null distribution). To assess, whether the restraint to single domain protein chains restraint reduces the sampled structure space, the distribution of the full CATH database was also included in the statistic. The result of the analysis on the CATH class level is shown in Figure 3.13.

A comparison of the corresponding frequencies in Figure 3.13, confirms that the single domain proteins are similarly distributed among the CATH classes as the full content of the database itself. Thus, it can be safely assumed that the subset of single-domain chain entries represents the space of known protein structures adequately. In contrast, the distribution of CATH classes among the outcome of the AllHits analysis deviates strongly from the latter two distributions:

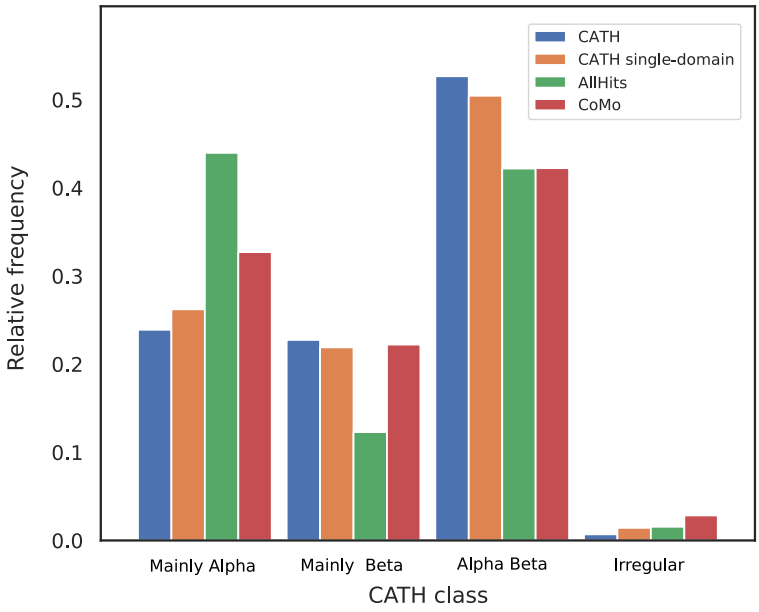


Figure 3.13: Distribution of PDB protein chains in CATH classes.
The bar plot shows the relative portion of the CATH classes in different datasets. Blue: The unfiltered complete CATH database. Orange: Only single-domain chains in the CATH database. Green: All hits from the scan (AllHits). Red: Hits filtered for redundancy and plausibility (CoMo)

A great fraction of these unfiltered hits belongs to proteins from class one "Mainly Alpha". The class two "Mainly Beta" on the other hand is underrepresented in the unfiltered hits. On the other hand, the hits resulting from the CoMo set possess a CATH class distribution that is highly similar to that of the full data set. This finding indicates that the removal of false positive hits, e.g. unspecific helix-only hits of class one members, is effective. Still, minor deviations of the CoMo set in reference to the null distributions can be seen: A slight overrepresentation of class one "Mainly Alpha" and four "Few Secondary Structures" can be observed. In class three "Alpha Beta" the candidate hits are slightly underrepresented.

The analysis on the hightest level of the CATH classification hierarchy shows four things: Firstly, most proteins belong to class three "Alpha Beta". Secondly, the used subset of single-domain chains represents the CATH database well. Thirdly, the chosen filters are effective. Fourthly, at least on the hightest level of the CATH classification, there is no striking enrichment of putative ancestral motifs in one of the classes.

The previous analysis showed that the number of proteins comprised of alpha and beta secondary structures surmount the number of proteins comprised of mainly alpha helices or mainly beta

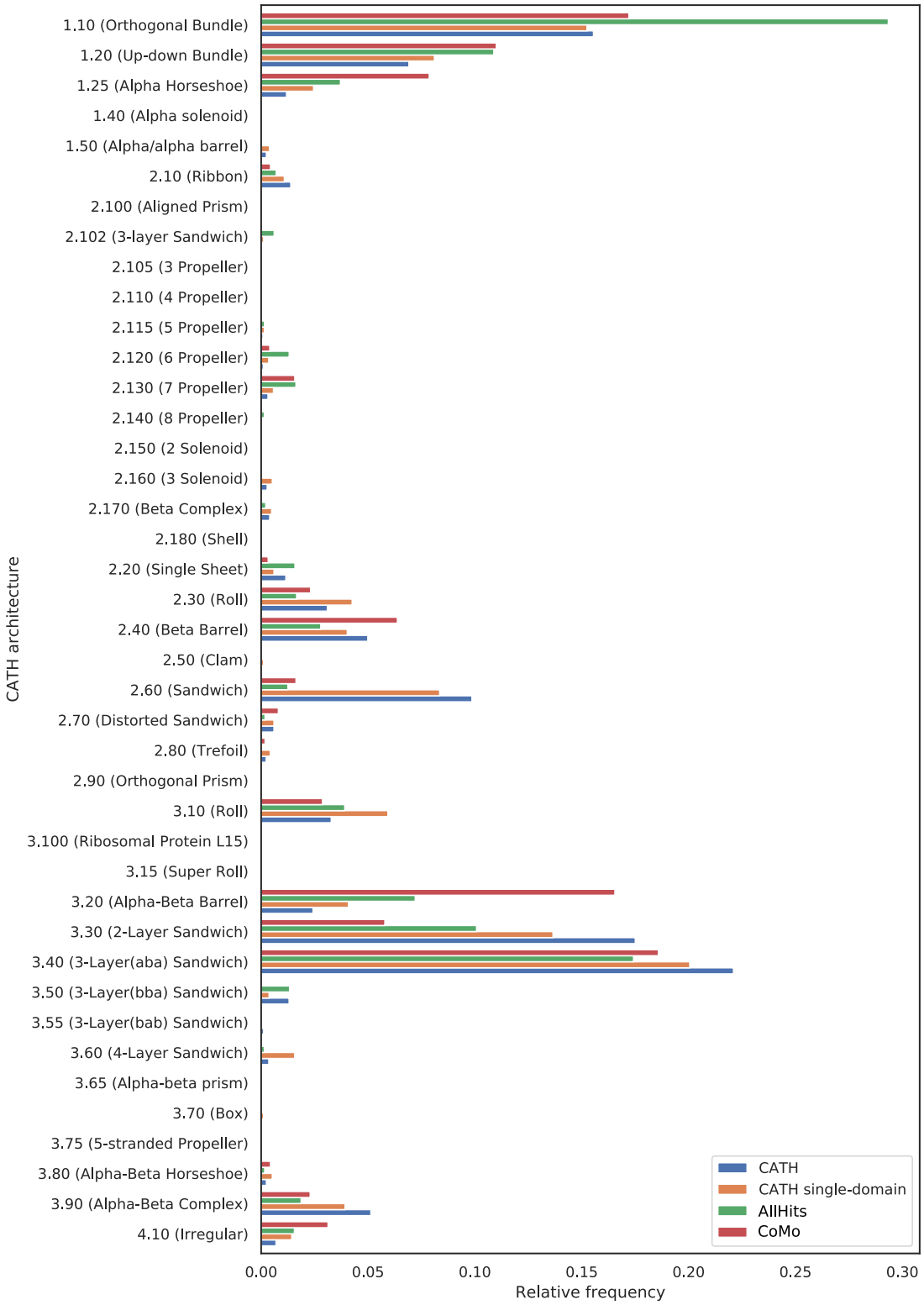


Figure 3.14: Distribution of CATH architectures in four data sets.
Each bar indicates the relative frequency of an architecture and the color encodes the dataset. Blue: The unfiltered complete CATH database. Blue: The unfiltered complete CATH database. Orange: Only single-domain chains in the CATH database. Green: All hits from the scan (AllHits). Red: Hits filtered for redundancy and plausibility (CoMo)

sheets: Still, the total of protein structures is distributed even among the three classes "Mainly Alpha", "Mainly Beta" and "Alpha Beta" at a ratio of roughly 1:1:2. Not necessarily, a similarly even distribution has to be expected for the next level of the CATH classification, which is the architecture. Thus, the analysis which was performed above for the class level, was now applied to CATH architectures.

As Figure 3.14 shows, the five most populated architectures at 35 % sequence identity are `3.40` (3-Layer(aba) sandwich), `1.10` (Orthogonal Bundle), `3.30` (2-Layer Sandwich), `2.60` (Sandwich), and `1.20` (Up-down Bundle). About 65 % of all domains assigned to CATH belong to one of these five architectures. Thus, one can expect that the hits belonging to the CoMo set will show a similar bias. Figure 3.14 shows that this is generally the case: Highly populated architectures also have high hit counts. However, the barplot also reveals that some architectures show overproportionally many hits.

To quantify the enrichment E, a log-ratio of the candidates frequency and the background frequency was calculated:

$$E_i = \log \frac{f(C_i)}{f(CSD_i)} \tag{3.3}$$

Here $f(C_i)$ is the frequency of all CoMo hits belonging to architecture i and $f(CSD_i)$ denotes the frequency of all CATH single-domain entries belonging to architecture i. Table 3.4 shows the result of this analysis.

The positively enriched architectures, ordered from high to low enrichment, are `2.140` (8 Propeller), `3.20` (Alpha-Beta Barrel), `1.25` (Alpha Horseshoe), `4.10` (Irregular), `2.130` (7 Propeller), `2.40` (Beta Barrel), `1.20` (Up-down Bundle), `1.10` (Orthogonal Bundle), and `2.70` (Distorted Sandwich). With the exception of `2.140`, a tendency of sparsely populated architectures being underrepresented can be observed. From the above mentioned highly populated the two "All Alpha" architectures `1.10` (Orthogonal Bundle) and `1.20` (Up-down Bundle) are enriched. Most of these architectures can be associated with an internal symmetry (e.g Alpha-Beta Barrel, Propellers) or a repetitive structure (Alpha Horseshoe, Up-down Bundle).

The analysis of hit vs. background frequencies relates to single CATH architectures. However,

Table 3.4: Relative frequencies of CATH architectures in different datasets.
The first column denotes the CATH architecture. The following columns give the relative frequencies (rf) of the CATH architecture in different sets. The column Enrichment gives the enrichment log-ratios calculated according to Equation 3.3. The rows are sorted according to the enrichment in descending order.

CATH Arch.	rf CATH (%)	rf CATH single-domain (%)	rf All hits (%)	rf CoMo (%)	**E** Enrichment
2.140	0.04	0.02	0.15	0.08	1.62
3.20	2.43	4.08	7.21	13.98	1.23
1.25	1.19	2.45	3.71	7.63	1.13
4.10	0.70	1.43	1.56	3.19	0.80
2.130	0.32	0.58	1.63	1.26	0.78
2.40	4.99	4.03	2.79	5.62	0.33
1.20	6.91	8.11	10.89	10.98	0.30
1.10	15.55	15.25	29.34	20.35	0.29
2.70	0.60	0.60	0.18	0.65	0.07
2.120	0.10	0.36	1.31	0.33	-0.07
3.40	22.10	20.05	17.42	18.78	-0.07
3.80	0.25	0.52	0.17	0.47	-0.10
3.10	3.28	5.93	3.91	3.54	-0.52
3.90	5.13	3.93	1.87	2.19	-0.58
3.30	17.51	13.66	10.08	6.87	-0.69
2.102	0.04	0.12	0.61	0.06	-0.75
2.20	1.16	0.60	1.58	0.28	-0.77
2.30	3.10	4.26	1.66	1.90	-0.81
2.10	1.39	1.08	0.70	0.37	-1.08
2.80	0.24	0.43	0.03	0.14	-1.12
3.50	1.31	0.37	1.33	0.06	-1.76
2.60	9.86	8.35	1.25	1.24	-1.91
3.70	0.01	0.11	0.01	0.02	-1.94
1.50	0.25	0.39	0.02	0.01	-3.88
2.170	0.40	0.49	0.22	< 0.01	-4.81
2.160	0.28	0.52	< 0.01	< 0.01	-4.87

as FragStatt links two architectures, it is of interest to consider which pairs of architectures were linked, and if some combinations turned out to be particularly abundant. This was done by counting how many motifs were detected for each combination of CATH architectures. The results of this evaluation are visualized as a network in Figure 3.15 and quantified in Table 3.5. As expected, relatively few combinations of architectures constitute the greatest part of all hits. The most abundant five combinations are

`3.20` (Alpha-Beta Barrel) $\leftrightarrow$ `3.40` (3-Layer(aba) Sandwich),

`1.10` (Orthogonal Bundle) $\leftrightarrow$ `3.40` (3-Layer(aba) Sandwich),

1.20 (Up-down Bundle) ↔ 1.25 (Alpha Horseshoe),

1.20 (Up-down Bundle) ↔ 1.10 (Orthogonal Bundle),

1.10 (Orthogonal Bundle) ↔ 3.30 (2-Layer Sandwich).

These five combinations sum up to a fraction of 57 % of all hits.

With a frequency of 25 %, hits between CATH architectures 3.20 (Alpha-Beta Barrel) and 3.40 (3-Layer(aba) Sandwich) are the most abundant ones in the all vs. all scan. The $(\beta\alpha)_2$ motif identified by Farías-Rico et al. (Farías-Rico et al., 2014) belongs to this group. In general a overrepresentation of the "Mainly Alpha" class can be observed: Four of the five most abundant combinations can be attributed to this class. Again, a preference for repetitive (Up-down Bundle, Alpha Horseshoe) and internally symmetrical architectures (Alpha-Beta Barrel) can be seen among the most abundant architecture combinations.

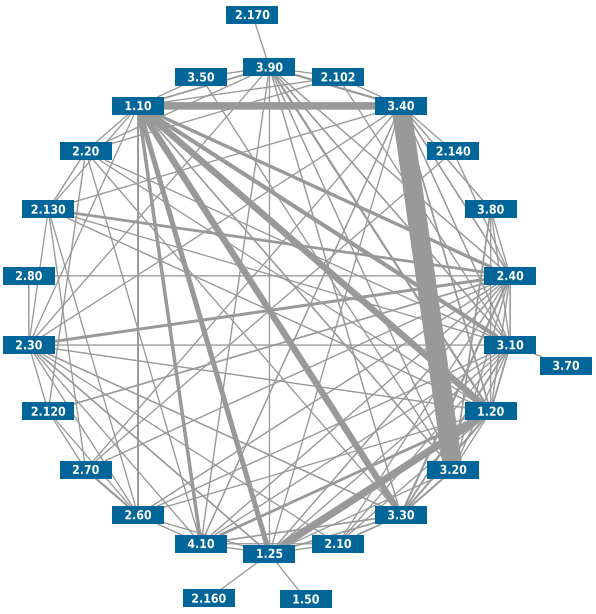


Figure 3.15: A survey of motifs shared between CATH architectures.
The nodes represent CATH architectures, which are connected by an edge, if FragStatt detected shared common motifs between members of the architectures. The width of the edges represents the number of detected motifs. Exact values can be found in Table 3.5.

The enrichment of alpha helical architectures can also be seen in the network (compare Figure 3.15) in the form of the highly connected architectures 1.10 (Orthogonal Bundle), 1.20 (Up-down Bundle) and 1.25 (Alpha Horseshoe). Of these architectures 1.10 (Orthogonal Bundle) seems to be the most central, showing strong connections to architectures from all classes,

Table 3.5: Number of hits detected by FragStatt when comparing two CATH architectures. The tables gives the count of the hits for each combination of CATH architectures. The column Fraction denotes for each combination the relative share regarding all hits in the CoMo set.

Arch. 1	Name 1	Arch. 2	Name 2	Count	Fraction (%)
3.20	Alpha-Beta Barrel	3.40	3-Layer(aba) Sandwich	3186	25.4
1.10	Orthogonal Bundle	3.40	3-Layer(aba) Sandwich	1095	8.7
1.20	Up-down Bundle	1.25	Alpha Horseshoe	1063	8.4
1.20	Up-down Bundle	1.10	Orthogonal Bundle	917	7.3
1.10	Orthogonal Bundle	3.30	2-Layer Sandwich	915	7.3
1.10	Orthogonal Bundle	1.25	Alpha Horseshoe	667	5.3
3.10	Roll	1.10	Orthogonal Bundle	499	3.9
4.10	Irregular	1.10	Orthogonal Bundle	388	3.1
2.40	Beta Barrel	1.10	Orthogonal Bundle	360	2.8
2.40	Beta Barrel	2.30	Roll	296	2.3
4.10	Irregular	1.20	Up-down Bundle	260	2.0
2.40	Beta Barrel	2.130	7 Propeller	253	2.0
3.90	Alpha-Beta Complex	1.20	Up-down Bundle	160	1.2
2.40	Beta Barrel	3.30	2-Layer Sandwich	155	1.2
3.90	Alpha-Beta Complex	3.40	3-Layer(aba) Sandwich	155	1.2
2.70	Distorted Sandwich	2.60	Sandwich	138	1.1
1.20	Up-down Bundle	3.30	2-Layer Sandwich	136	1.0
3.20	Alpha-Beta Barrel	3.30	2-Layer Sandwich	127	1.0
1.10	Orthogonal Bundle	2.60	Sandwich	115	0.9
3.10	Roll	3.30	2-Layer Sandwich	107	0.8
3.10	Roll	1.20	Up-down Bundle	82	0.6
3.90	Alpha-Beta Complex	1.10	Orthogonal Bundle	79	0.6
3.30	2-Layer Sandwich	2.10	Ribbon	79	0.6
1.20	Up-down Bundle	3.40	3-Layer(aba) Sandwich	78	0.6
3.20	Alpha-Beta Barrel	3.80	Alpha-Beta Horseshoe	77	0.6
2.40	Beta Barrel	2.120	6 Propeller	67	0.5
2.30	Roll	1.25	Alpha Horseshoe	66	0.5
1.25	Alpha Horseshoe	3.30	2-Layer Sandwich	64	0.5
2.40	Beta Barrel	3.40	3-Layer(aba) Sandwich	61	0.4
4.10	Irregular	3.30	2-Layer Sandwich	60	0.4
3.20	Alpha-Beta Barrel	1.10	Orthogonal Bundle	56	0.4
3.10	Roll	3.90	Alpha-Beta Complex	52	0.4
2.30	Roll	3.30	2-Layer Sandwich	44	0.3
3.10	Roll	3.40	3-Layer(aba) Sandwich	43	0.3
2.40	Beta Barrel	3.10	Roll	43	0.3
4.10	Irregular	3.40	3-Layer(aba) Sandwich	36	0.2
3.80	Alpha-Beta Horseshoe	3.40	3-Layer(aba) Sandwich	36	0.2
2.40	Beta Barrel	1.25	Alpha Horseshoe	32	0.2
2.80	Trefoil	2.30	Roll	29	0.2
3.90	Alpha-Beta Complex	3.30	2-Layer Sandwich	28	0.2

notably also to the second most abundant architecture (compare Table 3.4) in the CoMo set 3.40 (3-Layer(aba) Sandwich). As mentioned this architecture contributes to the most abundant combination, connecting it to the architecture 3.20 (Alpha-Beta Barrel). However, overall, the connectivity of 3.40 (3-Layer(aba) Sandwich) seems to be relatively low. A particularly high connectivity, on the other hand, can be observed for the architectures 2.40 (Beta Barrel), 3.10 (Roll) and 1.20 (Up-down Bundle). These architectures have in common that they consist of relatively simple and short repeating elements (beta strands and alpha helices) which can be found in many other architectures.

In general, while most of the hits were detected between only a few architectures, it can be seen that the network is nevertheless highly connected, suggesting that many less populated architecture combinations deliver positive results. The same analysis at the topology level was also carried out. The results can be found in the supplementary material in Table S1 and Figure S1.

In summary, the analysis presented in this section suggests that during the course of evolution some protein architectures were utilized particularly often while others seem to be rare. This natural uneven background distribution of CATH architectures can be also observed in the CoMo set. A normalization to the background frequencies showed that particularly the CATH architectures 3.20 (Alpha-Beta Barrel) and 1.25 (Alpha Horseshoe) (compare Table 3.4) seem to be enriched in the CoMo . Also, the architecture 2.140 (8 Propeller) is enriched, however its absolute abundance in the CoMo is small. In general, the majority of motifs in the CoMo is linked to a small number of CATH architectures. Among these architectures, a tendency for mainly alpha helical ones can be observed. The analysis of all CATH architecture combinations substantiates these findings; e.g. the combination 3.20 (Alpha-Beta Barrel) and 3.40 (3-Layer(aba) Sandwich) contributes 25 % of motifs in the CoMo set. Yet, a network analysis showed that many motifs between other CATH architectures were detected as well, indicating the existence of many less universal motifs. In conclusion, the CoMo set is highly uneven distributed amongst the protein architecture space and the presented findings question the universality of the detected motifs.

3.4.4 Length distribution of the CoMo motifs

A reasonable assumption is that motifs have at least the size of a small super-secondary structure element and are not larger than a protein domain. Thus, their length should range between 15 and 60 amino acid residues. This is why the all vs. all scan was restricted to the identification of fragments of this size. To investigate the length distribution, the histogram shown in Figure 3.16 was generated. The plot shows the relative frequency of motifs of a given size within the CoMo

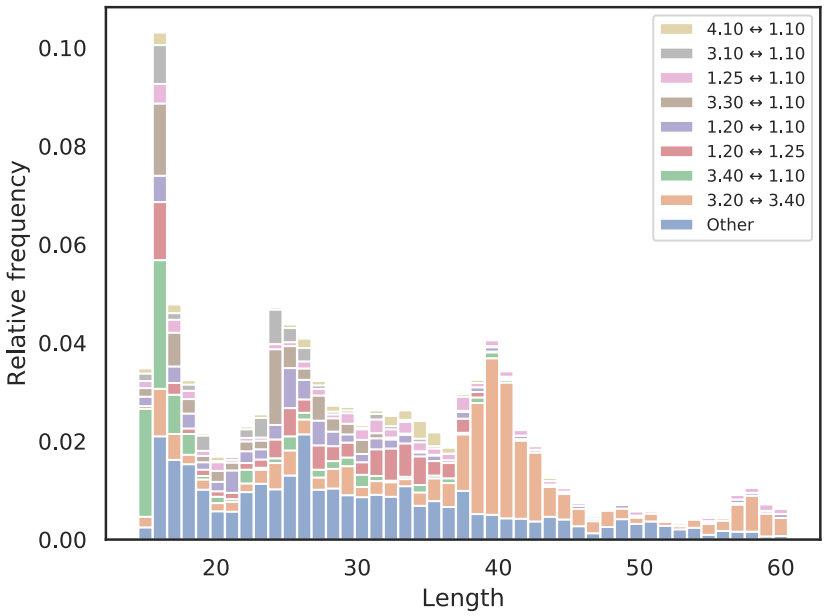


Figure 3.16: Distribution of the length of the detected motifs.
The histogram shows the relative frequency of the motifs according to their length. For each bin, the contribution of the eight most abundant combinations of CATH architectures (see legend) is given.

set. Three maxima at a length of approximately 18, 25 and 40 residues can be observed. A less pronounced enrichment can also be observed at a length of approximately 58 residues, however, in general motifs with a length greater than 45 residues seem to be rare. As demonstrated in the last section, the different architectures contribute unevenly to the CoMo set. Thus it is of interest to assess the length distribution of the most abundant architecture combinations individually. To achieve this in a single graphical representation, the histogram was plotted as a stacked bar plot. The height of each bar is the sum of the portions of the individual architecture combinations. It was decided to show the eight most abundant combinations individually (accounting for almost

70 % of the CoMo set, see Table 3.5) and to group the rest together as "Other". The histogram makes clear that motifs of length 40 and 58 residues are predominately found when comparing the architectures 3.40 (3-Layer(aba) Sandwich) ↔ 3.20 (Alpha-Beta Barrel). This finding is not unexpected, because this combination contributes over 25 % of all CoMo hits (compare Table 3.5).

In contrast, the motifs found by comparing the other architectures have a different length distribution with a maximum between 24 and 28 residues. Interestingly, 3.40↔3.20 pairs contribute less to this range of length. This finding further supports the idea that these two architectures are a special case because they share an unusually long motif. The length distribution of the motifs resulting from the comparison of all other architectures has two maxima around 18 and 25 residues.

3.4.5 Structure similarity of the CoMo motifs

A further characteristic feature of the motifs is their pairwise structural similarity which can be determined by means of the TM-score. FragStatt hits consist of two fragments from two proteins whose 3D structure is known having different CATH architectures. While the initial detection is sequence based, each pair of fragments is finally scored regarding its structural similarity using TM-align. TM-align delivers the TM-score which is normalized to the range 0 to 1.0, where 1.0 indicates a perfect match between the two structures. Domains with a TM-score > 0.5 can be generally assumed to belong to the same protein fold (Zhang and Skolnick, 2005; Xu and Zhang, 2010).

The question arises whether for the motifs in the CoMo set a higher structural similarity can be observed than one would expect by chance. For this analysis, a null model distribution is needed that consists of the TM-scores resulting from a comparison of unrelated fragments. In order to create this fragment set, residue positions were randomly chosen in the PDB entries contributing to the CoMo set. Subsequently fragments were excised with a length distribution following that of the CoMo fragments (compare Figure 3.16).

In Figure 3.17 the resulting length distribution is shown together with the length distribution

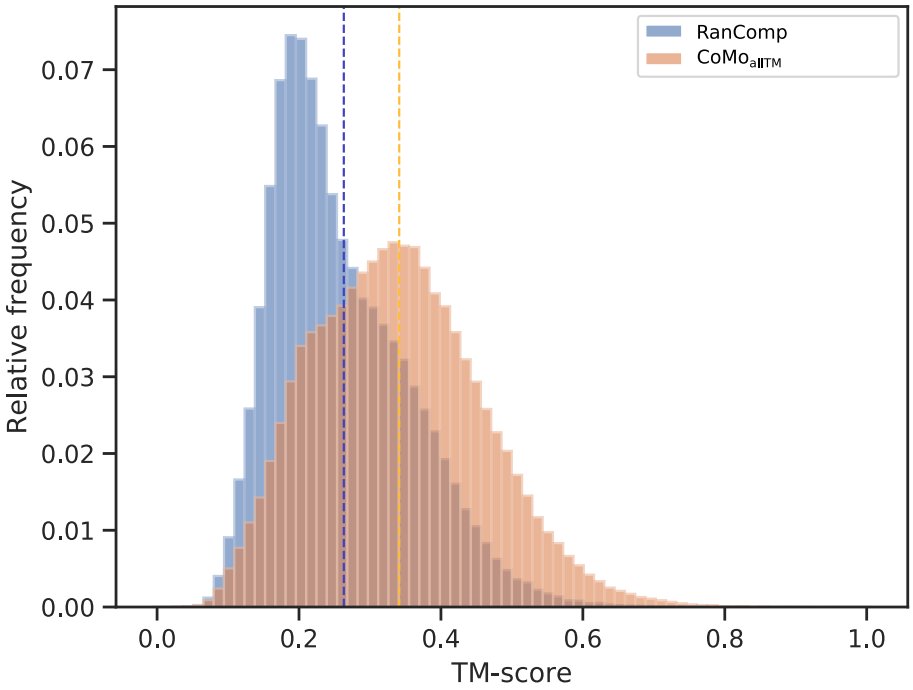


Figure 3.17: Histogram of TM-Scores.
The plot shows the relative frequencies of TM-Scores for the CoMo$_{allTM}$ set (CoMo without TM-score filter, orange) and for scores resulting from the comparison of randomly generated fragments (RanComp set, blue). For the CoMo$_{allTM}$ set a significantly higher TM-Score (mean value of 0.34) compared to the RanComp set (mean value of 0.26) can be observed. Mean values are highlighted with dotted lines.

of the CoMo$_{allTM}$ set. This set is identical to the CoMo set, except for omitting the TM-score filtering step. The lower part of the TM-score distribution would be missing, if the CoMo set was used for this analysis. The comparison of the two distributions makes clear that the comparison of randomly chosen fragments (RanComp) results in considerably lower TM-scores than observed in CoMo$_{allTM}$. The mean values are 0.26 and 0.34, respectively. Moreover, the RanComp distribution is right skewed similar a Poisson distribution, wheres CoMo$_{allTM}$ is nearly symmetrical. Both an unpaired t-test and a Wilcoxon signed-rank test gave highly significant results with a p-value $< 1E{-}99$.

The small shoulder of the CoMo$_{allTM}$ distribution at the mean of the RanComp distribution might be due to a certain fraction of structurally unrelated fragments, which are, however, not part of CoMo. In summary, these results conform that a great fraction of the motifs identified by means of FragStatt possess a larger TM-score than expected by chance.

3.4.6 Clustering the CoMo set to identify unique motifs

The motifs in the CoMo set are based on pairwise comparisons, i.e. they link exactly two proteins via a common motif. However, if the predecessors of the motifs originated in the pre-LUCA era, it might be that the sequence similarity of modern manifestations is too low to detect homology, even with a highly sensitive HMM based approach. Thus, it might be that the same or a highly similar motif in two different pairs of proteins was detected: Assume two similar pairwise motifs M_1 and M_2. Motif M_1 was found in the protein chains P_1 and P_2 and motif M_2 was found in the protein chains P_3 and P_4. When further assuming that no similar motif was found between the pairs (e.g. P_1 and P_3, P_1 and P_4, etc.) then there is no way to deduce from the CoMo set that M_1 and M_2 represent the same or a very similar motif. Consequently, to detect these cases, it is necessary to cluster all pairwise motifs M_i and get a set of *unique* motifs M_j^*, whereas a unique motif is defined as a set of pairwise motifs.

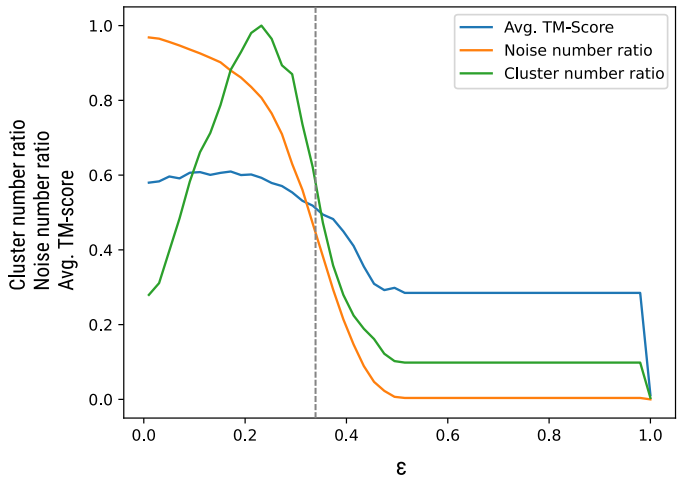


Figure 3.18: Histogram of TM-Scores.
The plot shows three characteristics of the clustering as a function of the DBSCAN neighborhood radius ϵ: The average TM-Score, the "cluster number ratio" which is the normalized number of clusters and the "noise number ratio" indicating the fraction of unclustered motifs. The threshold $\epsilon = 0.33$ chosen for subsequent analyses is indicated by a dotted line.

To reduce the redundancy among the fragments (two fragments $P(M_i)$ and $P(M_j)$ can be identical or highly similar) a sequence based CD-Hit clustering was applied. Only the cluster center fragments were submitted to the structure based clustering. The clustering of the pairwise motifs was carried out using the TM-score as a similarity measure. In this manner, by applying

an all vs. all approach the TM-scores between all individual fragments from all pairwise motifs M_i were calculated. The TM-Score based similarity matrix was converted to a distance matrix and used as input for the DBSCAN clustering algorithm. DBSCAN has two parameters: The neighborhood radius ϵ and the minimum number of cluster members, which was set to two. In order to find an optimal neighborhood radius, the radius ϵ was varied with an increment of 0.01 between 0.0 and 1.0 and three critical parameters were recorded. These were the number of clusters, the fraction of fragments which could be assigned to a cluster (i.e. noise ratio) and the average TM-score within clusters. Trading off the number of clusters, the number of clustered fragments, and the average TM-Score of the clusters in combination with a manual inspection of the clustering output showed that a value of 0.33 for ϵ gave best results: 159 clusters with an average TM-score of 0.52. Of the 7901 fragments 4054 fragments (51 %) could be assigned to a cluster leaving 3847 fragments (49 %) which could not be assigned to a cluster. After visual inspection 25 clusters were dismissed as of poor quality (due to helix only or too short superpositions), leaving 134 valid clusters M_i^* which define the CoMoClust set. All 134 clusters are set as listed in Table 3.6. As a cluster defines a motif the terms cluster and motif will be used synonymously in the following.

Table 3.6: CoMoClust set of 134 clusters each defining a putative ancestral motif.
ID i: Identifier of the cluster/motif M_i^*. #Cl.: Number of CATH classes the motif can be observed in. #Ar.: Number of CATH architectures the motif can be observed in. #To.: Number of CATH topologies the motif can be observed in. H/S/L (%): H(elix), S(heet) and L(oop) secondary struture content of the motif. Classes: List of classes the motif can be observed in. Architectures: List of architectures the motif can be observed in.

ID	#Cl.	#Ar.	#To.	H/S/L (%)	Classes	Architectures
1	4	10	13	59/ 2/39	1, 2, 3, 4	1.10, 1.20, 1.25, 2.40, 2.60, 3.10, 3.30, 3.40, 3.90, 4.10
2	3	8	9	59/ 0/41	1, 2, 3	1.10, 1.25, 2.40, 2.60, 3.10, 3.30, 3.40, 3.90
3	3	5	5	62/ 0/38	1, 2, 3	1.10, 1.25, 2.40, 3.10, 3.40
4	3	4	4	62/ 0/38	1, 3, 4	1.10, 3.10, 3.30, 4.10
5	2	4	4	48/ 0/52	1, 3	1.10, 3.20, 3.30, 3.40
6	3	4	4	75/ 0/25	1, 3, 4	1.10, 1.25, 3.10, 4.10
7	2	4	4	0/21/79	2, 3	2.130, 2.140, 2.70, 3.40
8	1	4	5	35/20/45	3	3.20, 3.40, 3.50, 3.90
9	2	4	4	2/33/65	2, 3	2.102, 2.20, 3.10, 3.40
10	1	4	4	0/57/43	2	2.120, 2.130, 2.140, 2.40
11	1	4	4	41/15/44	3	3.20, 3.30, 3.40, 3.80
12	3	4	4	56/ 0/44	1, 2, 3	1.10, 1.20, 2.30, 3.30
13	2	3	3	49/ 0/51	1, 3	1.10, 3.10, 3.30
14	1	3	3	64/ 0/36	3	3.20, 3.40, 3.90
15	1	3	3	77/ 0/23	1	1.10, 1.20, 1.25
16	1	3	3	1/49/50	2	2.130, 2.140, 2.40

ID	#Cl.	#Ar.	#To.	H/S/L (%)	Classes	Architectures
17	2	3	3	78/ 0/22	1, 2	1.10, 1.25, 2.160
18	2	3	4	3/17/80	2, 3	2.20, 3.30, 3.90
19	2	3	3	65/ 0/35	1, 2	1.10, 1.25, 2.40
20	1	3	4	2/52/46	2	2.30, 2.40, 2.80
21	3	3	3	61/ 0/39	1, 2, 3	1.10, 2.40, 3.10
22	3	3	3	63/ 0/37	1, 2, 3	1.10, 2.40, 3.30
23	1	3	3	41/ 0/59	3	3.20, 3.40, 3.90
24	3	3	3	65/ 0/35	1, 2, 4	1.10, 2.40, 4.10
25	1	3	3	3/47/50	2	2.120, 2.130, 2.60
26	1	3	3	28/19/53	3	3.20, 3.40, 3.90
27	2	3	5	67/ 0/33	1, 2	1.10, 1.20, 2.40
28	1	3	3	77/ 0/23	1	1.20, 1.25, 1.50
29	1	3	3	38/25/37	3	3.10, 3.20, 3.90
30	3	3	3	9/17/74	2, 3, 4	2.10, 3.30, 4.10
31	1	3	3	0/46/54	2	2.120, 2.130, 2.40
32	1	3	4	0/59/41	2	2.30, 2.40, 2.80
33	3	3	3	0/12/88	2, 3, 4	2.20, 3.30, 4.10
34	2	3	3	1/59/40	2, 3	2.30, 2.40, 3.10
35	2	3	4	72/ 0/28	1, 3	1.10, 3.30, 3.40
36	1	3	3	56/ 0/44	3	3.20, 3.40, 3.80
37	1	3	3	0/46/54	2	2.130, 2.40, 2.60
38	2	3	3	65/ 0/35	1, 4	1.10, 1.25, 4.10
39	2	3	3	0/ 0/100	2, 3	2.40, 2.70, 3.40
40	2	3	3	0/14/86	2, 3	2.40, 2.70, 3.40
41	1	3	3	38/ 0/62	3	3.20, 3.30, 3.40
42	1	3	3	0/60/40	2	2.30, 2.40, 2.80
43	1	3	3	39/20/41	3	3.20, 3.40, 3.90
44	1	3	3	34/ 6/60	3	3.20, 3.30, 3.40
45	3	3	4	56/ 1/43	1, 2, 3	1.10, 2.40, 3.30
46	2	3	4	76/ 0/24	1, 3	1.10, 1.20, 3.40
47	1	3	3	39/ 0/61	3	3.20, 3.40, 3.80
48	1	3	4	69/ 0/31	1	1.10, 1.20, 1.25
49	1	2	3	85/ 0/15	1	1.10, 1.20
50	1	2	2	40/11/49	3	3.20, 3.40
51	2	2	2	0/ 0/100	2, 3	2.60, 3.10
52	1	2	3	66/ 0/34	1	1.10, 1.25
53	1	2	2	42/12/46	3	3.20, 3.40
54	1	2	2	28/11/61	3	3.20, 3.40
55	1	2	2	51/ 0/49	3	3.10, 3.70
56	1	2	2	27/21/52	3	3.40, 3.90
57	1	2	2	35/ 0/65	3	3.20, 3.90
58	1	2	2	46/ 0/54	3	3.10, 3.30
59	2	2	2	48/ 0/52	1, 3	1.10, 3.10
60	1	2	2	61/ 0/39	1	1.10, 1.25
61	1	2	3	43/13/44	3	3.20, 3.40
62	1	2	2	0/39/61	2	2.40, 2.70
63	1	2	2	38/ 0/62	3	3.20, 3.40
64	1	2	3	75/ 0/25	1	1.10, 1.20
65	1	2	2	45/ 6/49	3	3.20, 3.40
66	1	2	2	75/ 0/25	1	1.10, 1.25
67	2	2	2	0/35/65	2, 3	2.10, 3.10

ID	#Cl.	#Ar.	#To.	H/S/L (%)	Classes	Architectures
68	2	2	2	0/ 0/100	2, 3	2.60, 3.10
69	1	2	2	73/ 0/27	1	1.10, 1.20
70	1	2	2	33/18/49	3	3.20, 3.40
71	1	2	2	39/ 0/61	3	3.20, 3.40
72	1	2	2	0/49/51	2	2.30, 2.40
73	1	2	2	45/15/40	3	3.30, 3.90
74	1	2	2	42/20/38	3	3.20, 3.40
75	2	2	2	0/56/44	2, 3	2.40, 3.30
76	2	2	2	68/ 0/32	1, 4	1.10, 4.10
77	1	2	2	18/10/72	3	3.20, 3.40
78	1	2	2	40/17/43	3	3.20, 3.80
79	1	2	2	32/20/48	3	3.20, 3.40
80	2	2	2	52/ 0/48	2, 3	2.30, 3.30
81	1	2	2	11/11/78	2	2.60, 2.70
82	1	2	2	39/15/46	3	3.20, 3.40
83	1	2	2	37/10/53	3	3.20, 3.40
84	2	2	2	21/22/57	2, 3	2.10, 3.30
85	1	2	2	42/11/47	3	3.20, 3.40
86	1	2	2	68/ 0/32	1	1.10, 1.20
87	1	2	4	78/ 0/22	1	1.10, 1.20
88	1	2	2	47/19/34	3	3.20, 3.40
89	1	2	2	48/ 0/52	3	3.20, 3.40
90	2	2	2	17/38/45	2, 3	2.40, 3.40
91	1	2	2	44/ 0/56	3	3.20, 3.40
92	2	2	2	0/61/39	2, 3	2.40, 3.30
93	1	2	2	32/14/54	3	3.20, 3.40
94	1	2	2	12/11/77	2	2.60, 2.70
95	1	2	2	33/ 6/61	3	3.20, 3.40
96	1	2	2	28/19/53	3	3.20, 3.40
97	1	2	2	44/10/46	3	3.20, 3.40
98	1	2	2	43/21/36	3	3.10, 3.30
99	1	2	2	70/ 0/30	1	1.10, 1.25
100	1	2	2	0/51/49	2	2.40, 2.60
101	1	2	2	8/41/51	2	2.30, 2.40
102	1	2	2	44/ 0/56	3	3.10, 3.30
103	1	2	2	33/19/48	3	3.20, 3.40
104	1	2	2	44/ 0/56	3	3.20, 3.40
105	1	2	2	52/ 0/48	3	3.20, 3.40
106	2	2	3	19/20/61	2, 3	2.10, 3.30
107	1	2	2	37/21/42	3	3.40, 3.90
108	1	2	3	55/ 0/45	3	3.20, 3.40
109	2	2	2	64/ 0/36	1, 3	1.10, 3.40
110	1	2	2	44/15/41	3	3.20, 3.40
111	2	2	2	17/22/61	2, 3	2.10, 3.30
112	1	2	2	52/ 0/48	3	3.20, 3.40
113	1	2	2	11/20/69	2	2.60, 2.70
114	1	2	3	86/ 0/14	1	1.10, 1.20
115	2	2	2	36/ 0/64	1, 3	1.20, 3.40
116	1	2	2	59/ 0/41	3	3.20, 3.40
117	2	2	3	0/70/30	2, 3	2.40, 3.30
118	1	2	2	47/ 0/53	3	3.20, 3.40

ID	#Cl.	#Ar.	#To.	H/S/L (%)	Classes	Architectures
119	1	2	2	48/11/41	3	3.20, 3.40
120	1	2	2	28/45/27	3	3.10, 3.30
121	1	2	2	61/ 0/39	3	3.20, 3.40
122	1	2	2	48/ 0/52	3	3.10, 3.30
123	1	2	2	42/17/41	3	3.20, 3.40
124	1	2	2	50/15/35	3	3.40, 3.90
125	1	2	2	30/24/46	3	3.40, 3.90
126	1	2	2	37/18/45	3	3.20, 3.40
127	1	2	2	40/24/36	3	3.40, 3.90
128	1	2	2	11/17/72	2	2.10, 2.40
129	1	2	3	53/ 2/45	3	3.20, 3.40
130	1	2	2	55/ 0/45	3	3.20, 3.40
131	1	2	5	70/ 0/30	1	1.10, 1.20
132	1	2	2	6/37/57	2	2.102, 2.20
133	1	2	2	56/ 0/44	3	3.20, 3.40
134	1	2	2	9/48/43	2	2.60, 2.70

Three clusters (M_1^*, M_2^*, M_3^*) connect more than four CATH architectures. Nine clusters connect four (M_4^*–M_{12}^*) CATH architectures, 36 clusters (M_{13}^*–M_{48}^*) connect three CATH architectures and the rest, summing up to 86 clusters (ID M_{49}^*–M_{134}^*), connect two CATH architectures. This means that over 90 % of the clusters respectively motifs occur only in three or less CATH architectures. The majority of detected motifs can not be observed in a wide range of architectures, in other words most of the detected motifs solely occur in a very limited set of architectures.

The motifs which show the most diverse utilization among architectures tend to have a alpha helix-rich secondary structure content (M_1^*–M_6^*, see Figure 3.20). In contrast, motifs with a more even distribution of secondary structure content tend to be found in fewer architectures (M_7^*, see Figure 3.20). Mainly beta sheet comprised motifs also seem to be less diversely distributed amongst architectures than alpha helix motifs are (M_{10}^*, see Figure 3.20).

The five most abundant CATH architectures among all motifs with at least three CATH architectures are `1.10` (Orthogonal Bundle), `3.40` (3-Layer(aba) Sandwich), `2.40` (Beta Barrel), `3.30` (2-Layer Sandwich), and `3.20` (Alpha-Beta Barrel) (compare Figure 3.19). This observation coincides with the frequencies of CATH architectures in the CoMo set as presented previously in Section 3.4.3, which substantiates the finding that most of the motifs are found in a small set of CATH architectures.

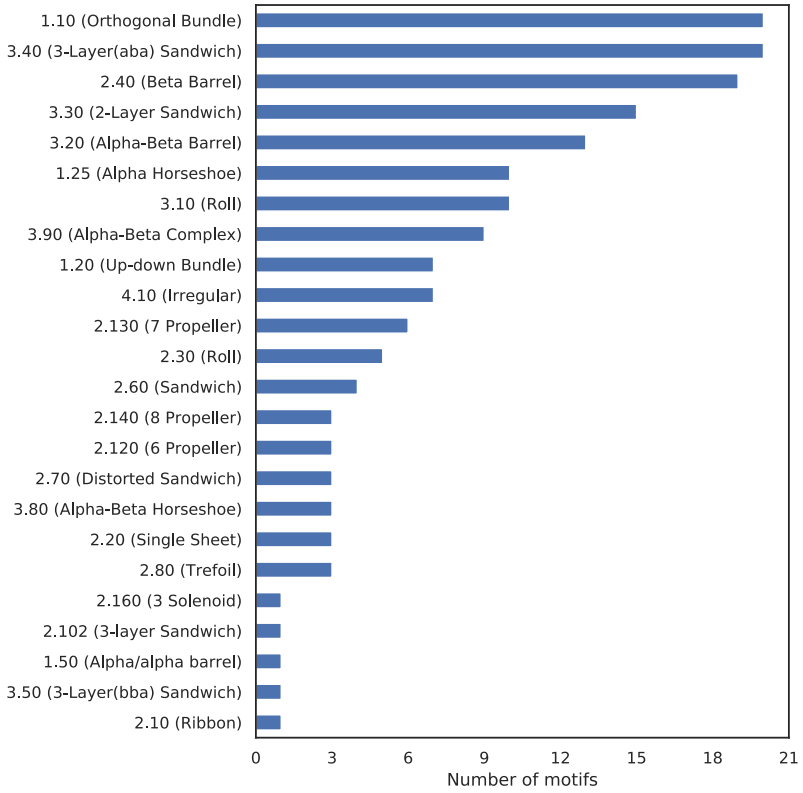


Figure 3.19: Frequency of CATH architectures in the CoMoClust set.
The bar plot visualizes in how many different unique motifs M_i^* a certain CATH architecture occurs.

The clustering of the CoMo set based on structure similarity showed that the CoMo set contains many structurally similar motifs. However, only few of the 134 identified motifs in the CoMoClust set can be found in more than three CATH architectures. Moreover, only slightly more than half of the motifs in the CoMo set could be assigned to a cluster. This suggests that the CoMo set contains many "special cases" of motifs which were only detected between one specific pair of proteins respectively CATH architectures. The findings from the clustering concord with the results from the analysis of the CATH distribution. It appears that only few motifs are universal to some degree. Still, even these most abundant motifs can only be found in a small set of CATH architectures.

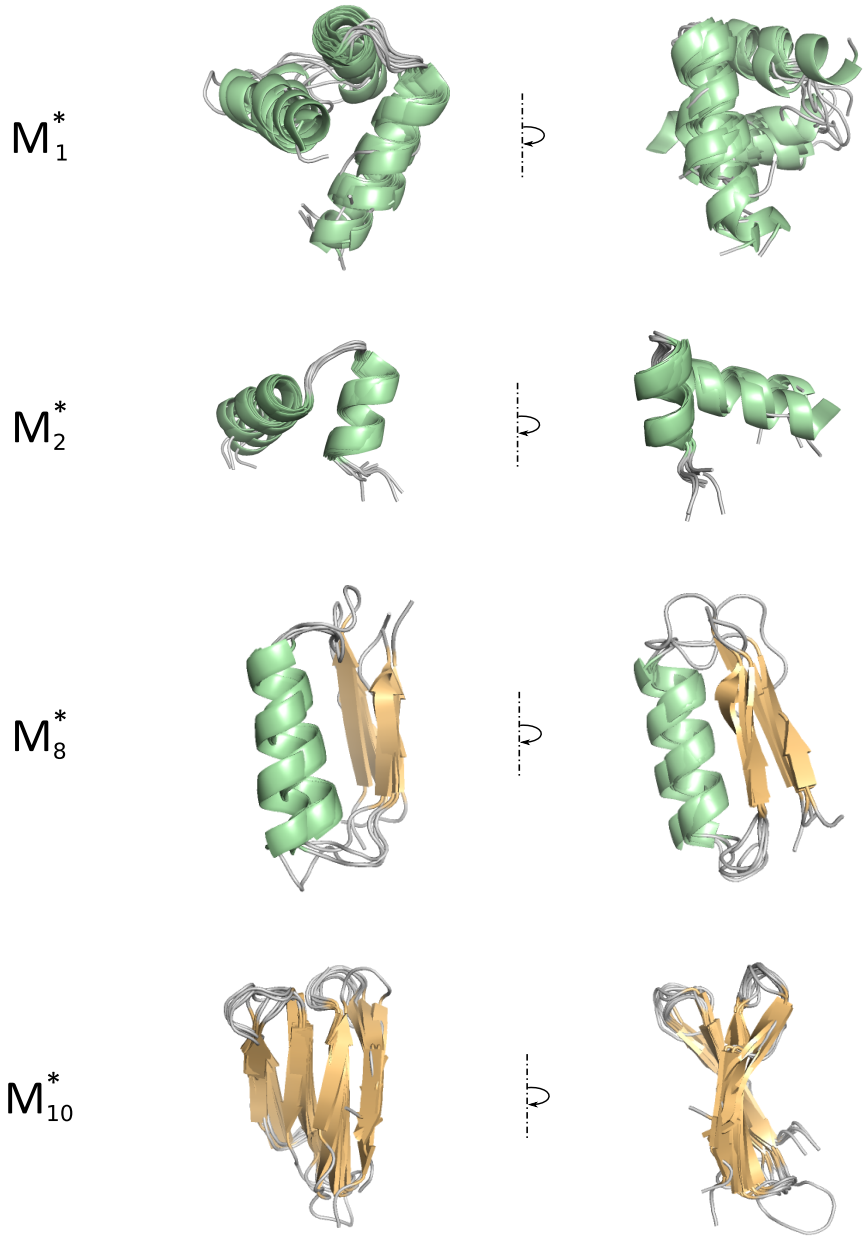


Figure 3.20: Examples of motifs from the CoMoClust set.
Selected motifs are shown as a superposition of their manifestations (fragments) in cartoon representation. M_1^* and M_2^* are typical alpha helix-rich motifs which are common in the CoMo and CoMoClust set. The motif M_8^* has a more even secondary structure content distribution, while M_{10}^* shows a beta sheet arrangement. For this exemplary illustration, motifs with a distinct secondary structure content were selected to show the structural diversity in the CoMoClust set.

3.4.7 Identifying proteins possessing more than one motif

If ancestral motifs have been the building blocks for the evolution of proteins domains, it should be possible to find some proteins that contain at least two or more of the identified motifs. This is why all PDB chains from the CoMo set were scanned for the occurrence of several motifs.

In general it is possible that multiple pairwise common motifs were detected for a single PDB protein chain. These PDB chains can be easily extracted from the CoMo set. However, the fact that for a certain protein chain P_1 common motifs were detected for more then one other protein chain, say for P_2 and P_3, does not automatically imply that two distinct motifs were detected.

In the CoMo set it is a common case that P_2 and P_3 are mapped to the same region on P_1, which means that the same motif was found in three protein chains. Thus it is necessary to filter these trivial cases and detect the cases with multiple distinct motifs. This can be done by identifying clusters of fragments for each PDB protein chain in the CoMo set (compare Figure 3.21). The protein P_1 may share motifs with other proteins P_2 and P_3. Their manifestations can either overlap and be located in the same region of P_1 (compare Figure 3.21, a) or cover different and distant protein regions (compare Figure 3.21, b).

It is not trivial to determine automatically whether the hits fall into a single region or in multiple regions if one wants to consider all motifs of the CoMo dataset. As shown, many of the motifs are highly similar and will thus map to the same region. On the other hand, the assessment of all these motifs is a must for a most comprehensive analysis.

Each manifestation is uniquely specified by a start and an end position. Thus, by using clustering based on a metric for this positions, the localization and the overlap between manifestations can be determined. More detailed, the task is to cluster intervals of natural numbers. The Hausdorff metric is well suited to determine their distance (for details see Chapter 2, Equation 2.25). Thus, for each P_i the start and end positions of all manifestations of the detected motifs were determined and their pairwise Hausdorff distances were calculated. Based on this distance matrix, which was determined for each protein of CoMo the manifestations were clustered by means of DBSCAN. Proteins possessing more than one distinct region occupied by manifestations were identified. The corresponding regions will be called motif regions in the following.

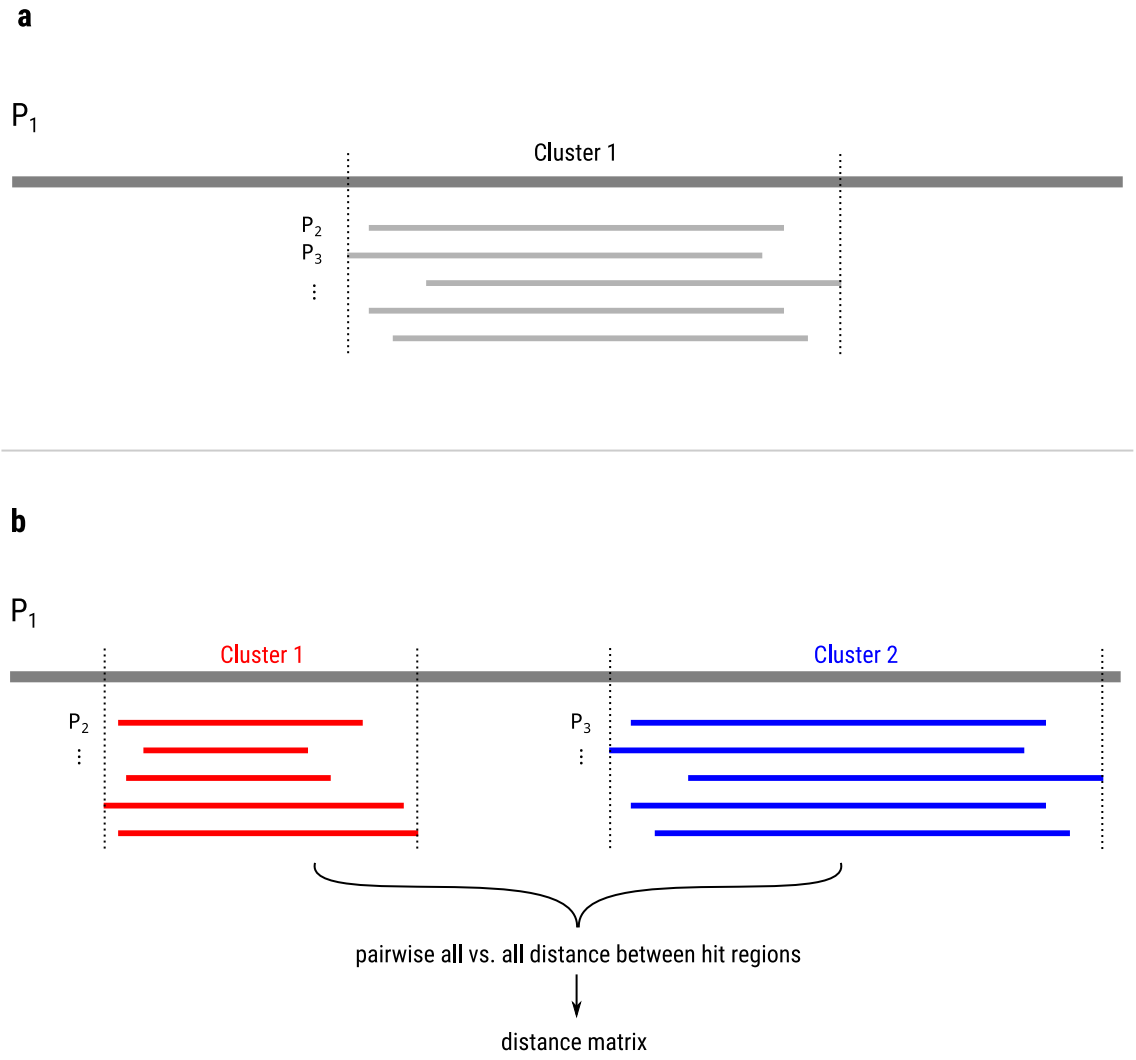


Figure 3.21: Clustering of motif manifestations on a protein chain basis.
a Several manifestations of similar motifs occur at the same region of protein P_1. The hit region (start and end position of the hit) cluster in one contiguous region of P_1. **b** The hits cluster in two distinct regions of P_1 called cluster 1 and cluster 2. To apply a clustering algorithm, a distance matrix based on pairwise distances between the hit regions is calculated.

In total, 135 proteins with more than two distant motif regions could be identified and all cases were inspected manually. 22 protein chains were discarded as of a low quality of the superpositions of the motifs (helix only or too short) leaving 113 cases which belong to 15 different CATH topologies (compare Figure 3.22). The majority (90 %) of the cases belong to the three CATH topologies `3.20.20` (TIM Barrel, 53 %), `3.40.50` (Rossmann fold, 21 %), and `1.25.40` (Serine Threonine Protein Phosphatase 5, 16 %). The 113 cases were classified according to three criteria *repetitive*, *overlapping* and *fragmented*. Repetitive indicates that the same motif was found

multiple times at different positions of the protein chain (compare Figure 3.23). Overlapping indicates that the positions of the motifs overlap (compare Figure 3.24). Fragmented indicates that for a bigger motif smaller sub-fragments were detected and interpreted as an individual motif region (compare Figure 3.25). The classification into these categories was done by manual inspection.

106 of the 113 cases were repetitive and mainly based on motifs detected between TIM barrel and the Rossmann fold. Among the four cases that were non-repetitive and non-fragmented was an *in silico* designed protein (PDB-ID `4J29_A`). As no further data have been provided by the authors, this protein was discarded. The remaining three cases are listed in Table 3.7 and will be presented in the following.

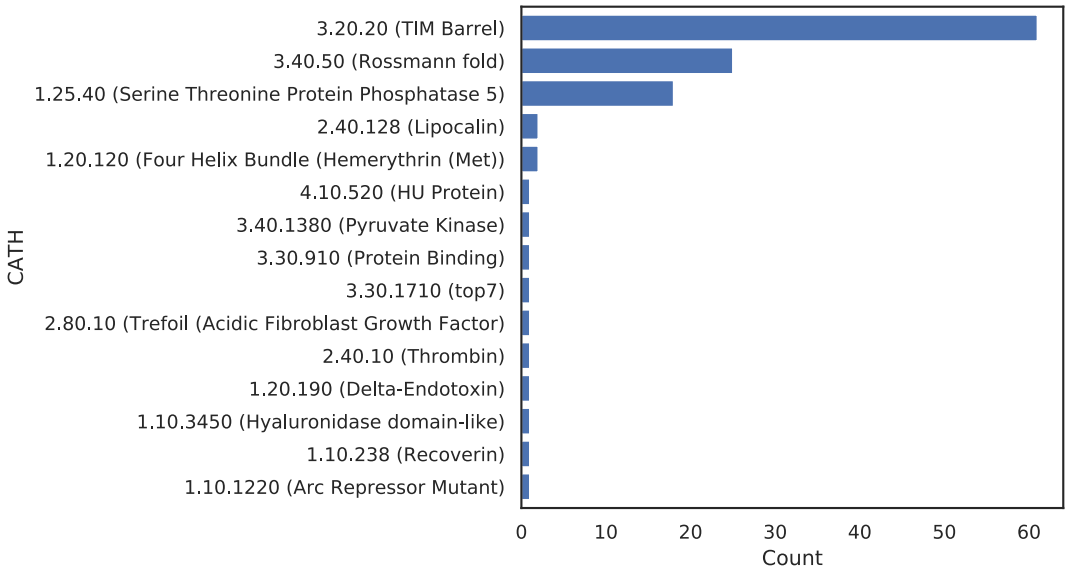


Figure 3.22: CATH topologies of putative multi-motif proteins.
90 % of the detected cases belong to the three CATH topologies `3.20.20` (TIM Barrel, 53 %), `3.40.50` (Rossmann fold, 21 %) and `1.25.40` (Serine Threonine Protein Phosphatase 5, 16 %). The histogram shows absolute counts of the different topologies.

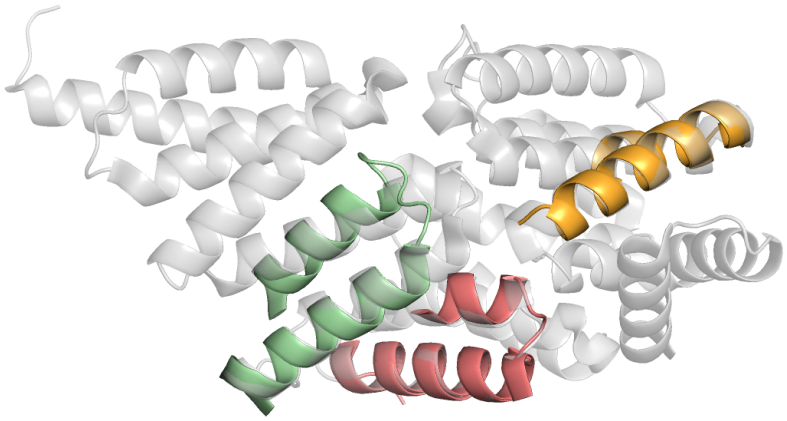


Figure 3.23: A protein possessing a repetitive motif.
The transcription factor MalT, domain III (PDB-ID `1HZ4_A`, gray, transparent) contains three regions featuring similar motifs, show in green, red, and orange. All three motifs are helix-loop-helix combinations.

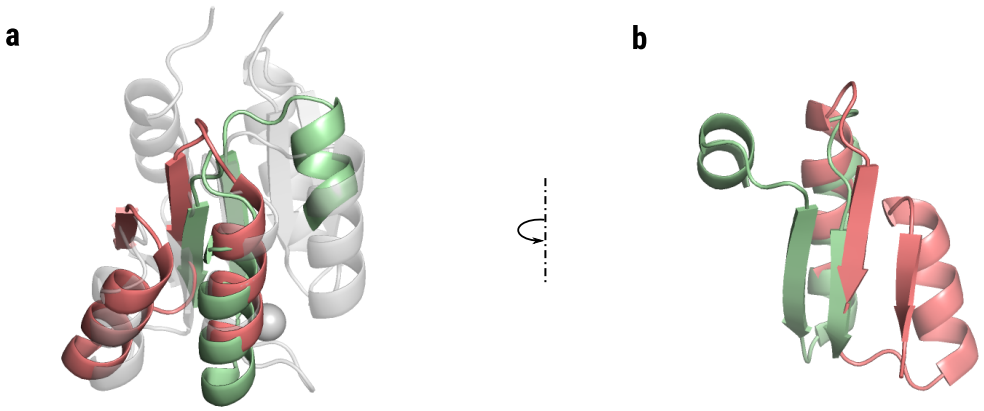


Figure 3.24: A protein possessing two overlapping and repetitive motifs.
(a) The sporulation response regulator Spo0F (PDB-ID `1SRR_C`, gray, transparent) contains two overlapping motifs shown in green and red. **b** The motifs share a $(\beta\alpha)_2$ structure.

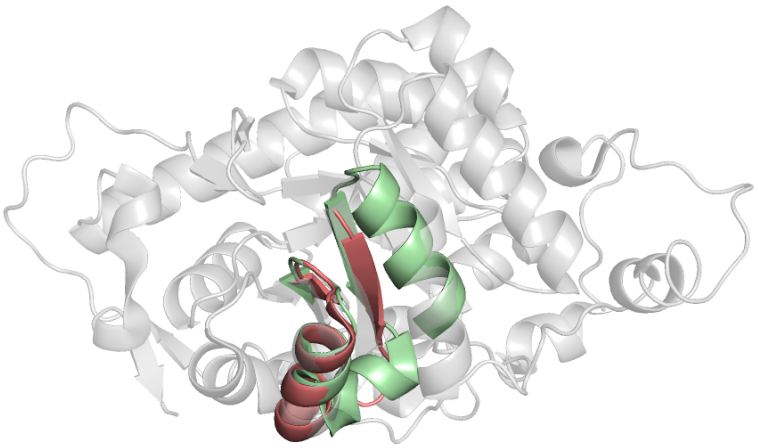


Figure 3.25: Example of fragmented motifs.
For the mandelate dehydrogenase MDH (PDB-ID 1P4C_A, gray, transparent), two regions featuring motifs were detected whereas on region includes the other, i.e. a smaller motif is part of a bigger motif. For each detected region a representative fragment is shown in green and red. One motif features a sheet-helix-sheet structure and the other one a $\beta\alpha_2$ structure.

Table 3.7: Putative multi-motif proteins.
The first column denotes the PDB-ID of proteins which represent a putative case of multi-motif utilization. The second column gives the CATH classification of these proteins. The other columns give for each motif region a list of the PDB-IDs which also feature a manifestation of the motif.

PDB ID	CATH	Motif region 1	Motif region 2	Motif region 3
2QFC_A	1.25.40.10	5K98_B,1.10.260.40 3EUS_B,1.10.260.40	3D3B_A,1.10.940.10 2RKL_A,1.20.5.420	1OM2_A,1.20.960.10
2XBL_B	3.40.50.10490	2NDP_A,4.10.520.10	1JCN_A,3.20.20.70	-
2YVA_A	3.40.50.10490	2NDP_A,4.10.520.10	1I4N_A,3.20.20.70	-

The transcriptional regulator PlcR (PDB-ID 2QFC_A) contains three distinct and separated motifs (compare Figure 3.26). This protein belongs to the CATH superfamily *Tetratricopeptide repeat (TPR) domain* and all superfamily members share the repetition of TPR motifs. Motif region 1 and motif region 2 describe a TPR motif. For the first motif region, the TPR-like motif is found in two DNA-binding proteins (PDB-ID 5K98_B and PDB-ID 3EUS_B), both belonging to the CATH superfamily of *lambda repressor-like DNA-binding domains*. The second TPR-like motif region links PlcR to a transcription related protein (PDB-ID 3D3B_A) and a lipid transport related protein (PDB-ID 2RKL_A), which belong to the CATH superfamily of *NusB-like proteins* and the CATH superfamily of *Immunoglobulin FC, subunit C proteins*, respectively. In conclu-

sion, motif region 1 and 2 both feature a TPR-like structure and can be associated with the same general type of motif. The third motif region links PlcR to a mitochondrial protein (PDB-ID 1OM2_A), which belongs to the CATH superfamily *Mitochondrial outer membrane translocase complex, subunit Tom20 domain.* This motif is structurally distinct from motifs of motif regions 1 and 2, however, as it seems to form an independent and isolated folding unit, it is questionable whether it shouldn't be classified as an individual domain, which would turn PlcR into a multi-domain protein. In this case, the motif would have not been detected as it would not suffice the definition of a putative ancestral motif.

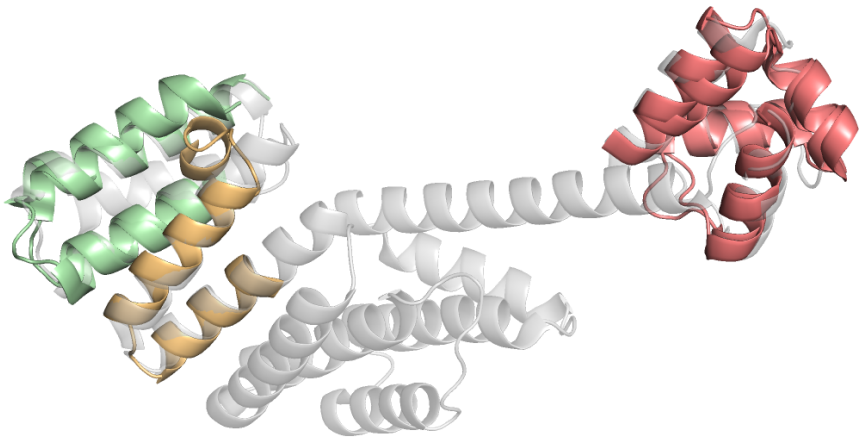


Figure 3.26: Multiple motifs of the protein PlcR (PDB-ID 2QFC_A).
The manifestations of two repetitive TPR-like motifs are shown in green and orange. The manifestations of a third, distinct motif are shown in red.

The isomerase GmhA (PDB-ID 2XBL_B) and the initiator-associating protein diaA (PDB-ID 2YVA_A) share 38 % identical residues (EMBOSS Needle, Madeira et al. 2019, Alignment S1). Both proteins possess a Rossmann fold and belong to the CATH superfamily *Glucose-6-phosphate isomerase like protein.* In both cases (Figure 2.27 a and b) a beta-alpha-beta motif is detected, linking GmhA and diaA to two TIM barrel proteins (PDB-ID 1JCN_A and PDB-ID 1I4N_A, motif region 2). The TIM barrel proteins belong to the CATH superfamily *Aldolase class I.* The second motif in the GmhA and diaA, a alpha-beta motif, links them to a DNA-binding protein (PDB-ID 2NDP_A) which belongs to the CATH superfamily of *IHF-like DNA-binding proteins.*

In conclusion, out of the 309 PDB protein chains in the CoMo set, only one (0.3 %) unique case, represented by two proteins (GmhA and diaA) belonging to the same CATH superfamily, could

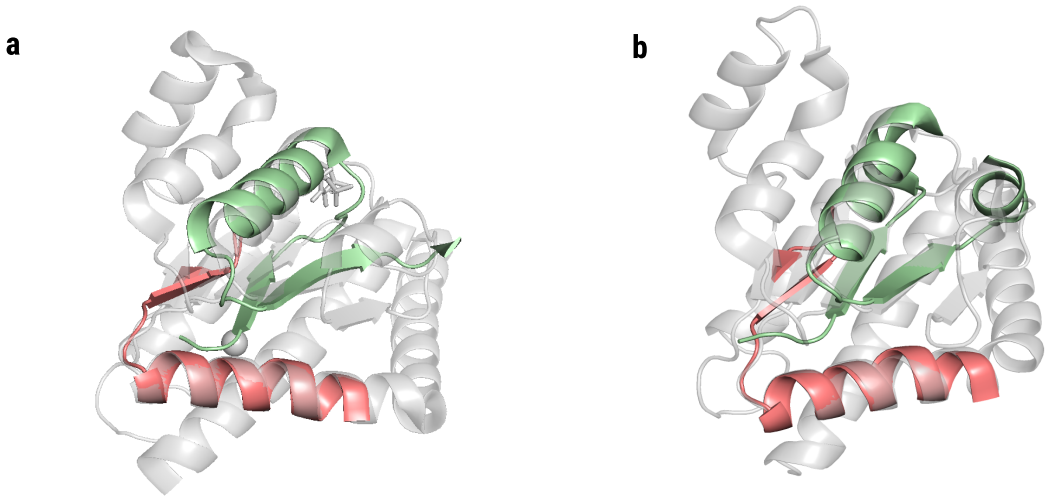


Figure 3.27: Multiple putative motifs in GmhA (PDB-ID 2XBL_B) and diaA (PDB-ID 2YVA_A). GmhA (a) and diaA (b) share a sequence identity of 38 %. For both proteins the same two motifs were detected: A beta-alpha-beta motif (green) and an alpha-beta motif (red).

be identified. This finding strongly argues against the frequent reuse of sub-domain sized motifs evolved since the pre-LUCA era.

Chapter 4

Discussion

4.1 Alternative Approaches for the Detection of Ancestral Protein Motifs

Since Dayhoff and Eck postulated the hypothesis that modern proteins emerged from the recombination of smaller reoccurring fragments (Eck and Dayhoff, 1966a,b), different studies have shown that repetition does indeed play a fundamental role in evolution (Blaber and Lee, 2012; Broom et al., 2012). It is generally assumed that gene duplication followed by fusion and diversification is a "major force in evolution" (Magadum et al., 2013). For instance the $(\beta\alpha)_8$ barrel fold supposedly evolved from the accretion of four quarter-barrel $(\beta\alpha)_2$ fragments (Richter et al., 2010) (compare Figure 4.1). Similar cases of a conserved internal symmetry can be observed in various protein folds (e.g beta propellers, beta barrels, TPR repeat folds, etc.), which further substantiates Dayhoff and Eck's hypothesis, that large globular protein domains can emerge from relatively small protein fragments. Based on this observation the idea of proteins built from smaller modules was developed. Similar as each element of a mathematical vector space can be constructed from its basis vectors, the idea of a set of *protein modules* as the basis of the protein space emerged.

As more an more individual relationships between ancient folds were reported, in the recent years the focus in this research field shifted towards a more general comprehensive approach and efforts were made to identify a basic set of putative ancient protein modules, with the ProVoc set (Alva

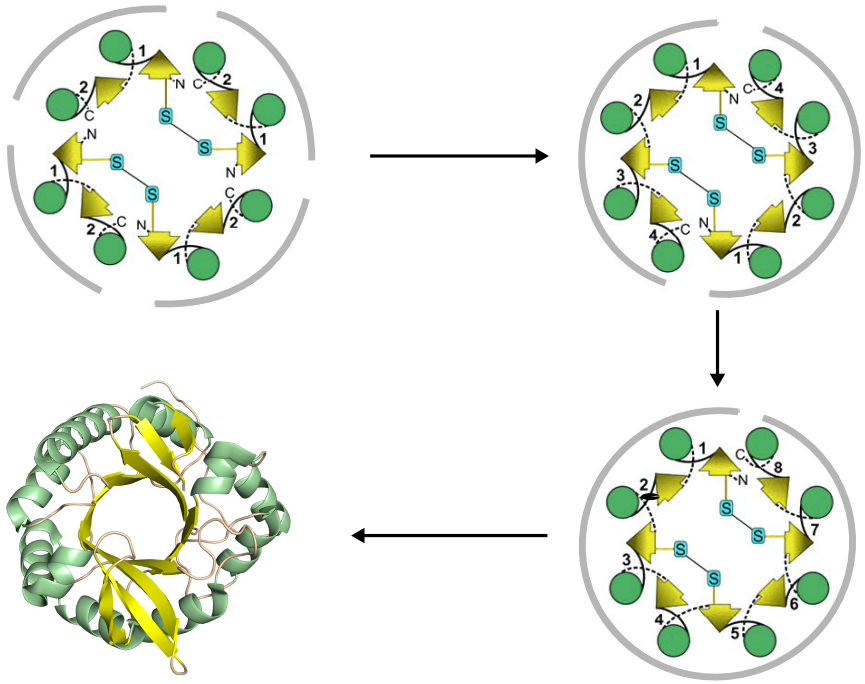


Figure 4.1: Evolutionary scenario of the $(\beta\alpha)_8$ barrel fold.
The three-stage model of Richter et al. suggests the formation of the $(\beta\alpha)_8$ barrel fold from four $(\beta\alpha)_2$ fragments. In the first step, four quarter barrels form a tetrameter facilitated by the formation of disulfide bridges. Subsequently gene duplication and fusion leads to a dimer of $(\beta\alpha)_4$ and finally a monomeric $(\beta\alpha)_8$ protein, the ancestor of modern $(\beta\alpha)_8$ barrel proteins. Figure adapted from Richter et al. (2010).

et al., 2015b) representing one of the most recognized ones. Recently, Ferruz et al. presented a set of putative natural building blocks collected in their database Fuzzle (Ferruz et al., 2020). The CoMo set generated by means of FragStatt constitutes a further orthogonal study aimed at the identification of putative ancestral protein motifs.

4.2 Parameters Affecting The Sensitivity of FragStatt

When designing FragStatt, the main focus was to maximize its sensitivity for the detection of very weak homologous signals. Former studies aimed at the detection of putative ancestral motifs relied on the classical HMM based software HHsearch (Alva et al., 2015b, 2010; Farías-Rico et al., 2014). While being equally sensitive, HHblits is faster compared to HHsearch (Remmert et al., 2012), which allows to employ the computationally expensive cascading of HHblits runs. It has been shown that cascading of HMM searches can be used to further increase sensitiv-

ity (Kaushik et al., 2016). In this manner, the HHblits cascade was implemented. FragStatt computes two HHblits cascades and combines them into a single graph and searches for paths of pairwise alignments. By mapping all paths back to the root sequences it is able to combine many individual pairwise alignments of HHblits. The traceback method implemented in the component SWiFD identifies regions on the root proteins which are frequently hit. In this manner, individually non-significant pairwise HHblits hits can be included in the analysis. False positive hits, i.e "random hits" will not produce a continuous traceback in the match matrix evaluated by SWiFD. The dotplot from the case of the ancestral barrel fragment treated in Section 3.3.1 illustrates this property of SWiFD (compare Figure 3.5). This implicit noise filtering allows to apply very high E-value thresholds (defaults to 1) in the initial HHblits cascade hits. The identified tracebacks are translated into a pairwise motif consisting of two fragments which is submitted to a final scoring for structural similarity which helps to further reduced the amount of false positive motifs.

The mentioned former studies applied a relatively conservative probability threshold for HHsearch. In the case of ProVoc, the authors mentioned that lowering this threshold leads to additional putative motifs (Alva et al., 2015b). The implicit filtering methods implemented in FragStatt allow to explore the option of lowering the thresholds. However, the method of cascading HMM searches as implemented in FragStatt comes with a drawback. The links FragStatt deduces, consist of a cascade of multiple HHblits hits. Each of the hits has an individual HHblits score, E-value and probability. It is not trivial to evaluate the combined significance for the complete path. For this, it would be necessary to create a statistical model of the cascading approach. In the current implementation of FragStatt however, it was decided to solely use the match count as stored in the match matrix to calculate a cumulative raw score for each traceback.

The choice of a suitable database for the HHblits cascade is crucial. The computational costs of the HHblits cascade scale exponentially with an increasing depth of the cascade. For example, if in the first iteration 500 hits were generated for each of these 500 hits again a HHblits run is carried out. If each subsequent run produces 500 hits, this means that in the third iteration 250,000 runs need to be carried out. In a fourth iteration 125 million runs would be needed, etc.

Depending on the maximum number of children per iteration the possible number of iterations is reached fairly quickly, i.e. after three or four iterations. Thus it is crucial to reach the desired point in sequence space within a minimum amount of hops (iterations). This can be achieved by maximizing the distance each hop travels in sequence space. The average distance between hops again is a property defined by the HHblits database (compare Figure 4.2). A very fine grained database shows a small average distance between its elements (HMMs), reducing oder minimizing the distance the HHblits cascade can travel per hop. Consequently, with such a database many hops (iterations of the HHblits cascade) will be needed to reach the desired point it sequence space (i.e reach P_2 starting from P_1 and vice versa). In this case, the possible number of iterations can be exceeded and it is not possible to calculate the HHblits cascade in a reasonable time. On the contrary, choosing a coarse database can have the effect that the distance in sequence space between the HMMs can not be bridged with a HHblits hit, i.e. HHblits can not detected a similarity between the HMMs. In this case, it is not possible to make a link between two proteins P_1 and P_2 at all. These two examples show that it is vital to choose a database which has a suitable average distance between its elements. For example it turned out that the UniClust database (Mirdita et al., 2017) is too fine-grained to be used on a large-scale with FragStatt. The Pfam database on the other hand has proven to be suitable.

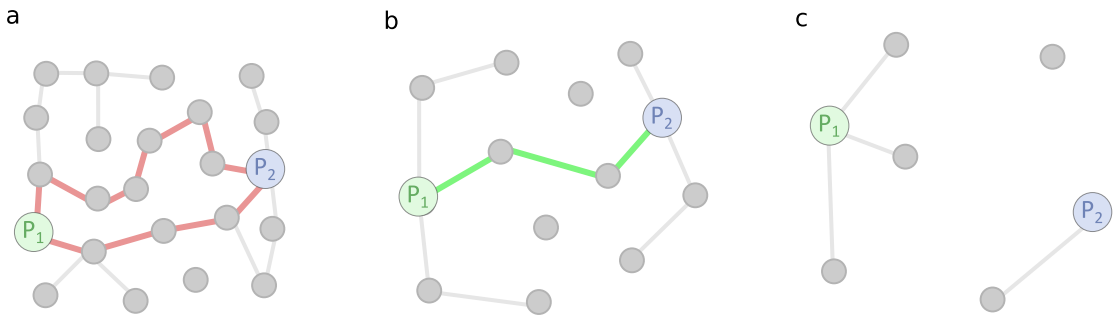


Figure 4.2: Consequences of the choice of the HHblits database.
a A fine-grained HHblits database results in many "short distance hits" leading to long paths. In this case, a high number of iterations of the HHblits cascade is required. Often this exceeds the available computation resources. **b** With a suitable choice of the database, paths from P_1 to P_2 can be detected without exceeding the limit for the depth of the HHblits cascade. **c** A coarse-grained database permits to travel wide distances in the sequence space with a single hop. However, this can prevent that a path from P_1 to P_2 forms, as the required distance to close the path can not be bridged with a single HHblits hit.

4.3 Assessing the Capabilities of FragStatt

When introducing improved algorithms (e.g. MSA algorithms, structural alignment algorithms, etc.) it is common to assess their performance using an well-proven benchmark dataset. However, for specifically tailored algorithms like FragStatt there is no universal benchmark dataset available. It was thus decided to use previously reported motifs as benchmark cases to obtain a qualitative assessment of FragStatts capabilities. In this manner, it was shown that FragStatt is able to detect a previously reported ancestral $(\beta\alpha)_8$ barrel fragment (Farías-Rico et al., 2014) and correctly identify the ancestral relationship between the $(\beta\alpha)_8$ barrel fold and the flavodoxin-like fold. Furthermore in this case FragStatt delivers additional insight: The results indicate that the examined $(\beta\alpha)_8$ barrel originates from the repetition of the same (or homologous) $(\beta\alpha)_8$ fragment. This concords with the evolutionary model of $(\beta\alpha)_8$ barrels proposed by Richter et al. (Richter et al., 2010) and the findings from a computational analysis studying the origin of $(\beta\alpha)_8$ barrels (Söding et al., 2006).

Moreover, due to its cascading nature, FragStatt delivers additional information in the form of intermediate nodes. It was shown that the intermediate nodes of the paths detected by FragStatt concord with the findings of Farías-Rico et al. In general, the information on the intermediate nodes can be used to gain deeper insights into the relationship of a specific pair of proteins: The reported intermediate Pfam nodes can be used to identify links to additional superfamilies and deduced possible evolutionary trajectories.

As a second data basis to evaluate the performance of FragStatt the ProVoc set was considered. From the 40 ProVoc motifs, 34 (85 %) could be identified by FragStatt. Six motifs could not be identified, presumably for two reasons: Firstly, Alva et al. exerted additional manual selection when compiling the ProVoc by evaluating results from clustering HHsearch hits. A possible manual intervention can not be reproduced by FragStatt. Secondly, the data bases differ, the MSAs used to determine ProVoc were generated by means of PSI-BLAST. In contrast FragStatt utilized the Pfam database. It is also notable that the six motifs which could not be detected by FragStatt seem to be relatively uncommon as in each case these motifs occur in only two different SCOP folds (Table 3.3). Furthermore, for the Fuzzle database Ferruz et al. reported

that they could only identify 37 of the motifs from the ProVoc set (Ferruz et al., 2020). The methodology of Fuzzle and ProVoc is almost identical except for the usage of different variants of the SCOP database. This supports the conjecture that a possible manual intervention of Alva et al. and the varying choice of data basis are the reasons why not all motifs of the ProVoc set could be detected by FragStatt. In general, the fact that FragStatt can detect 85 % of the ProVoc motifs, despite using a different data basis, strongly suggests that the motifs identified by two alternative approaches are shared between different protein folds.

4.4 A comparison of CoMo, ProVoc, and Fuzzle

The ProVoc set and the Fuzzle database are both founded on simple HHsearch hits between SCOP domains. The CoMo set, however, is based on the HHblits cascade and the CATH database. The different design choices were made because of the following two reasons: Firstly, as explained in the previous section, the HHblits cascade can be expected to improve the detection sensitivity. Secondly, most similar studies relied on the SCOP database. To explore the effect of the selected structural classification scheme on the detection of putative ancestral protein motifs, it was decided to use the CATH database in this work. By using an alternative approach, a comparison of the orthogonal findings allows one to substantiate the reliability of the results.

The motifs of the ProVoc set are spread over 130 different SCOP folds which constitutes approximately 11 % of the 1,194 considered SCOP folds (Alva et al., 2015b). A slightly higher fraction can be observed for the CoMo set in respect to the CATH database: The motifs are spread over 245 of 1,391 (18 %) considered CATH topologies. The low abundance of the motifs in the ProVoc set conflicts with the proposed hypothesis of universal building blocks. A main consideration in the design of FragStatt was to allow the inclusion of low probability HHblits hits to detect more motifs. Although the motifs in the CoMo set seem to be slightly more abundant than those from the ProVoc set, the utilization of these motifs seems to be far from universal. Alva et al. mention that their set of motifs, as of conservatively chosen thresholds, constitutes a lower bound of ancestral motifs. For the CoMo approach it was tried to extend the homology detection sensitivity. At the same time, however, a slightly more conservative threshold for the

structural similarity was applied: A TM-score > 0.55 for the CoMo set and a TM-score > 0.5 for the ProVoc set. The higher TM-score threshold was chosen as it delivered better results for the clustering of the motifs. In this regard, the Fuzzle database was created with a relatively low TM-score threshold of 0.3 in combination with a RMSD threshold of 3 Å. Moreover, no upper or lower length threshold was used; as a consequence the average motif length in the Fuzzle database is 64 residues and the motifs cover 519 of 1,221 considered SCOP folds which equals to a fraction of approximately 43 %. However, the TM-score threshold of 0.3 is close to the maximum of 0.2 of the null distribution of randomly sampled TM-scores.

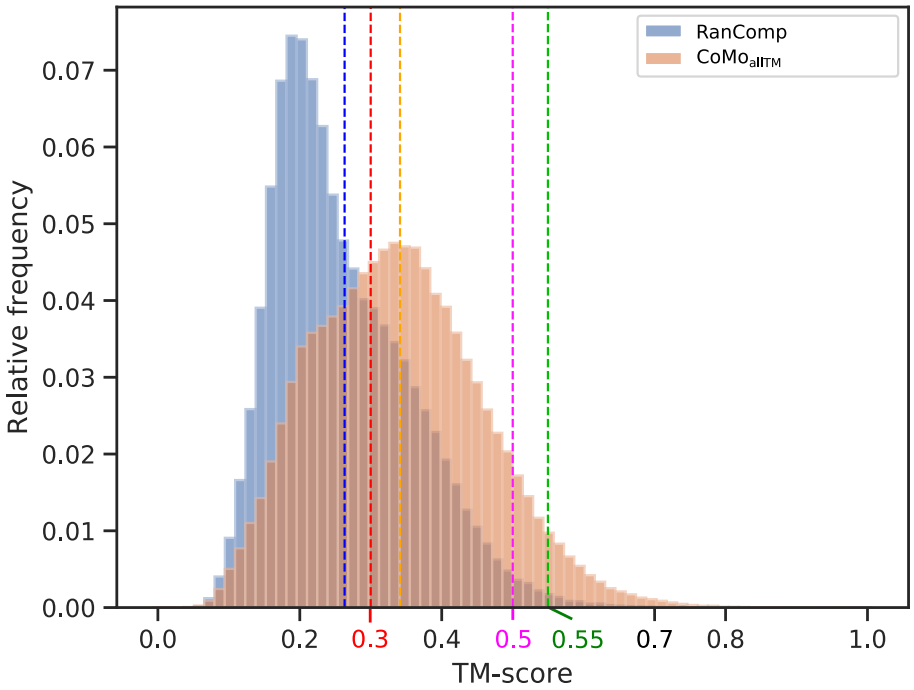


Figure 4.3: Histogram of TM-scores.
The plot shows the relative frequencies of TM-scores for the detected motifs, i.e. pairs of fragments (orange), and for scores resulting from the comparison of randomly generated fragments (blue) as presented in Figure 3.17. For the detected pairs a significantly higher TM-score (mean value of 0.34, orange dotted) compared to the randomly generated set of pairs (mean value of 0.26, orange dotted) can be observed. The threshold of 0.3 chosen for the Fuzzle database is indicated with a red dotted line. The thresholds used for the ProVoc set and the CoMo set are indicated with purple and green dotted lines.

Figure 4.3 shows the TM-score distribution of randomly sampled fragments and the motifs detected by FragStatt. The TM-score distribution was previously treated in Section 3.4.5. It is striking that the false positive rate increases drastically when choosing a TM-score threshold below 0.5. Therefore it can be assumed that the threshold of 0.3 poses a limit of what is a

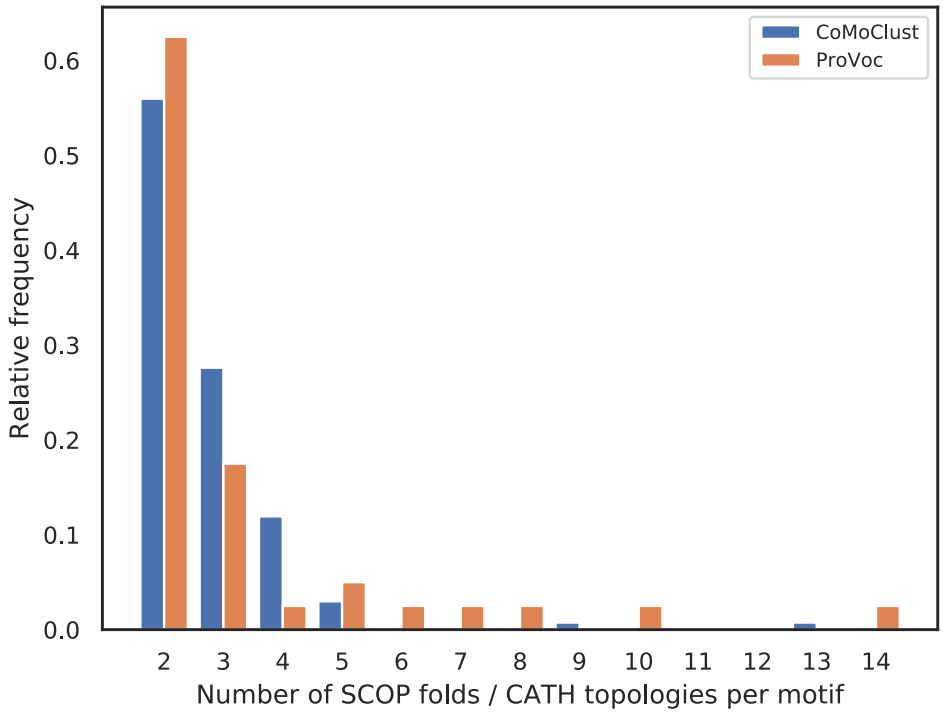


Figure 4.4: Number of folds/topologies per motif in the CoMoClust set and the ProVoc set. The plot shows the distribution of the number of SCOP folds (ProVoc) or CATH topologies (CoMoClust) in which a motif could be found. The distribution shows a power law like shape: Less universal motifs are common and only few motifs occur in many topologies or folds.

reasonable choice for the required lowest structural similarity between the fragments constituting a motif. At the same time it is remarkable that even with such a low TM-score threshold putative ancestral motifs were detected in less than half of the SCOP folds. Further the analysis of TM-score distributions presented in Section 3.4.5 clearly shows that the motifs detected by FragStatt show a significantly higher TM-score than one would except by chance. This rebuts concerns that common motifs between different folds may just be a result of mere chance.

A way to evaluate the universality of the detected motifs is to assess the fold/topology distribution of individual motifs, i.e. the number of folds or topologies in which the motifs occur. These numbers were presented in Table 3.6 for the CoMoClust set and are given in Figure 3.9 for the ProVoc set. Figure 4.4 shows a histogram of these counts. It can be seen that there are only few motifs which cover many folds or topologies (a maximum of 13 for the CoMoClust set

and 14 for the ProVoc set). The majority of motifs occur in only two, three or four folds or topologies. Consequently universal motifs seem to be relativity rare. For the Fuzzle database, Ferruz et al. used a network-based approach instead of classical clustering to group the detected pairwise motifs. The nodes represent proteins and links represent motifs; 2 % of their most connected nodes possess 80 % of the links. Further, the authors of Fuzzle report that the number of links per node in their network follows a power-law distribution. This finding concords with the distributions observed for the ProVoc set and the CoMoClust set.

In conclusion, the CoMo set and the ProVoc set suggest that the detected motifs are not as widely spread among modern proteins as one would expect. If one was to pick a random structure from the PDB database, the chance that it includes a proposed ancestral motif of the ProVoc set or CoMo set is small (the CoMo set includes 2870, and the ProVoc set 239 proteins, while the PDB contains over 35,000 unique proteins at 30 % sequence identity). The Fuzzle approach shows that lowering thresholds for structural similarity increases the number of detected motifs. However, it seems that the occurrence of the majority of motifs still concentrates on a relatively small set of folds or topologies. This concords with the striking dominance of motifs in the CoMo set linking the TIM Barrel and the Rossmann fold CATH topology, which constitute nearly a quarter of all detected motifs (Table S1).

The prevalence of the detected motifs seems to be relatively low which leads to the question of the compatibility of the motifs amongst each other, i.e. how many cases of multi-motif utilization can be observed. Alva et al. reported that they could detect multiple cases of repeated use of the same motif (e.g. repeating TPR motifs). However, they could not observe the utilization of multiple different motifs in the same protein. For the CoMo set multiple cases of different motifs occurring in one protein chain could be detected. However, most of these cases were detected in inherently repetitive protein topologies (e.g TPR like motifs or quarter $(\beta\alpha)_8$ barrel like motifs). In the CoMo set only one convincing case of multi-motif utilization could be identified (compare Figure 3.27). In contrast, for the Fuzzle 1,155 protein domains were reported to contain multiple motifs. This – compared to the ProVoc set and CoMo set – high number most likely results from the low structure similarity threshold required for the motifs in the Fuzzle database. Furthermore, it seems that similar motifs and motifs occurring in inherently repetitive motifs

were not filtered out. For example the repeated utilization of TPR like motifs and quarter $(\beta\alpha)_8$ barrel like motifs, stated as exemplary cases by Ferruz et al., were neither considered multi-motif cases in this work nor for the ProVoc set.

4.5 Protein Modules as the Building Blocks of Evolution

"Nature is a tinkerer, not an inventor." With this phrase François Jacob condensates in his famous 1977 essay (Jacob, 1977) the point that evolution is mainly driven by the reuse of existing and available "construction" material and that the *de novo* emergence of genetic material is rare. The results from this work and former studies substantiate the conjecture that at least to some extent the reuse of small motifs played a role in the origin of protein folds and that the origins of some folds possibly goes back to the formation of RNA-motif complexes in the RNA world era (Alva et al., 2015b). However, the relatively low abundance of the motifs in modern proteins does not provide sufficient evidence to conclude a general modular origin of proteins. This leads to two opposing hypotheses: On the one hand it could be the case that a large portion of ancient motifs is not detectable anymore in recent proteins (at least with the available methods), which would render the currently known motifs a small subset representing the few conserved ones. On the other hand, if one assumes that the detected motifs represent a large fraction of the ancient building blocks, the idea of a modular assembly of protein folds is difficult to sustain, as there is only a very limited number of folds which can be constructed from a combination of the known motifs (e.g. the long-known cases of $(\beta\alpha)_8$ barrels or TPR repeat folds). In the following these hypotheses will be discussed briefly.

The first option, that only a small subset of ancestral motifs is known yet, leads to the question of how many motifs there could be. For example Alva et al. assume that the number of detected motifs, will reach one hundred in the next decades (Alva et al., 2015b). However, this contrasts with the high number of motifs in the Fuzzle database and highlights the inherently different stringency criteria applied in different approaches. From the perspective of stringency, the CoMo approach can be located between the Fuzzle and the ProVoc study. This estimation also complies with the fact that the number of 134 unique motif clusters in the CoMo set is higher than the

number of motifs in the ProVoc but lower than the number of motifs in the Fuzzle database.

The varying number of motifs resulting from different approaches reveals a basic problem for the detection of ancestral protein motifs: There is no agreed upon definition of an ancestral protein motif. For example there is no consent regarding the required structural similarity or the length limit of the motifs. The lack of a clear definition makes it difficult to estimate the number of detectable motifs. However, similar to the estimate of Alva et al. the CoMo approach suggests a few hundred detectable motifs. Provided that more sensitive motif detection methods exist in the future, it can only be speculated whether the additional motifs, will cover the rest of the fold space.

The second option, that most of the existing motifs are already known, leads to the question of how folds which do not contain any ancestral motifs emerged. As stated before, all three approaches found motifs in a small or at most moderate number of SCOP folds or CATH topologies: The fractions were 11 % for the ProVoc set, 18 % for the CoMo set, and 43 % for the Fuzzle database. However, independently of the approach, an enrichment of folds proposed to be ancient can be observed in all cases. The authors Caetano et al. proposed a set of the five most ancient folds based on SCOP (Caetano-Anolles et al., 2009). Mapping them to the CATH database showed that two of the proposed SCOP folds are subsumed in the *Rossmann fold* topology of CATH. In this manner, the four supposedly most ancient CATH topologies are *Rossmann fold*, *TIM Barrel*, *Trp Operon Repressor* and *Alpha-Beta Plaits*. In the CoMo set all four CATH topologies can be observed. Similary, the authors of ProVoc and Fuzzle report an enrichment of the corresponding SCOP folds, as well. The seemingly low abundance or lack of putative ancestral motifs for most other folds could be explained if these had evolved after the most ancient folds, i.e. in the post-RNA world era. A relatively recent study finds that random sequences yield beneficial effects when expressed in bacteria (Neme et al., 2017). The authors report that for 25 % of random sequences introduced in *E. coli* an enriched expression level and a positive effect on the growth rate could be observed. In this manner, they concluded that "random sequences are an abundant source of bioactive RNAs or peptides" (Neme et al., 2017). If this finding can be substantiated, it would render the possibility of a *de novo* emergence of protein folds more likely than currently assumed. In this line of reasoning the lack of ancestral

motifs in such folds could be explained.

In conclusion, the results from this work and the other discussed studies support the hypothesis that some of the most ancient protein folds evolved within the era of the RNA world from smaller protein motifs. However, there is not sufficient evidence to conclude a similar evolutionary origin for the complete fold space. It might be the case that only a few ancient folds emerged in the RNA world, while for younger protein folds from the post-RNA world, another mode of origin possibly by *de novo* emergence comes into question.

References

Abdelhamid, Y., P. Brear, J. Greenhalgh, X. Chee, T. Rahman, and M. Welch (2019). Evolutionary plasticity in the allosteric regulator-binding site of pyruvate kinase isoform PykA from *Pseudomonas aeruginosa*. *Journal of Biological Chemistry*, 294(42):15505–15516.

Abriata, L. A., G. E. Tamò, and M. Dal Peraro (2019). A further leap of improvement in tertiary structure prediction in CASP13 prompts new routes for future assessments. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1100–1112.

Alexandrov, N. N. and D. Fischer (1996). Analysis of topological and nontopological structural similarities in the PDB: new examples with old structures. *Proteins: Structure, Function, and Bioinformatics*, 25(3):354–365.

Altschul, S. F. and E. V. Koonin (1998). Iterated profile searches with PSI-BLAST–a tool for discovery in protein databases. *Trends in Biochemical Sciences*, 23(11):444–447.

Altschul, S. F., T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.

Alva, V., M. Remmert, A. Biegert, A. Lupas, and J. Söding (2010). A galaxy of folds. *Protein Science*, 19(1):124–130.

Alva, V., J. Söding, and A. N. Lupas (2015a). elife-09410-fig3-data1-v2.docx. *Elife*, 4:e09410. https://elifesciences.org/articles/09410#fig3.

Alva, V., J. Söding, and A. N. Lupas (2015b). A vocabulary of ancient peptides at the origin of folded proteins. *Elife*, 4:e09410.

Andrade, M. A., C. Perez-Iratxeta, and C. P. Ponting (2001). Protein repeats: structures, functions, and evolution. *Journal of Structural Biology*, 134(2-3):117–131.

Berman, H. M., T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, et al. (2002). The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, 58(6):899–907.

Berman, H. M., T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, et al. (2019). PDB FTP server. ftp://ftp.wwpdb.org/pub/pdb/derived_data/.

Betts, H. C., M. N. Puttick, J. W. Clark, T. A. Williams, P. C. Donoghue, and D. Pisani (2018). Integrated genomic and fossil evidence illuminates life's early evolution and eukaryote origin. *Nature Ecology & Evolution*, 2(10):1556–1562.

Blaber, M. and J. Lee (2012). Designing proteins from simple motifs: opportunities in top-down symmetric deconstruction. *Current Opinion in Structural Biology*, 22(4):442–450.

Božič, A. L., A. Šiber, and R. Podgornik (2013). Statistical analysis of sizes and shapes of virus capsids and their resulting elastic properties. *Journal of Biological Physics*, 39(2):215–228.

Branden, C. I. and J. Tooze (2012). *Introduction to protein structure*. Garland Science.

Brocchieri, L. and S. Karlin (2005). Protein length in eukaryotic and prokaryotic proteomes. *Nucleic Acids Research*, 33(10):3390–3400.

Bronson, J. (2019). bidict: The bidirectional mapping library for Python. https://github.com/jab/bidict.

Broom, A., A. C. Doxey, Y. D. Lobsanov, L. G. Berthin, D. R. Rose, P. L. Howell, B. J. McConkey, and E. M. Meiering (2012). Modular evolution and the origins of symmetry: reconstruction of a three-fold symmetric globular protein. *Structure*, 20(1):161–171.

Caetano-Anolles, G., M. Wang, D. Caetano-Anollés, and J. E. Mittenthal (2009). The origin, evolution and structure of the protein world. *Biochemical Journal*, 417(3):621–637.

Carugo, O. and S. Pongor (2001). A normalized root-mean-spuare distance for comparing protein three-dimensional structures. *Protein Science*, 10(7):1470–1473.

CATH Team (2020). CATH webserver. http://www.cathdb.info/browse/tree.

Chaudhuri, I., J. Söding, and A. N. Lupas (2008). Evolution of the β-propeller fold. *Proteins: Structure, Function, and Bioinformatics*, 71(2):795–803.

Chavent, M. (2004). A Hausdorff distance between hyper-rectangles for clustering interval data. In *Classification, Clustering, and Data Mining Applications*, Pp. 333–339. Springer.

Chothia, C. (1992). One thousand families for the molecular biologist. *Nature*, 357(6379):543–544.

Chothia, C. and J. Gough (2009). Genomic and structural aspects of protein evolution. *Biochemical Journal*, 419(1):15–28.

Cock, P. J., T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. de Hoon (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–3.

Coles, M., S. Djuranovic, J. Söding, T. Frickey, K. Koretke, V. Truffault, J. Martin, and A. N. Lupas (2005). AbrB-like transcription factors assume a swapped hairpin fold that is evolutionarily related to double-psi β barrels. *Structure*, 13(6):919–928.

Csaba, G., F. Birzele, and R. Zimmer (2009). Systematic comparison of SCOP and CATH: a new gold standard for protein structure analysis. *BMC Structural Biology*, 9(1):23.

Darwin, C. (1859). *On the Origin of Species*. London: John Murray.

Doolittle, W. F. (2000). Uprooting the tree of life. *Scientific American*, 282(2):90–95.

Eck, R. V. and M. O. Dayhoff (1966a). *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Silver Spring.

Eck, R. V. and M. O. Dayhoff (1966b). Evolution of the structure of ferredoxin based on living relics of primitive amino acid sequences. *Science*, 152(3720):363–366.

Eddy, S. R. (2020). HMMER. http://hmmer.org.

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–7.

El-Gebali, S., J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, et al. (2018). The Pfam protein families database in 2019. *Nucleic Acids Research*, 47(D1):D427–D432.

EMBL-EBI (2020). What are profile hidden Markov models (HMMs). https://ebi.ac.uk/training/online/course/pfam-database-creating-protein-families/what-are-profile-hidden-markov-models-hmms. License: Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

European Bioinformatics Institute (2020). Pfam FTP server. ftp://ftp.ebi.ac.uk/pub/databases/Pfam.

Facebook (2012–2020). RocksDB: A persistent key-value store for flash and RAM storage. https://rocksdb.org/.

Farías-Rico, J. A., S. Schmidt, and B. Höcker (2014). Evolutionary relationship of two ancient protein superfolds. *Nature Chemical Biology*, 10(9):710–5.

Ferruz, N., F. Lobos, D. Lemm, S. Toledo-Patino, J. A. Farías-Rico, S. Schmidt, and B. Höcker (2020). Identification and analysis of natural building blocks for evolution-guided fragment-based Protein Design. *Journal of Molecular Biology*.

Fetrow, J. S. and A. Godzik (1998). Function driven protein evolution. A possible proto-protein for the RNA-binding proteins. In *Pacific Symposium on Biocomputing*, volume 3, Pp. 485–496.

Fox, N. K., S. E. Brenner, and J.-M. Chandonia (2014). SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309.

Fox, N. K., S. E. Brenner, and J.-M. Chandonia (2020). SCOP web server. https://scop.berkeley.edu/downloads.

Fu, Z., G. Indrisiunaite, S. Kaledhonkar, B. Shah, M. Sun, B. Chen, R. A. Grassucci, M. Ehrenberg, and J. Frank (2019). The structural basis for release-factor activation during translation termination revealed by time-resolved cryogenic electron microscopy. *Nature Communications*, 10(1):1–7.

Gilbert, W. (1986). Origin of life: The RNA world. *Nature*, 319(6055):618–618.

Glansdorff, N., Y. Xu, and B. Labedan (2008). The last universal common ancestor: emergence, constitution and genetic legacy of an elusive forerunner. *Biology direct*, 3(1):29.

Gulbis, J. M., Z. Kelman, J. Hurwitz, M. O'Donnell, and J. Kuriyan (1996). Structure of the C-terminal region of p21WAF1/CIP1 complexed with human PCNA. *Cell*, 87(2):297–306.

Hall, B. K. (2012). *Homology: The hierarchial basis of comparative biology*. Academic Press.

Hofmockel, S. (2015). pyrocksdb: Python bindings for RocksDB. https://github.com/stephan-hof/pyrocksdb.

Holm, L. and C. Sander (1995). Dali: a network tool for protein structure comparison. *Trends in Biochemical Sciences*, 20(11):478–480.

Horning, D. P. and G. F. Joyce (2016). Amplification of RNA by an RNA polymerase ribozyme. *Proceedings of the National Academy of Sciences*, 113(35):9786–9791.

Huang, P.-S., K. Feldmeier, F. Parmeggiani, D. A. F. Velasco, B. Höcker, and D. Baker (2016). De novo design of a four-fold symmetric TIM-barrel protein with atomic-level accuracy. *Nature Chemical Biology*, 12(1):29.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Höcker, B., S. Schmidt, and R. Sterner (2002). A common evolutionary origin of two elementary enzyme folds. *FEBS Letters*, 510(3):133–5.

Illergård, K., D. H. Ardell, and A. Elofsson (2009). Structure is three to ten times more conserved than sequence—a study of structural response in protein cores. *Proteins: Structure, Function, and Bioinformatics*, 77(3):499–508.

Jacob, F. (1977). Evolution and tinkering. *Science*, 196(4295):1161–1166.

Joyce, G. F. (2002). The antiquity of RNA-based evolution. *Nature*, 418(6894):214–221.

Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 34(5):827–828.

Katoh, K., K. Misawa, K. Kuma, and T. Miyata (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–66.

Kaushik, S., A. G. Nair, E. Mutt, H. P. Subramanian, and R. Sowdhamini (2016). Rapid and enhanced remote homology detection by cascading hidden Markov model searches in sequence space. *Bioinformatics*, 32(3):338–344.

Kessel, A. and N. Ben-Tal (2010). *Introduction to proteins: Structure, function, and motion.* CRC Press.

Kolodny, R. and N. Linial (2004). Approximate protein structural alignment in polynomial time. *Proceedings of the National Academy of Sciences*, 101(33):12201–12206.

Kolodny, R., L. Pereyaslavets, A. O. Samson, and M. Levitt (2013). On the universe of protein folds. *Annual Review of Biophysics*, 42:559–582.

Kolodny, R., D. Petrey, and B. Honig (2006). Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction. *Current Opinion in Structural Biology*, 16(3):393–398.

Koonin, E. V. (2003). Comparative genomics, minimal gene-sets and the last universal common ancestor. *Nature Reviews Microbiology*, 1(2):127–136.

Koonin, E. V. (2005). Orthologs, paralogs, and evolutionary genomics. *Annu. Rev. Genet.*, 39:309–338.

Koonin, E. V., Y. I. Wolf, and G. P. Karev (2002). The structure of the protein universe and genome evolution. *Nature*, 420(6912):218–223.

Kopec, K. O. and A. N. Lupas (2013). β-Propeller blades as ancestral peptides in protein evolution. *PLoS One*, 8(10).

Kuhlman, B., G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker (2003). Design of a novel globular protein fold with atomic-level accuracy. *Science*, 302(5649):1364–1368.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, Pp. 707–710.

Levitt, M. and C. Chothia (1976). Structural patterns in globular proteins. *Nature*, 261(5561):552–558.

Levitt, M. and M. Gerstein (1998). A unified statistical framework for sequence comparison and structure comparison. *Proceedings of the National Academy of Sciences*, 95(11):5913–5920.

Li, W. and A. Godzik (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–9.

Looger, L. L., M. A. Dwyer, J. J. Smith, and H. W. Hellinga (2003). Computational design of receptor and sensor proteins with novel functions. *Nature*, 423(6936):185–190.

Madeira, F., Y. M. Park, J. Lee, N. Buso, T. Gur, N. Madhusoodanan, P. Basutkar, A. R. Tivey, S. C. Potter, R. D. Finn, et al. (2019). The EMBL-EBI search and sequence analysis tools APIs in 2019. *Nucleic Acids Research*, 47(W1):W636–W641.

Magadum, S., U. Banerjee, P. Murugan, D. Gangapur, and R. Ravikesavan (2013). Gene duplication as a major force in evolution. *Journal of Genetics*, 92(1):155–161.

McKinney, W. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, eds., Pp. 51 – 56.

Meier, M. (2017). hh_reader.py. https://github.com/soedinglab/hh-suite/blob/master/scripts/hh_reader.py.

Mirdita, M., L. von den Driesch, C. Galiez, M. J. Martin, J. Söding, and M. Steinegger (2017). Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Research*, 45(D1):D170–D176.

Mitchell, A., H.-Y. Chang, L. Daugherty, M. Fraser, S. Hunter, R. Lopez, C. McAnulla, C. McMenamin, G. Nuka, S. Pesseat, et al. (2014). The InterPro protein families database: the classification resource after 15 years. *Nucleic Acids Research*, 43(D1):D213–D221.

Murzin, A. G., S. E. Brenner, T. Hubbard, and C. Chothia (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540.

Nagano, N., E. G. Hutchinson, and J. M. Thornton (1999). Barrel structures in proteins: Automatic identification and classification including a sequence analysis of TIM barrels. *Protein Science*, 8(10):2072–84.

Neme, R., C. Amador, B. Yildirim, E. McConnell, and D. Tautz (2017). Random sequences are an abundant source of bioactive RNAs or peptides. *Nature Ecology & Evolution*, 1(6):1–7.

Notredame, C., D. G. Higgins, and J. Heringa (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–17.

Ohno, S. (2013). *Evolution by Gene Duplication*. Springer Science & Business Media.

Opitz, C. A., M. Kulke, M. C. Leake, C. Neagoe, H. Hinssen, R. J. Hajjar, and W. A. Linke (2003). Damped elastic recoil of the titin spring in myofibrils of human myocardium. *Proceedings of the National Academy of Sciences*, 100(22):12688–12693.

Orengo, C., A. Bateman, and V. Uversky (2014). *Protein families: Relating protein sequence, structure, and function*, Wiley Series in Protein and Peptide Science. Wiley.

Orengo, C. A., J. E. Bray, D. W. Buchan, A. Harrison, D. Lee, F. M. G. Pearl, I. Sillitoe, A. E. Todd, and J. M. Thornton (2002). The CATH protein family database: a resource for structural and functional annotation of genomes. *Proteomics*, 2:11–21.

Park, J., K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia (1998). Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210.

Pearson, W. R. (2013). An introduction to sequence similarity ("homology") searching. *Current Protocols in Bioinformatics*, 42(1):3–1.

Peixoto, T. P. (2014). The graph-tool python library. *figshare*.

Pesquita, C. (2017). Semantic similarity in the gene ontology. In *The Gene Ontology Handbook*, Pp. 161–173. Humana Press, New York, NY.

Pfam Team (2020). Pfam webserver. https://pfam.xfam.org/.

Remmert, M., A. Biegert, A. Hauser, and J. Söding (2012). HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2):173.

Remmert, M., A. Biegert, D. Linke, A. N. Lupas, and J. Söding (2010). Evolution of outer membrane β-barrels from an ancestral $\beta\beta$ hairpin. *Molecular biology and evolution*, 27(6):1348–1358.

Richter, M., M. Bosnali, L. Carstensen, T. Seitz, H. Durchschlag, S. Blanquart, R. Merkl, and R. Sterner (2010). Computational and experimental evidence for the evolution of a $(\alpha\beta)_8$-barrel protein from an ancestral quarter-barrel stabilised by disulfide bonds. *Journal of Molecular Biology*, 398(5):763–773.

Russell, A. (2001). On the evolution of protein folds: are similar motifs in different protein folds the result of convergence, insertion, or relics of an ancient peptide world? *Journal of Structural Biology*, 134:191–203.

Schrödinger, LLC (2019). The PyMOL Molecular Graphics System.

SCOP Team (2020). SCOP webserver statistics. http://scop.mrc-lmb.cam.ac.uk/stats.

Shafee, T. (2016). Summary of protein structure (primary, secondary, tertiary, and quaternary) using the example of PCNA. https://commons.wikimedia.org/wiki/File:Ortholog_paralog_analog_(homologs).svg. License: Attribution 4.0 International (CC BY 4.0).

Shafee, T. (2018). Ancestral gene duplication. https://en.wikipedia.org/wiki/File:Protein_structure_(full).png. License: Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

Shannon, P., A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–504.

Shindyalov, I. N. and P. E. Bourne (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(9):739–747.

Shindyalov, I. N. and P. E. Bourne (2000). An alternative view of protein fold space. *Proteins: Structure, Function, and Bioinformatics*, 38(3):247–260.

Siew, N., A. Elofsson, L. Rychlewski, and D. Fischer (2000). MaxSub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785.

Skolnick, J., A. K. Arakaki, S. Y. Lee, and M. Brylinski (2009). The continuity of protein structure space is an intrinsic property of proteins. *Proceedings of the National Academy of Sciences*, 106(37):15690–15695.

Smith, T. F., M. S. Waterman, et al. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Söding, J., M. Remmert, and A. Biegert (2006). HHrep: de novo protein repeat detection and the origin of TIM barrels. *Nucleic Acids Research*, 34(suppl_2):W137–W142.

Steinegger, M., M. Meier, M. Mirdita, H. Voehringer, S. J. Haunsberger, and J. Soeding (2019a). HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, 473.

Steinegger, M., M. Meier, M. Mirdita, H. Voehringer, S. J. Haunsberger, and J. Soeding (2019b). HHsuite: pre-built databases. http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/.

Student (1908). The probable error of a mean. *Biometrika*, Pp. 1–25.

Söding, J. (2005). Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21(7):951–60.

Söding, J. and A. N. Lupas (2003). More than the sum of their parts: on the evolution of proteins from peptides. *Bioessays*, 25(9):837–46.

Theobald, D. L. (2010). A formal test of the theory of universal common ancestry. *Nature*, 465(7295):219–222.

UniProt Consortium (2019). UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515.

UniProt Consortium (2020). UniRef UniProt release 2020_01. https://www.uniprot.org/statistics/UniRef.

Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, and Reddy (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272.

Waterhouse, A. M., J. B. Procter, D. M. Martin, M. Clamp, and G. J. Barton (2009). Jalview Version 2 – a multiple sequence alignment editor and analysis workbench. *Bioinformatics*, 25(9):1189–91.

Webb, E. C. (1992). *Enzyme nomenclature 1992. Recommendations of the nomenclature committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*, volume 6. San Diego: Academic Press.

Weichenberger, C. X., A. Palermo, P. P. Pramstaller, and F. S. Domingues (2017). Exploring approaches for detecting protein functional similarity within an Orthology-based framework. *Scientific Reports*, 7(1):1–15.

Weiss, M. C., F. L. Sousa, N. Mrnjavac, S. Neukirchen, M. Roettger, S. Nelson-Sathi, and W. F. Martin (2016). The physiology and habitat of the last universal common ancestor. *Nature Microbiology*, 1(9):1–8.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in Statistics*, Pp. 196–202. Springer.

Woese, C. (1998). The universal ancestor. *Proceedings of the National Academy of Sciences*, 95(12):6854–6859.

Xu, J. and Y. Zhang (2010). How significant is a protein structure similarity with TM-score= 0.5? *Bioinformatics*, 26(7):889–895.

Zaher, H. S. and P. J. Unrau (2007). Selection of an improved RNA polymerase ribozyme with superior extension and fidelity. *RNA*, 13(7):1017–1026.

Zemla, A. (2003). LGA: a method for finding 3D similarities in protein structures. *Nucleic Acids Research*, 31(13):3370–3374.

Zhang, Y. and J. Skolnick (2004). Scoring function for automated assessment of protein structure template quality. *Proteins*, 57(4):702–10.

Zhang, Y. and J. Skolnick (2005). TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–9.

Acknowledgement

First and foremost, I would like to thank my supervisor Prof. Dr. Rainer Merkl for his constant support and guidance throughout this thesis and all other research projects I was involved in during the course of my doctorate. Without his excellent scientific advice, encouraging words, and his ingenuous way of problem solving, this work would not have been possible.

I am also very grateful to Prof. Dr. Reinhard Sterner for his mentorship during this thesis and the excellent advice he provided on biochemistry related questions.

I would like to thank Prof. Dr. Wolfram Gronwald for beeing part of the examination committee and reviewing this thesis.

Also, I am grateful to Prof. Dr. Jens Meiler for the valuable scientific advice he gave me as a mentor in the context of the Regensburg International Graduate School of Life Sciences.

Financing by the *SFB960* is gratefully acknowledged.

My special thanks go to Julian Nazet and Dr. Kristina Heyn. It has been a pleasure to share an office with such companionable, clever, and fun colleagues. I'm very grateful for all the helpful scientific discussions with them and the advice they provided in all kinds of situations.

My heartfelt thanks goes to all current and former members of the Merkl and Sterner groups. It has been a privilege to be a part of such a competent, collegial, friendly and endearing group. Special thanks go to Florian Semmelmann and Thomas Kinateder for countless inspiring, encouraging, and enjoyable discussions during coffee breaks in the last years. Also, I'd like to thank my former office colleagues Dr. Patrick Löffler and Dr. Maximilian Plach for their help and advice, and the warm welcome they gave me when I joined the group.

Finally, my deep and sincere gratitude goes to my family. I'd like to thank my sisters, my brother and especially my mother for their constant encouragement and support they gave me.

Supplementary Material

Table S1: Number of hits detected by FragStatt in respect of CATH topologies.

Topo 1	Name 1	Topo 2	Name 2	Count	Fraction (%)
3.20.20	TIM Barrel	3.40.50	Rossmann fold	2986	23.8
1.20.960	Mitochondrial Import Receptor Subun...	1.25.40	Serine Threonine Protein Phosphatas...	353	2.8
2.30.30	SH3 type barrels.	2.40.50	OB fold (Dihydrolipoamide Acetyltra...	274	2.2
3.30.70	Alpha-Beta Plaits	1.10.10	Arc Repressor Mutant, subunit A	271	2.2
1.20.5	Single alpha-helices involved in co...	1.25.40	Serine Threonine Protein Phosphatas...	262	2.1
1.25.40	Serine Threonine Protein Phosphatas...	1.20.120	Four Helix Bundle (Hemerythrin (Met...	252	2.0
1.10.150	DNA polymerase; domain 1	3.40.50	Rossmann fold	251	2.0
2.40.10	Thrombin, subunit H	1.10.10	Arc Repressor Mutant, subunit A	224	1.8
4.10.520	HU Protein; Chain A	1.10.10	Arc Repressor Mutant, subunit A	215	1.7
1.25.40	Serine Threonine Protein Phosphatas...	1.10.3450	Hyaluronidase domain-like	211	1.7
1.25.40	Serine Threonine Protein Phosphatas...	1.10.10	Arc Repressor Mutant, subunit A	187	1.5
3.20.20	TIM Barrel	3.40.980	Molybdenum Cofactor Biosythetic Enz...	161	1.3
1.10.760	Cytochrome Bc1 Complex; Chain D, do...	3.40.50	Rossmann fold	147	1.2
2.70.230	Glycoprotein D; Chain: A;	2.60.40	Immunoglobulin-like	138	1.1
2.130.10	Methylamine Dehydrogenase; Chain H	2.40.10	Thrombin, subunit H	135	1.1
1.10.3910	SP0561-like	3.40.50	Rossmann fold	123	1.0
1.20.5	Single alpha-helices involved in co...	3.90.20	Hemagglutinin Ectodomain; Chain B	122	1.0
2.130.10	Methylamine Dehydrogenase; Chain H	2.40.128	Lipocalin	118	0.9
1.20.58	Methane Monooxygenase Hydroxylase; ...	1.10.287	Helix Hairpins	117	0.9
4.10.280	MYOD Basic-Helix-Loop-Helix Domain,...	1.20.5	Single alpha-helices involved in co...	106	0.8
1.10.10	Arc Repressor Mutant, subunit A	3.40.1570	Heme iron utilization protein-like ...	105	0.8
1.10.10	Arc Repressor Mutant, subunit A	3.40.50	Rossmann fold	99	0.8
1.10.10	Arc Repressor Mutant, subunit A	3.30.160	Double Stranded RNA Binding Domain	95	0.8
1.10.10	Arc Repressor Mutant, subunit A	1.20.225	Bacteriocin As-48; Chain A	94	0.8
2.60.120	Jelly Rolls	1.10.10	Arc Repressor Mutant, subunit A	92	0.7
1.10.10	Arc Repressor Mutant, subunit A	3.30.240	CRO Repressor	89	0.7
3.30.70	Alpha-Beta Plaits	3.20.20	TIM Barrel	83	0.7
1.10.10	Arc Repressor Mutant, subunit A	3.40.190	D-Maltodextrin-Binding Protein; dom...	80	0.6
3.80.30	pyruvate-formate lyase- activating ...	3.20.20	TIM Barrel	77	0.6
1.20.58	Methane Monooxygenase Hydroxylase; ...	1.10.10	Arc Repressor Mutant, subunit A	69	0.6
1.20.5	Single alpha-helices involved in co...	3.10.100	Mannose-Binding Protein A; Chain A	68	0.5
3.30.429	Macrophage Migration Inhibitory Fac...	1.10.10	Arc Repressor Mutant, subunit A	67	0.5
1.25.40	Serine Threonine Protein Phosphatas...	2.30.30	SH3 type barrels.	66	0.5
1.20.1280	Monooxygenase	1.10.10	Arc Repressor Mutant, subunit A	65	0.5
1.10.760	Cytochrome Bc1 Complex; Chain D, do...	3.10.450	Nuclear Transport Factor 2; Chain: ...	65	0.5
3.30.40	Herpes Virus-1	2.10.110	Cysteine Rich Protein	63	0.5
1.10.620	Ribonucleotide Reductase, subunit A	1.20.1260	Ferritin	62	0.5
1.20.58	Methane Monooxygenase Hydroxylase; ...	1.25.40	Serine Threonine Protein Phosphatas...	62	0.5
1.10.357	Tetracycline Repressor; domain 2	3.10.350	Membrane-bound Lytic Murein Transgl...	57	0.5
1.25.40	Serine Threonine Protein Phosphatas...	1.10.940	N-utilizing Substance Protein B Hom...	57	0.5
1.25.40	Serine Threonine Protein Phosphatas...	3.30.2320	hypothetical protein PF0899 fold	56	0.4
1.20.1260	Ferritin	1.10.357	Tetracycline Repressor; domain 2	56	0.4
1.10.10	Arc Repressor Mutant, subunit A	3.10.180	2,3-Dihydroxybiphenyl 1,2-Dioxygena...	55	0.4
1.20.190	Delta-Endotoxin; domain 1	1.25.40	Serine Threonine Protein Phosphatas...	51	0.4
1.10.437	Apoptosis Regulator Bcl-x	1.25.10	Leucine-rich Repeat Variant	50	0.4
1.10.260	434 Repressor (Amino-terminal Domai...	3.10.350	Membrane-bound Lytic Murein Transgl...	50	0.4
2.40.160	Porin	3.30.1300	Pantoate–beta-alanine Ligase; Chai...	50	0.4
1.10.260	434 Repressor (Amino-terminal Domai...	3.40.1570	Heme iron utilization protein-like ...	47	0.4
1.10.10	Arc Repressor Mutant, subunit A	1.20.120	Four Helix Bundle (Hemerythrin (Met...	46	0.4
1.20.930	Transcription Elongation Factor S-I...	1.25.10	Leucine-rich Repeat Variant	46	0.4
1.20.5	Single alpha-helices involved in co...	3.40.50	Rossmann fold	45	0.4
3.10.580	CBS-domain	3.90.1280	CBS domain Like	45	0.4
2.40.10	Thrombin, subunit H	2.120.10	Neuraminidase	44	0.4
3.10.450	Nuclear Transport Factor 2; Chain: ...	1.10.238	Recoverin; domain 1	44	0.4
1.10.10	Arc Repressor Mutant, subunit A	3.10.350	Membrane-bound Lytic Murein Transgl...	42	0.3
1.20.5	Single alpha-helices involved in co...	4.10.220	Light-harvesting Protein	41	0.3
1.10.1220	Arc Repressor Mutant	3.10.20	Ubiquitin-like (UB roll)	41	0.3

2.30.29	PH-domain like	3.30.70	Alpha-Beta Plaits	37	0.3
1.10.10	Arc Repressor Mutant, subunit A	3.20.20	TIM Barrel	35	0.3
3.90.20	Hemagglutinin Ectodomain; Chain B	3.40.30	Glutaredoxin	35	0.3
3.10.20	Ubiquitin-like (UB roll)	2.40.128	Lipocalin	34	0.3
1.10.20	Histone, subunit A	3.30.70	Alpha-Beta Plaits	32	0.3
1.10.260	434 Repressor (Amino-terminal Domai...	3.30.240	CRO Repressor	32	0.3
1.25.40	Serine Threonine Protein Phosphatas...	1.10.287	Helix Hairpins	31	0.2
1.20.5	Single alpha-helices involved in co...	4.10.860	DNA Excision Repair, Uvrb; Chain A	30	0.2
2.40.10	Thrombin, subunit H	1.10.357	Tetracycline Repressor; domain 2	30	0.2
4.10.520	HU Protein; Chain A	3.30.429	Macrophage Migration Inhibitory Fac...	30	0.2
1.20.1440	de novo design (two linked rop prot...	1.25.10	Leucine-rich Repeat Variant	30	0.2
3.90.550	Spore Coat Polysaccharide Biosynthe...	3.40.50	Rossmann fold	29	0.2
1.25.40	Serine Threonine Protein Phosphatas...	2.40.128	Lipocalin	29	0.2
1.10.260	434 Repressor (Amino-terminal Domai...	1.25.40	Serine Threonine Protein Phosphatas...	29	0.2
3.40.50	Rossmann fold	3.80.10	Leucine-rich repeat, LRR (right-han...	29	0.2
1.20.58	Methane Monooxygenase Hydroxylase; ...	1.10.3060	Helical scaffold and wing domains o...	29	0.2
4.10.520	HU Protein; Chain A	1.10.1220	Arc Repressor Mutant	28	0.2
1.10.260	434 Repressor (Amino-terminal Domai...	1.20.58	Methane Monooxygenase Hydroxylase; ...	28	0.2
1.10.260	434 Repressor (Amino-terminal Domai...	2.40.10	Thrombin, subunit H	28	0.2
3.20.20	TIM Barrel	3.30.1710	top7, de novo designed protein	28	0.2
3.90.226	2-enoyl-CoA Hydratase; Chain A, dom...	3.40.50	Rossmann fold	28	0.2
2.20.25	N-terminal domain of TfIIb	2.130.10	Methylamine Dehydrogenase; Chain H	28	0.2
4.10.520	HU Protein; Chain A	1.10.8	Helicase, Ruva Protein; domain 3	28	0.2
2.30.30	SH3 type barrels.	2.80.10	Trefoil (Acidic Fibroblast Growth F...	27	0.2
1.10.8	Helicase, Ruva Protein; domain 3	1.20.120	Four Helix Bundle (Hemerythrin (Met...	27	0.2
1.20.5	Single alpha-helices involved in co...	4.10.260	G Protein Gi Gamma 2	27	0.2
3.30.2310	YaeB-like fold	2.40.128	Lipocalin	26	0.2
3.90.70	Cathepsin B; Chain A	2.40.50	OB fold (Dihydrolipoamide Acetyltra...	25	0.2
1.10.260	434 Repressor (Amino-terminal Domai...	3.10.180	2,3-Dihydroxybiphenyl 1,2-Dioxygena...	25	0.2
3.40.525	Phosphatidylinositol Transfer Prote...	1.10.8	Helicase, Ruva Protein; domain 3	25	0.2
1.20.1440	de novo design (two linked rop prot...	1.10.10	Arc Repressor Mutant, subunit A	24	0.2
4.10.520	HU Protein; Chain A	3.40.50	Rossmann fold	24	0.2
3.30.429	Macrophage Migration Inhibitory Fac...	3.10.20	Ubiquitin-like (UB roll)	24	0.2
2.60.120	Jelly Rolls	2.40.128	Lipocalin	23	0.2
3.20.20	TIM Barrel	3.40.1380	Pyruvate Kinase; Chain: A, domain 1	23	0.2
2.120.10	Neuraminidase	2.40.128	Lipocalin	23	0.2
1.10.357	Tetracycline Repressor; domain 2	3.30.240	CRO Repressor	22	0.2
1.10.287	Helix Hairpins	1.20.120	Four Helix Bundle (Hemerythrin (Met...	22	0.2
1.10.1220	Arc Repressor Mutant	3.30.1660	Dodecin subunit-like	21	0.2
1.20.5	Single alpha-helices involved in co...	3.30.1310	Ybab; Chain: A;	21	0.2
1.10.150	DNA polymerase; domain 1	1.25.40	Serine Threonine Protein Phosphatas...	21	0.2
1.10.1220	Arc Repressor Mutant	3.30.2310	YaeB-like fold	21	0.2
1.10.3210	Hypothetical protein af1432	1.25.40	Serine Threonine Protein Phosphatas...	21	0.2
1.20.58	Methane Monooxygenase Hydroxylase; ...	1.10.3450	Hyaluronidase domain-like	21	0.2
3.30.70	Alpha-Beta Plaits	3.10.20	Ubiquitin-like (UB roll)	20	0.2
1.10.357	Tetracycline Repressor; domain 2	3.40.1570	Heme iron utilization protein-like ...	19	0.2
1.10.720	Transcription Termination Factor Rh...	3.40.30	Glutaredoxin	19	0.2
3.30.1300	Pantoate–beta-alanine Ligase; Chai...	2.40.128	Lipocalin	18	0.1
3.90.1820	LDH C-terminal domain-like	3.40.50	Rossmann fold	18	0.1
3.90.20	Hemagglutinin Ectodomain; Chain B	1.10.287	Helix Hairpins	17	0.1
1.20.1270	Substrate Binding Domain Of Dnak; C...	1.10.287	Helix Hairpins	17	0.1
3.20.20	TIM Barrel	3.90.226	2-enoyl-CoA Hydratase; Chain A, dom...	17	0.1
3.30.1490	Dna Ligase; domain 1	1.10.238	Recoverin; domain 1	16	0.1
4.10.520	HU Protein; Chain A	1.10.238	Recoverin; domain 1	16	0.1
1.10.760	Cytochrome Bc1 Complex; Chain D, do...	1.20.120	Four Helix Bundle (Hemerythrin (Met...	16	0.1
2.130.10	Methylamine Dehydrogenase; Chain H	1.20.5	Single alpha-helices involved in co...	15	0.1
1.10.1660	Multidrug-efflux Transporter Regula...	3.10.350	Membrane-bound Lytic Murein Transgl...	15	0.1
2.40.30	Elongation Factor Tu (Ef-tu); domai...	2.30.30	SH3 type barrels.	15	0.1
2.70.70	Glucose Permease (Domain IIA)	2.40.50	OB fold (Dihydrolipoamide Acetyltra...	15	0.1
2.30.29	PH-domain like	1.20.5	Single alpha-helices involved in co...	15	0.1
2.40.128	Lipocalin	3.40.190	D-Maltodextrin-Binding Protein; dom...	15	0.1
1.20.5	Single alpha-helices involved in co...	3.30.429	Macrophage Migration Inhibitory Fac...	14	0.1
4.10.520	HU Protein; Chain A	3.30.70	Alpha-Beta Plaits	14	0.1
4.10.280	MYOD Basic-Helix-Loop-Helix Domain,...	1.10.287	Helix Hairpins	14	0.1
1.10.1220	Arc Repressor Mutant	2.40.128	Lipocalin	14	0.1
1.10.1220	Arc Repressor Mutant	3.90.1520	H-NOX domain	14	0.1
1.20.5	Single alpha-helices involved in co...	3.30.910	Protein Binding, DinI Protein; Chai...	14	0.1
3.30.70	Alpha-Beta Plaits	2.40.128	Lipocalin	14	0.1
1.25.40	Serine Threonine Protein Phosphatas...	1.10.375	Human Immunodeficiency Virus Type 1...	14	0.1
4.10.520	HU Protein; Chain A	1.10.760	Cytochrome Bc1 Complex; Chain D, do...	14	0.1
3.30.1330	60s Ribosomal Protein L30; Chain: A...	2.40.128	Lipocalin	13	0.1
1.10.1660	Multidrug-efflux Transporter Regula...	3.30.240	CRO Repressor	13	0.1
3.40.980	Molybdenum Cofactor Biosythetic Enz...	3.90.226	2-enoyl-CoA Hydratase; Chain A, dom...	13	0.1
1.10.260	434 Repressor (Amino-terminal Domai...	2.60.120	Jelly Rolls	13	0.1
3.10.450	Nuclear Transport Factor 2; Chain: ...	3.30.1660	Dodecin subunit-like	13	0.1
1.20.58	Methane Monooxygenase Hydroxylase; ...	3.90.20	Hemagglutinin Ectodomain; Chain B	13	0.1

4.10.520	HU Protein; Chain A	1.10.287	Helix Hairpins	12	0.1
3.10.580	CBS-domain	3.40.50	Rossmann fold	12	0.1
2.60.200	Tumour Suppressor Smad4	3.10.20	Ubiquitin-like (UB roll)	12	0.1
3.20.20	TIM Barrel	2.40.128	Lipocalin	12	0.1
3.90.20	Hemagglutinin Ectodomain; Chain B	1.20.120	Four Helix Bundle (Hemerythrin (Met...	12	0.1
1.10.20	Histone, subunit A	3.30.2310	YaeB-like fold	12	0.1
1.10.1660	Multidrug-efflux Transporter Regula...	3.10.180	2,3-Dihydroxybiphenyl 1,2-Dioxygena...	12	0.1
1.10.10	Arc Repressor Mutant, subunit A	2.40.128	Lipocalin	12	0.1
1.10.3910	SP0561-like	3.40.630	Aminopeptidase	12	0.1
1.20.1260	Ferritin	1.10.287	Helix Hairpins	12	0.1
2.40.128	Lipocalin	1.10.238	Recoverin; domain 1	11	0.1
4.10.520	HU Protein; Chain A	2.40.128	Lipocalin	11	0.1
2.20.25	N-terminal domain of TfIIb	2.120.10	Neuraminidase	11	0.1
1.10.238	Recoverin; domain 1	3.40.50	Rossmann fold	11	0.1
1.10.620	Ribonucleotide Reductase, subunit A	1.20.910	Heme Oxygenase; Chain A	11	0.1
2.40.128	Lipocalin	3.40.30	Glutaredoxin	11	0.1
3.10.580	CBS-domain	3.20.20	TIM Barrel	11	0.1
2.40.10	Thrombin, subunit H	2.140.10	Methanol Dehydrogenase; Chain A	11	0.1
1.10.760	Cytochrome Bc1 Complex; Chain D, do...	3.30.1740	first zn-finger domain of poly(adp-...	11	0.1
3.20.20	TIM Barrel	3.40.30	Glutaredoxin	11	0.1
1.10.10	Arc Repressor Mutant, subunit A	3.10.290	Structural Genomics Hypothetical 15...	11	0.1
1.10.357	Tetracycline Repressor; domain 2	3.10.180	2,3-Dihydroxybiphenyl 1,2-Dioxygena...	10	0.1
3.30.1660	Dodecin subunit-like	2.40.128	Lipocalin	10	0.1
1.10.287	Helix Hairpins	4.10.860	DNA Excision Repair, Uvrb; Chain A	10	0.1
3.90.20	Hemagglutinin Ectodomain; Chain B	3.40.50	Rossmann fold	10	0.1
1.10.760	Cytochrome Bc1 Complex; Chain D, do...	3.40.1350	Trna Endonuclease; Chain: A, domain...	10	0.1
3.40.1530	hypothetical protein tt1805	1.10.10	Arc Repressor Mutant, subunit A	10	0.1

```
########################################
#
# Length: 203
# Identity:      77/203 (37.9%)
# Similarity:   118/203 (58.1%)
# Gaps:          12/203 ( 5.9%)
# Score: 352.0
#
########################################

2YVA_A         1 -MQERIKACFTESI-QTQIAAAEALPD-----AISRAAMTLVQSLLNGNK     43
                 |:.|.....|.|| :.|...|..|.|      .:.:.|...:.|:..|.|
2XBL_B         1 SMENRELTYITNSIAEAQRVMAAMLADERLLATVRKVADACIASIAQGGK     50

2YVA_A        44 ILCCGNGTSAANAQHFAASMINRFETERPSLPAIALNTDNVVLTAIANDR     93
                 :|..|||.|||:|||.|...::||..:||.|||:||.||..:||||.||.
2XBL_B        51 VLLAGNGGSAADAQHIAGEFVSRFAFDRPGLPAVALTTDTSILTAIGNDY    100

2YVA_A        94 LHDEVYAKQVRALGHAGDVLLAISTRGNSRDIVKAVEAAVTRDMTIVALT    143
                 .:::::::||:|||:.||||:..||.|.|.:|:.|...|..:.||.|..|
2XBL_B       101 GYEKLFSRQVQALGNEGDVLIGYSTSGKSPNILAAFREAKAKGMTCVGFT    150

2YVA_A       144 GYDGGELAGLLGPQDVEIRIPSHRSARIQEMHMLTVNCLCDLIDNTLFPH    193
                 |..|||:..|.   |:.:.:||..:.:|||.|::..:.:|.|:::::|..
2XBL_B       151 GNRGGEMRELC---DLLLEVPSADTPKIQEGHLVLGHIVCGLVEHSIFGK    197

2YVA_A       194 QDD    196
                 |
2XBL_B       198 Q--    198
```

Alignment S1: Alignment of PDB-ID 2XBL_B and PDB-ID 2YVA_A.
Alignment generated using EMBOSS Needle (Madeira et al., 2019)

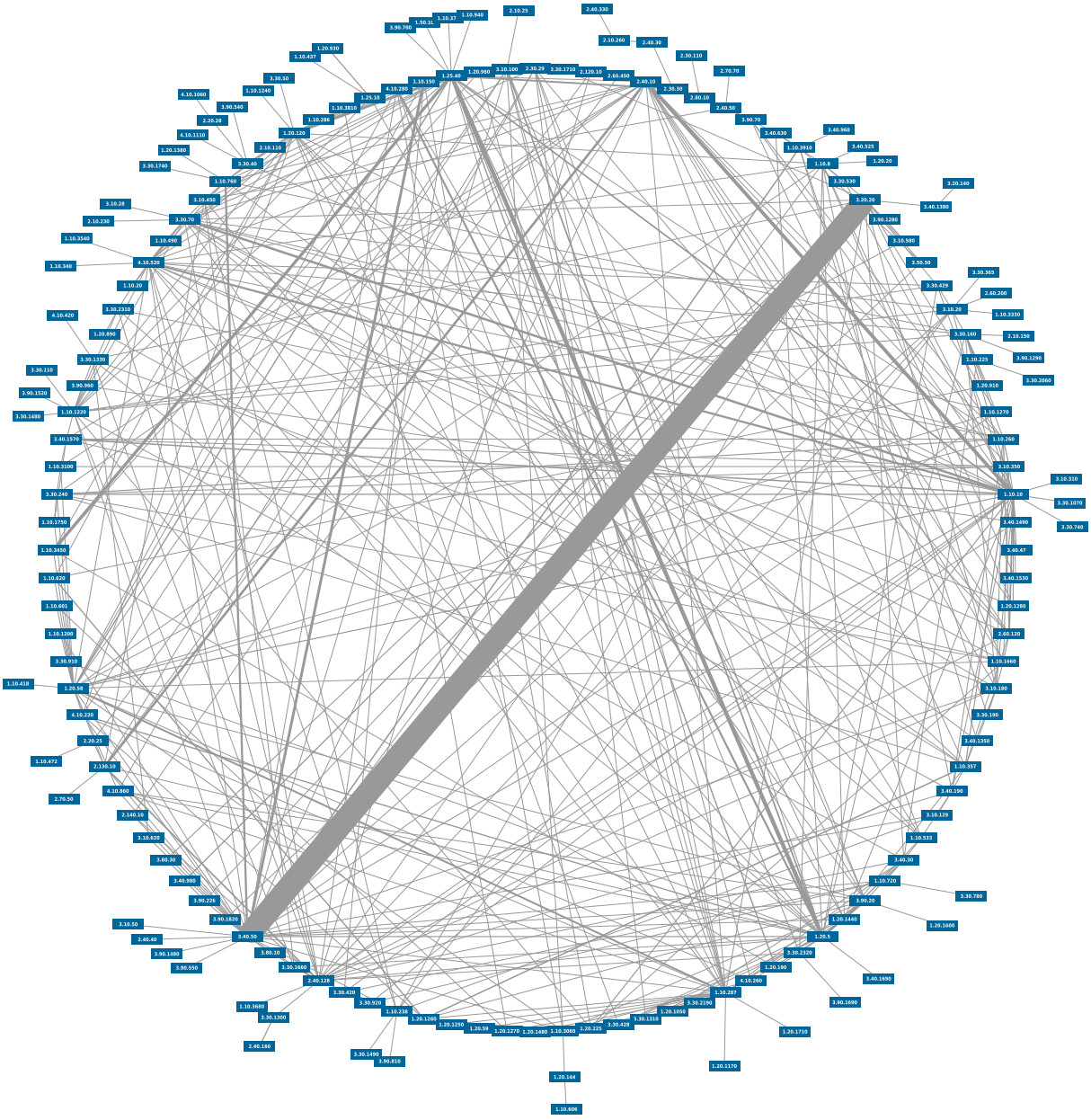


Figure S1: A survey of motifs shared between CATH topologies.
The nodes represent CATH topologies, which are connected by an edge, if FragStatt detected shared common motifs between members of the topologies. The width of the edges represents the number of detected motifs. Exact values can be found in Table S1.