Dissertation zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Ludwig-Maximilians-Universität München

Decomposing Protein Sequence Space into Domains

Markus Meier

aus

Kösching

2016

Erklärung

Diese Dissertation wurde im Sinne von § 7 der Promotionsordnung vom 28. November 2011 von Herrn Dr. Johannes Söding betreut.

Eidesstattliche Versicherung

Diese Dissertation wurde eigenständig und ohne unerlaubte Hilfe erarbeitet.

München, 9. September 2016

Markus Meier

Dissertation eingereicht am	27.09.2016
Erstgutachter:	Dr. Johannes Söding
Zweitgutachter:	Prof. Dr. Julien Gagneur
Mündliche Prüfung am	16.12.2016

Contents

List of Figures

List of Tables

Summary

Domains are the structural subunits of proteins. They are considered to be the basic units of folding, evolution and function [61]. Understanding the domain structure helps to improve the functional annotation of proteins [43, 37], tertiary protein structure prediction [20], protein engineering [29] and protein mutagenesis [45]. Domains are the minimal functional units of a protein. To elucidate the cellular functions of a protein we need to understand the molecular functions of its domains.

Protein sequences are annotated by the matches to domains in domain family databases such as Pfam [26], SCOP [44] or CATH [55]. However, existing domain databases cover about half of the known sequence space and encompass only a small fraction of all protein domain families [49] [60]. Here, we developed an algorithm based on a Bayesian statistical model, called Pdom, with which we can consistently decompose the entire protein sequence space into its evolutionary units. Pdom predicts domains on the basis of an all-against-all search of protein Hidden Markov models with HHblits [52]. An alignment of HHblits indicate the shared homologous region between the query and a template. This shared homologous region encompasses usually one or multiple domains shared by query and template. We can infer the domain borders of the query from its alignments to different templates. For this purpose, we can use the probabilities for the beginning and ending of an homologous region calculated by HHblits.

HHblits is an iterative protein homology detection tool that is more sensitive, generates more accurate alignments and is faster than its best competitors, PSI-BLAST [10] and HMMER3 [39]. HHblits searches with a query Hidden Markov Model (HMM) against a database of HMMs of multiple sequence alignments. For the clustering of the Uniprot sequence database to the uniprot20 HMM database, we expect the sequences within a cluster to cover each other by at least 90%. Therefore, the HMMs of the uniprot20 are very conservative with few mostly very similar sequences contained in each cluster. For the domain decomposition with Pdom we wanted to increase the sensitivity of HHblits to detect more remote homologs. For this purpose, we enriched our database clusters by jumpstarting HHblits with each cluster alignment in the uniprot20 and adding significant matches to the cluster alignments. The resulting uniprot_boost1 database (*boost1* for one iteration with HHblits) has 14x more sequences per cluster than uniprot20. The effective number of sequences is raised from 1.18 to 3.78. We developed efficient sparse compression and alignment handling algorithms that keep memory size nearly the same. HHblits finds with the uniprot_boost1 20% more homologs in three iterations compared

to the uniprot20. The alignments of HHblits with the uniprot_boost1 have a 11.32% and 17.39% increased per-residue precision and sensitivity, respectively. The structural models of global alignments with fixed query template pairs against the uniprot_boost1 have on average a 4.9% higher TMscore with the native structure. Especially, the structural models of queries with a TMscore ≤ 0.3 in the uniprot20 can be improved with the uniprot_boost1. In a simple homology modeling pipeline with free template selection we showed that the TMscore of models built with the uniprot_boost1 is on average 4.9% higher.

In summary, HHblits finds with the diversity-enriched uniprot_boost1 database more homologs and generates more accurate alignments that lead to better structure models than with the uniprot20. It further widens the gap to HHblits' competitors HMMER and PSI-BLAST. We showed that the performance of HHblits with the uniprot_boost1 can be transferred to the downstream application of homology modeling. The uniprot_boost1 is capable to become a default database for HHblits and may impact sequence-based predictions of evolutionarily conserved properties, such as secondary or tertiary structure, disorder, catalytic sites, post-translational modifications, short linear motifs, or interaction interface.

Protein domains occur in different proteins with different protein architectures. Pairwise alignments of a query protein against multiple homologous template proteins reveal with the different alignment start and end positions the boundaries of protein domains in the query protein. Some of those alignments encompass a single domain, others might encompass multiple domains. On the basis of an all-against-all search of HHblits within the uniprot_boost1, we decomposed the protein sequence space into its domains with Pdom. We compared the predictions of Pdom, ADDA [34] and Pfam to SCOP annotations mapped onto full length protein sequences. ADDA applies the same fundamental idea as Pdom. ADDA predicts domains on the basis of an all-against-all search of protein sequences with BLAST. Pfam uses manually curated seed alignments that incorporate available data from literature.

On average ADDA covers 32%, Pfam covers 80% and Pdom covers 75% of the reference domain annotations. ADDA predicts domain start and end sites within 20 residues in 15% of the reference domain start and end sites. Pfam annotates domain start sites within 20 residues in 67% of the reference domain start sites and annotates domain end sites within 20 residues in 58% of the reference domain end sites. Pdom predicts domain start sites within 20 residues in 50% of the reference domain start sites and predicts domain end sites within 20 residues in 45% of the reference domain end sites.

The seed alignments of Pfam have a very high quality due to the manual curation effort. But those seed alignments are limited to domains analyzed in literature. With our fully automatic approach in Pdom we are able to find new domains in the protein sequence space. The clustered database of Pdom's domain predictions, UniDom, has the potential to become a fundamental tool for homology-based protein sequence annotation efforts.

Introduction

Already in 1973 Wetlaufer et al. proposed the existence of stable and compact units of protein structure that could fold on their own [63]. Those structural subunits, in modern parlance called domains, are considered to be the basic units of folding, evolution and function [61]. The majority of globular domains are between 40 and 400 residues in size, with a mean length of about 170 residues [67]. In approximately $5 - 20\%$ of multi-domain proteins one domain can be found to be inserted within or interlaced with another domain [67] (see figure 1).

Domains are the minimal functional units of a protein. Understanding the domain structure helps to improve its functional annotation [43, 37], its tertiary structure prediction [20], protein engineering [29] and mutagenesis [45]. To elucidate the cellular functions of a protein we need to understand the molecular functions of its domains.

Experimental methods struggle with the analysis of large, multi-domain proteins [16]. Current methods analyze each domain separately [38, 18]. Experimental approaches for identifying structural domains are mostly based on limited proteolysis, but this requires significant quantities of proteins, and a large amount of time and effort.

The UniProt Reference Clusters (UniRef) provide clustered sets of protein sequences from the UniProt and selected UniParc records in order to obtain a complete coverage of the sequence space at several clustering depths while hiding redundant sequences.

The UniRef100 (UniProt Reference Clusters) protein sequence database consists of 78 201 882 sequences (April, 2016) clustered for 100% pairwise sequence identity. Its size is rapidly increasing due to technological advances in next-generation sequencing [54, 40] (see figure 2). A vast majority of those new protein sequences do not have an annotated function. Protein sequences are usually annotated by the matches to domains in domain family databases such as Pfam [26], SCOP [44] or CATH [55].

Domain definition on basis on structure, e.g. CATH and SCOP, is limited by the scarce structure information. The Protein Data Bank (PDB) was established in 1971 with 7 structures. The database has grown to more than 118 000 structures (May 6th, 2016) [68]. However, when filtered for 70% pairwise sequence identity, removing only sequences that are covered by at least 90%, less than 38 000 protein sequences are left (statistic from our PDB70 HHsuite database pipeline).

In 1995 Murzin et al. published a structural classification of proteins (SCOP) [44]. Alexei Murzin and his colleagues inspect and compare the structures in the Protein Data Bank (PDB) [14] to produce an accurate, hierarchical classification of protein domains

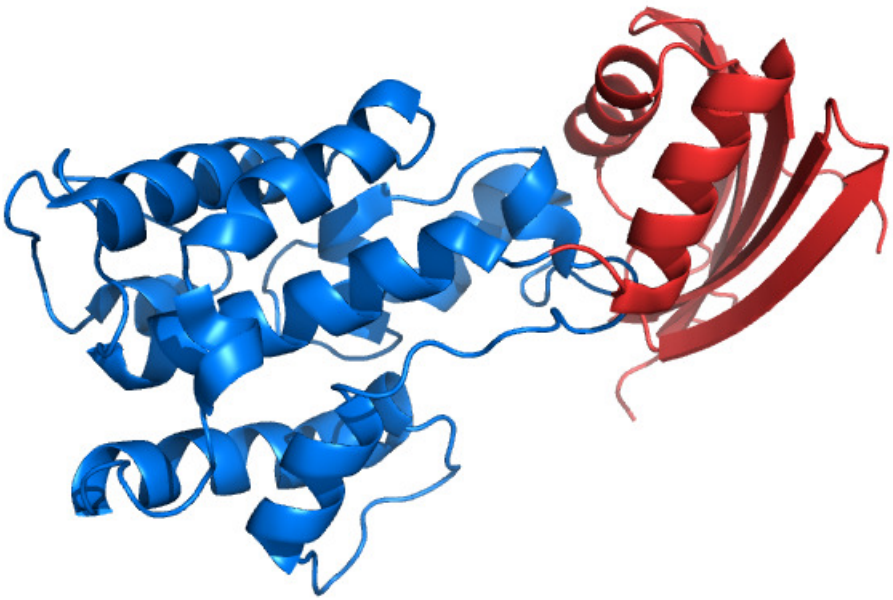

Figure 1: Protein api92 from Yersinia pseudotuberculosis (PDB identifier: 2IJR) with two CATH domains colored in red and blue; the blue domain is nested in the red domain

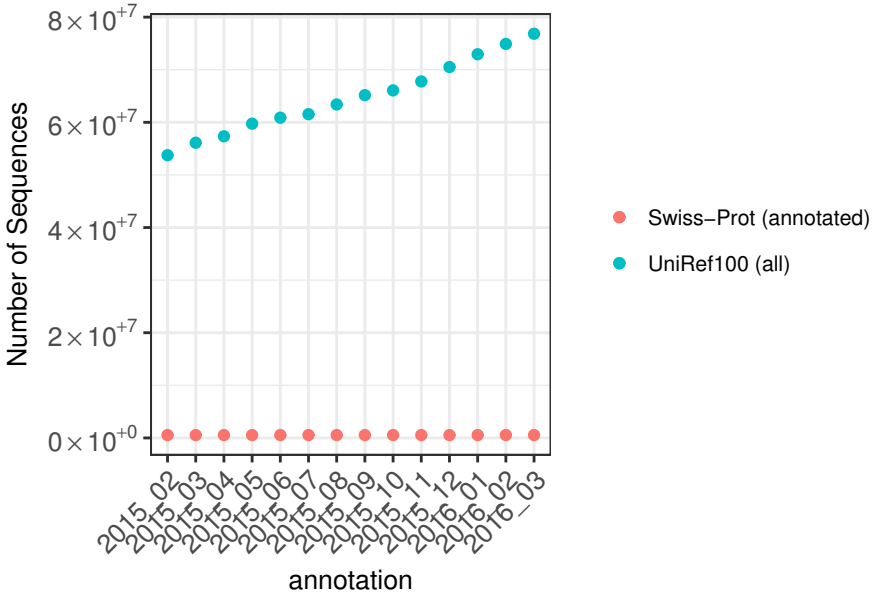

Figure 2: Number of sequences in the UniRef100 protein sequence database over the last twelve months

based on evolution and structure. The SCOP database classifies into four levels:

- **Class:** Domains with the same composition of secondary structure elements, e.g. all alpha-helices.
- **Fold:** Domains with the same major secondary structures in the same arrangement with the same topological connections.
- **Superfamily:** Domains whose structures and functions suggest a common evolutionary origin.
- **Family:** Domains with residue identities of $\geq 30\%$ or residue identities $\leq 30\%$, but similar functions and structures.

A similar approach is the CATH database published in 1997 by Orengo et al. [55]. CATH chops the PDB into structural domains and sorts in a hierarchical protein domain classification. The chopping is done by automatic methods supervised by humans. The CATH hierarchy knows four levels:

- **Class:** Domains with the same composition of secondary structure elements, e.g. mainly alpha-helices.
- **Architecture:** Domains with topologies that share roughly the same spatial arrangement of secondary structures.
- **Topology:** Domains with the same fold, but not sufficient evidence for a decent from a common ancestor.
- **Homologous Superfamily:** Domains with evidence for a common ancestor.

SCOP and CATH share the disadvantage that their domains need a structural reference in the Protein Data Bank. The latest release of SCOP encompassed 1 962 superfamilies (SCOP 1.75, June 2009). The latest release of CATH encompassed 2 738 Homologous Superfamilies (CATH 4.0, March 2013). Both databases provide a classification that indicates a potential common evolutionary origin. If two domains share the same family in these classifications they are similar and likely to share the same function.

Another domain database is Pfam [26]. Pfam is built on manually curated seed multiple-sequence alignments. Each of those multiple-sequence alignments represents a domain family. Families with similar alignments and families with similar linked structures in the PDB are grouped to a so called clan. Pfam matches 76.1% of sequences and 54.8% of residues in the UniProt protein sequence database.

The un-matched part of the Uniprot is referred to as 'dark matter' of protein space [51] [49].

In summary, structure data is scarce and is slowly growing compared to sequence data. Therefore, the bioinformatical challenge of dissecting proteins sequences into structural domains gets more important. De-novo domain predictors without reference databases such

as PDB, Pfam, CASP or SCOP offer the opportunity to discover new domains in the dark matter of protein space.

There are two groups of de-novo domain predictors from amino-acid sequence: statistical or machine-learning and 3D model-based methods.

3D model-based methods build an ab-inito 3D model of the protein and parse the protein domains from the structure (SnapDRAGON [28], RosettaDom [41], OPUS-DOM [64]). Ab-initio models are built with time-consuming molecular dynamics simulations given just the protein sequences. These methods are not suitable for large scale analysis of un-characterized sequences in the protein space.

The statistical and machine-learning methods can be subdivided into two groups:

- comparative sequence analysis: DoBo [25], ADDA [34], EVEREST [50]
- direct boundary prediction: Scooby-Domain [46], KemaDom [19], Armadillo [23]

Most direct boundary prediction methods aim to identify domain boundary regions such as domain linkers, exploiting their sequence and structural biases [24]. Most of those predictors apply machine learning techniques like Neural Networks, Support Vector Machines or Random Forests to decide between domain and linker regions. Linker regions are usually disordered parts of the protein between domains.

Domains with the same evolutionary origin might be found in different domain architectures (see figure 3). Therefore, comparative sequence analysis domain predictors can derive domain definitions of sequence segments generated from starts and ends of pairwise alignments of an all-against-all search within a protein database. Afterwards the sequence segments are usually clustered to domain families.

Francois Jacob stated in 1977 that nature is a tinkerer not an inventor to explain re-occurring domains in different domain architectures. His idea was that nature copied working domains, slightly modified them and combined them to develop new functions. Recent research indicates that genes may undergo recombination to produce complex domain architectures via gene fusion [42], gene fission [42, 47], domain duplication and domain swapping [47, 15, 22, 35, 12] during evolution.

As of 2016, there are three main competitors for protein homology search and alignment tools: PSI-BLAST [10], HMMer [39] and HHblits [52]. Remmert et al. showed that HHblits is more sensitive in detecting remote homologs, is faster and generates more accurate alignments than PSI-BLAST and HMMer [52]. HHblits searches in several iterations with a query Hidden Markov profile (HMM) against a database of HMM profiles from multiple sequence alignments. After each iteration the query HMM is updated with the found homologs. The commonly used database with HHblits is the uniprot20. It contains the sequences from the UniProt sequence database clustered for up to 20% pairwise sequence identity. Only sequences that cover each other by at least 90% are included in one cluster. This clustering is done by MMseqs [32]. The clusters of the uniprot20 contain on average five sequences with an effective number of 1.18 sequences. The effective number of sequences, short NEFF, considers the redundancy within a cluster. A cluster with several hundred but identical sequences has one effective sequence.

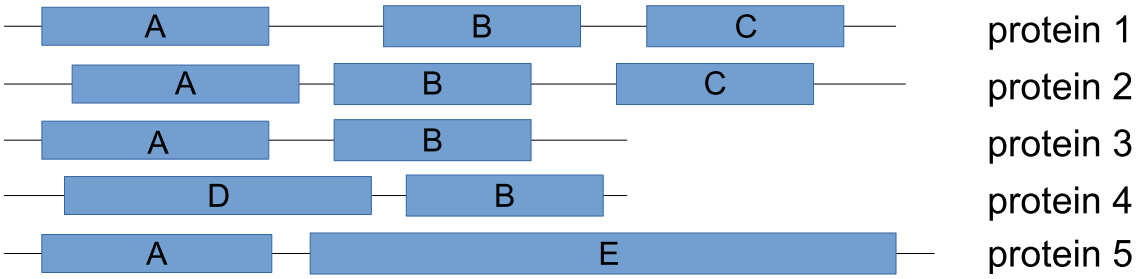

Figure 3: Illustration of domain architectures of different proteins evolutionarily related by single domains. Protein 1 and 2 are close homologs and will have almost full-length sequence alignments. With the additional more distantly related homologs of protein 3, 4 and 5 it is possible to chop protein 1 and 2 into their domains.

Chapter 1 introduces the diversity enriched uniprot_boost1 HMM database for HHblits. In order increase the detection sensitivity of more remote homologs and the alignment quality, we developed the uniprot_boost1. We enriched our database clusters by jump-starting HHblits with each cluster alignment in the uniprot20 and adding significant local matches from the uniprot20 to the cluster alignments.

Chapter 2 introduces our newly developed domain predictor Pdom that uses the more sensitive uniprot_boost1 HMM database. In a general Bayesian approach we take advantage of an all-against-all comparison of HMMs with HHblits to chop the complete sequence space into domains. Instead of fixed alignment start and end points like in PSI-BLAST we can use the alignment start and end probability at each position calculated for each alignment in HHblits to get more accurate domain predictions.

Chapter 3 summarizes additional work on the HHsuite and its databases.

Chapter 1

A Diversity-Enriched HMM Database for HHblits

1.1 Abstract

HHblits is an iterative protein homology detection tool that is more sensitive, generates more accurate alignments and is faster than PSI-BLAST and HMMer3. HHblits searches with a query Hidden Markov Model (HMM) against a database of HMMs of multiple sequence alignments. The HMMs of the uniprot20 database are based on a very conservative clustering of the UniProt sequence database with few mostly very similar sequences contained in each cluster. Here, we enrich our database clusters by jumpstarting HHblits with each cluster alignment in the uniprot20 and adding significant matches to the cluster alignments. The resulting uniprot_boost1 database (*boost1* for one iteration with HHblits) has about 14 times more sequences per cluster than uniprot20. The effective number of sequences was raised from 1.18 to 3.78. We developed efficient sparse compression and alignment handling algorithms that keep memory size nearly the same. HHblits finds with the uniprot_boost1 about 20% more homologs in three iterations. The alignments of HHblits with the uniprot_boost1 have a 11.32% and 17.39% increased per-residue precision and sensitivity, respectively. The structural models of global alignments with fixed query template pairs against the uniprot_boost1 have on average a 4.9% higher TMscore with the native structure. Especially, the structural models of queries with a TMscore ≤ 0.3 in the uniprot20 can be improved with the uniprot_boost1. In a simple homology modeling pipeline with free template selection we showed that the TMscore of models built with the uniprot_boost1 is on average 4.9% higher.

1.2 Method

The uniprot20 HHsuite database is the UniProt sequence database [11] clustered by MMseqs [33]. Sequences that cover each other with at least 80% and a pairwise sequence identity of at least 20% are clustered together. This leads to very conservative clusters with few sequences, but well conserved and almost full-length multiple sequence alignments (MSA). The clusters are functionally pure. Each sequence of the UniProt appeared in only one cluster of the uniprot20. Here, we searched with each cluster of the uniprot20 through the uniprot20 for homologous sequences with HHblits in one iteration. The found homologous sequences were added to the query alignment in a diversity-enriched database (see figure 1.1). The increase in diversity of these clusters improved the detection sensitivity of remote homologs.

1.3 Memory and Efficiency

The uniprot20 has 4 866 021 multiple sequence alignments (MSA) with about five sequences per cluster (release October 2012). Each sequence in the UniProt appears exactly once in the uniprot20. The raw diversity-enriched database has the same number of MSAs but about 100 sequences per multiple sequence alignment (without the filter described in section 1.3.1).

1.3.1 Sequence Filter

We observed that huge multiple sequence alignments in the diversity-enriched database cause a performance bottleneck in the calculation of the corresponding HMMs during the runtime of HHblits. Therefore, we removed redundant information in the sequence space by clustering the sequences in the whole database by 70% sequence identity with MMseqs and keeping just the representative sequences of the clusters. The filtered diversity-enriched database contains on average 70 sequences per cluster.

1.3.2 Compressed Redundant HHsuite Database

Each sequence appears on average 14 times in the filtered diversity-enriched database, therefore the raw MSAs in the a3m format are significantly bigger (224 GB) than the corresponding uniprot20 MSAs (15 GB). The enormous size makes it difficult to share and to use on average workstations. In order to reduce the size, we developed a compressed binary format for HHsuite databases with redundant sequences (see figure 1.2). The idea is to separate sequence and alignment information. Each sequence in an MSA can be represented as blocks of matches, insertions or deletions. The MSA can be reconstructed with this alignment information and the complete protein sequences.

Figure 1.1: Generation of the uniprot_boost1: To each multiple sequence alignment in the uniprot20 we add homologous sequences found in one search iteration through the uniprot20 with HHblits. The set of diversity enriched multiple sequence alignments is the uniprot_boost1. Those multiple sequence alignments contain on average 14 times more sequences with on average 3.78 effective sequences (NEFF).

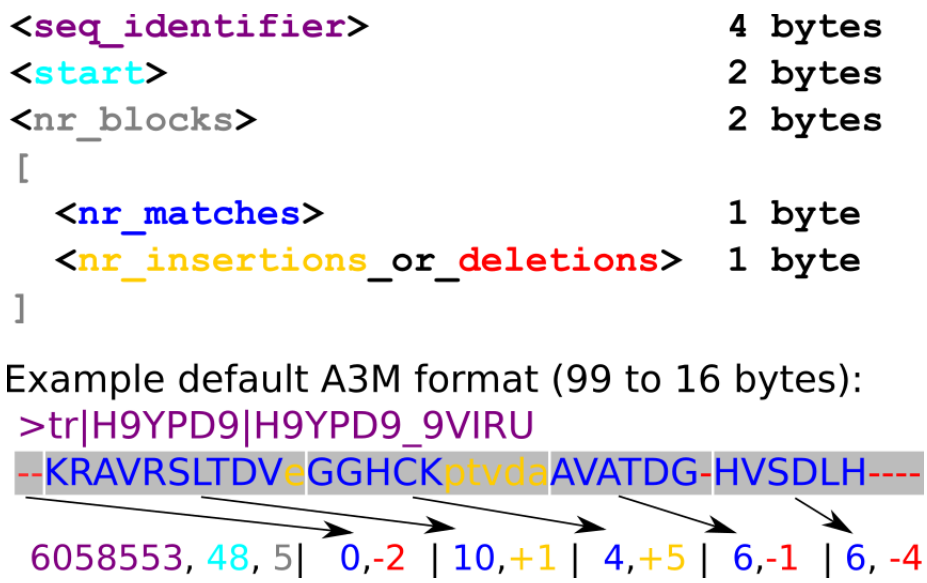

Figure 1.2: We used the illustrated compression scheme for each sequence in a multiple sequence alignment in the uniprot_boost1. Each sequence in the MSA has an identifier that points to an entry in a database with the complete sequence. Therefore, we need to store the same sequence occurring in multiple MSAs just once. For each sequence we need to save the position of the first aligned residue. Furthermore, the number and size of consecutive rows of matches, insertions and deletions of each sequence in the alignment are needed to reconstruct the alignment.

part	uniprot20	uniprot_boost1
a3m	15 GB	14 GB
hhm	3.4 GB	-
cs219	1.4 GB	1.4 GB
header	-	1 GB
sequence	-	2.5 GB
total	**19.8 GB**	**18.9 GB**

Table 1.1: Size of the different parts of the uniprot20 and uniprot_boost1 databases.

1.4 Results

1.4.1 Size

The default uniprot20 consists of three parts:

- The multiple sequence alignments in the a3m format [5]
- The HMM profiles translated into a discretized set of 219 column states (cs219) used for the prefilter
- The precalculated HMMs for large multiple sequence alignments in the hhm format

The uniprot_boost1 database consists of four parts:

- The alignment information of the multiple sequence alignments in the binary ca3m format
- The HMM profiles translated into a discretized set of 219 column states (cs219) used for the prefilter
- The protein sequences
- The headers to all proteins with the protein identifier and annotations

The uniprot_boost1 and uniprot20 have about the same total size of roughly 20 GB (see table 1.1). The raw MSAs of the uniprot_boost1 in the a3m format encompass 224 GB, since each protein occurs on average in 14 different MSAs. The database is compressed by separating the alignment information and the protein sequences (see section 1.3.2), so each protein sequence is stored just once. Additionally, the binary encoding of the alignment information is very space-efficient.

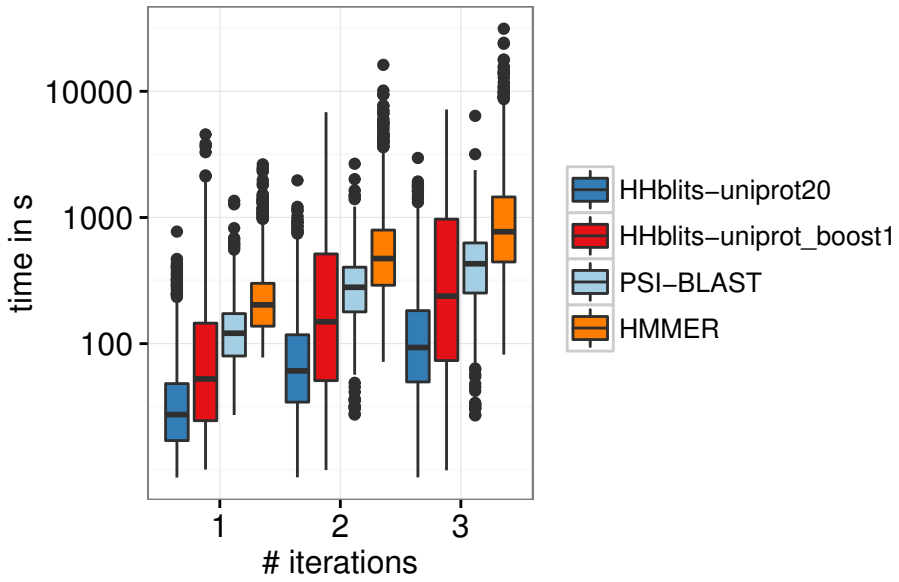

Figure 1.3: Runtime of HMMer and PSI-BLAST on the UniProt sequence database (release October 2012) and HHblits with the corresponding uniprot20 and uniprot_boost1 HMM databases.

1.4.2 Runtime

We randomly selected 1 644 sequences from the UniProt and measured the runtime of HMMer and PSI-BLAST through the UniProt sequence database, and of HHblits through the corresponding uniprot20 and uniprot_boost1 databases (release October 2012). The results are shown in figure 1.3 and in table 1.2). Three iterations through the uniprot_boost1 with HHblits are on average 3.7 times slower than three iterations through the uniprot20 with HHblits, 0.37 times slower than three iterations through the UniProt with PSI-BLAST, but 2.1 times faster than three iterations through the UniProt with HMMer.

1.4.3 Diversity in the Clusters

We calculated the number of effective sequences (NEFF) to determine the amount of homologous information in a multiple sequence alignment. The NEFF statistic is calculated as the exponential of the Shannon entropy averaged over all columns of the multiple sequence alignment [48]. The NEFF is a real value ranging from 1 to 20.

The average NEFF for the uniprot20 is 1.18 and 3.78 for the uniprot_boost1.

A NEFF of 1.0 means that there is just one sequence in the cluster or a set of identical sequences. In the following we call these clusters singletons.

In the uniprot20 3 549 885 of 4 866 021 clusters are singletons. In the uniprot_boost1 936 770 clusters are still singletons (see figure 1.4A).

We jump-started HHblits with the singletons in the uniprot_boost1 and searched in

	#iterations	average time in s
hhblits-uniprot_boost1	1	185
hhblits-uniprot_boost1	2	544
hhblits-uniprot_boost1	3	777
hhblits-uniprot20	1	46
hhblits-uniprot20	2	106
hhblits-uniprot20	3	163
HMMer	1	273
HMMer	2	765
HMMer	3	1445
PSI-BLAST	1	141
PSI-BLAST	2	319
PSI-BLAST	3	491

Table 1.2: Mean runtimes for PSI-BLAST and HMMer against the UniProt sequence database, and HHblits against the corresponding uniprot20 and uniprot_boost1 HMM databases.

one iteration through the uniprot_boost1 for homologs, so we get an impression of a uniprot_boost2. In the uniprot_boost2 832 483 clusters are still singletons (see figure 1.4B).

Short sequences (< 100 residues) are a 0.65 times less frequent among the singletons of the uniprot20 compared to the singletons of the uniprot_boost1 and 0.95 times less frequent among the uniprot_boost1 singletons compared to the uniprot_boost2 singletons (see figure 1.4C).

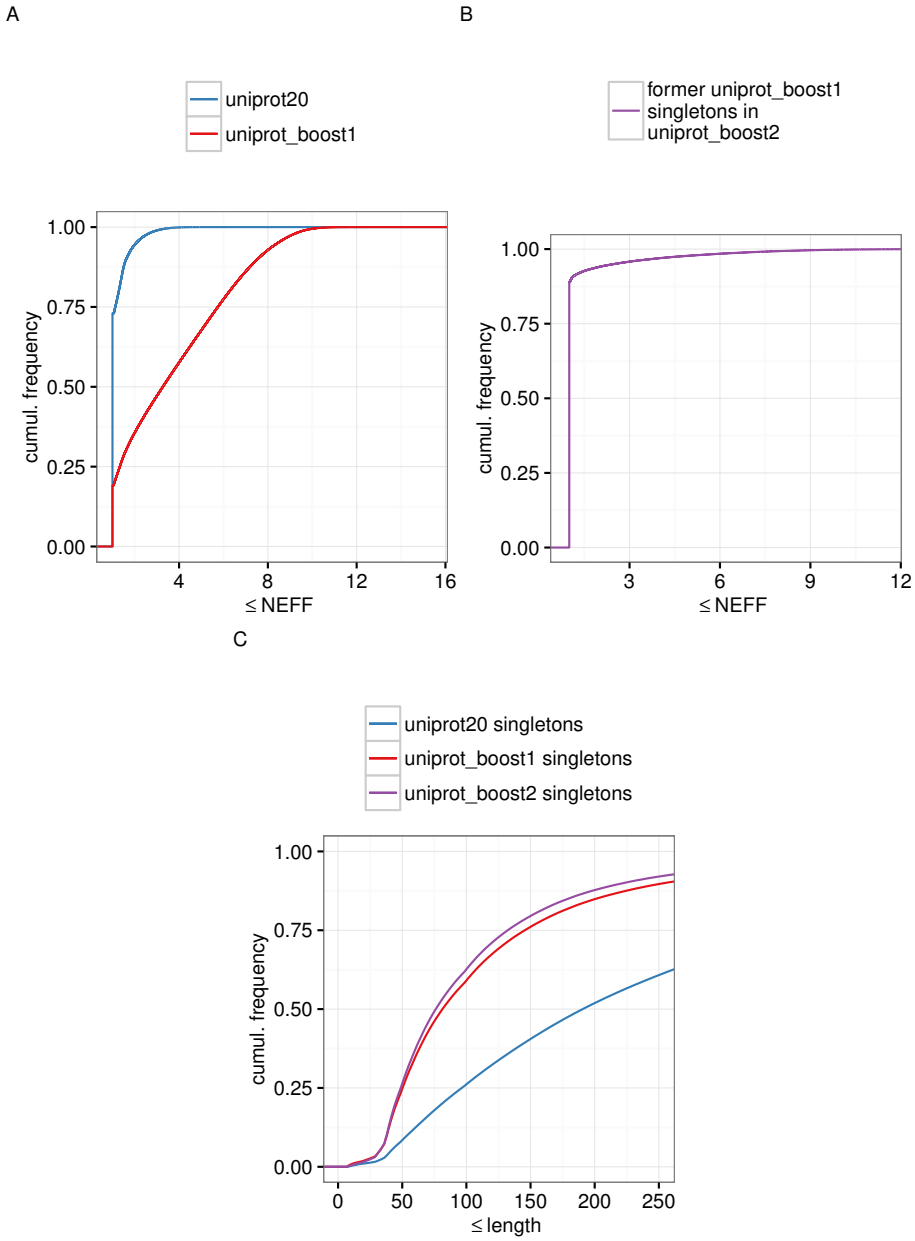

Figure 1.4: A) distribution of the number of effective sequences (NEFF) in the uniprot20 and the uniprot_boost1. In the uniprot20 around 75% of the MSAs have a NEFF of 1. In the uniprot_boost1 this fraction can be reduced to around 20%. B) distribution of NEFFs in the uniprot_boost2 for clusters with one effective sequence in the uniprot_boost1. Around 10% of the clusters in the uniprot_boost1 with one effective sequence were enriched in the uniprot_boost2. C) distribution of the lengths of MSAs (number of match states in the Hidden Markov profile) with a NEFF of 1 in the uniprot20, uniprot_boost1 and uniprot_boost2.

1.4.4 Homology Detection Sensitivity

Dataset We filtered the structural domain database SCOP (release 1.75) for a maximum pairwise sequence identity of 20% and excluded only templates that were covered by at least 90%. We split the domains by fold in a test (80%, 5 287 domains) and an optimization set (20%, 1 329 domains). The test set is referred to as SCOP20 in the following. We searched with the homology detection tools PSI-BLAST, HMMER and HHblits for each query of the SCOP20 through the SCOP20. We can measure the homology detection sensitivity in a ROCX plot, since we know all potential homologs for each query in the SCOP20 by means of the SCOP classification (see chapter).

PSI-BLAST - UniProt For each of the SCOP20 sequences we searched in 1, 2 and 3 iterations with PSI-BLAST through the UniProt sequence database merged with the SCOP20 sequence database. The intermediate profile output of PSI-BLAST seems to be buggy, since the results were much better with direct searches through the merged sequence database.

HHblits - uniprot20 We built the HMM benchmark database for HHblits with the SCOP20 sequences resembling the diversity of the uniprot20. For this purpose, we mapped the SCOP20 sequences to existing uniprot20 clusters. We chose the cluster with a BLAST search of the SCOP20 sequence against the consensus sequences of the uniprot20 MSAs.

For each of the SCOP20 sequences we searched with HHblits in 1, 2 and 3 iterations through uniprot20 merged with the corresponding SCOP20 HMM database.

The intermediate profile output of PSI-BLAST did not work, so we had to search with HHblits also through the merged database.

HHblits - uniprot_boost1 We built the HMM benchmark database for HHblits with the SCOP20 sequences resembling the diversity of the uniprot_boost1. For this purpose, we mapped the SCOP20 sequences to existing uniprot_boost1 clusters. We chose the cluster with a BLAST search of the SCOP20 sequence against the consensus sequences of the uniprot_boost1 MSAs.

For each of the SCOP20 sequences we searched with HHblits in 1, 2 and 3 iterations through uniprot_boost1 merged with the corresponding SCOP20 HMM database.

The intermediate profile output of PSI-BLAST did not work, so we had to search with HHblits also through the merged database.

HMMER - UniProt For each of the SCOP20 sequences we searched with jackhmmer in 1, 2 and 3 iterations through the UniProt sequence database merged with the SCOP20 sequence database.

The intermediate profile output of PSI-BLAST did not work, so we had to search with jackhmmer also through the merged database.

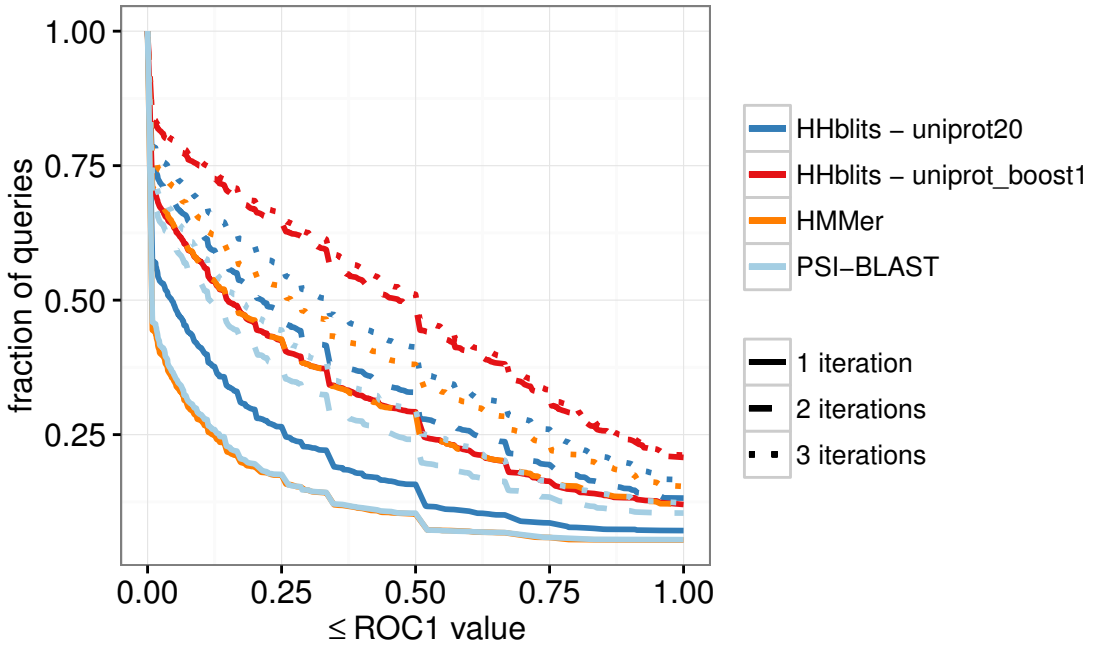

Figure 1.5: Homology detection sensitivity benchmark in the ROC1 plot comparing PSI-BLAST and HMMER on the UniProt sequence database and HHblits on the corresponding uniprot20 and uniprot_boost1 databases.

ROCX In order to measure the homology detection sensitivity, we calculated the ROCX statistic. The Receiver Operating Characteristic up to the X^{th} false positive hit (ROCX) is the fraction of true positive hits of all possible true positive hits found until the X^{th} false positive hit appears in the sorted list of hits.

We defined true positive hits to share the same superfamily and true negative hits to disagree with the query in at least the fold. We excluded the self-hit from the statistic as well as hits within Rossman-like folds (c.2-c.5, c.27 and 28, c.30 and 31) and the four- to eight-bladed β-propellers (b.66-b.70), which are probably evolutionarily related [58]. HMMER reported just in one of three queries more than one false positive hit even with very generous settings, so we used the ROC1 statistic.

Result HHblits finds in three iterations through the uniprot_boost1 about 20% more homologs than through the uniprot20 (area under the ROC1 curve; see figure 1.5). The increase in sensitivity from the second to the third iteration through the uniprot_boost1 is just 1.4%.

HHblits finds in three iterations through the uniprot_boost1 about 56% and 28% more homologs than PSI-BLAST and HMMER, respectively, through the UniProt sequence database.

1.4.5 Alignment Quality

Dataset We used structural alignments with TMalign [69] as gold standard to measure the quality of the generated alignments of HHalign and HMMER. We did an all-against-all comparison of the domain structures in the SCOP20 set within the same superfamily, but not the same family. We selected the alignment pairs (query and template) with a TM-score ≤ 0.8 and > 0.5 to get a non-trivial set of homologs (35 132 alignment pairs).

The TM-score is a real number between 0.0 and 1.0, with structurally more similar templates having higher TM-scores [69]. Protein pairs with a TM-score > 0.5 are mostly in the same fold while those with a TM-score < 0.5 are mainly not in the same fold [65].

HHalign - uniprot20 We built HMMs for the query sequences with HHblits searches in three iterations through the uniprot20. The template sequences were aligned with HHalign to the most similar cluster in the uniprot20. We want to estimate the alignment quality of HHblits independent of the homology detection sensitivity. Therefore, we enforce the alignments between query and template HMM with HHalign that uses the same alignment algorithms as HHblits. We used different thresholds for the Maximum ACcuracy alignment (*-mact* option; 0.01, 0.1, 0.35, 0.5) to adjust the alignment greediness. A low threshold generates more global alignments that are very sensitive but less precise.

HHalign - uniprot_boost1 We built the HMMs for the query sequences with HHblits searches in two iterations through the uniprot_boost1, since the homology detection sensitivity benchmark showed that most homologs are already found after the second iteration. The template sequences were aligned HHalign to the most similar cluster in the uniprot_boost1 database. We want to estimate the alignment quality of HHblits independent of the homology detection sensitivity. Therefore, we enforce the alignments between query and template HMM with HHalign that uses the same alignment algorithms as HHblits.

HMMER We built the HMMs for the query sequences with jackhmmer searches in five iterations (default setting) through the UniProt sequence database. The homology detection benchmark showed that the prefilter of HMMER hides potential homologs from the database. We built for each hmmsearch run an individual database that contained just one template sequence and turned off all acceleration heuristics (option *–max*), so we could ensure an alignment by hmmsearch if possible.

Benchmark For each tool we calculated the average per-residue precision and per-residue sensitivity compared to the structural alignments of TMalign.

$$sensitivity = \frac{TP}{TP + FN} \tag{1.1}$$

$$precision = \frac{TP}{TP + FP} \tag{1.2}$$

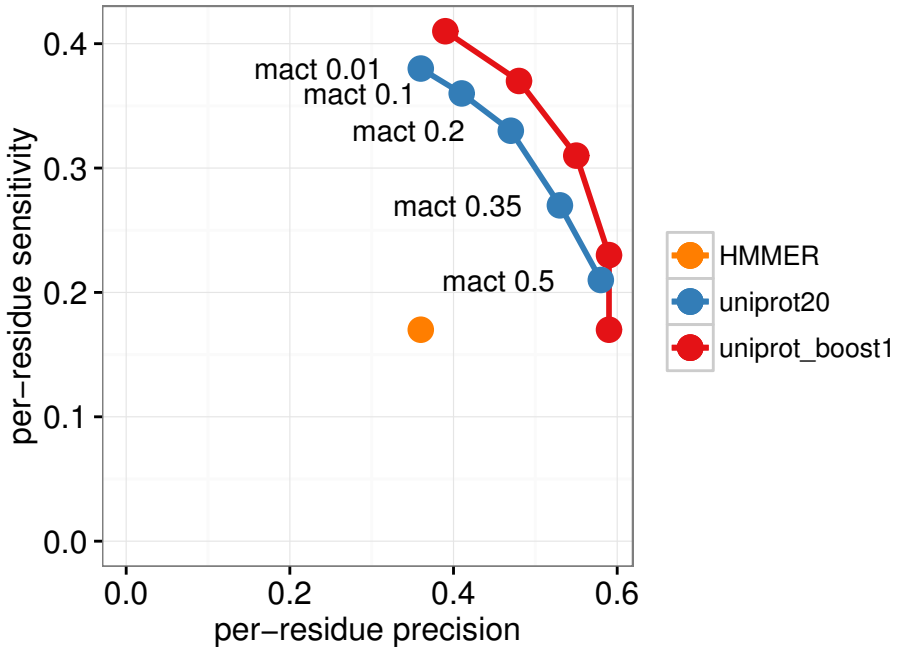

Figure 1.6: Alignment quality benchmark with average per-residue precision and sensitivity with respect to structural reference alignments by TMalign. We compared alignments calculated by HMMER and by HHalign on profiles generated with the uniprot20 and with the uniprot_boost1 databases for different *mact* values (alignment greediness setting).

In this case True-Positive (TP) means aligned residues that are also aligned in the reference alignment. False-Negative (FN) means not aligned residues that are aligned in the reference alignment. False-Positive (FP) means aligned residues that are not aligned in the reference alignment. Sensitivity defines what fraction of the reference alignment is captured by the calculated alignment. Precision defines what fraction of the calculated alignment is actually correctly aligned with respect to the reference alignment.

Result The overall alignment quality in the benchmark is low, since we focused on difficult alignments in this benchmark (see figure 1.6). We observed that the curve for the different thresholds in the Maximum ACcuracy alignment with the uniprot_boost1 lies above the curve with the uniprot20 in figure 1.6. For the mact of 0.35 (HHalign's default setting) the average per-residue sensitivity and precision of the uniprot_boost1 alignments improve by 11.32% and 17.39%, respectively, compared to the uniprot20 alignments and by 63.89% and 58.82%, respectively, compared to the HMMER alignments (see table 1.3).

For global alignments (mact 0.01) the per-residue precision and sensitivity of the uniprot_boost1 alignments improve by 8.3% and 7.9%, respectively, compared to the uniprot20. The improvement is greater for difficult alignments with low sequence identity (see table 1.4). The sequence identity is taken from the structural reference alignments.

tool	database	mact	precision	sensitivity	$\sqrt{precision \cdot sensitivity}$
HMMER	UniProt		0.36	0.17	0.25
HHalign	uniprot_boost1	0.01	0.39	0.41	0.40
HHalign	uniprot_boost1	0.1	0.48	0.37	0.42
HHalign	uniprot_boost1	0.2	0.55	0.31	0.41
HHalign	uniprot_boost1	0.35	0.59	0.23	0.36
HHalign	uniprot_boost1	0.5	0.59	0.17	0.32
HHalign	uniprot20	0.01	0.36	0.38	0.37
HHalign	uniprot20	0.1	0.41	0.36	0.38
HHalign	uniprot20	0.2	0.47	0.33	0.39
HHalign	uniprot20	0.35	0.53	0.27	0.38
HHalign	uniprot20	0.5	0.58	0.21	0.35

Table 1.3: Average per-residue sensitivity and precision compared to structural reference alignments.

		uniprot_boost1		uniprot20	
alignment identity	# alignments	precision	sensitivity	precision	sensitivity
$x \leq 0.1$	13618	0.25	0.27	0.22	0.23
$0.1 < x \leq 0.2$	20264	0.46	0.49	0.44	0.47
$0.2 < x \leq 0.3$	1208	0.64	0.69	0.64	0.68
$0.3 < x \leq 0.4$	41	0.78	0.84	0.79	0.84

Table 1.4: Average per-residue sensitivity and precision of the HHalign alignments with a mact of 0.01 split up by the alignment identity of the structural reference alignments.

1.4.6 Model Quality

Models with fixed Query-Template Pairs

The comparison of sequence alignments to structural alignments is a good indicator for the general performance of a sequence alignment tool. However, it does not consider alternative similar structural alignments. In the following benchmark we compared the predicted structure induced by the calculated alignment to the native structure.

Benchmark We used the alignments of the alignment quality benchmark calculated with a threshold for the Maximum ACcuracy alignment of 0.01 and built the corresponding structural models with Modeller [62]. We compared the models to their respective native structure by TM-score [69].

Result We calculated the TM-scores of structural models to the corresponding native structures. The models were built with Modeller on global (mact 0.01) HHalign alignments with fixed query-template pairs of the SCOP20 (intra-superfamily, inter-family, TM-score ≤ 0.8 and > 0.5 of the native structures). For the uniprot_boost1 the query HMMs were built with HHblits searches in two iterations through the uniprot_boost1 and the corresponding template HMMs resemble the diversity of the uniprot_boost1. For the uniprot20 the query HMMs were built with HHblits searches in three iterations through the uniprot20 and the corresponding template HMMs resemble the diversity of the uniprot20.

Queries, whose models on the basis of the uniprot20 and a TM-score of ≤ 0.3, have significantly better models with the uniprot_boost1 (see figure 1.7). The average TM-score of the uniprot_boost1 and uniprot20 models is 0.43 and 0.41 respectively. This is an improvement of 4.9%.

Homology Modeling Benchmark

In the previous benchmark we analyzed models with fixed query-template alignment pairs. This did not consider that we might find other templates for queries with the uniprot_boost1 compared to the uniprot20. Therefore, we set up a complete but still simple homology modeling pipeline to take this effect into account.

Data - uniprot20 We filtered the chain sequences in the PDB (release February 2015; SEQRES section) for maximal 30% pairwise sequence identity. We removed only sequences with at least 90% coverage to another sequence (24 558 sequences). We built multiple sequence alignments for PDB chains with homologous sequences found in 3 search iterations with HHblits through the uniprot20 database. For these multiple sequences we built the HHsuite database pdb30_uni20.

For each multiple sequence alignment of a PDB chain in the pdb30_uni20 we searched with HHblits in one iteration through the pdb30_uni20 database for potential homologs. We used the best homolog (excluding the self-hit) to build a structural model with Modeller. We compared the structural model to the native structure with the program TMscore.

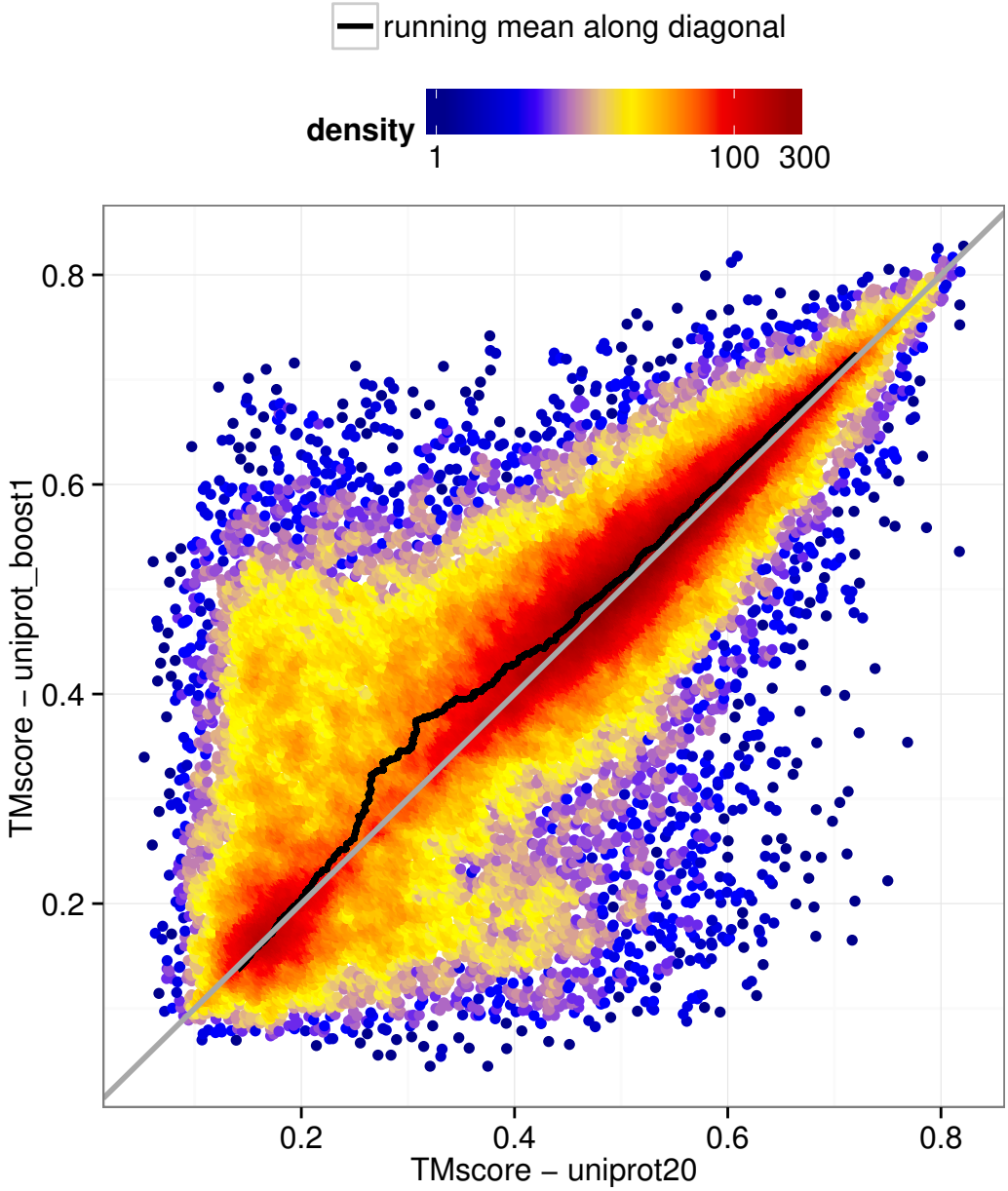

Figure 1.7: We compared the TM-scores of 35 132 structural models built with Modeller on global (mact 0.01) HHalign alignments with fixed query-template pairs of the SCOP20 (intra-superfamily, inter-family, TM-score ≤ 0.8 and > 0.5 of the native structures). For the uniprot_booost1 the query HMMs were built with HHblits searches in one iteration through the uniprot_boost1 and the corresponding template HMMs resembled the diversity of the uniprot_boost1. For the uniprot20 the query HMMs were built with HHblits searches in three iterations through the uniprot20 and the corresponding template HMMs resembled the diversity of the uniprot20.

	# queries	pdb30_boost1 - TM-score	pdb30_uni20 - TM-score
different template	16260	0.59	0.56
same template	8256	0.74	0.72

Table 1.5: Average TM-scores for 24 558 structure models of the pdb30_boost1 and pdb30_uni20 split up by queries that report a different best-scoring template and those that report the same best-scoring template in both databases.

Data - uniprot_boost1 We filtered the chain sequences in the PDB (release February 2015; SEQRES section) for maximal 30% pairwise sequence identity. We removed only sequences with at least 90% coverage to another sequence (24 558 sequences). We built multiple sequence alignments for these PDB chains with homologous sequences found in two search iterations with HHblits through the uniprot_boost1 database, since the homology detection sensitivity benchmark showed that most homologs are already found after the second iteration. For these multiple sequences we built the HHsuite database pdb30_boost1.

For each multiple sequence alignment of a PDB chain in the pdb30_boost1 we searched with HHblits in one iteration through the pdb30_boost1 database for potential homologs. We used the best homolog (excluding the self-hit) to build a structural model with Modeller. We compared the structural model to the native structure with the program TMscore.

Result Models of the pdb30_uni20 and TM-score of ≤ 0.25 can be improved by the models with the pdb30_boost1 (see figure 1.8). The average TM-score on the pdb30_boost1 and pdb30_uni20 is 0.64 and 0.61, respectively. This leads to an overall improvement of 4.9%. Templates with a high TM-score ≥ 0.75 in the pdb30_uni20 worsen in the corresponding TM-score of the pdb30_boost1.

Some queries report a different best-scoring template in the pdb30_uni20 than in the pdb30_boost1 (see table 1.5). Here we observed the following:

- 66% of the queries report different highest-scoring templates with the pdb30_boost1 compared to the pdb30_uni20.
- The average TM-score for both pdb30 databases is 25% - 35% higher for queries that report the same template than for queries that report a different template.
- The average TM-score of queries that report the same template can be improved in the pdb30_boost1 by 2.78%.
- The average TM-score of queries that report a different template can be improved in the pdb30_boost1 by 5.36%.

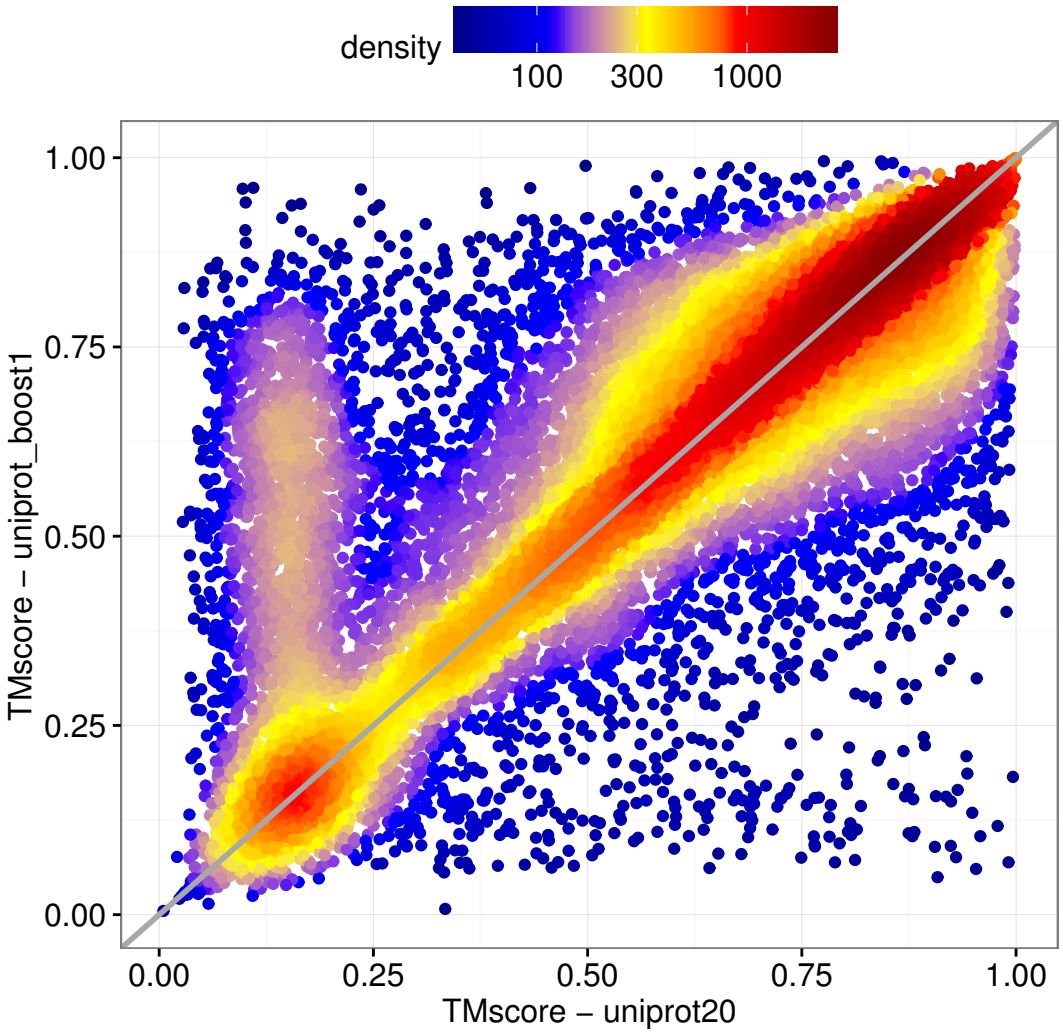

Figure 1.8: Comparison of 24 558 structure models built in a simple homology modeling pipeline with the the pdb30_boost1 and the pdb30_uni20 databases. We calculated the TM-score for all models with respect to their native structure. Both databases are built on the sequences of the PDB filtered for maximal 30% pairwise sequence identity. The MSAs for the pdb30_boost1 were built in two search iterations of HHblits through the uniprot_boost1. The MSAs for the pdb30_uni20 were built in three search iterations of HHblits through the uniprot20.

1.5 Discussion and Conclusion

We enriched the very conservative clusters of the uniprot20 with homologs found by HHblits in one search iteration through the uniprot20 to build the diversity-enriched uniprot_boost1 database. The clusters of the uniprot_boost1 contain about 14 times more sequences. The effective number of sequences is increased from 1.18 to 3.78. This increase in diversity leads to several performance improvements for HHblits.

Three iterations through the uniprot_boost1 are on average 3.7 times slower and find about 20% more homologs than three iterations through the uniprot20.

Three search iterations with HHblits through the uniprot_boost1 are about 0.37 times slower and find about 56% more homologs than three search iterations with PSI-BLAST through the UniProt.

Three search iterations with HHblits through the uniprot_boost1 are about 2.1 times faster and find about 32% more homologs than three search iterations with HMMER through the UniProt.

For default alignment greediness setting (mact of 0.35) the average per-residue sensitivity and precision of the uniprot_boost1 alignments improve by 11.32% and 17.39%, respectively, compared to the uniprot20 alignments and by 63.89% and 58.82%, respectively, compared to the HMMER alignments (see table 1.3). Global alignments (mact 0.01) of sequences with less than 10% sequence identity are improved by 28% in precision and 5% in sensitivity.

The structure models of global alignments with the uniprot_boost1 have a 4.9% higher TMscore than with the uniprot20. Especially models on the basis of the uniprot20 with a TMscore of < 0.3 can be improved with the models on the basis of the uniprot_boost1.

In a simple pipeline for protein homology modeling detection we showed that we are able to detect other templates for 66% of the queries with on average 5.4% higher TMscores with the uniprot_boost1 compared to the uniprot20. Models of queries that report the same template with both databases have a 2.8% higher TMscore due to the improved alignment quality with the uniprot_boost1.

The uniprot_boost1 improves especially models on the basis of the uniprot20 that have a TMscore < 0.25, however models with the uniprot20 that have a TMscore > 0.75 decreased slightly in the TMscore. In HHpred, our in-group developed homology modeling server [36], we apply a neural network to select the best template specifically for homology modeling. It might be useful to train a new neural network to select the best templates from uniprot20 and uniprot_boost1 models.

The generation of the uniprot_boost1 clusters with HHblits took 150 000 CPU hours on the Amazon Web Services. A faster but less sensitive approach might be the generation with iterative MMseqs profile-profile searches.

In summary, HHblits finds with the diversity-enriched uniprot_boost1 database more homologs and generates more accurate alignments that lead to better structure models than with the uniprot20. It further widens the gap to HHblits' competitors HMMER and PSI-BLAST. We showed that the performance of HHblits with the uniprot_boost1 can be transferred to the downstream application of homology modeling. The uniprot_boost1 is capable to become another default database for HHblits and may impact sequence-based

predictions of evolutionarily conserved properties, such as secondary or tertiary structure, disorder, catalytic sites, post-translational modifications, short linear motifs, or interaction interface. In chapter 2 we will use the uniprot_boost1 as a substrate to design an algorithm to decompose the complete protein sequence space into domains.

Chapter 2

Chopping Protein Sequence Space into Domains

2.1 Abstract

Domains are the basic structural, functional and evolutionary units of proteins. To elucidate the cellular functions of a protein we need to understand the molecular functions of its domains. Protein sequences are annotated by the matches to domains in domain family databases such as Pfam, SCOP or CATH. However, existing domain databases cover less than half of the known sequence space and encompass only a small fraction of all protein domain families. Here, we developed an algorithm based on a Bayesian statistical model, called Pdom, with which we can consistently decompose the entire protein sequence space into its evolutionary units. We showed that, for proteins of known structure, this decomposition agrees very well with the domain definitions based on their structure, function and evolution. We hope that the resulting database of protein domain families, UniDom, will become the basis for homology-based protein sequence annotation efforts.

2.2 Introduction

Domains are the functional subunits of proteins. During evolution they are copied and reused in different proteins. This leads to domains that appear in different proteins with different domain compositions (see figure 3). Due to their functions, domains are essential for the cell and the organism. That is why domains are under evolutionary pressure to maintain their function. Domain border prediction tools that apply comparative sequence analysis exploit the observation that the homologous region shared between two proteins usually starts with a common domain start and ends with a common domain end. By performing an all-against-all comparison within the protein sequence space, domain border predictors try to infer the domain boundaries from starts and ends of the generated pairwise alignments. Those alignments indicate the shared homologous region between two proteins, usually the shared domains. There are three domain border prediction tools that apply comparative sequence analysis: DoBo [25], ADDA [34] and EVEREST [50].

ADDA

ADDA [34] performs an all-against-all search with BLAST [10] within a protein sequence database. For each query sequence a multiple sequence alignment is calculated from the pairwise alignments of BLAST.

ADDA assumes that most of the templates share a single domain with the query that is captured by the pairwise alignment. Therefore, two residues i and j of the query are considered to be in the same domain, if they are frequently aligned together in the different templates.

A residue correlation matrix is derived from the multiple sequence alignment (see figure 2.1). For each pair of positions i and j in the query, the residue correlation matrix counts how often they were aligned simultaneously in the different templates. ADDA splits the query into putative domains to maximize the counts in the residue correlation matrix within the resulting domains and to minimize the counts in the residue correlation matrix between the resulting domains.

Afterwards, ADDA selects iteratively from this set of putative domains on basis of an objective function that considers the likelihood of alignments to end or start within a domain and the likelihood of alignments to cover multiple domains.

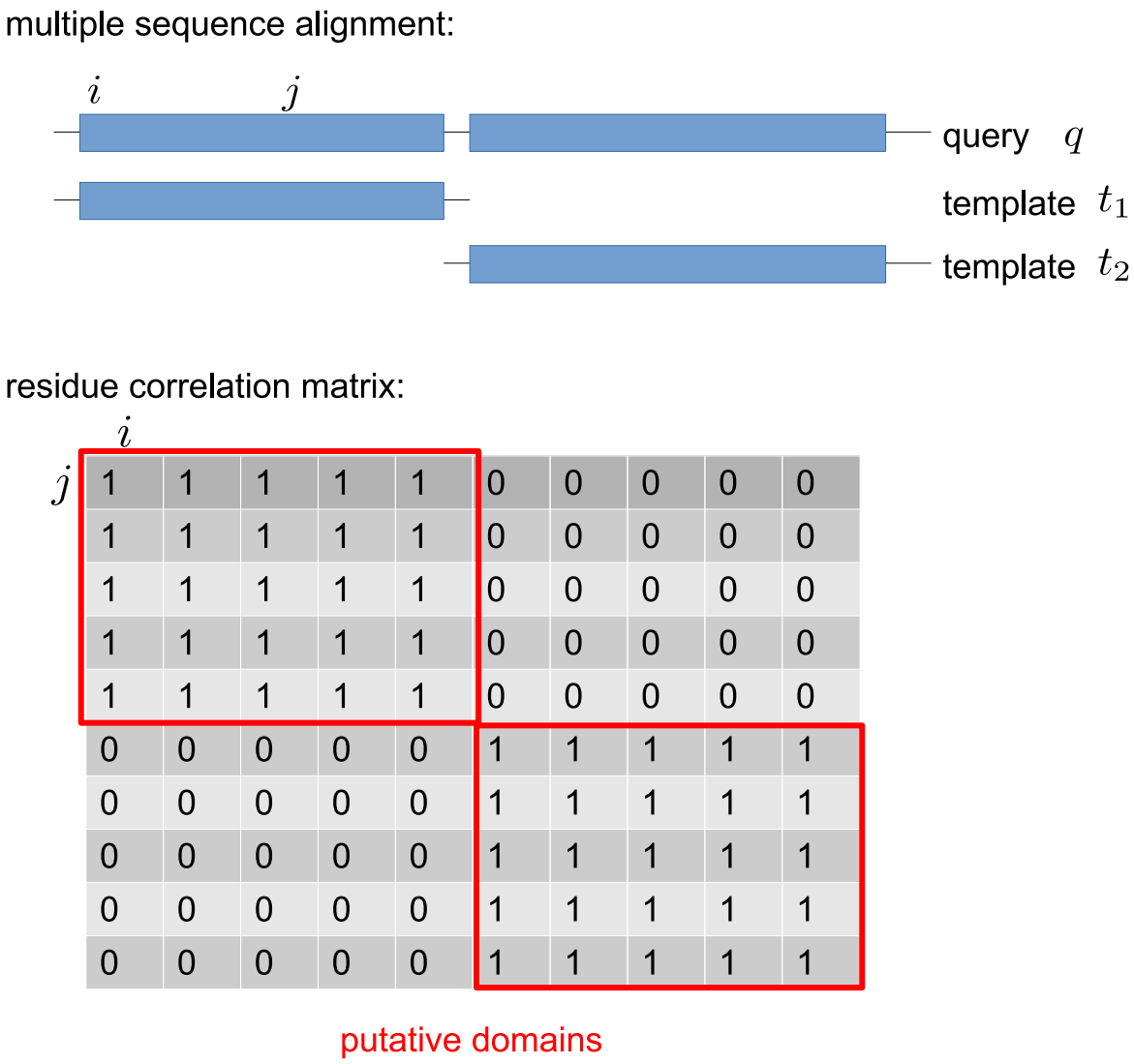

Figure 2.1: ADDA performs an all-against-all search with BLAST within a protein sequence database. A residue correlation matrix is generated for each query. Each cell in the matrix contains the counts for the pair of residues i and j in the query how often they were aligned simultaneously in the different templates. ADDA splits the query into domains. Each split separates the residue correlation matrix in four quadrants. The splits are chosen to maximize the counts within the putative domains (upper left and lower right quadrant) and minimize the counts between the two putative domains (upper right and lower left quadrant). Each putative domain may be split further into smaller putative domains. The final set of putative domains is chosen on basis of an objective function that considers the likelihood of alignments to end or start within a domain and the likelihood of alignments to cover multiple domains.

DoBo

DoBo [25] calculates a multiple sequence alignment for the query with a PSI-BLAST search [10].

Within the multiple sequence alignment DoBo searches for domain boundary signals.

A domain boundary signal is a gap which begins at the N- or C-terminal end of a sequence in the multiple sequence alignment and extends continuously for at least 45 residues. The location of the domain boundary signal is the first non-gap residue in the sequence.

DoBo classifies the set of domain boundary signals in:

- False boundary signals occurring in a single domain protein.
- Near boundary signals occurring within 20 residues of any domain boundary in a multi-domain protein.
- Away boundary signals occurring more than 20 residues away of any domain boundary in a multi-domain protein.

DoBo uses two Support Vector Machines (SVM) to decide between these three cases. The first SVM decides between false boundary or true boundary signals. The second SVM decides for the true boundary signals between near boundary or away boundary signals.

Both SVMs use the same set of features:

- The SVMs consider for each residue in a window of 41 residues centered around the signal site with the normalized frequencies of the 20 residues plus a gap, the predicted likelihood of the residue to be part of an alpha-helix, a beta-strand or a loop, and the predicted likelihood to be buried or to be exposed.
- The SVMs take as features the position of the signal with respect to the N terminal (residue index divided by 100), the position with respect to the C terminal (protein length minus residue index divided by 100) and the number of boundary signal sites within 5 residues.
- Another feature is the length of the sequence divided by 100.
- DoBo calculates the total number of signals generated by all of the sequences in the multiple sequence alignment within a 11 residue window of the signal site. This local sum was calculated for each residue in the sequence and then converted to z-scores. The final feature is the z-score for the signal site.

EVEREST

In order to predict domains, EVEREST [50] filters a protein sequence database for redundancy and repeated consecutive sequence segments are removed. EVEREST performs an all-against-all search within this database with BLAST [10].

The alignment of every query template pair with E-score < 100 from the all-against-all BLAST search is recalculated with an iterative variant of the Smith-Waterman algorithm [56]. Significantly similar aligned sequence segments together with their **sequence similarity** are collected in a segment database. EVEREST's workflow is illustrated in figure 2.2.

For each pair of template segments on the same query, an **overlap similarity** is calculated, that is the length of their intersection divided by the length of their union. The segments on the same query are clustered into groups according to their overlap similarity. EVEREST requires each pair of segments within the same group to have an overlap similarity of ≥ 0.5.

The groups are clustered by their sequence similarity of their segments by average linkage clustering. EVEREST starts with a single cluster for each group. In multiple iterations the two most similar clusters are merged until there is just one cluster left. EVEREST keeps track of each intermediate merged cluster. Those merged clusters are candidate domain families.

EVEREST applies a boosting algorithm [21] to select appropriate candidate families. The machine learing approach takes cluster intrinsic features, e.g. size of the cluster, similarity of the clusters that were merged, variance of the sequence lengths in the cluster ([50] mentions just these example features).

For each selected candidate family a multiple sequence alignment is calculated with Clustal-W [59]. Each MSA is subsequently transformed to a Hidden Markov model with HMMER [39].

HMMER searches for each HMM of the selected candidate families through the protein sequence database. From the aligned sequence segments, a segment database is re-generated. Segments created by the same HMM have a **sequence similarity** equal to the sum of their E-Scores from HMMER. With this new database a new iteration can be started. After three iterations, overlapping domain families are merged.

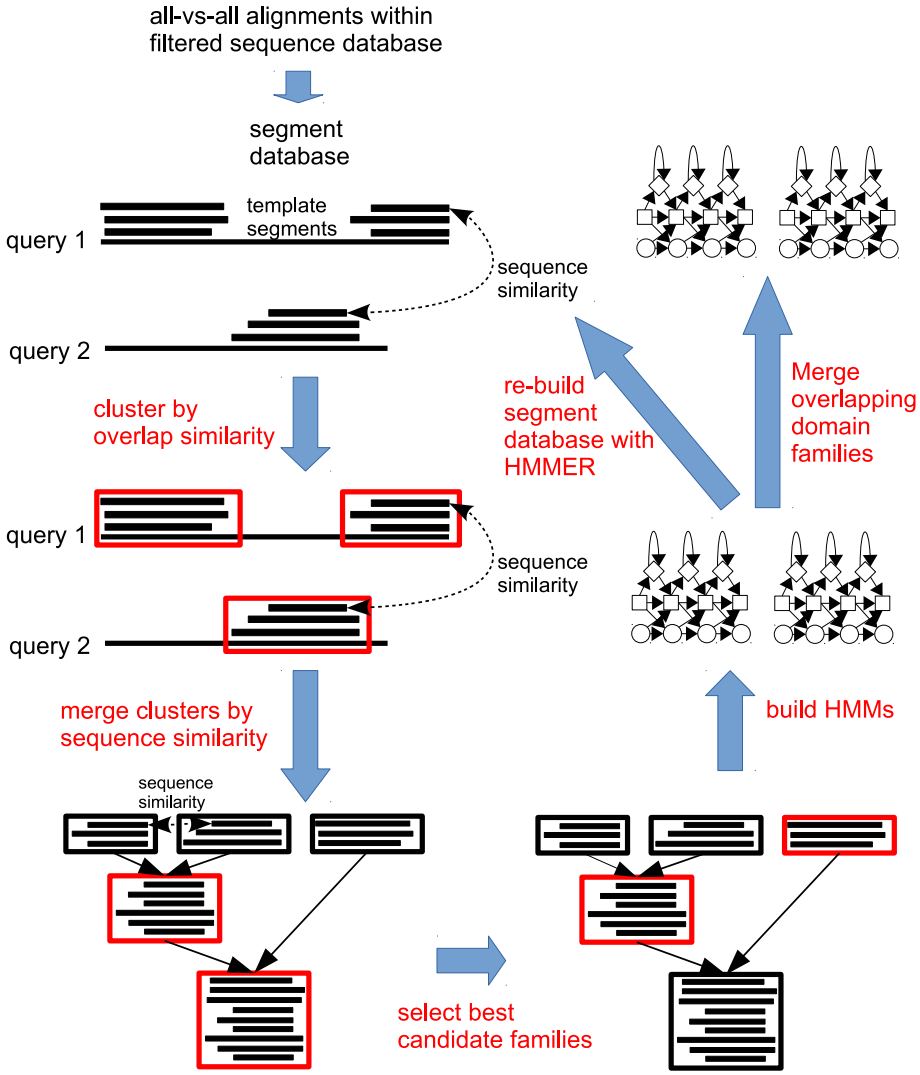

Figure 2.2: EVEREST performs an all-against-all search within a protein sequence database. From this search a database of the aligned template segments is generated together with the pairwise sequence similarity between segments. The template segments for a query are clustered by their overlap similarity, that is the length of their intersection divided by the length of their union. Those groups of segments are hierarchically merged by the sequence similarity of their segments. A machine learning approach selects the best candidate domain families of these merged clusters. For each selected domain family a Hidden Markov model is calculated. With these HMMs and HMMER a new segment database may be built for a new iteration or overlapping domain families are merged.

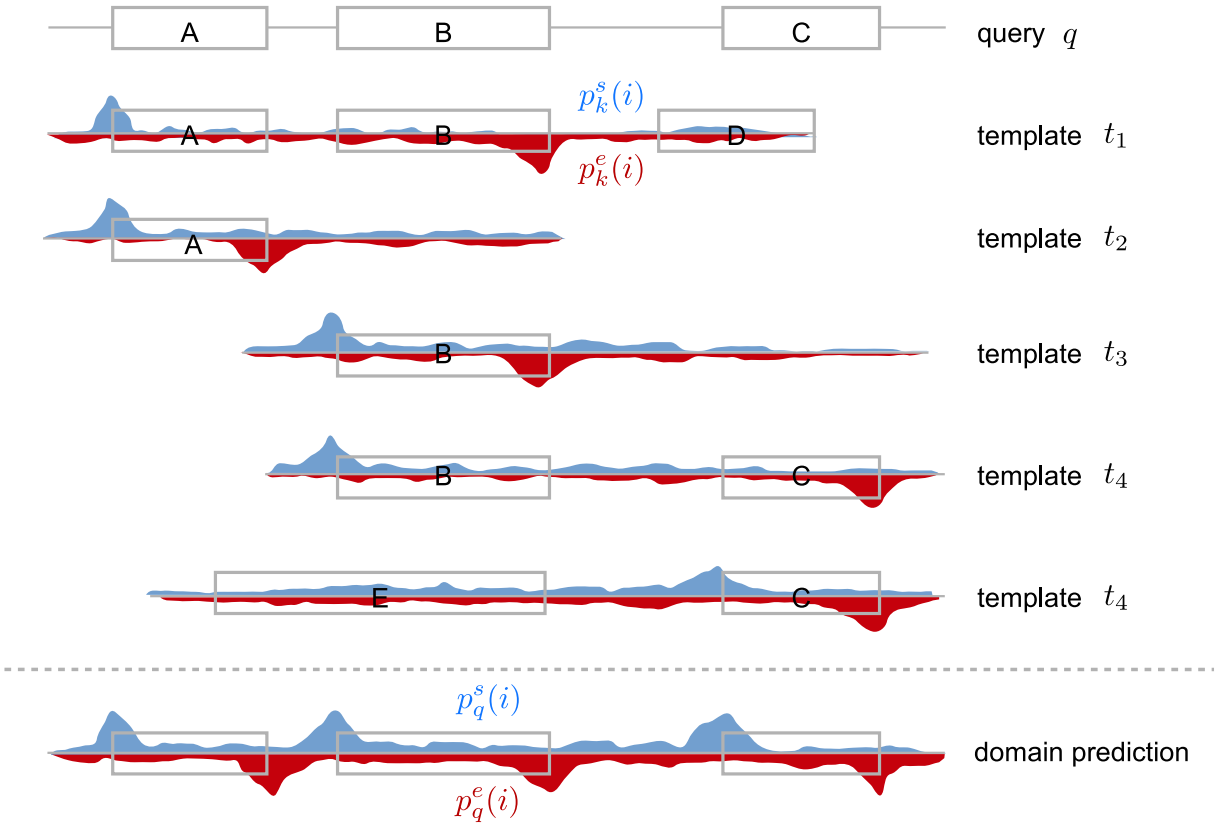

Figure 2.3: Identifying domain borders in a query sequence by alignments against all known sequences. By calculating the probabilities for the alignment start and end of the homologous region (blue and red distributions) in a large number of sequence alignments, the precise domain starts and ends in the query sequence can be determined in a Bayesian approach. The more diverse the found domain architectures of the template, the better the domain decomposition. More diverse domain architectures are more remotely related and harder to find with homology detection tools like HHblits.

Pdom

In the following, we will describe the method of our domain predictor Pdom. We calculated the all-against-all alignments with HHblits within the HMM database uniprot_boost1 (see chapter 1). For each alignment between a query and a template, we get the alignment start and end probabilities for each pair of residues in the query and template. We can combine the alignment start and end probabilities of several templates to a domain prediction in a Bayesian statistical model (see figure 2.3). This approach allows us to consider inserted domains (see section 2.3.3). Since we calculate domain predictions for every cluster in the uniprot_boost1, we can include the domain prediction of the templates in further iterations to maximize the consistency between the domain predictions (see section 2.3.4).

2.3 Method

In chapter 1 we introduced the uniprot_boost1 database a large set of 4 866 021 protein profile Hidden Markov Models (HMM) that encompassed the known protein sequence space (release October 2012, see chapter 1). We performed an all-against-all search with HHblits within this database. Let q be the query profile HMM, since we queried the rest of the database for homologs to this query sequence. The detected homologous database profile HMMs are called templates $t_1, \ldots, t_K$.

HHblits returns the following information about the alignment between the query q and the template t_k

1. the probability P_{qk} that query q and template t_k are homologous;
2. the alignment backward probability $b_{qk}(i, j)$ that the alignment *starts* at position i of the query q and position j of the template t_k;
3. the alignment forward probability $f_{qk}(i, j)$ that the alignment *ends* at position i of the query q and position j of the template t_k;
4. the alignment posterior probability $p(q_i \diamond t_{k,j}|A_{qk})$ that position i in the query q and position j in the template t_k are aligned.

This information, abbreviated as $A_{qk} = (P_{qk}, b_{qk}, f_{qk}, p(q_i \diamond t_{k,j}|A_{qk}))$, is calculated by HHblits.

The goal of the statistical framework presented in the following is to extract from the all-versus-all pairwise alignments A_{qk} the domain start posterior probabilities $p_q^s(i)$ and domain end posterior probabilities $p_q^e(i)$ for all sequences q in the database at each position i. The domain start posterior probability $p_q^s(i)$ is the likelihood for a domain start in the query q at position i given the pairwise alignments A_{qk}. The domain end posterior probability $p_q^s(i)$ is the likelihood for a domain start in the query q at position i given the pairwise alignments A_{qk}.

2.3.1 Notation

q	query HMM
$t_1, \ldots, t_K$	template HMM with which q is aligned
L_q, L_k	length of query sequence q and template sequence t_k
i, j, l	indices for positions in q or t_k
σ, σ'	labels for domain start s or end e position. Without inserted domains: $\sigma, \sigma' \in \{s, e\}$. With inserted domains: $\sigma, \sigma' \in \{s^0, e^0, s^1, e^1, \ldots\}$ (see section 2.3.3 for explanation of superindices)
P_{qk}	probability that q and t_k are homologous, calculated by HHblits
$b_{qk}(i, j)$	alignment backward probability that the alignment *starts* at position i of the query q and position j of the template t_k, calculated from the Backward algorithm in HHblits
$f_{qk}(i, j)$	alignment forward probability that the alignment *ends* at position i of the query q and position j of the template t_k, calculated from the Forward algorithm in HHblits
$p(q_i \diamond t_{k,j} \vert A_{qk})$	alignment posterior probability that position i of query q and position j of template t_k are aligned, calculated from the Backward/-Forward algorithm in HHblits
$\mathcal{A}_q = \{A_{q1}, \ldots, A_{qK}\}$	pairwise alignments between q and templates $t_1, \ldots, t_K$. A_{qk} subsumes all alignment information, $A_{qk} = (P_{qk}, b_{qk}, f_{qk}, p(q_i \diamond t_{k,j} \vert A_{qk}))$.
$\mathcal{Y}_q = \{y_1, \ldots, y_{2D}\}$	set of ordered domain start s and end e points in the query. Each point $y_\nu = [\sigma, i]_\nu$ consists of a label $\sigma \in \{s, e\}$ and of the position $i \in \{0, \ldots, L_q\}$ of the ν'th start or end point. D is the number of domains.
$p([\sigma, i] \vert \ldots)$	$p([\sigma, i] \in \mathcal{Y}_q \vert \ldots)$, probability for $[\sigma, i]$ to be a domain start/end point
$p([\sigma, i]_\nu \vert [\sigma', j]_{\nu-1} \ldots)$	probability for label $[\sigma, i]$ to directly succeed label $[\sigma', j]$
s_{qk}, e_{qk}	start and end positions of common homologous region (*not* aligned region) between q and t_k; must coincide with domain start/end positions
$q \sim t_k$	q is homologous to t_k; example: $p(q \sim t_k \vert A_{qk}) = P_{qk}$
$F_{\sigma,i}$	Forward probability for domain start ($\sigma = s$) or end ($\sigma = e$)
$B_{\sigma,j}$	Backward probability for domain start ($\sigma = s$) or end ($\sigma = e$)
$\text{pred}(\sigma), \text{succ}(\sigma)$	set of states that are possible predecessors / successors to σ
$p_q^\sigma(i)$	probability that a domain starts ($\sigma = s$) or ends ($\sigma = e$) at position i in the query

2.3.2 Forward-Backward algorithm for domain start and end probabilities

The sequence of a protein shall be classified in linker and domain regions. Linker regions are the unconserved or less conserved residues before and after a domain. Those regions are usually structurally disordered. Every protein sequence q starts with a linker at position 1, possibly of length 0, and it ends with a linker at position L_q. Therefore, the following virtual domain end and start points are given,

$$\boxed{\begin{aligned} y_0 &:= [e, 0] \\ y_\dagger &:= [s, L_q + 1] \end{aligned}} \tag{2.1}$$

to fix the boundary conditions of the dynamic programming problem.

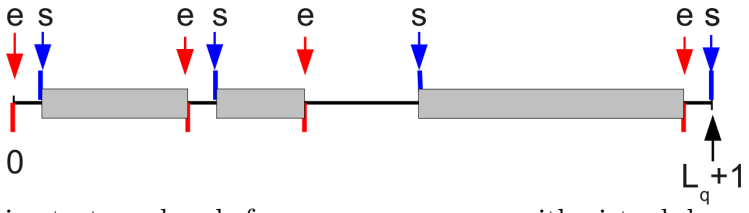

Figure 2.4: Domain starts and ends for a query sequence with virtual domain end and start at position 0 and $L_q + 1$, respectively.

With the exception of inserted domains, every domain is followed by a linker, and every linker is followed either by another domain or the end of the sequence (see figure 2.4).

We applied the forward-backward algorithm to compute the posterior probability $p([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q)$ for a domain start $\sigma = s$ and end $\sigma = e$ at position $i \in \{1, \ldots, L_q\}$, given the alignments $\mathcal{A}_q$ of the query with the K templates (see figure 2.5). Each path through the dynamic programming matrix shown in figure 2.5 represents one domain decomposition of the query protein.

We define the domain forward probabilities $F_{\sigma,i}$ as the sum of the probabilities of all paths $\{y_0 \ldots y_\nu\}$ starting at $y_0 = [e, 0]$ and ending at $y_\nu = [\sigma, i]$ in the matrix, with $\sigma \in \{s, e\}$:

$$F_{\sigma,i} := \sum_{\text{all paths } y_1 \ldots y_{\nu-1}} p(y_1 \ldots y_{\nu-1}, y_\nu = [\sigma, i] | \, y_0, \mathcal{A}_q) \;=\; p([\sigma, i] | \, y_0, \mathcal{A}_q) \tag{2.2}$$

where $y_1 \ldots y_\nu$ is a valid path with ordered domain starts and domain ends at positions $i_0 < i_1 < \ldots < i_\nu$. Analogously, we define the backward probabilities $B_{\sigma,j}$ as the sum of probabilities of all paths $\{y_\nu \ldots y_\dagger\}$ starting at $y_\nu = [\sigma, i]$ and ending at $y_\dagger = [s, L_q + 1]$:

$$B_{\sigma,i} := \sum_{\text{all paths } y_{\nu+1} \ldots y_{2D}} p(y_{\nu+1} \ldots y_{2D+1} | \, y_\nu = [\sigma, i], \mathcal{A}_q) \;=\; p(y_\dagger | \, y_\nu = [\sigma, i], \mathcal{A}_q) \tag{2.3}$$

The domain posterior probabilities $p([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q)$ for a domain start or end at position $i \in \{1, \ldots, L_q\}$ can be written as:

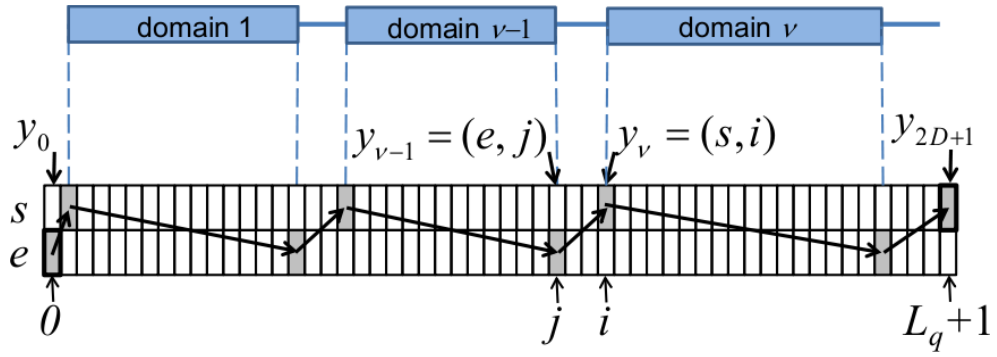

Figure 2.5: Dynamic programming algorithm that does not take into account inserted domains. Each path through the matrix represents a domain decomposition of the query protein. By dynamic programming, the optimal path and the posterior probabilities for domain starts s and ends e at any position i can be found.

$$p([\sigma, i]|\, y_0, y_\dagger, \mathcal{A}_q) = \frac{p([\sigma, i], y_\dagger|\, y_0, \mathcal{A}_q)}{p(y_\dagger|\, y_0, \mathcal{A}_q)}\,. \tag{2.4}$$

The denominator is $F_{s,L_q+1} = B_{e,0}$. The numerator can be split into two terms:

$$\begin{aligned} p([\sigma, i], y_\dagger, |\, y_0, \mathcal{A}_q) &= \sum_{\text{all paths } \{y_1 \ldots y_{2D}\} \text{ through } [\sigma,i]} p(y_1 \ldots y_{2D+1}|\, y_0, \mathcal{A}_q) \\ &= \sum_{\text{all paths } \{y_1 \ldots y_{2D}\} : y_\nu = [\sigma,i]} p(y_1 \ldots y_{\nu-1}, y_\nu = [\sigma, i]|\, y_0, \mathcal{A}_q)\; p(y_{\nu+1} \ldots y_{2D+1}|\, y_\nu = [\sigma, i], \mathcal{A}_q) \\ &= F_{\sigma,i}\, B_{\sigma,i} \end{aligned} \tag{2.5}$$

Therefore, the posterior probability is

$$\boxed{p_q^\sigma(i) = p([\sigma, i]|y_0, y_\dagger, \mathcal{A}_q) = \frac{F_{\sigma,i}\, B_{\sigma,i}}{F_{s,L_q+1}}} \tag{2.6}$$

The initialization is given by equation (2.1). The iteration equation for the domain start positions of the forward matrix is:

$$\begin{aligned} F_{s,i} &= \sum_{\text{all paths } y_1 \ldots y_{\nu-1}} p(y_1 \ldots y_{\nu-1}, y_\nu = [s, i]|y_0, \mathcal{A}_q) \\ &= \sum_{j=0}^{i} \sum_{\text{all paths } y_1 \ldots y_{\nu-2}} p(y_1 \ldots y_{\nu-2}, y_{\nu-1} = [e, j]|y_0, \mathcal{A}_q)\, p(y_\nu = [s, i]|y_{\nu-1} = [e, j], \mathcal{A}_q) \\ &= \sum_{j=0}^{i} F_{e,j}\, g_q(s, i|e, j) \end{aligned} \tag{2.7}$$

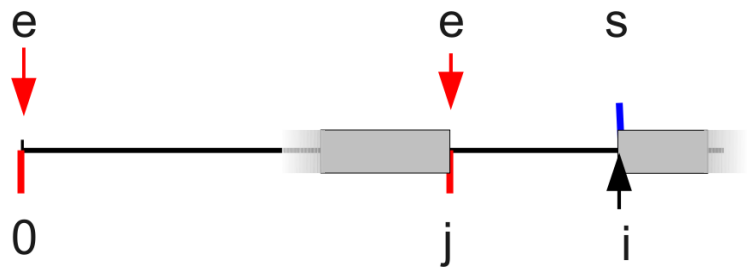

Figure 2.6: Illustration of the Forward iteration equation (2.7). We can calculate the Forward probability $F_{s,i}$ for a domain start at position i by summing up the joint probability of all preceding positions j to be a domain end $F_{e,j}$ and the transition probability $g_q(s,i|e,j)$ for adjacent domain end at position j and domain start at position i.

where we used the following abbreviation:

$$\boxed{g_q(\sigma,i|\sigma',j) := p(y_\nu = [\sigma,i]|y_{\nu-1} = [\sigma',j],\mathcal{A}_q)} \tag{2.8}$$

The iteration equation for the domain end positions of the forward matrix is analogous to the previous equation. In summary, we get

$$\boxed{\begin{aligned} F_{s,i} &= \sum_{j=0}^{i-1} F_{e,j}\, g_q(s,i|e,j) \\ F_{e,i} &= \sum_{j=0}^{i-1} F_{s,j}\, g_q(e,i|s,j) \end{aligned}} \tag{2.9}$$

The backward iteration equations are derived in the same way:

$$\begin{aligned} B_{s,i} &= \sum_{\text{all paths } y_{\nu+1}\ldots y_{2D}} p(y_{\nu+1}\ldots y_{2D+1}|y_\nu = [s,i],\mathcal{A}_q) \\ &= \sum_{j=i+1}^{L_q+1} \sum_{\text{all paths } y_{\nu+2}\ldots y_{2D}} p(y_{\nu+1} = [e,j], y_{\nu+2}\ldots y_{2D+1}|y_\nu = [s,i],\mathcal{A}_q) \\ &= \sum_{j=i+1}^{L_q+1} \sum_{\text{all paths } y_{\nu+2}\ldots y_{2D}} p(y_{\nu+2}\ldots y_{2D+1}|y_{\nu+1} = [e,j],\mathcal{A}_q)\; p(y_{\nu+1} = [e,j]|y_\nu = [s,i],\mathcal{A}_q) \\ &= \sum_{j=i}^{L_q+1} B_{e,j}\; g_q(e,j|s,i) \end{aligned} \tag{2.10}$$

and analogously for $B_{e,i}$. In summary, the backward iteration equations are

$$\boxed{\begin{aligned} B_{s,i} &= \sum_{j=i+1}^{L_q+1} B_{e,j}\, g_q(e,j|s,i) \\ B_{e,i} &= \sum_{j=i+1}^{L_q+1} B_{s,j}\, g_q(s,j|e,i) \end{aligned}} \tag{2.11}$$

The derivation of $g_q(\sigma, i|\sigma', j)$ is shown in section B.2.2.

2.3.3 Extension to inserted domains

In about $5-20\%$ of multi-domain proteins at least one domain is inserted within or interlaced with another domain [67]. We introduced start and stop states s^λ, e^λ for each level of nesting $\lambda = 0, 1, 2, \ldots$ to model those cases of inserted domains (see figure 2.7).

Forward-Backward algorithm for inserted domain boundary prediction

Equations (2.2)-(2.6) for the definition of the forward, backward and posterior domain probabilities are unmodified, except that they now apply to states $\sigma, \sigma' \in \{s^0, e^0, s^1, e^1, s^2, e^2, \ldots\}$. The iteration equations for forward and backward probabilities are generalized:

$$\boxed{\begin{aligned} F_{\sigma,0} &= I(\sigma = e^0) \\ B_{\sigma,L_q+1} &= I(\sigma = s^0) \\ F_{\sigma,i} &= \sum_{\sigma' \in \mathrm{pred}(\sigma)} \sum_{j=0}^{i-1} F_{\sigma',j}\, g_q(\sigma, i|\sigma', j) \\ B_{\sigma,i} &= \sum_{\sigma' \in \mathrm{succ}(\sigma)} \sum_{j=i+1}^{L_q+1} B_{\sigma',j}\, g_q(\sigma', j|\sigma, i) \end{aligned}} \tag{2.12}$$

where the definition of $g_q(\sigma', j|\sigma, i,)$ in equation (2.8) is the same as before but applies to $\sigma, \sigma' \in \{s^0, e^0, s^1, e^1, s^2, e^2, \ldots\}$. $\mathrm{pred}(\sigma)$ and $\mathrm{succ}(\sigma)$ are the sets of possible predecessor and successor states of σ, respectively, as shown in figure 2.7b. Example given, $\mathrm{pred}(s^0) = \{e^0\}, \mathrm{pred}(e^0) = \{s^0, e^1\}$.

In equation (2.6) the state s is changed to s^0:

$$p_q^\sigma(i) = p([\sigma, i]|y_0, y_\dagger, \mathcal{A}_q) = \frac{F_{\sigma,i}\, B_{\sigma,i}}{F_{s^0,L_q+1}}\,. \tag{2.13}$$

We derive $g_q(\sigma, i|\sigma', j)$ for inserted domains in section B.3.

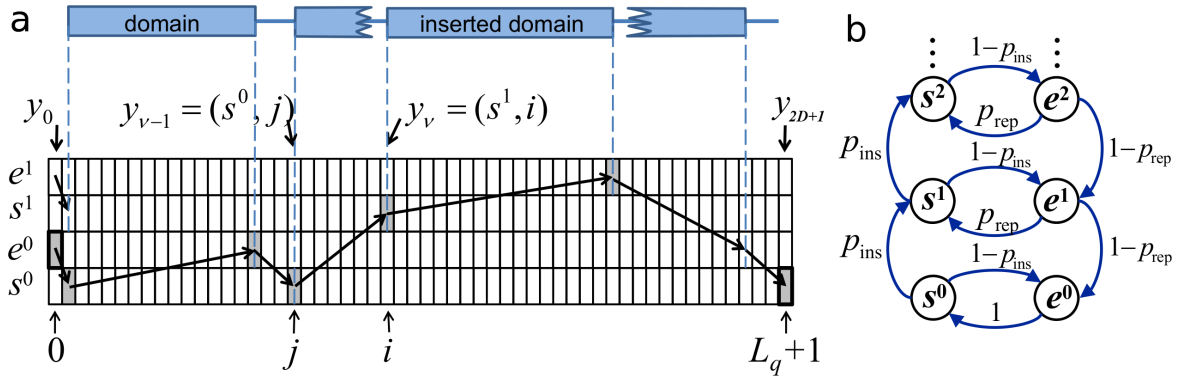

Figure 2.7: Dynamic programming algorithm that takes possible inserted domains into account. (**a**) Each path through the matrix represents a domain decomposition of the query protein. Into a domain between states s^0 and e^0, a domain can be inserted whose start and end positions are labeled with s^1 and e^1. (**b**) Within inserted domains, further domains can be inserted in a nested fashion. Domain starts are labeled $s^0, s^1, s^2, \ldots$, domain ends are labeled $e^0, e^1, e^2, \ldots$. The upper indices denote the level of nested insertions. The probability for a domain insertion is p_{ins}, the probability for an inserted domain to be followed by another inserted domain on the same nesting level is p_{rep}.

2.3.4 Consistency iterations

Our goal is to iteratively improve the consistency between predictions for all pairs of sequences by using the information on domain starts and ends in template t_k from the previous iteration to improve the prediction of domain starts and ends of query q. The first iteration corresponds to what has been described so far. After a few consistency-increasing iterations, the domain start and end probability profiles should converge.

Updating the query-template alignments

In the first iteration of the algorithm, predictions were based on the alignment start and end probabilities , p^σ_{qk} (equations (B.1), (B.2)). In subsequent iterations, we use the *domain* start and end posterior probabilities p^σ_k,

$$p^\sigma_k(i) := p([\sigma, i] \in \mathcal{Y}_{t_k} \,|\, y_0, y_\dagger, \mathcal{A}_{t_k}) \tag{2.14}$$

derived in the previous iteration, in order to improve the *alignment* start and end probability estimates p'^σ_{qk}. In the following, we assume that the alignment forward and backward probabilities $f_{qk}(i,j)$ and $b_{qk}(i,j)$ are approximately independent of the domain start and end posterior probabilities $p^\sigma_k(i)$, since these integrate information gathered over many

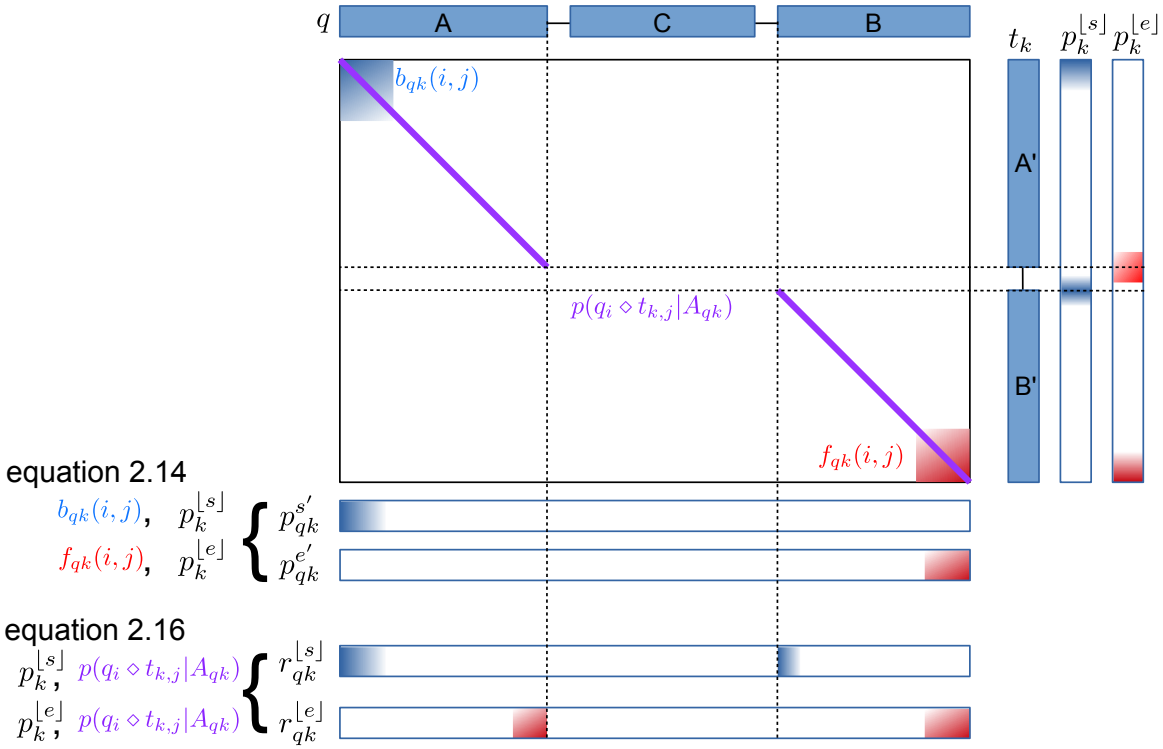

Figure 2.8: The simple update of the alignment start probabilities $b_{qk}(i,j)$ with the predicted domain start probabilities $p_k^{\lfloor s \rfloor}(j)$ of the previous iteration leads to improved alignment start profiles $p_{qk}^{s'}(i)$ (see section 2.3.4). Combining the alignment probability $p(q_i \diamond t_{k,j}|A_{qk})$ and the predicted domain boundaries $p_k^{\lfloor s \rfloor}(j)$ of the previous iteration in the calculation of $r_{qk}^{\lfloor \sigma \rfloor}(i)$ leads to new domain start signals after domains in the query not shared with the template t_k (see section 2.3.4). The same procedure can be applied analogously to the alignment and domain end probabilities.

alignments:

$$
\begin{aligned}
p_{qk}^{s}{}'(i) &\approx \frac{\sum_{j=1}^{L_k} b_{qk}(i,j)\, p_k^s(j)}{\sum_{i'=1}^{L_q}\sum_{j=1}^{L_k} b_{qk}(i',j)\, p_k^s(j)} \\
p_{qk}^{e}{}'(i) &\approx \frac{\sum_{j=1}^{L_k} f_{qk}(i,j)\, p_k^e(j)}{\sum_{i'=1}^{L_q}\sum_{j=1}^{L_k} f_{qk}(i',j)\, p_k^e(j)}
\end{aligned}
\tag{2.15}
$$

We transfer the information about domain starts and ends from the templates onto the query.

Transferring information about domain boundaries within alignments

The previous modification to subsequent iterations is not sufficient to attain full consistency between the domain decomposition of all queries in the database. Consider a query sequence

with three domains, A, C and B and a template with homologous domains A' and B' aligned to it. Suppose the template has reliably predicted domain boundaries between A' and B'. Both sequences have domains A and B in common, improving the estimate of the alignment start and end positions will not improve the prediction of the domain end of A and the domain start of B in the query, even though the alignment would allow to predict these boundaries in the query from their positions in the template (see figure 2.8).

We transfer the information about domain starts and ends from the templates onto the query by using the alignment posterior probability $p(q_i \diamond t_{k,j}|A_{qk})$ that position i of q and position j of t_k are aligned:

$$r_{qk}^{\lfloor\sigma\rfloor}(i) = p(s_{qk} \leq i \leq e_{qk} \wedge [\sigma, i] \in \mathcal{Y}_q | A_{qk}, p_k^{\lfloor\sigma\rfloor}) = \sum_{j=1}^{L_q} p(q_i \diamond t_{k,j}|A_{qk})\, p_k^{\lfloor\sigma\rfloor}(j) \tag{2.16}$$

We define $\lfloor\sigma\rfloor$ by $\lfloor s^\lambda \rfloor := s$ and $\lfloor e^\lambda \rfloor := e$, where s and e signify any of the start or end states, respectively. We further define r_{qk}^* as the probability that the alignment A_{qk} includes column i of q:

$$r_{qk}^*(i) = p(s_{qk} \leq i \leq e_{qk}|A_{qk}) = \sum_{j=1}^{L_q} p(q_i \sim t_{k,j}|A_{qk}) \tag{2.17}$$

For the consistency iterations we use almost the same procedure to compute the next iteration of $p_q^s(i)$, including the forward-backward algorithm. But instead of computing $g_q(\sigma, i|\sigma', j)$ as defined in equation (2.8), we needed to calculate

$$g_q(\sigma, i|\sigma', j) = p([\sigma, i]_\nu | [\sigma', j]_{\nu-1}, \mathcal{A}_q, {p^s_{1\ldots K}}', p^e_{1\ldots K})', \tag{2.18}$$

which differs from equation (2.8) only by conditioning also on the domain start and end posterior probability densities $p_k^s(j)$ and $p_k^e(j)$ from the previous iteration (equation (2.14)). The information in $p_k^s(j)$ and $p_k^e(j)$ is included by ${p_{qk}^s}'(i)$, ${p_{qk}^e}'(i)$, $r_{qk}^s(i)$, $r_{qk}^e(i)$, and $r_{qk}^*(i)$.

We derived $g_q(\sigma, i|\sigma', j)$ for the consistency iteration in section B.4.

2.3.5 Algorithm for Domain Border Prediction

We calculate a domain decomposition with the following algorithm. For this purpose, we calculate the mean posterior domain probability for each σ and position i in a window of $i \pm \delta$:

$$p^*([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q) = \sum_{j=i-\delta}^{i+\delta} p([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q) \left(1.0 - \frac{|i-j|}{\delta}\right)$$

We determine for each position i and each σ the most likely precursor border $[j, \sigma']$ in the matrix $v([\sigma, i])$:

$$v([\sigma, i]) = \max_{\substack{j<i \\ \sigma' \in \mathrm{pred}(\sigma)}} F(\sigma', j) g(i, \sigma | j, \sigma') \frac{p([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q)}{p^*([\sigma, i] \in \mathcal{Y}_q | \mathcal{A}_q)}$$

For each entry we save the most likely precursor $[j, \sigma']$. Beginning from $i = L_q + 1$ and and $\sigma = s^0$ we trace back the most likely path through the matrix $v([\sigma, i])$, that ends in $\sigma = e^0$ and $i = 0$.

2.3.6 HHblits - Alignment Information Output

HHblits builds high-quality MSAs starting from single sequences or from MSAs. It transforms the input into a query HMM and iteratively searches through a database of HMMs by adding significantly similar sequences to the query HMM for the next search iteration. HHblits scans a discretized representation of the database Hidden Markov profiles, the column state sequences, with a fast prefilter. HMMs whose column state sequences pass the prefilter are aligned to the query HMM using the fast Viterbi HMM-HMM alignment algorithm. HHblits realigns all high scoring Viterbi alignments in a second stage using the more accurate, but slower Maximum ACcuracy (MAC) algorithm.

During the Maximum Accuracy algorithm the backward and forward algorithms are applied. These algorithms calculate the alignment start and end probabilities, respectively. The alignment posterior probability for position i in the query q and position j in the template t_k is calculated from the alignment start and end probabilities.

Alignment Matrix Compression

The alignment start, alignment end and alignment posterior probabilities together with the probability that query q and template t_k are homologs are used by Pdom. For our final goal to split the whole known sequence space into domains we developed a compressed binary format to output these matrices with HHblits (see figure 2.9).

Floats are usually saved in 32 bits. The last bit is for the sign, the following eight bits are for the exponent and the remaining 23 bits are for the mantissa. This representation is defined in the Standard for Floating-Point Arithmetic by the Institute of Electrical and Electronics Engineers (IEEE 754).

The float value can be calculated from its bits in the following way:

$$(-1)^{sign} \cdot 2^{exponent-127} \cdot fraction \tag{2.19}$$

$$exponent = \sum_{i=0}^{7} b_{23+i} 2^i \tag{2.20}$$

$$mantissa = 1 + \sum_{i=0}^{22} b_{22-i} 2^{-i-1} \tag{2.21}$$

For our binary representation of probabilities, we do not need the full scope of 32 bits and can reduce the representation to eight bits, the usual size of a char (see figure 2.10). The compressed float representation can display a minimal float value of $3.0517578 \cdot 10^{-5}$ and a maximal float value of 1.9375 with a precision of $1.907349 \cdot 10^{-6}$.

```
<query_name>\0<query length|unsigned short int>
#first ali
    <template name>\0<template length|unsigned short int><ali prob|char><ali sim|unsigned short int>
    #ali start probabilites
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
        ...
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
    <0|unsigned short int>
    #ali end probabilities
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
        ...
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
    <0|unsigned short int>
    #ali posterior probabilities
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
        ...
        <query_pos|unsigned short int> <template_pos|unsigned short int><prob|char>...<prob|char>\0
    <0|unsigned short int>
#second ali
    <template name>\0<template length|unsigned short int><ali prob|char><ali sim|unsigned short int>
    #alignment start probabilites
    ...
```

Figure 2.9: Compression of alignments for Pdom in HHblits; The output includes the name of the query, the length of the query and for each template the template name, the length of the template, the probability that query and template are homologous, the similarity between query and template and the three matrices for alignment start probabilities, alignment end probabilities and alignment posterior probabilities. The matrices are printed for each query position in lines of consecutive template positions with probabilities above 0.01.

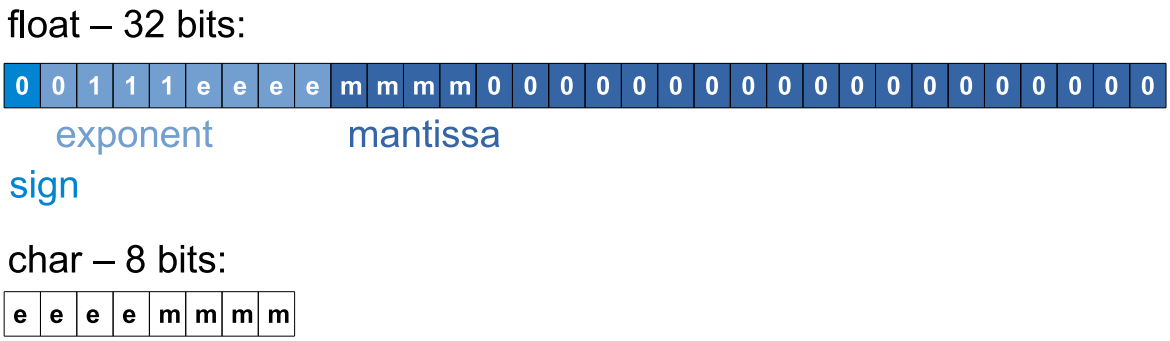

Figure 2.10: Floats are usually saved in 32 bits. The last bit is for the sign, the following eight bits are for the exponent and the remaining 23 bits are for the mantissa. For our binary representation of probabilities, we do not need the full precision of 32 bits and can reduce the representation to eight bits, the usual size of a char. Therefore, we use only the bits indicated by e and m.

2.3.7 Training, Testing and Benchmark Set

We filtered the SCOP (release 1.75) to a maximum 25% pairwise sequence identity and filtered only pairs with at least 90% coverage. We mapped the filtered SCOP sequences to UniRef50 sequences with SWIPE, a fast Smith-Waterman alignment implementation [53]. We included only domain annotations with an e-value $< 10^{-5}$, and only proteins that have annotations in releases of Pfam and ADDA, so we can compare Pdom to those domain predictors. The selected domains were filtered for a maximal overlap of ten residues. We split up the proteins by their SCOP domain folds into a training, testing and benchmark set.

The final benchmark set consists of 914 proteins with 1379 domain annotations. The final test set for parameter optimization of Pdom consists of 240 proteins with 393 domain annotations.

Pfam

We used the domain annotation of Pfam-A (version 27.0) [26].

ADDA

We used the latest domain predictions of ADDA (October 2012, `http://genserv.anat.ox.ac.uk/downloads/adda/`) [34].

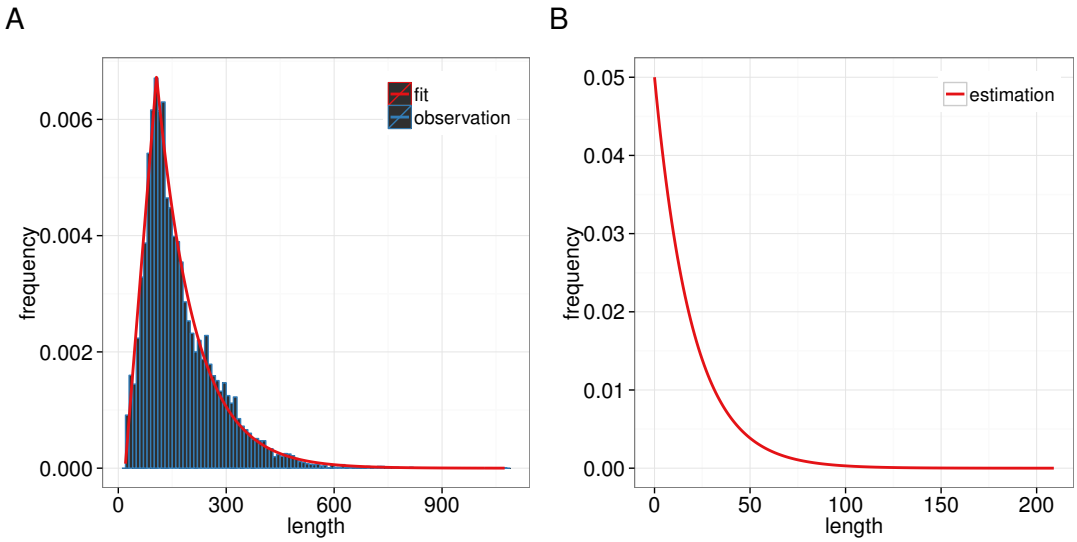

Figure 2.11: (A) Training with fit of the domain length distribution; (B) Estimation of the linker length distribution

2.4 Optimization and Training

2.4.1 Training of Domain and Linker Lengths

The algorithm requires prior knowledge about the linker length distribution, the domain length distribution and the length distributions for domain fragments separated by domain insertions (see section B.2.2). The algorithm assumes different distributions for the different nesting levels λ. There are not enough examples for higher nesting levels in SCOP, therefore we assumed the same distributions for those higher nesting levels.

We trained the domain length and the length of domain fragments split by domain insertions from SCOP annotations (SCOPe version 2.05 [27]; filtered to a maximum of 95% pairwise sequence identity). We fitted the distribution

$$f(x,a,c,d) = \begin{cases} z^{-1}\frac{x-a}{c-a} & \text{if } x \leq c \\ z^{-1}e^{-d(x-c)} & \text{if } x > c \end{cases}$$

with:

$$z^{-1} = \frac{1-e^{-d}}{0.5(c-a-1)+1}$$

to these trained statistics with the L-BFGS-B algorithm [17] (see figure 2.11A and figure 2.12).

SCOP separates proteins in the PDB completely into domains without linker regions. Therefore, we assumed a reasonable distribution for the linker lengths (see figure 2.11B).

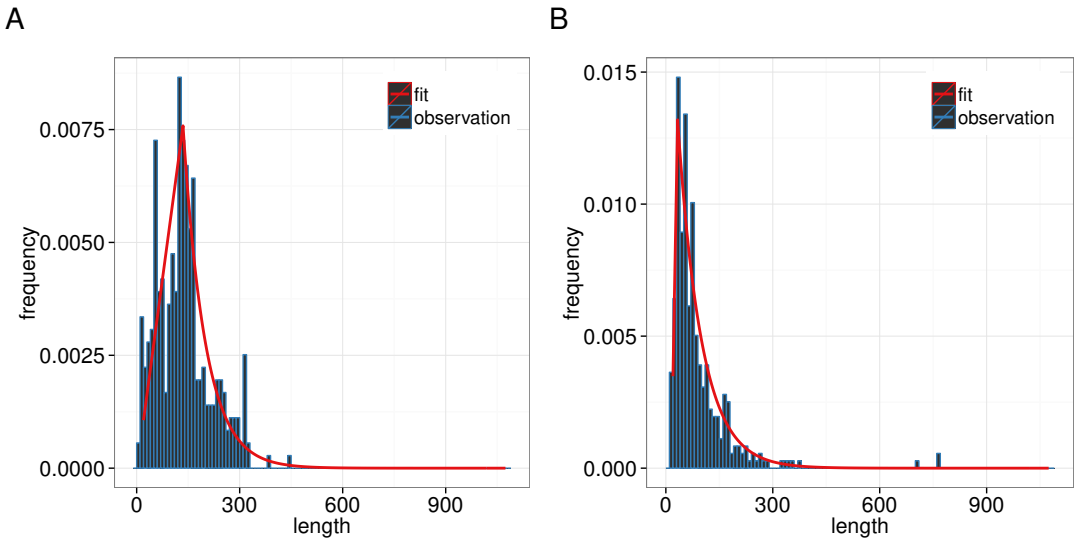

Figure 2.12: Training with fit of the left (A) and right (B) domain fragment length distributions. Those fragments are caused by insertions.

2.4.2 Scores for Benchmark and Optimization

Domain Boundary Shift For each annotated domain start in the benchmark set we pick the closest predicted domain start. The distance between annotated and predicted domain start is called shift. The predictions are more accurate the lower the shift. The same is done for domain ends.

Domain starts close to the N-terminus and domain ends close to the C-terminus are easier to predict. Therefore, we limited the benchmark to annotated domain borders at least 40 residues away from the N-/C-terminal site of the protein. However, this score does not penalize too short domain predictions (see figure 2.13).

Domain Coverage For each annotated domain in the reference set we calculate the coverage with each predicted domain. For each annotation we pick the best-covering prediction. Each prediction may be used only once. This statistic does not penalize too long domain predictions for proteins sparsely covered by reference domain annotations (see figure 2.14).

Optimization Score We developed an optimization score that captures how well the domain prediction matches the annotation. For each reference domain of the query we pick the best covering predicted domain. No predicted domain can be used twice to cover two reference domains. For each predicted domain of the query we pick the best covering reference domain. No annotated reference domain can be used twice to cover two predicted domains. The final score is the mean coverage of all references and predictions. The score is designed to penalize too short and too long domain predictions (see figure 2.15).

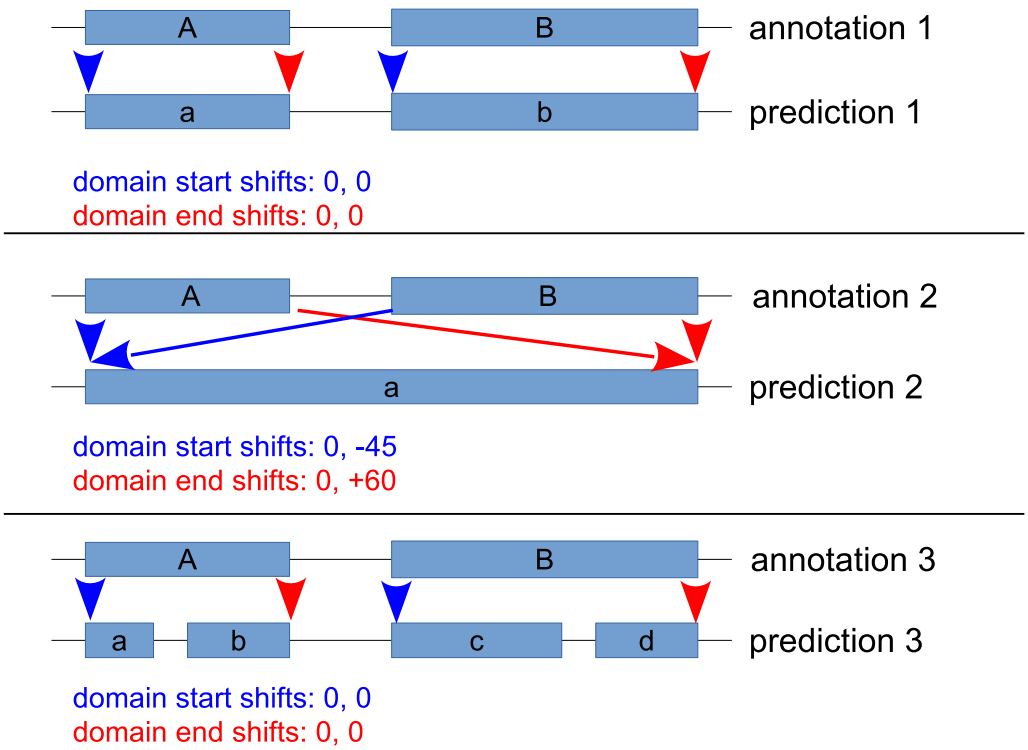

Figure 2.13: For each domain start in the reference set we calculate the distance to the closest predicted domain border in this domain boundary shift benchmark.

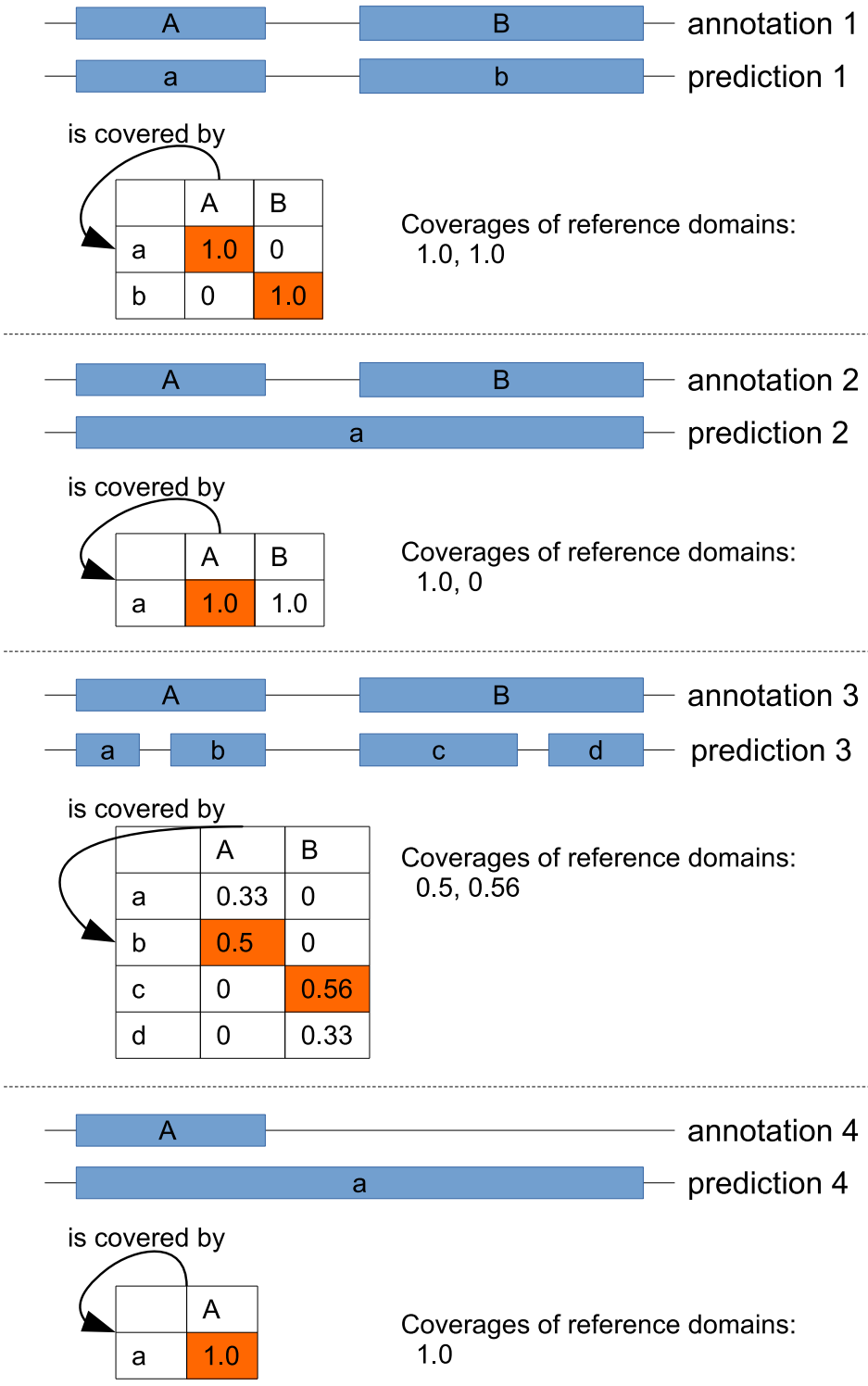

Figure 2.14: For each annotated domain in the reference set we calculate the coverage with each predicted domain. For each annotation we choose one prediction that covered the annotation best. Each prediction may be used only once.

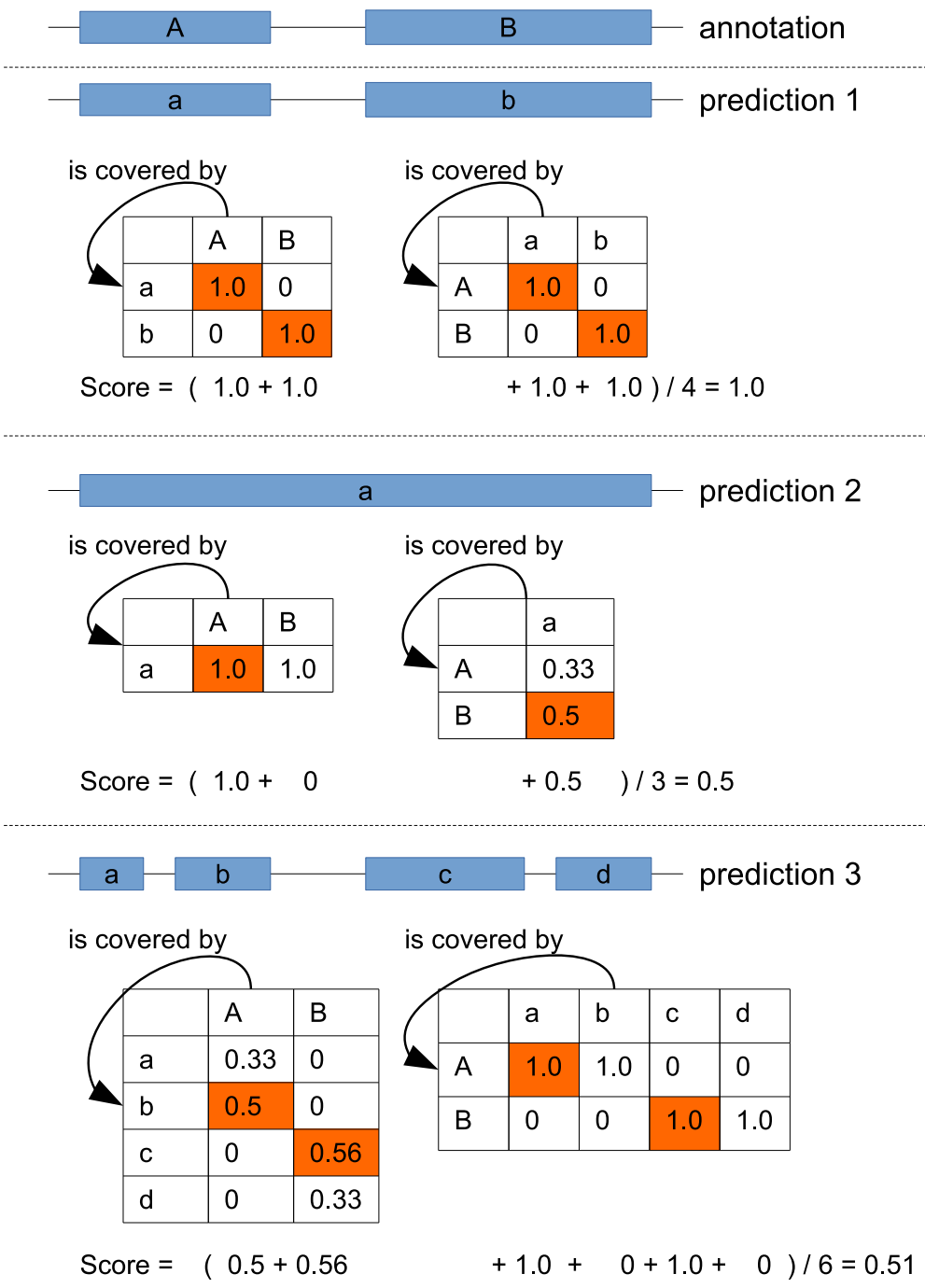

Figure 2.15: The illustrated score picks for each reference domain of the query the best covering predicted domain. No predicted domain may be used to cover two reference domains. The same is done the other way around. The final score is the mean coverage of all references and predictions.

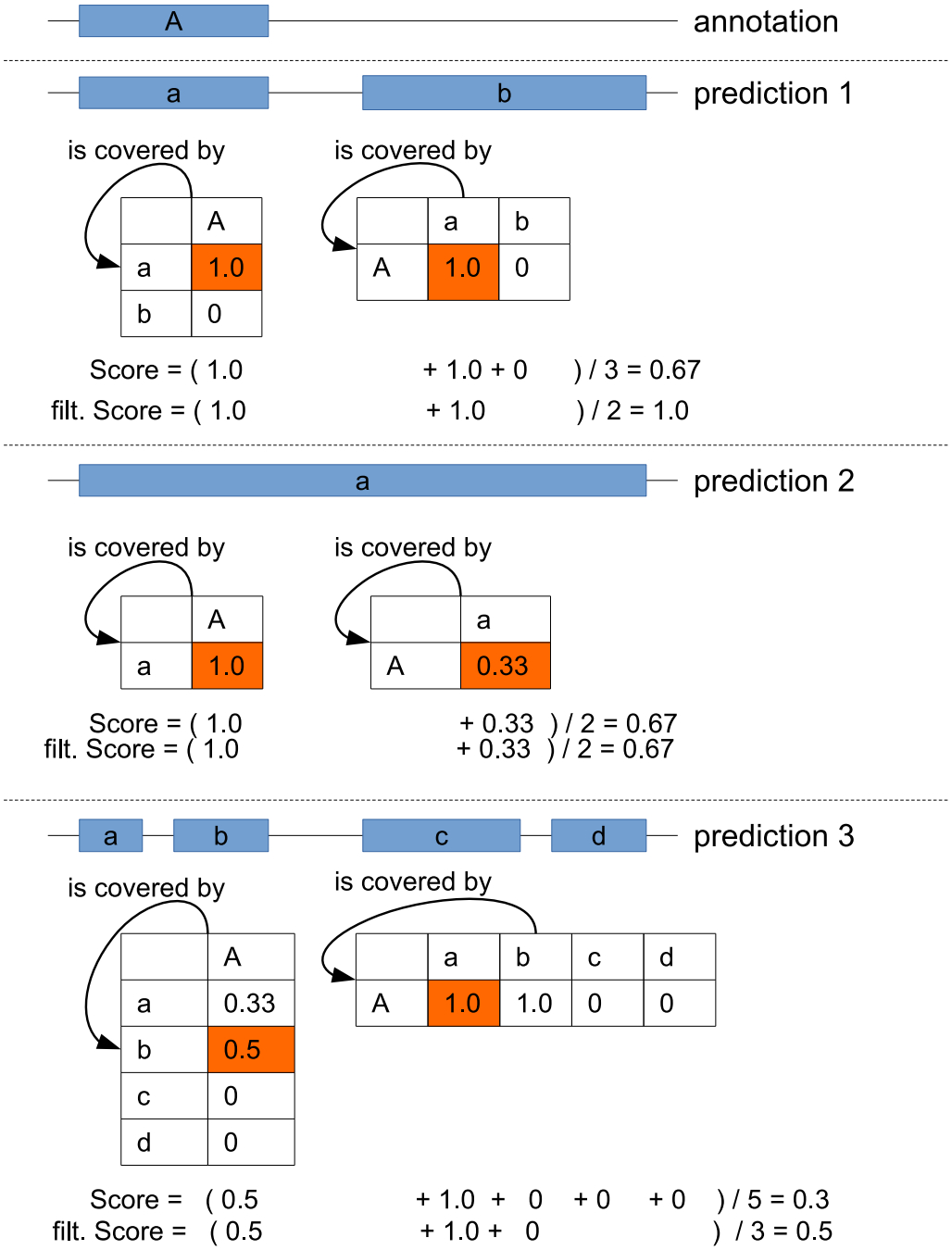

Figure 2.16: The score shown in figure 2.15 assumes that all reference domains for the query are known. Therefore, we introduced a filtered score that fixes this disadvantage. We include only the coverage of predictions in the statistic that could theoretically be covered by an annotation above a threshold of 10%.

The score assumes a perfect reference annotation, in which every domain of every protein is accurately annotated. Therefore, we introduce a filtered score that fixes this disadvantage. We include only the coverage of predictions in the mean coverage that can be covered by an annotation above a threshold of 10% (see figure 2.16).

This score penalizes too long predictions and too short predictions, therefore it is suitable for optimization.

2.4.3 Parameter Optimization

Alignment Parameters of HHblits The alignment parameters of HHblits were optimized to generate accurate final alignments, but the intermediate alignment start and end distributions used by Pdom were never directly optimized.

We focused on the following three parameters of HHblits:

- shift: profile-profile score offset
- gapf: gap open penalty for deletes
- gapg: gap open penalty for inserts

For our optimization we assumed the same setting for the gap open penalty for deletes and inserts.

Template Weighting Parameter α The different alignments that lead to a domain prediction might contain redundant information. Therefore, we weight the contribution of each alignment to the Bayes factor (see equation (B.4)). The weighting w_{qk}^{σ} is regulated by the parameter α (see section B.2.3). An α of zero means that all templates contribute equally in the Bayes factor. Higher values of α decrease the contribution of templates with similar alignment information. α might depend on the alignment parameters, therefore we optimized them together.

Results We performed a grid search for optimal parameter settings. For each pair of parameters we calculated the corresponding alignments of our optimization set with HHblits and calculated the domain predictions with Pdom.

For all tested parameter combinations of gap and shift an α of 0.0 performs best. There are two parameter combinations with similar high scores gap 0.6 with shift -1.0 and gap 0.0 with shift -1.0 (see figure 2.17). For gap 0.0 we have more alignments than for gap 0.6 (see figure 2.18A).

A statistical model can be disturbed by noise. In the case of Pdom, noise are alignments that do not start or end at actual domain boundaries. With the change in the default parameters of HHblits we might get more of them. We defined informative alignments to have a sum of alignment start probabilities ± 20 residues around an annotated domain start > 0.2 or a sum of alignment end probabilities ± 20 residues around an annotated domain end > 0.2. So we have at least a small signal for an annotated domain boundary in

the informative alignment. We calculated the fraction of informative alignments for our domain prediction with respect to the optimization reference for all alignment parameter combinations (see figure 2.18B). The fraction of informative alignments is higher for gap 0.6 with shift −1.0 than for gap 0.0 with shift −1.0.

More remote domain architectures help us to infer domain boundaries. Alignments of proteins with more remote domain architectures are more difficult, since the homologous region of the query and template might be several residues away from the N-terminal and C-terminal protein site. We calculated the mean sum of alignment start probabilities in a window of ±20 residues around annotated domain starts at least 40 residues away from the query's N-terminal site and a greater portion of the alignment start probabilities at least 40 residues away from the template's N-terminal site (see figure 2.19A). We did the corresponding statistic for difficult alignment ends (see figure 2.19B).

The parameter combination of gap 0.6 with shift −1.0 performed for difficult alignment starts and ends moderately well, whereas the parameter combination of gap 0.0 with shift −1.0 performed for difficult alignment ends well but for difficult alignment starts badly.

Therefore, we decided to use a shift of −1.0, a gap penalty of 0.6 and an alpha of 0.0. By default, HHblits uses a gap of 0.6 with a shift of −0.03.

2.5 Results

Domain Boundary Shift ADDA predicts domain start and end sites within 20 residues in 15% of the reference domain start and end sites (see figure 2.20). Pfam annotates domain start sites within 20 residues in 67% of the reference domain start sites and annotates domain end sites within 20 residues in 58% of the reference domain end sites.

The first iteration of Pdom predicts domain start sites within 20 residues in 50% of the reference domain start sites and annotates domain end sites within 20 residues in 37% of the reference domain end sites. After the second iteration predicted domain starts are within 20 residues in 50% of the reference domain start sites and predicted domain ends are within 20 residues in 45% of the reference domain end sites.

Domain Coverage The mean coverage of domains in the benchmark by predictions of ADDA was 32%, 80% by Pfam, 74% and 75% by Pdom after the first iteration and after the second iteration, respectively (see figure 2.21).

2.5.1 Analysis of Good Predictions

In table 2.1 we show several examples of good predictions generated by Pdom after the second iteration.

Q6N2W5 encompasses two SCOP domains according to our mapping (see table 2.2). The prediction of Pdom after the second iteration covers the SCOP domains almost perfectly. However, the predicted domain Q6N2W5_5 includes the long linker between the two SCOP

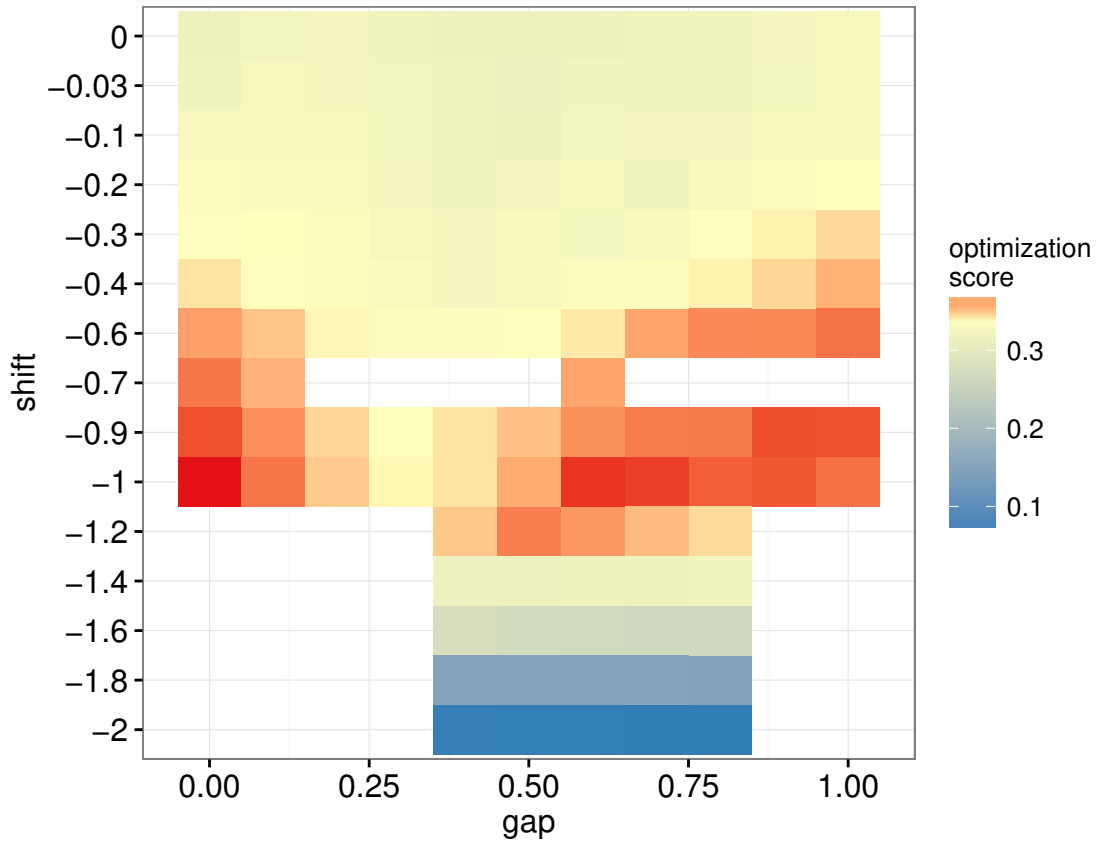

Figure 2.17: The filtered optimization score for domain predictions for different alignment parameter settings of HHblits. The parameter combinations of gap penalty 0.6 with shift −1.0 and gap penalty 0.0 with shift −1.0 performed similarly well. HHblits uses by default the gap penalty 0.6 and shift −0.03.

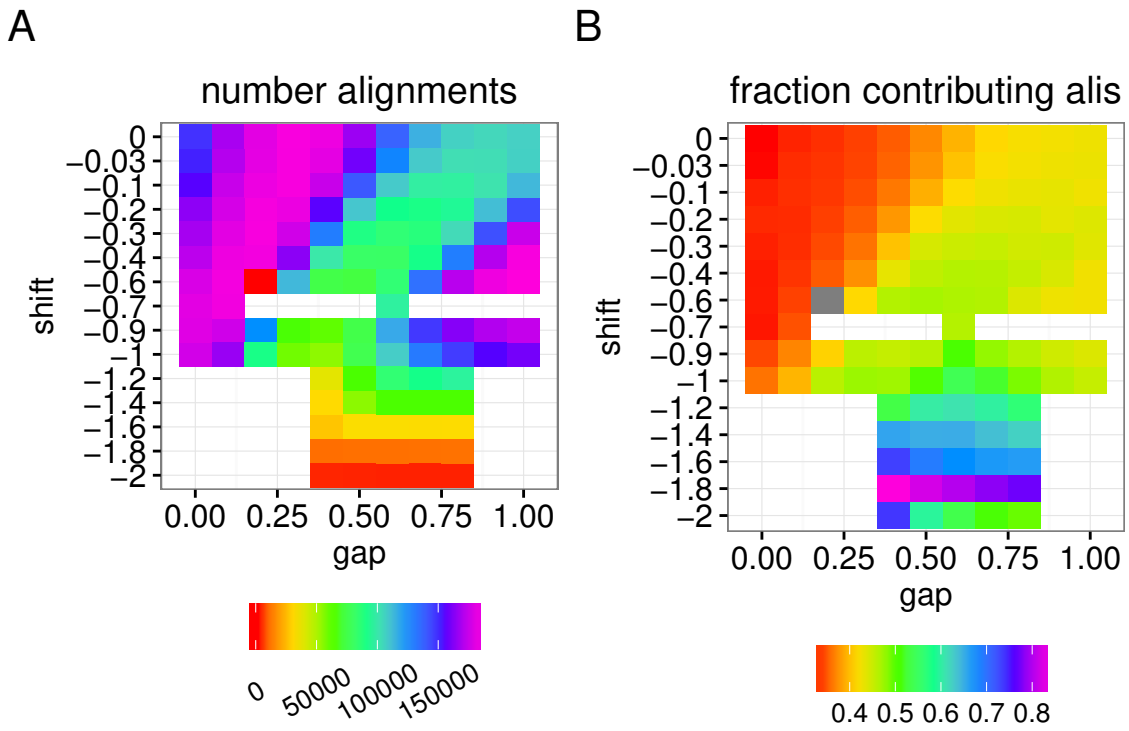

Figure 2.18: For different alignment parameters of HHblits the number of alignments (A) and the fraction of contributing alignments (B). A statistical model can be disturbed by noise. In the case of Pdom noise are alignments that do not start or end at actual domain boundaries. They do not contribute to the prediction. We identified two optimal parameter combinations with respect to the optimization score (see figure 2.17). With respect to the ratio of contributing alignments the gap penalty of 0.6 with shift -1.0 performs better than the gap penalty of 0.0 with shift -1.0.

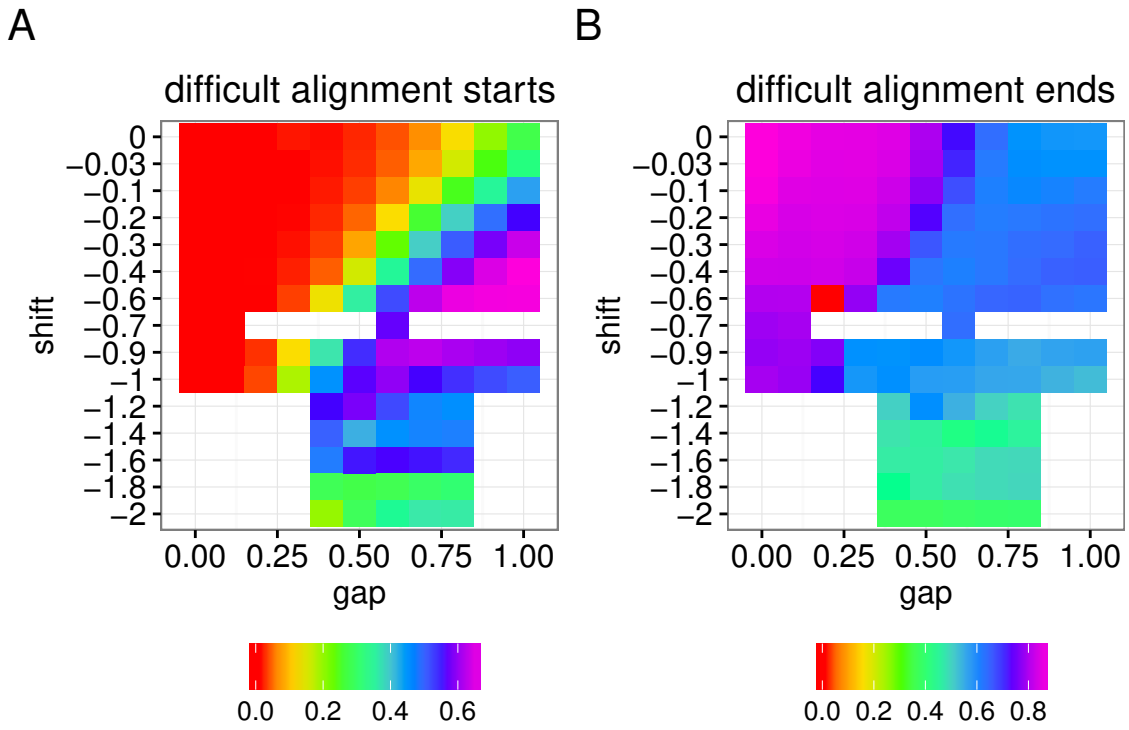

Figure 2.19: More remote domain architectures help us to infer domain boundaries. Alignments of proteins with more remote domain architectures are more difficult, since the homologous region of the query and template might be far off the for alignments trivial N-terminal and C-terminal protein site. For different alignment parameters of HHblits we calculated in (A) the mean of the summed alignment start probabilities at least 40 residues away from the N-terminal site and ±20 residues around an annotated domain start at least 40 residues away from the query's N-terminal site. In (B) we calculated the mean of the summed alignment end probabilities at least 40 residues away from the template's C-terminus and ±20 residues around an annotated domain end at least 40 residues away from the query's C-terminus. We identified two optimal parameter combinations with respect to the optimization score (see figure 2.17). With respect to the alignment quality of proteins with more remote domain architectures the gap penalty 0.6 with shift −1.0 performs better than the gap penalty 0.0 with shift −1.0.

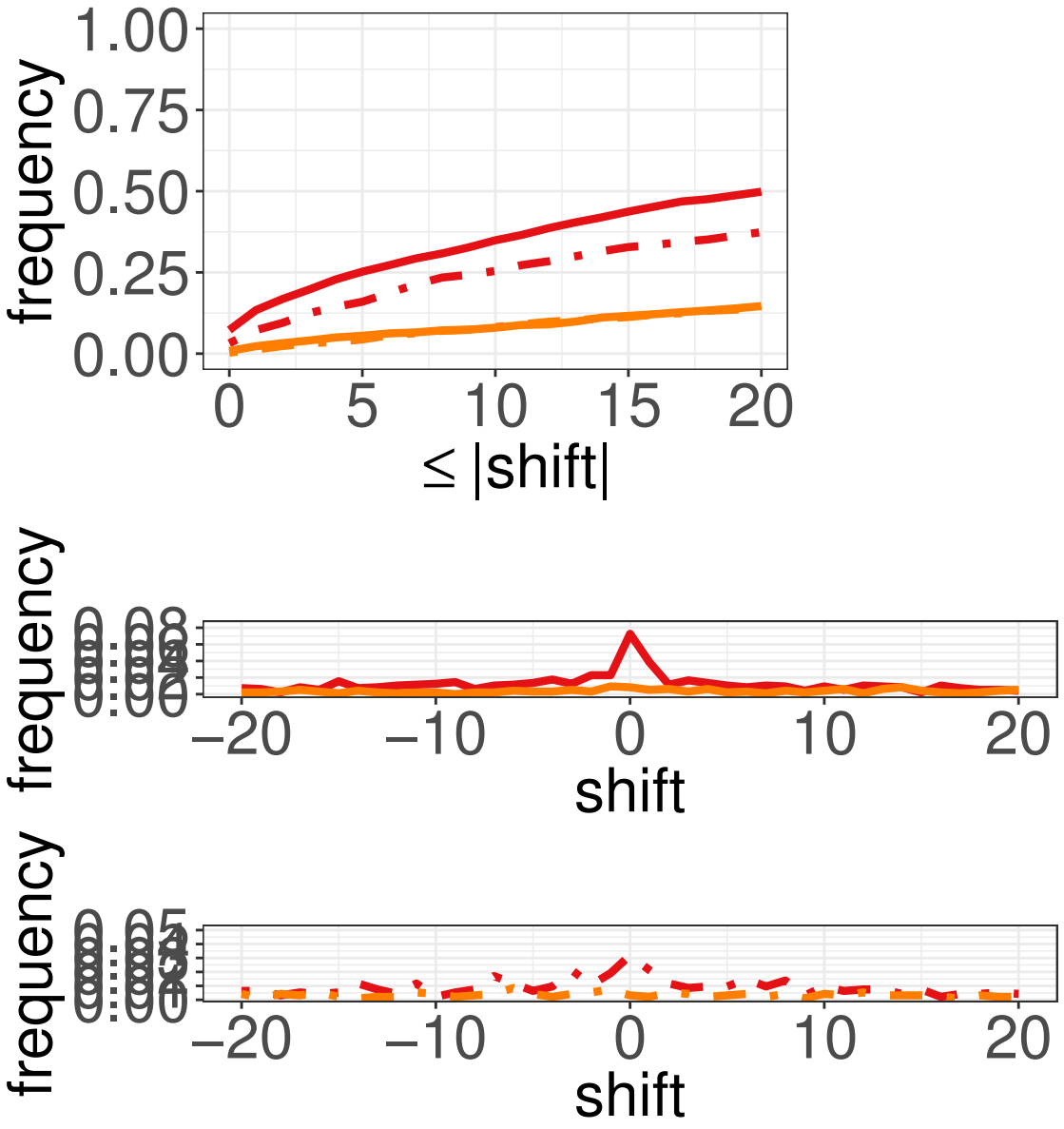

Figure 2.20: Domain boundary shift benchmark for benchmark domains with starts and ends at least 40 residues away from the corresponding protein's C- and N-terminus for the domain predictors ADDA, Pfam and Pdom.

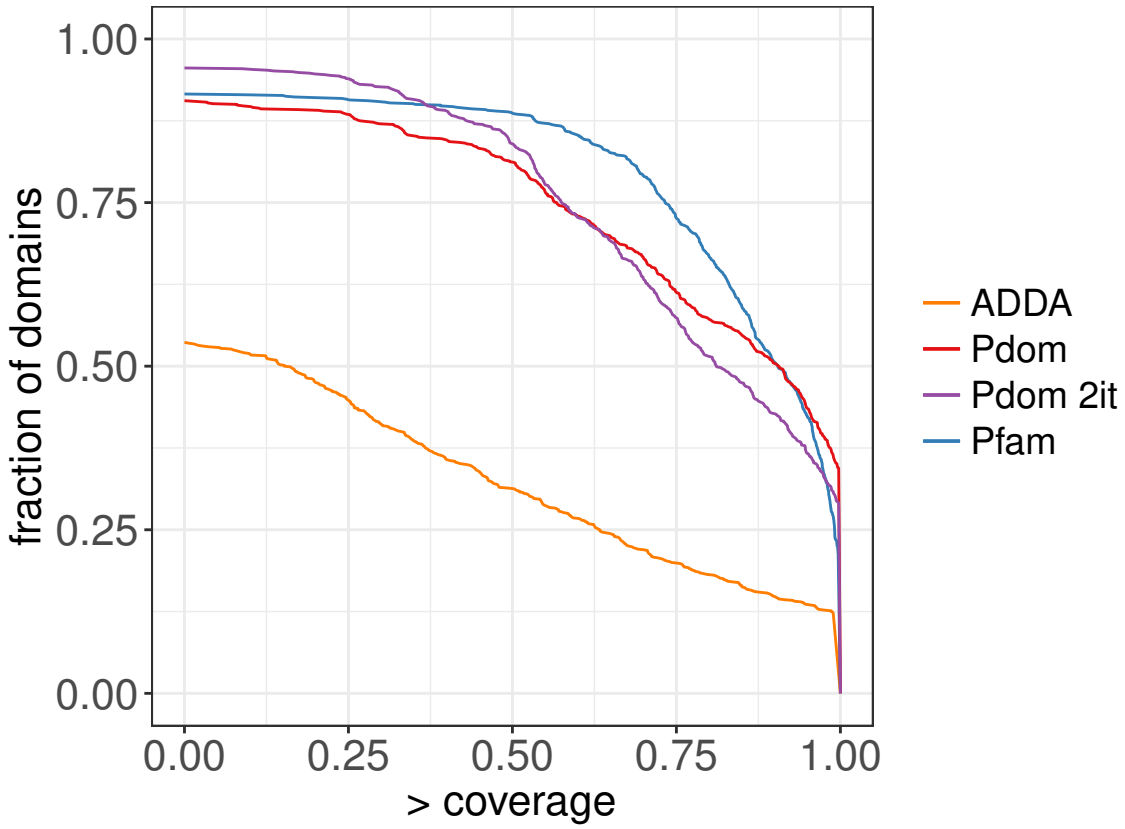

Figure 2.21: Coverage benchmark for the domain predictors ADDA, Pfam and Pdom. Despite the manually ensured quality of Pfam the automatic predictions of Pdom are quite close in this benchmark.

protein	domain	source	family	borders
Q9HJM0	d1f0ya2	SCOP	c.2.1.6	3-216
Q9HJM0	d1f0ya1	SCOP	a.100.1.3	219-314
Q9HJM0	Q9HJM0_1	Pdom 2nd iteration		1-220
Q9HJM0	Q9HJM0_2	Pdom 2nd iteration		221-314
Q7D0H0	d1ixca1	SCOP	a.4.5.37	4-76
Q7D0H0	d1i6aa_	SCOP	c.94.1.1	93-289
Q7D0H0	Q7D0H0_1	Pdom 2nd iteration		4-75
Q7D0H0	Q7D0H0_2	Pdom 2nd iteration		78-295
Q8KEN1	d1mb3a_	SCOP	c.23.1.1	6-123
Q8KEN1	d1gxqa_	SCOP	a.4.6.1	131-226
Q8KEN1	Q8KEN1_1	Pdom 2nd iteration		5-114
Q8KEN1	Q8KEN1_2	Pdom 2nd iteration		133-227
Q7NMA6	d1mb3a_	SCOP	c.23.1.1	14-129
Q7NMA6	d1gxqa_	SCOP	a.4.6.1	143-234
Q7NMA6	Q7NMA6_1	Pdom 2nd iteration		13-119
Q7NMA6	Q7NMA6_2	Pdom 2nd iteration		139-234
Q8TJN6	d1vmea2	SCOP	d.157.1.3	38-256
Q8TJN6	d1e5da1	SCOP	c.23.5.1	260-403
Q8TJN6	Q8TJN6_1	Pdom 2nd iteration		1-258
Q8TJN6	Q8TJN6_2	Pdom 2nd iteration		259-404

Table 2.1: Examples for good Pdom predictions and their SCOP annotation

		SCOP	
protein	**domain**	**family**	**borders**
Q6N2W5	d1bxda_	d.122.1.3	476-580
Q6N2W5	d1a2oa1	c.23.1.1	645-745
		Pdom 2nd iteration	
Q6N2W5	Q6N2W5_1		1-254
Q6N2W5	Q6N2W5_2		255-362
Q6N2W5	Q6N2W5_3		363-470
Q6N2W5	Q6N2W5_4		471-582
Q6N2W5	Q6N2W5_5		587-751
Q6N2W5	Q6N2W5_6		754-1021
Pdom 2nd iteration with uniform linker length distribution			
Q6N2W5	Q6N2W5_1		207-314
Q6N2W5	Q6N2W5_2		471-581
Q6N2W5	Q6N2W5_3		643-751
Q6N2W5	Q6N2W5_4		914-1021

Table 2.2: SCOP annotation and Pdom prediction for protein Q6N2W5

domains. This behavior could not be explained by the alignments. Therefore, it seems to be a problem with the linker length distribution. We assumed a uniform linker length distribution In a simple test and generated a much better prediction by Pdom (see table 2.2). The linker length distribution is an estimation at the moment (see section 2.4.1). We will train the linker length distribution from the linker lengths of the prediction after the second iteration. For this purpose, an initial relaxed linker length distribution should be applied, so linkers can be introduced but not enforced.

2.5.2 Analysis of Bad Predictions

Bad Predictions in the 1st Iteration of Pdom

Q9KEQ2 encompasses two SCOP domain annotations (see table 2.22). Pdom predicted one domain partially overlapping both SCOP domains. No calculated alignment included a complete domain (see figure 2.23). For a higher shift (-0.03) there is more noise for the domain start and end probabilities, but there are also full length domain alignments (see figure 2.24). During the parameter optimization for HHblits (shift and gap penalty) we improved the overall prediction accuracy with Pdom, however for some queries we decreased the accuracy.

In this case, the resulting alignment starts and ends indicate fragments of complete domains. Queries that are only aligned with domain fragments cannot be handled during the runtime of Pdom. If we want to avoid those cases, we have to do it before Pdom. If they are detectable, those queries could be recalculated with different parameter settings for HHblits. Perhaps, we can select optimal parameters for the query with a machine learning

SCOP			
protein	**domain**	**family**	**borders**
Q9TYQ8	d1khba2	c.109.1.1	45-303
Q9TYQ8	d1khba1	c.91.1.1	304-620
Pdom 1st iteration			
Q9TYQ8	Q9TYQ8_1		54-171
Q9TYQ8	Q9TYQ8_2		172-290
Q9TYQ8	Q9TYQ8_3		310-384
Q9TYQ8	Q9TYQ8_4		385-449
Q9TYQ8	Q9TYQ8_5		450-526
Q9TYQ8	Q9TYQ8_6		531-613

Table 2.3: SCOP annotation and Pdom prediction for protein Q9TYQ8

approach like a neural network depending on features of the multiple sequence alignment of the query.

Q9TYQ8 encompasses two SCOP domain annotations (see table 2.3). Both SCOP domains were derived from the protein 1khb in the PDB. Pdom predicts six domains. Those splits have no structural foundation (see figure 2.26 and figure 2.27). In this case, noisy alignment start and end signals (see figure 2.25) led to wrong domain predictions.

For the prediction of the domains of one query we include the alignments to different templates. Those alignments may contain conflicting information. It might be beneficial to introduce some kind of outlier detection in Pdom to prohibit alignments that encompass truncated domains to disturb the prediction. This outlier detection is a challenging task, since it could be difficult to decide between single domain alignments compared to multi-domain alignments and truncated domain alignments compared to single domain alignments.

SCOP			
protein	**domain**	**family**	**borders**
Q9KEQ2	d1xc3a1	c.55.1.10	4-121
Q9KEQ2	d1z05a2	c.55.1.10	129-292
	Pdom 1st iteration		
Q9KEQ2	Q9KEQ2_1		59-244

Figure 2.22: SCOP annotation and Pdom prediction for protein Q9KEQ2

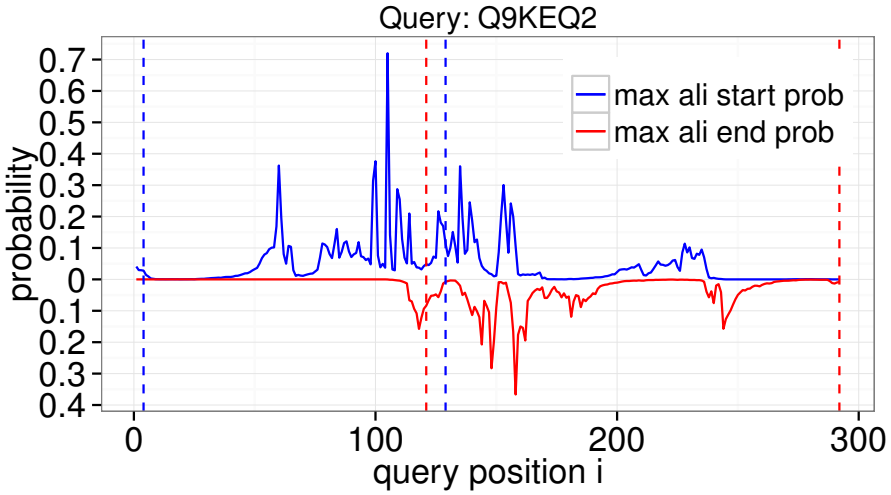

Figure 2.23: Maximum alignment start and end profile over all templates with annotated SCOP domain start and end positions (dashed lines) of protein Q9KEQ2.

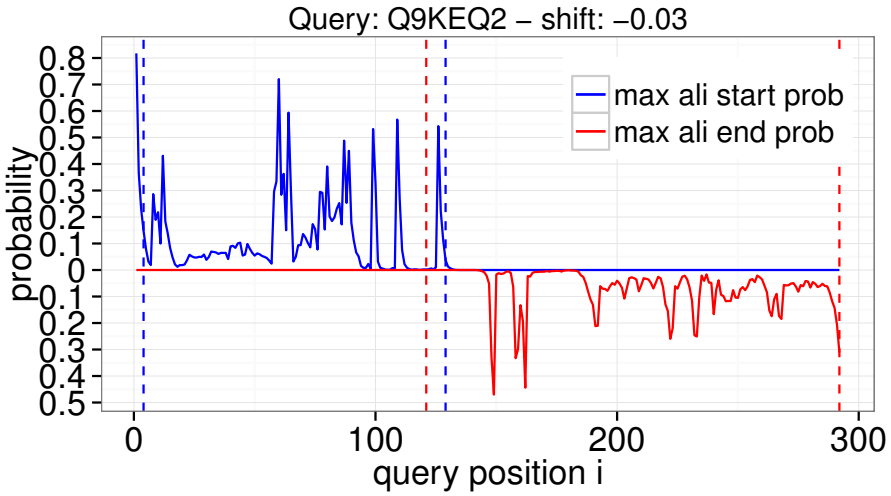

Figure 2.24: Maximum alignment start and end profile over all templates with annotated SCOP domain start and end positions (dashed lines) of protein Q9KEQ2 with lower alignment shift 0.03.

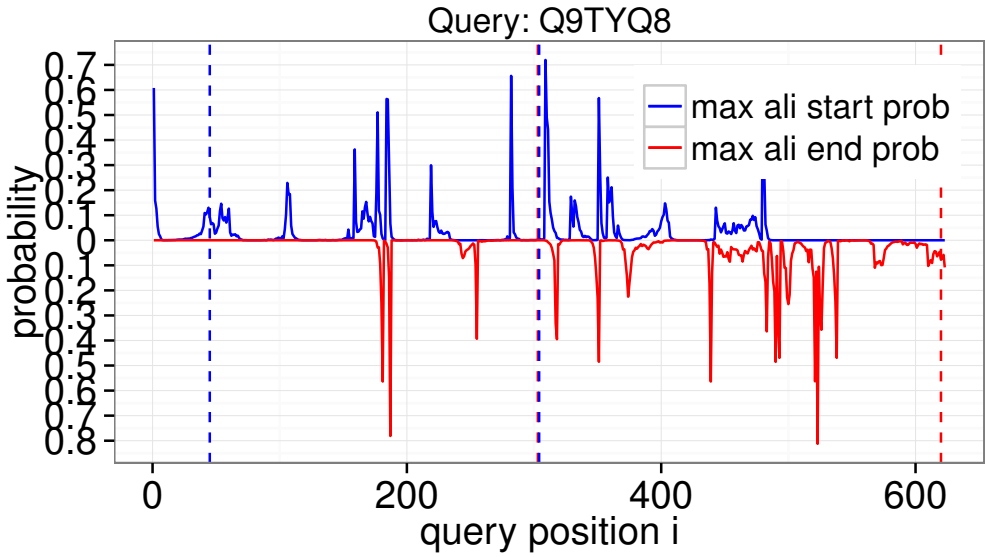

Figure 2.25: Maximum alignment start and end profile over all templates with annotated SCOP domain start and end positions (dashed lines) for protein Q9TYQ8.

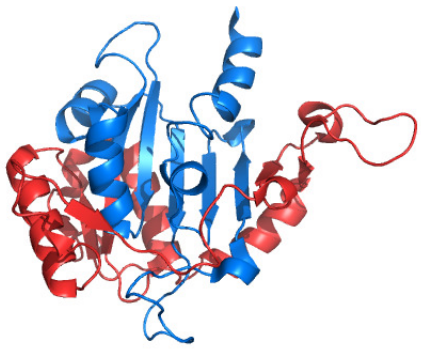

Figure 2.26: Split of domain 1khba2 according to the domain prediction of Pdom in the 1st iteration for the protein Q9TYQ8

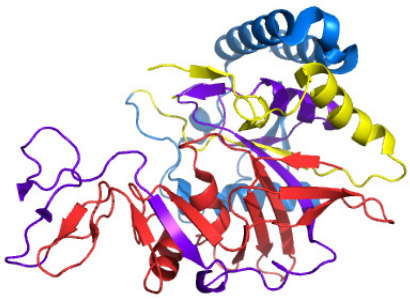

Figure 2.27: Split of domain 1khba1 according to the domain prediction of Pdom in the 1st iteration for the protein Q9TYQ8

	SCOP		
protein	**domain**	**family**	**borders**
Q7UHV2	d1umqa_	a.4.1.12	154-204
Q7UHV2	d1dbwa_	c.23.1.1	32-144
	Pdom 1st iteration		
Q7UHV2	Q7UHV2_1		75-133
Q7UHV2	Q7UHV2_2		141-221
	Pdom 2nd iteration		
Q7UHV2	Q7UHV2_1		75-138

Table 2.4: SCOP annotation and Pdom prediction for protein Q7UHV2

Bad Cases in the 2nd Iteration of Pdom

Q7UHV2 has two domains according to the SCOP mapping (see table 2.4).

After the first iteration Pdom predicts both domains quite accurately. After the second iteration the prediction of the second domain is completely missing.

The maximum alignment start and end profile over all templates for this query show that there is no alignment encompassing the second domain (see figure 2.28). We use r^*_{qk} in the consistency iterations to hide query positions that are not covered by alignments (see section 2.3.4). r^*_{qk} prohibits the inference of the second domain. The suppression by r^*_{qk} is in those cases too strong.

Q82BT2 and Q9VC63 each encompass two instances of domains from the SCOP family c.37.1.12. After the first iteration of Pdom those domains were completely covered. In the second iteration of Pdom those domains were split. We mapped the splits of the domains to the corresponding structures (see figures 2.31, 2.32, 2.33 and 2.34).

For one of those structures there is a domain prediction at the PDB by Domain Parser that suggests an inserted domain in the region where our splits occur [66]. We did not find evidence for this inserted domain in the alignments, that means there was no alignment starting and ending at the suggested inserted domain borders.

All alignments, except of one, have alignment start and end probability peaks at the annotated domain start and end sites or outside the annotated domain. Therefore, there was no domain boundary predicted within the annotated domain after the first iteration of Pdom. After the second iteration there were several peaks in the distributions of $r^{\lfloor\sigma\rfloor}_{qk}$ (see section section 2.3.4) around the position of the single wrong alignment start. Those peaks were transferred from homologous domains that found the same or similar truncated templates, even if they did not predict a domain border at those positions. Through the accumulated wrong signal from different templates the annotated domains were split after the second iteration of Pdom. The sequences of those multiple sequence alignments were similar in sequence, but differ in length at the N-terminal site (e.g. cluster VAVPUCEBA, see figure 2.35). According to the annotation at the GenBank [13] those sequences were not complete.

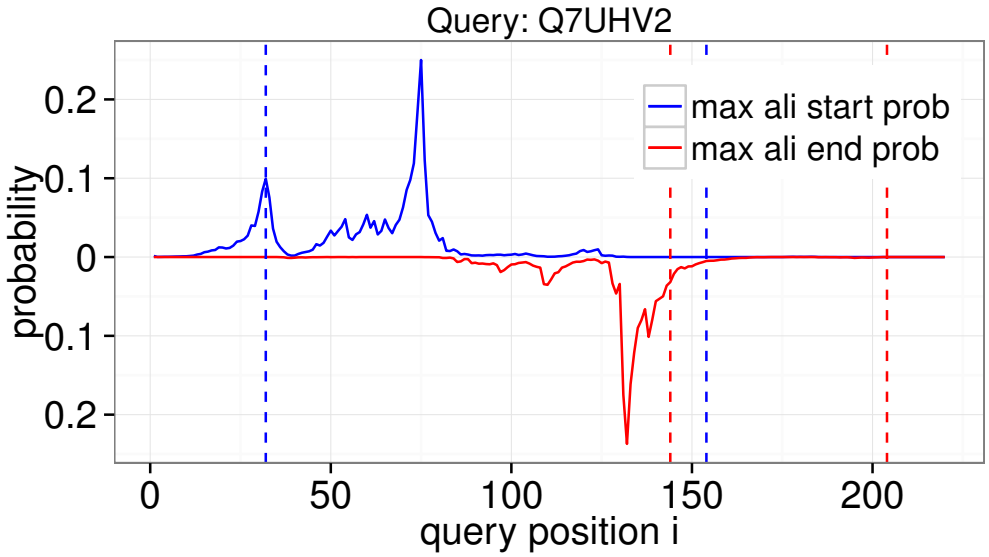

Figure 2.28: Maximum alignment start and end profile over all templates with annotated SCOP domain start and end positions (dashed lines)

The clusters of truncated sequences can impact the predictions of Pdom in the consistency iterations. Therefore, it is useful to build a uniprot20 excluding truncated sequences for Pdom. As of August 2016, about 10% of the sequences in the UniProt are truncated. From this database a new diversity-enriched database may be recalculated. This requires that the annotation of segmented sequences in the UniProt is complete. For the clustering of the uniprot20 we require the sequences within a cluster to cover each other by at least 90%. Truncated sequences can be clustered to their complete homologs using an asymmetrical condition that the longer sequence covers the shorter sequence by at least 90%.

Q7QG29 encompasses three SCOP domains. The prediction for the first domain was split into two parts in both iterations of Pdom. We transferred the split on the structure of d1gz8a_ (see figure 2.36). CATH split the SCOP domain d1gz8a_ on the structure 1gz8 in the PDB. The additional domain border of CATH was close to ours. We analyzed the alignments that caused the additional domain border. The three causing templates were probably not truncated. Therefore, the prediction of Pdom might be accurate and the SCOP annotation could be wrong. SCOP classifies domains if they see structural subunits reoccurring in different structures with different domain architectures. Therefore, if two domains appear always together in the scarce structural data of the PDB, they will be classified as one domain in SCOP.

The predicted split of domain d1nz6a_ in the second iteration of Pdom was caused by the templates HEWKELABA. Most sequences in this multiple sequence alignment were annotated in the UniProt to be truncated.

SCOP			
protein	**domain**	**family**	**borders**
Q82BT2	d1l7vc_	c.37.1.12	45-228
Q82BT2	d1jj7a_	c.37.1.12	313-525
	Pdom 1st iteration		
Q82BT2	Q82BT2_1		1-311
Q82BT2	Q82BT2_2		312-521
	Pdom 2nd iteration		
Q82BT2	Q82BT2_1		1-151
Q82BT2	Q82BT2_2		152-247
Q82BT2	Q82BT2_3		294-410
Q82BT2	Q82BT2_4		411-521

Figure 2.29: SCOP annotation and Pdom prediction for protein Q82BT2

SCOP			
protein	**domain**	**family**	**borders**
Q9VC63	d1oxxk2	c.37.1.12	513-712
Q9VC63	d1ji0a_	c.37.1.12	1154-1301
	Pdom 1st iteration		
Q9VC63	Q9VC63_1		1-278
Q9VC63	Q9VC63_2		279-386
Q9VC63	Q9VC63_3		387-494
Q9VC63	Q9VC63_4		495-720
Q9VC63	Q9VC63_5		726-920
Q9VC63	Q9VC63_6		921-1028
Q9VC63	Q9VC63_7		1029-1136
Q9VC63	Q9VC63_8		1137-1365
	Pdom 2nd iteration		
Q9VC63	Q9VC63_1		1-278
Q9VC63	Q9VC63_2		279-386
Q9VC63	Q9VC63_3		387-494
Q9VC63	Q9VC63_4		495-623
Q9VC63	Q9VC63_5		624-711
Q9VC63	Q9VC63_6		713-920
Q9VC63	Q9VC63_7		921-1028
Q9VC63	Q9VC63_8		1029-1136
Q9VC63	Q9VC63_9		1137-1251
Q9VC63	Q9VC63_10		1252-1365

Figure 2.30: SCOP annotation and Pdom prediction for protein Q9VC63

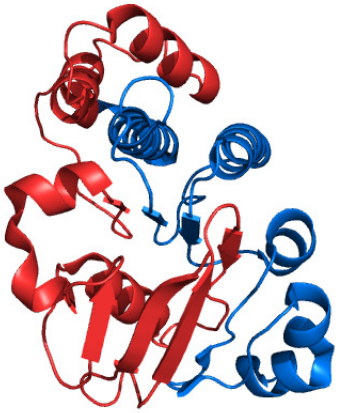

Figure 2.31: Split of domain d1l7vc_ according to the domain prediction of Pdom in the 2nd iteration for the protein Q82BT2

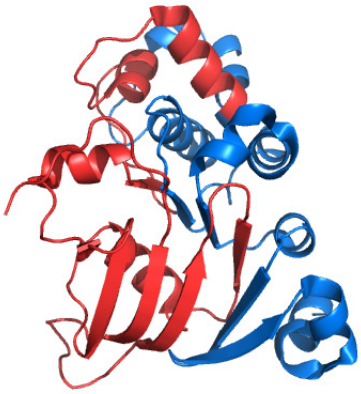

Figure 2.32: Split of domain d1jj7a_ according to the domain prediction of Pdom in the 2nd iteration for the protein Q82BT2

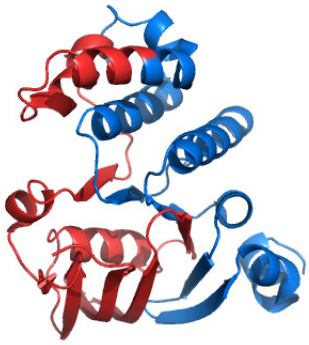

Figure 2.33: Split of domain d1ji0a_ according to the domain prediction of Pdom in the 2nd iteration for the protein Q9VC63

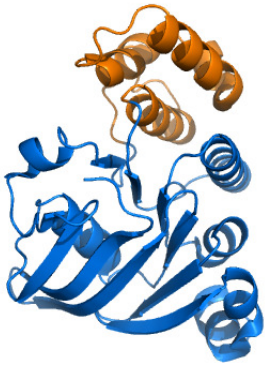

Figure 2.34: Split of domain d1ji0a_ according to DomainParser

```
#UP20|VAVPUCEBA|11|438 Microcin C ABC transporter ATP-binding protein
>tr|C9NV91|C9NV91_9VIBR ABC transporter ATP-binding protein
-----MAEAIMCHRPVSRSQAKQKVLELFNLVHLPNPEQAYTKYPHEFSGGQLQRIMIAMALINEPDILIADEPTTALDVTVQAEVLNLI...
>tr|Q7OIN2|Q7OIN2_9PSED ATP binding protein
----------mtVHGVARHDAGLRAVALLHQVGLKDPEQLLGRYPHELSGGMCQRVLIAIALANDPPLLIADEPTSALDVSVQRQILD...
>tr|F3DBL3|F3DBL3_9PSED Peptide ABC transporter ATP-binding protein
SVARQIGETLLLHRGISGREAQKRIVELLEMVGIQQPEKRLKAYPHELSGGQRQRVMIAMALACEPELLVADEPTTALDVTVQRKILLLL...
>tr|E3F447|E3F447_KETVY ABC peptide transporter, fused ATPase domains
----------------------MVELLHQVGIRDPESRLGAYPHQLSGGQRQRVMIAMALSNDPKLLIADEPTTALDVTIQAQILELL...
>tr|D7HZG1|D7HZG1_PSESS Peptide ABC transporter, ATP-binding protein
----------------------MEMVGIQQPEKRLKAYPHELSGGQRQRVMIAMALACEPELLVADEPTTALDVTVQRKILLLL...
>tr|F3DJ51|F3DJ51_9PSED ABC transporter
-------------------------------------------------MIAIALAGNPRLIIADEPTSALDVTVQRKILDHL...
>tr|G5LBN0|G5LBN0_SALET Dipeptide transport ATP-binding protein DppF
-------------------------------------------------MCXXRQRVMIAMALLTRPELLIADEPTTALDVSVQAQILSLL...
>tr|G5FYY4|G5FYY4_9PSED Putative uncharacterized protein
-----------------------------------------------------MLIADEPTTALDVTVQKRLLELL...
>tr|D4DTU5|D4DTU5_NEIEG Putative uncharacterized protein
-------------------------------------------------MIAMAVAAEPELLIADEPTTALDVAVQAQILDLL...
>tr|F3E1W1|F3E1W1_9PSED Microcin C ABC transporter ATP-binding protein YejF
-----------------------------------------------------MADEPTTALDVTVQLKILELL...
>tr|F3HW54|F3HW54_PSESF Microcin C ABC transporter ATP-binding protein YejF
-----------------------------------------------------PTTALDVTVQLKILELL...
```

Figure 2.35: Dubious cluster VAVPUCEBA of the uniprot20 HHsuite database; The sequences are probably truncated at the N-terminus; F3DBL3 (CAA56798.1; GeneBank) is annotated as incomplete

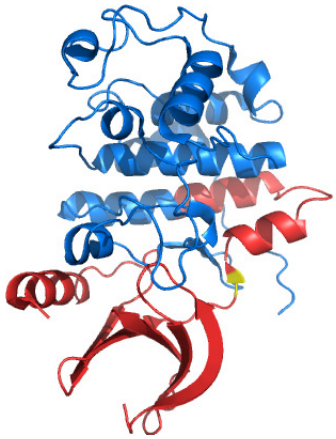

Figure 2.36: Split of domain d1gz8a_ according to the domain prediction of Pdom in the 2nd iteration for the protein Q7QG29; The yellow residue indicates the domain split by CATH

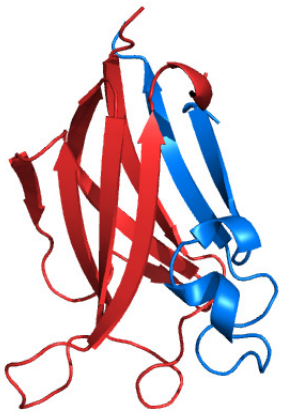

Figure 2.37: Split of domain d1d5ra2 according to the domain prediction of Pdom in the 2nd iteration for the protein Q7QG29

The predicted split of domain d1d5ra2 in the second iteration of Pdom was caused by the templates XATCUWABA and DORLEHABA. The sequence H2P0X7 was aligned over the complete length of cluster XATCUWABA and is annotated to be a fragment. DORLEHABA consists of sequences annotated to be truncated. Therefore, this split might also be caused by truncated clusters . However, the split on the structure of d1d5ra2 might show a C-terminal decorator (blue structure in figure 2.37).

Truncated sequence clusters can obscure the domain prediction with Pdom especially during the consistency iterations. This problem can be handled as as described in the previous paragraph (Q82BT2 and Q9VC63).

2.5.3 Web server

We developed a django web server [2] for Pdom with the final goal to offer users a comprehensive atlas of our predicted domains (see `http://141.5.103.163/`). At the moment users can investigate the predicted domains of the benchmark set and the underlying HHblits alignments (see figure 2.38). In the future we want to offer the option to search with queries through our domain atlas or to predict domains from scratch for single queries.

The web server uses MySQL [6] as the back-end database. It distributes jobs with Celery [1]. The job management is organized with a Redis database [7]. The server is hosted in the cloud of the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen [3].

	SCOP		
Q7QG29	d1gz8a_	d.144.1.7	42-263
Q7QG29	d1d5ra2	c.45.1.1	462-631
Q7QG29	d1nz6a_	a.2.3.1	1188-1277
	Pdom 1st iteration		
Q7QG29	Q7QG29_1		36-147
Q7QG29	Q7QG29_2		148-257
Q7QG29	Q7QG29_3		270-461
Q7QG29	Q7QG29_4		462-648
Q7QG29	Q7QG29_5		649-759
Q7QG29	Q7QG29_6		760-867
Q7QG29	Q7QG29_7		868-975
Q7QG29	Q7QG29_8		976-1083
Q7QG29	Q7QG29_9		1084-1192
Q7QG29	Q7QG29_10		1193-1287
	Pdom 2nd iteration		
Q7QG29	Q7QG29_1		36-147
Q7QG29	Q7QG29_2		148-259
Q7QG29	Q7QG29_3		260-461
Q7QG29	Q7QG29_4		462-584
Q7QG29	Q7QG29_5		585-629
Q7QG29	Q7QG29_6		631-724
Q7QG29	Q7QG29_7		725-771
Q7QG29	Q7QG29_8		777-911
Q7QG29	Q7QG29_9		912-1019
Q7QG29	Q7QG29_10		1020-1127
Q7QG29	Q7QG29_11		1128-1235
Q7QG29	Q7QG29_12		1236-1279

Table 2.5: SCOP annotation and Pdom prediction for protein Q7QG29

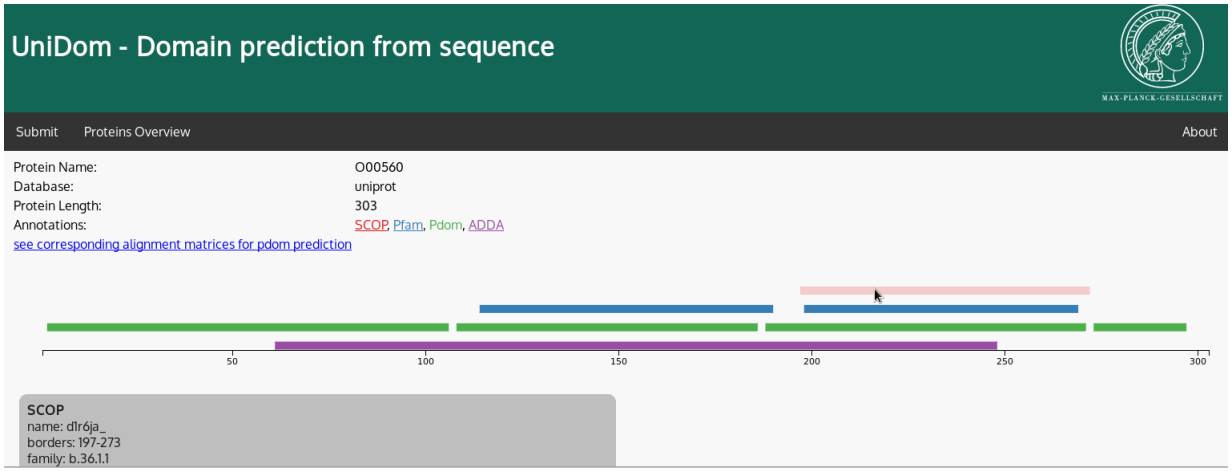

Figure 2.38: The illustration of the predicted domains for protein O00560 in the Pdom webserver

2.6 Discussion and Conclusion

There are three automatic predictors for protein domain boundaries that relied on all-against-all alignments, namely ADDA, DoBo and EVEREST. DoBo was only available as a webtool that is not designed for the complete decomposition of the protein sequence space. EVEREST reconciles its domains to known protein families from SCOP and Pfam, therefore we cannot benchmark EVEREST as a de-novo domain predictor. ADDA was used by Pfam to build the automatically generated Pfam-B database. ADDA used BLAST to build the all-against-all alignments. With the release of Pfam 29.0 the maintenance of the Pfam-B was stopped.

Pfam-A encompasses multiple sequence alignments of manually curated seed alignments. Those alignments are built on the basis of evolutionary, structural and functional knowledge.

Our automatic Bayesian domain predictor Pdom worked on the basis of the all-against-all alignments by HHblits within the clustered UniProt. The only prior knowledge we used was the domain and linker length distributions from annotated domains in SCOP.

ADDA predicted domain start and end sites within 20 residues in 15% of the reference domain start and end sites in our benchmark set. In PfamA the annotated domain start sites were within 20 residues in 67% of the reference domain start sites, the annotated domain end sites were within 20 residues in 58% for the reference domain end sites. The first iteration of Pdom predicted domain start and end sites within 20 residues of the reference domain start and end sites in 50% and 37%, respectively. In the second iteration of Pdom the predicted domain start and end sites were within 20 residues of the reference domain start and end sites in 50% and 45%, respectively.

The mean coverage of domains in the benchmark was 32% by predictions of ADDA, 80% by annotations of Pfam, 74% and 75% by predictions of Pdom in the first iteration and in the second iteration, respectively.

We showed that our predictions corresponded very well to the structural domain

definition of SCOP. The predictions of Pdom might be sufficiently accurate to predict domains for the analysis of yet undiscovered domains.

We observed several challenges in the domain decomposition by Pdom. There were presumably incompletely sequenced or falsely predicted protein sequences that clustered to incomplete multiple sequence alignments in the uniprot20 HHsuite database. The late start or early stop of those clusters leads to truncated alignments. If the truncation lied within a domain, we might infer wrong domain start and end sites. That had a huge impact for further iterations by Pdom. In the second iteration the wrong signal over all homologs accumulate and cause a wrong domain decomposition in all homologs. Therefore, it is necessary to filter the database for reliable clusters. However, we observed that for the different suggested filters there were still enough truncated clusters to cause problems in the second iteration of Pdom.

2.7 Outlook

A challenge for Pdom are incomplete sequences in the database that lead to alignments that start or end inside a domain. These wrong alignment starts and stops cause truncated domain predictions in the consistency iterations. There are several ways to handle them:

- Apply filters for clusters in the database for remove clusters dominated by truncated or invalid sequences.
- Add truncated sequences to clusters of full length sequences during the clustering of the uniprot20 HMM database
- Rebuild uniprot20 and the corresponding uniprot_boost1 only with sequences from the UniProt without sequence caution labels (example given: segments, wrong initiation site, wrong termination site)

A hierarchy of domain families based on a clustering can improve our detection of dubious clusters and offers new use cases for the resulting domain database:

- Cluster domains to domain families and hyper-families
- Merge possible domain fragments in the cluster analysis

For the publication of Pdom, we have to compare Pdom additionally against DoBo and EVEREST. Our competitors DoBo, ADDA and EVERST are not able to handle inserted domains properly, so we should find some reliable show cases for this feature.

Furthermore, we can improve the prediction of Pdom by including additional information:

- Integration of disorder prediction
- Integration of secondary structures in HHblits alignments and Pdom predictions

Chapter 3

HHsuite

The software package HHsuite offers several tools for protein homology detection and alignments, notably HHblits and HHsearch. HHblits searches iteratively through a database of profile Hidden Markov models (HMM) while updating the query HMM with homologous hits. A prefilter selects hits for the computationally more expensive Viterbi alignment algorithm. High scoring hits of the Viterbi algorithm are realigned with the Maximum Accuracy algorithm. HHblits is usually used with an all-encompassing protein database like the UniProt. HHblits is more sensitive, generates more accurate alignments and is faster than its best competitors, PSI-BLAST and HMMER3. In contrast, HHsearch searches (non-iteratively) through a database without applying a prefilter, and is accordingly more sensitive than HHblits, hence it is frequently used in the template selection for homology modeling of protein structures with protein domain databases or structure databases like SCOP, CATH or the PDB. For the reasons outlined above, the HHsuite has become essential in computational biology. I maintained the software and the building pipelines for the required HMM databases for selected, common protein databases.

3.1 Code Improvement

Different students with different goals without an overall concept contributed to HHsuite. The resulting code was stable, but inconsistent spaghetti code with intermediate hotfixes and many occurrences of the same or similar code fragments at different positions.

For the generation of the uniprot_boost1 and for the necessary calculations of Pdom, we wanted to run HHblits in parallel over many queries. Therefore, I decided to do a re-factoring for this purpose and fix additional flaws:

- Removed global variables
- Replaced pthreads with openMP in the parallelization over templates
- Included option to search through multiple databases in HHblits and HHsearch in one call

- Introduced consistent database format for HHblits and HHsearch
- Transformed HHblits, HHsearch and HHalign to classes with inheritance
- Cleaned up compilation calls in Makefiles
- Changed from Makefiles to cmake
- Introduced output logger in HHsuite
- Improved performance in the Hidden Markov profile and amino acid transition calculation
- Added feature to exclude template residues from the alignment in HHalign
- Parallelized over several queries with openMP in hhblits_omp

The code of the HHsuite is now openly available at GitHub [4], so we will be able to provide bug fixes swiftly. The project is set up under Travis CI [8] for continuous integration.

Ffindex is a tool written by Andreas Hauser. It allows to build from single files a database. This database consists of two files: the ffindex-file and the ffdata-file. The ffdata-file contains the single files concatenated and separated by a null byte. The ffindex-file contains an index for each single file where it starts in the ffdata-file and its length in bytes. The ffdata-file is usually opened with mmap. Mmap is a tool in most Unix distributions that allows to load a file or parts of a file in the virtual memory. The kernel can pre-load parts of the file depending on the usage of the accessing program. Therefore, the file handling is accelerated especially if the file access is sorted so that there is no forward and backward jumping in the ffdata-file.

ffindex_apply and ffindex_apply_mpi are binaries that take entries from an ffindex database [30], pipe them on the standard input stream to a script or another binary and catch the resulting standard output stream to a new ffindex database. It was necessary to adjust the logging in the scripts and the binaries in HHsuite to print log messages only on the standard error stream and to be able to read the input from the standard input stream.

The calculation of the column state sequences for the HHsuite databases with many cstranslate calls by ffindex_apply_mpi was too slow. Therefore, cstranslate can now process multiple sequence alignments in an ffindex database in parallel with openMP.

3.2 Bug-Fixes

We fixed several bugs in the HHsuite since the publication in 2012:

- Segmentation faults in the hash class of HHsuite with new compiler versions (gcc 4.8.1)

- Wrongly initialized sparse encoding of the Viterbi backtrace matrix
- Wrong calculation of consensus sequences with unknown residue X
- Wrong treatment of the ANY state in the alignment filter
- Memory leaks in HHblits, HHsearch and HHalign
- Incomplete initialization of alignment matrices in Align.pm for too long sequences
- Size of Viterbi matrix was not correctly re-allocated for large sequences

Since 2014, I took also care of user requests for the HHsuite. hast du fr

3.3 HHsuite Databases Pipelines

For the HHsuite, especially for HHsearch, we offer several well known protein databases also as HMM databases. We developed for each of these databases an automatic pipeline that keeps the corresponding database up to date. Until 2014 the automatic pipelines were maintained by the administrator of the toolkit webserver in Tübingen. Due to changes in the infrastructure and the HHsuite database format, those pipelines were rewritten from scratch.

Each HHsuite database consists of multiple sequence alignments, pre-calculated Hidden Markov profiles for multiple sequence alignments with many sequences generated by hhmake and column state sequences for each multiple sequence alignment generated by cstranslate (see section A.1).

3.3.1 Protein Data Bank - pdb70

The pdb70 contains multiple sequence alignments for a representative set of proteins in the Protein Data Bank [14]. It is widely used for homology modeling by HHpred [36] and other prediction servers. The sequences are taken from the SEQRES section in the PDB files and are filtered for maximum 70% pairwise sequence identity. Only sequences that cover each other by at least 90% are filtered. This filtering is done with all-against-all BLAST [9] searches. The pdb70 is updated every Wednesday morning, where new sequences are added and the 1000 oldest entries are re-calculated with the newest uniprot20 database.

In 2014 the default PDB archive format changed to a new format, mmCIF, to avoid the limits of the old punch card format. In 2015 we started to adjust our scripts (pdb2fasta.pl, hhmakemodel.pl, addss.pl) for this new format. The responsible master student, Harald Vöhringer, was supervised by me. Those scripts will be used in the future in our automatic pipeline. Additionally, the student changed the filtering of the PDB from BLAST to MMseqs [33].

3.3.2 Pfam

Pfam [26] consists of manually curated multiple sequence alignments that represent domain families. In the automatic pipeline we check every week for a new release of Pfam and build for every seed alignment a multiple sequence alignment in three iterations against the newest uniprot20 database. On those multiple sequence alignments we build an HHsuite database.

3.3.3 UniProt - uniprot20

The UniProt [11] is a comprehensive protein sequence database. Until 2012, we used kClust [31] for the clustering of this database to build our uniprot20 HHblits database. Beginning with 2012, the Ph.D. student Maria Hauser started the development of MMseqs, a faster, more robust and modular all-against-all sequence-sequence search tool. MMseqs was published in 2015. The master student Lars von den Driesch developed during his master thesis a pipeline for MMseqs to generate the needed clustering for the uniprot20. I implemented the pipeline for the generation of the multiple sequence alignments from this clustering and the following steps to build an HHsuite database.

After the building of the uniprot20, we can evaluate its performance in almost automatic pipelines that evaluate the homology detection sensitivity in the ROCX benchmark and the alignment quality with HHblits.

3.3.4 UniProt - uniprot_boost1

Given the uniprot20, we developed a pipeline to generate the corresponding uniprot_boost1 database. The estimated runtime on the newly clustered uniprot20 database with about 10^7 clusters is $195 \cdot 10^3$ CPU hours. So far we have not calculated this database again, but worked instead on a faster build pipeline with iterative profile-profile searches with MMseqs.

Appendix A

Supplementary - uniprot_boost1

A.1 Discretized Column States of the uniprot_boost1 for the uniprot20 Database

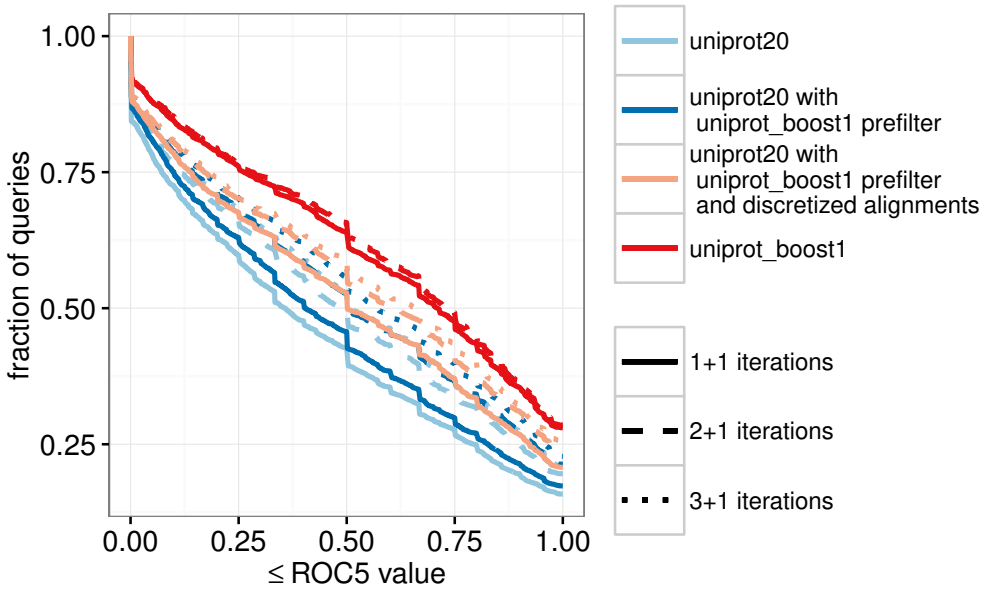

Figure A.1: ROC5 plot to measure the the homology detection sensitivity of HHblits with the uniprot20, the uniprot_boost1, the uniprot20 with the discretized column states of the uniprot_boost1 for the prefilter and the uniprot20 with the discretized column states of the uniprot_boost1 for the prefilter and the column scores.

HHblits is an iterative protein homology detection tool that is more sensitive, generates more accurate alignments and is faster than PSI-BLAST and HMMer3. HHblits searches with a query Hidden Markov Model (HMM) against a database of HMMs of multiple sequence alignments. The HMMs of the uniprot20 database are based on a very conservative clustering of the UniProt sequence database with few, mostly very similar sequences contained in each cluster. In chapter 1, we enriched our database clusters by jumpstarting HHblits with each cluster alignment in the uniprot20 and adding significant matches to the cluster alignments. The resulting uniprot_boost1 database is much more diverse with about 14 times more sequences per cluster and the effective number of sequences was raised from 1.18 to 3.78.

In three iterations through the uniprot_boost1, we find about 20% more homologs than with the uniprot20. However, three iterations through the uniprot_boost1 are on average 3.7 times slower. Therefore, our aim was to improve the faster uniprot20 database with results from the uniprot_boost1.

A database for HHblits consists of three parts:

- The multiple sequence alignments in the a3m format [5]
- The HMM profiles translated into a discretized set of 219 column states (cs219) used for the prefilter
- The precalculated HMMs for large multiple sequence alignments in the hhm format [5]

Every cluster in the uniprot20 database was used as a seed alignment for a cluster in the uniprot_boost1 database. Therefore, every cluster in the uniprot20 has a corresponding cluster in the uniprot_boost1 and we can exchange the cs219 sequences of the uniprot20 database with the cs219 sequences of the uniprot_boost1. This should improve the sensitivity of the prefilter, since cs219 sequences of the uniprot_boost1 should capture the information of the much more diverse clusters in the uniprot_boost1.

Additionally, we implemented the feature to use the profile of the discretized column states for the calculation of the column score in HHblits' Viterbi and Maximum Accuracy alignment algorithms:

The column score between position i of the query Hidden Markov profile q and position j of the template Hidden Markov profile t is the scalar product of the profiles of amino acid frequencies at those positions:

$$score(i,j) = \sum_{a=1}^{20} q(i,a) \cdot t(j,a) \tag{A.1}$$

In contrast to the setup in section 1.4.4, we searched with HHblits in one, two and three iterations through the database followed by a single search iteration through the reference SCOP HMM database.

We measured the performance of the following setups in the ROC5 plot (see figure A.1):

- HHblits searches through the uniprot_boost1
- HHblits searches through the uniprot20
- HHblits searches through the uniprot20 with the discretized column states of the uniprot_boost1 for the pre-filter
- HHblits searches through the uniprot20 with the discretized column states of the uniprot_boost1 for the pre-filter, the Viterbi alignment and the Maximum Accuracy alignment

The homology detection sensitivity of HHblits with the uniprot20 can be improved with the discretized column states of the uniprot_boost1. We expected that they increase the sensitivity of the pre-filter. But more astonishingly, they already capture enough information of the uniprot_boost1 Hidden Markov profiles to be used for the calculation of the Viterbi and Maximum Accuracy alignments in HHblits.

The column scores of the query profile to the 219 discretized column states could be pre-calculated, so the overall runtime of HHblits was decreased. In the meanwhile, Martin Steinegger re-implemented the Viterbi algorithm with faster SIMD instructions. This implementation was no longer compatible to the scoring against the discretized column state sequences of the database HMMs. Therefore, this feature is not included in HHblits 3.0.

Appendix B

Supplementary - Pdom

B.1 Notation

q	query HMM
$t_1, \ldots, t_K$	template HMM with which q is aligned
L_q, L_k	length of query sequence q and template sequence t_k
i, j, l	indices for positions in q or t_k
σ, σ'	labels for domain start s or end e position. Without inserted domains: $\sigma, \sigma' \in \{s, e\}$. With inserted domains: $\sigma, \sigma' \in \{s^0, e^0, s^1, e^1, \ldots\}$
P_{qk}	probability that q and t_k are homologous, calculated by HHblits
$b_{qk}(i, j)$	alignment backward probability that the alignment *starts* at position i of the query q and position j of the template t_k, calculated from the Backward algorithm in HHblits
$f_{qk}(i, j)$	alignment forward probability that the alignment *ends* at position i of the query q and position j of the template t_k, calculated from the Forward algorithm in HHblits
$p(q_i \diamond t_{k,j} \vert A_{qk})$	alignment posterior probability that position i of query q and position j of template t_k are aligned, calculated from the Backward/-Forward algorithm in HHblits
$p^s_{qk}(i)$	probability of alignment between q and t_k to start at position $i \in \{1, \ldots, L_q\}$, computed with the Backward algorithm in `hhblits`
$p^e_{qk}(i)$	probability of alignment between q and t_k to end at position $i \in \{1, \ldots, L_q\}$, computed with the Forward algorithm in `hhblits`
$\mathcal{A}_q = \{A_{q1}, \ldots, A_{qK}\}$	pairwise alignments between q and t_k. A_{qk} subsumes all alignment information, $A_{qk} = (P_{qk}, b_{qk}, f_{qk}, p(q_i \diamond t_{k,j} \vert A_{qk}))$.
w^σ_{qk}	weight of p^σ_{qk}, where $\sigma \in \{s, e\}$
$p_{e^0 \vert s^0}(i-j)$	probability for length of domain at level $0 = i - j + 1$ ($p_{\text{dom}}(i-j+1)$) (input)
$p_{s^0 \vert e^0}(i-j)$	probability of linker length $= i - j - 1$ ($p_{\text{link}}(i-j-1)$ (input)
$p_{s^1 \vert e^1}(i-j)$	probability for length of inserted domain $= i - j + 1$
$p_{s^1 \vert s^0}(i-j)$	length distributions of N-terminal parts of split domains
$p_{e^0 \vert e^1}(i-j)$	length distributions of C-terminal parts of split domains
$L_{\min}$	minimum length of a domain
$\mathcal{Y}_q = \{y_1, \ldots, y_{2D}\}$	set of ordered domain start s and end e points in the query. Each point $y_\nu = [\sigma, i]_\nu$ consists of a label $\sigma \in \{s, e\}$ and of the position $i \in \{0, \ldots, L_q\}$ of the ν'th point. D is the number of domains.

$p([\sigma, i]\|\ldots)$	$p([\sigma, i] \in \mathcal{Y}_q\|\ldots)$, probability for $[\sigma, i]$ to be a domain start/end point
$p([\sigma, i]_\nu\|[\sigma', j]_{\nu-1}\ldots)$	probability for label $[\sigma, i]$ succeeding label $[\sigma', j]$
s_{qk}, e_{qk}	start and end positions of common homologous region (*not* aligned region) between q and t_k; must coincide with domain start/end positions
$q \sim t_k$	q is homologous to t_k; example: $p(q \sim t_k\|A_{qk}) = P_{qk}$
$F_{\sigma,i}$	Forward probability for domain start ($\sigma = s$) or end ($\sigma = e$)
$B_{\sigma,j}$	Backward probability for domain start ($\sigma = s$) or end ($\sigma = e$)
p_{ins}	Probability for a domain insertion
p_{rep}	Probability for an inserted domain following a previous inserted domain
$\text{pred}(\sigma), \text{succ}(\sigma)$	Set of states that are possible predecessors / successors to σ
$p_q^\sigma(i)$	$= p([\sigma, i]\| y_0, y_\dagger, ...)$, probability that a domain starts ($\sigma = s$) or ends ($\sigma = e$) at position i in the query

B.2 Domain Prediction without Insertions

B.2.1 Calculation of $p^e_{qk}(i)$ and $p^s_{qk}(i)$

The probability $p^e_{qk}(i)$ that the alignment between query q and template t_k ends at position i in q is obtained from the Forward-Backward algorithm implemented in HHblits [57, 52]. Let us call the forward probability in HHblits $f_{qk}(i,j)$, which can be understood as the probability for the alignment to *end* at position i in query q and position j in template t_k. Then the probability for the alignment to *end* at query position i is:

$$p^e_{qk}(i) = \frac{\sum_{j=1}^{L_k} f_{qk}(i,j)}{\sum_{i'=1}^{L_q}\sum_{j=1}^{L_k} f_{qk}(i',j)} \tag{B.1}$$

L_k is the number of residues in t_k.

The backward probability in HHblits $b_{qk}(i,j)$ can be understood as the probability for the alignment to *start* at position i in query q and position j in template t_k. Analogously, the probability for the alignment to *start* at query position i is

$$p^s_{qk}(i) = \frac{\sum_{j=1}^{L_k} b_{qk}(i,j)}{\sum_{i'=1}^{L_q}\sum_{j=1}^{L_k} b_{qk}(i',j)} \tag{B.2}$$

B.2.2 Calculation of $g_q(\sigma, i|\sigma', j) := p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, \mathcal{A}_q)$

In the following we will abbreviate $y_\nu = [\sigma, i]$ by $[\sigma,i]_\nu$, $y_\nu \neq [\sigma, i]$ by $\neg[\sigma,i]_\nu$ and $y_{\nu-1} = [\sigma, j]$ by $[\sigma',j]_{\nu-1}$ and so on. The function $g_q(\sigma, i|\sigma', j) = p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, \mathcal{A}_q)$ gives the probability of observing a start or end site σ at i, given that the *preceding* end or start site is $[\sigma', j]$ and given the alignments $\mathcal{A}_q = \{A_{q1}, \ldots, A_{qK}\}$.

We get with Bayes' theorem:

$$\begin{aligned} g_q(\sigma, i|\sigma', j) &= \frac{p(\mathcal{A}_q|[\sigma,i]_\nu, [\sigma',j]_{\nu-1})\; p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{\text{numerator} + p(\mathcal{A}_q|\neg[\sigma,i]_\nu, [\sigma',j]_{\nu-1})\; p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \\ g_q(\sigma, i|\sigma', j) &= \left[1 + \frac{p(\mathcal{A}_q|\neg[\sigma,i]_\nu, [\sigma',j]_{\nu-1})\; p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p(\mathcal{A}_q|[\sigma,i]_\nu, [\sigma',j]_{\nu-1})\; p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}))}\right]^{-1} \end{aligned} \tag{B.3}$$

We approximate the odds ratio for the K alignments by a product of odds ratios with different weight factors w^σ_{qk}. These correct for redundancy between alignments and are calculated as discussed in section B.2.3:

$$g_q(\sigma, i|\sigma', j) = \left[1 + \frac{p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \prod_{k=1}^{K} \left(\frac{p(A_{qk}|\neg[\sigma,i]_\nu, [\sigma',j]_{\nu-1})}{p(A_{qk}|[\sigma,i]_\nu|[\sigma',j]_{\nu-1})}\right)^{w^\sigma_{qk}}\right]^{-1} \tag{B.4}$$

The first factor can be calculated from the probability distributions for linker length and domain length, since it does not depend on any alignment. For $\sigma = s$ and $\sigma' = e$ we obtain

$$\begin{aligned} p([e,i]_\nu|[s,j]_{\nu-1}) &= p(\sigma_\nu = e|\sigma_{\nu-1} = s)\, p(i_\nu = i|\sigma_\nu = e, [s,j]_{\nu-1}) \\ &= p_{\text{dom}}(i - j + 1) \end{aligned} \tag{B.5}$$

Similarly to the previous equation, we get

$$\begin{aligned} p([s,i]_\nu|[e,j]_{\nu-1}) &= p(\sigma_\nu{=}s|\sigma_{\nu-1}{=}e)\, p(i_\nu{=}i|\sigma_\nu{=}s,[e,j]_{\nu-1}) \\ &= p_{\text{link}}(i-j-1) \end{aligned} \tag{B.6}$$

$p(\sigma_\nu{=}e|\sigma_{\nu-1}{=}s)=1$ and $p(\sigma_\nu{=}s|\sigma_{\nu-1}{=}e)=1$, since a domain start is always followed by a domain end and a domain end is always followed by a domain start in the simple case without inserted domains.

The probability P_{qk} that query q and template t_K are homologous

The Bayes factor for each alignment in equation (B.4) is rewritten by applying Bayes' theorem on both numerator and denominator,

$$\begin{aligned} \frac{p(A_{qk}|\neg[\sigma,i]_\nu,[\sigma',j]_{\nu-1})}{p(A_{qk}|[\sigma,i]_\nu,[\sigma',j]_{\nu-1})} &= \frac{p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1},A_{qk})\,\cancel{p(A_{qk}|[\sigma',j]_{\nu-1})}\,p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},A_{qk})\,\cancel{p(A_{qk}|[\sigma',j]_{\nu-1})}\,p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \\ &= \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1})}\,\frac{p(\neg[\sigma,i]_\nu|[\sigma',j]_{\nu-1},A_{qk})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},A_{qk})} \end{aligned} \tag{B.7}$$

We transform the denominator in the second odds ratio by summing over two states, query and template are homologous $q\sim t_K$ and query and template are not homologous $q\not\sim t_K$. In the latter case the alignment does not contribute any information about the connection between i and j:

$$\begin{aligned} &p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},A_{qk}) \\ &= p([\sigma,i]_\nu|q{\sim}t_k,[\sigma',j]_{\nu-1},A_{qk})p(q{\sim}t_k|A_{qk}) + p([\sigma,i]_\nu|q{\not\sim}t_k,[\sigma',j]_{\nu-1},A_{qk})p(q{\not\sim}t_k|A_{qk}) \\ &= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},p^s_{qk},p^e_{qk})P_{qk} + p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})(1-P_{qk}) \end{aligned} \tag{B.8}$$

The numerator in the second odds term in equation (B.7) is in fact just 1 minus the denominator. Therefore, the Bayes factor in equation (B.7) becomes

$$\begin{aligned} &\frac{p(A_{qk}|\neg[\sigma,i]_\nu,[\sigma',j]_{\nu-1})}{p(A_{qk}|[\sigma,i]_\nu,[\sigma',j]_{\nu-1})} \\ &= \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{1-p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \times \left(\frac{1}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},p^s_{qk},p^e_{qk})P_{qk} + p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})(1-P_{qk})} - 1 \right) \end{aligned} \tag{B.9}$$

We abbreviate the odds ratio

$$R_{qk}(\sigma,i|\sigma',j) = \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},p^s_{qk},p^e_{qk})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \tag{B.10}$$

This allows us to write equation (B.9) as

$$\frac{p(A_{qk}|\neg[\sigma,i]_\nu,[\sigma',j]_{\nu-1})}{p(A_{qk}|[\sigma,i]_\nu,[\sigma',j]_{\nu-1})} = \frac{(R_{qk}(\sigma,i|\sigma',j)P_{qk}+1-P_{qk})^{-1} - p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{1-p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \tag{B.11}$$

Substituting equation (B.11) into (B.4) yields

$$g_q(\sigma, i|\sigma', j) = \left[1 + \frac{1 - p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \prod_{k=1}^{K} \left(\frac{(R_{qk}(\sigma, i|\sigma', j)P_{qk} + 1 - P_{qk})^{-1} - p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{1 - p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \right)^{w^\sigma_{qk}} \right]^{-1} \tag{B.12}$$

Calculation of $R_{qk}(\sigma, i|\sigma', j)$

We rewrite the numerator of $R_{qk}(\sigma, i|\sigma', j)$ in equation (B.10) as,

$$p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_{qk}, p^e_{qk}) = \frac{p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})}{p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})} \tag{B.13}$$

Calculation of $p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$

We use the information of p^s_{qk}, p^e_{qk} in $p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ by summing over all possible pairs of start positions $l = s_{qk}$ and end positions $m = e_{qk}$ of the homologous region between q and t_k:

$$p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk}) = \sum_{l=1}^{L_q} \sum_{m=l}^{L_q} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|l = s_{qk}, m = e_{qk})\, p(l = s_{qk}, m = e_{qk}|p^s_{qk}, p^e_{qk}) \tag{B.14}$$

In the first probability inside the sum, $[\sigma,i]_\nu$ and $[\sigma',j]_{\nu-1}$ depend on p^s_{qk}, p^e_{qk} only through $l = s_{qk}, m = e_{qk}$. We can therefore drop the conditioning on p^s_{qk}, p^e_{qk}. The double sum can be split into three partial sums, $\sum_{(1)}, \sum_{(2)}, \sum_{(3)}$, characterized by the following relations:

$$\begin{aligned} &(1)\ \ l < m \leq j < i \\ &(2)\ \ l \leq j < i \leq m \\ &(3)\ \ j < i \leq l < m \end{aligned}$$

Those three cases are all-encompassing, since j is a direct predecessor of i and therefore l and m cannot lie between j and i.

In the following we use the approximations:

$$\begin{aligned} p(l = s_{qk}|p^s_{qk}) &\approx p^s_{qk}(l) \\ p(m = e_{qk}|p^e_{qk}) &\approx p^e_{qk}(m) \end{aligned}$$

Additionally, we use the approximation:

$$p(l = s_{qk}, m = e_{qk}|p^s_{qk}, p^e_{qk}) \approx p(l = s_{qk}|p^s_{qk})\, p(m = e_{qk}|p^e_{qk})$$

Calculation of $p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ for $l < m \leq j < i$

In the first partial sum, it follows from $l < m \leq j < i$ that $[\sigma,i]_\nu$ depends only on $[\sigma',j]_{\nu-1}$ and $[\sigma',j]_{\nu-1}$ depends only on $m = e_{qk}$, therefore

$$\begin{aligned}
\sum\nolimits_{(1)} &= \sum_{m=1}^{j}\sum_{l=1}^{m} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|\cancel{l=s_{qk}}, m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= \sum_{m=1}^{j}\sum_{l=1}^{m} p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})\, p([\sigma',j]_{\nu-1}|m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{m=1}^{j} p([\sigma',j]_{\nu-1}|m=e_{qk}) \sum_{l=1}^{m} p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{m=1}^{j} p([\sigma',j]_{\nu-1}|m=e_{qk})\, p^e_{qk}(m) \\
&= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{m=1}^{j} p([\sigma',j]|[e,m])\, p^e_{qk}(m) \qquad \text{(B.15)}
\end{aligned}$$

Calculation of $p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ for $l \leq j < i \leq m$

In the second partial sum, it follows from $l \leq j < i \leq m$ that $[\sigma,i]_\nu$ depends on $l = s_{qk}$ only through $[\sigma',j]_{\nu-1}$, and $[\sigma',j]_{\nu-1}$ depends on $m = e_{qk}$ only through $[\sigma,i]_\nu$, and therefore

$$\begin{aligned}
\sum\nolimits_{(2)} &= \sum_{l=1}^{j}\sum_{m=i}^{L_q} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|\cancel{l=s_{qk}}, m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= \sum_{l=1}^{j}\sum_{m=i}^{L_q} p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, m=e_{qk})\, p([\sigma',j]_{\nu-1}|l=s_{qk}, m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&\approx \sum_{l=1}^{j}\sum_{m=i}^{L_q} p([\sigma',j]_{\nu-1}|l=s_{qk}, m=e_{qk})\, p^s_{qk}(l)\, p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, m=e_{qk})\, p^e_{qk}(m) \qquad \text{(B.16)}
\end{aligned}$$

We apply Bayes' theorem to exchange $m = e_{qk}$ and $[\sigma', j]$ in the first probability in equation (B.16),

$$p([\sigma',j]_{\nu-1}|l=s_{qk}, m=e_{qk}) = \frac{p(m=e_{qk}|[\sigma',j], \cancel{l=s_{qk}})\, p([\sigma',j]|l=s_{qk})}{p(m=e_{qk}|l=s_{qk})} \qquad \text{(B.17)}$$

We drop the conditioning on $l = s_{qk}$ in the first probability of the numerator, since $m = e_{qk}$ depends on $l = s_{qk}$ only through $[\sigma',j]_{\nu-1}$. Similarly, we apply Bayes' theorem to exchange $[\sigma,i]_\nu$ and $m = e_{qk}$ in the third probability in equation (B.16),

$$p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, m=e_{qk}) = \frac{p(m=e_{qk}|[\sigma,i]_\nu, \cancel{[\sigma',j]_{\nu-1}})\, p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}{p(m=e_{qk}|[\sigma',j]_{\nu-1})} \qquad \text{(B.18)}$$

We drop the conditioning on $[\sigma',j]_{\nu-1}$ in the first probability of the numerator, since $m=e_{qk}$ depends on $[\sigma',j]_{\nu-1}$ only through $[\sigma,i]_\nu$. When inserting equations (B.17) and (B.18) into equation (B.16), the terms $p(m=e_{qk}|[\sigma',j]_{\nu-1})$ in the numerator of equation (B.17) and the denominator of (B.18) cancel out, yielding

$$\sum\nolimits_{(2)} = p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l) \sum_{m=i}^{L_q} \frac{p(m=e_{qk}|[\sigma,i]_\nu)}{p(m=e_{qk}|l=s_{qk})}\, p^e_{qk}(m) \quad \text{(B.19)}$$

Calculation of $p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ for $j < i \leq l < m$

In the third partial sum, it follows from $j < i \leq l < m$ that $[\sigma,i]_\nu$ and $[\sigma',j]_{\nu-1}$ are independent of $m=e_{qk}$ given $l=s_{qk}$, and therefore

$$\begin{aligned}
\sum\nolimits_{(3)} &= \sum_{l=i}^{L_q}\sum_{m=l}^{L_q} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|l=s_{qk}, \cancel{m=e_{qk}})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= \sum_{l=i}^{L_q} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|l=s_{qk}) \sum_{m=l}^{L_q} p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\
&= \sum_{l=i}^{L_q} p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|l=s_{qk})\, p^s_{qk}(l) \\
&= \sum_{l=i}^{L_q} p(l=s_{qk}|[\sigma,i]_\nu, \cancel{[\sigma',j]_{\nu-1}})\, p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})\, \frac{p([\sigma',j]_{\nu-1})}{p(l=s_{qk})}\, p^s_{qk}(l) \\
&= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})\, \frac{\cancel{p([\sigma',j])}}{\cancel{p([\sigma,i])}} \sum_{l=i}^{L_q} p([\sigma,i]|l=s_{qk})\, p^s_{qk}(l)
\end{aligned} \quad \text{(B.20)}$$

We assume approximately the same priors for domain start and end sites, so the ratio before the sum cancels out.

Calculation of $p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ for $l < m \leq j$

In the first partial sum, it follows from $l < m \leq j$ that $[\sigma',j]_{\nu-1}$ depends on $l = s_{qk}$ only through $m = e_{qk}$, and therefore

$$\begin{aligned}\sum\nolimits_{(1')} &= \sum_{m=1}^{j}\sum_{l=1}^{m} p([\sigma',j]|\cancel{l=s_{qk}}, m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\ &= \sum_{m=1}^{j} p([\sigma',j]|m=e_{qk}) \sum_{l=1}^{m} p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\ &= \sum_{m=1}^{j} p([\sigma',j]|m=e_{qk})\, p^e_{qk}(m)\end{aligned} \tag{B.21}$$

Calculation of $p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})$ for $l \leq j \leq m$

In the second partial sum, it follows from $l \leq j \leq m$ that $m = e_{qk}$ depends on $l = s_{qk}$ only through $[\sigma',j]_{\nu-1}$, and therefore

$$\begin{aligned}\sum\nolimits_{(2')} &= \sum_{l=1}^{j}\sum_{m=j}^{L_q} p([\sigma',j]|l=s_{qk}, m=e_{qk})\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\ &= \sum_{l=1}^{j}\sum_{m=j}^{L_q} \frac{p(m=e_{qk}|[\sigma',j], \cancel{l=s_{qk}})\, p([\sigma',j]|l=s_{qk})}{p(m=e_{qk}|l=s_{qk})}\, p(l=s_{qk}, m=e_{qk}|p^s_{qk}, p^e_{qk}) \\ &\approx \sum_{l=1}^{j}\sum_{m=j}^{L_q} \frac{p(m=e_{qk}|[\sigma',j])\, p([\sigma',j]|l=s_{qk})}{p(m=e_{qk}|l=s_{qk})}\, p^s_{qk}(l)\, p^e_{qk}(m) \\ &= \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l) \sum_{m=j}^{L_q} \frac{p(m=e_{qk}|[\sigma',j])}{p(m=e_{qk}|l=s_{qk})}\, p^e_{qk}(m)\end{aligned} \tag{B.22}$$

Calculation of $p([\sigma',j]_{\nu-1}|p^s_{qk},p^e_{qk})$ for $j \le l < m$

In the third partial sum, it follows from $j \le l < m$ that $[\sigma', j]$ is independent of $m = e_{qk}$ given $l = s_{qk}$, and therefore

$$\begin{aligned}\sum\nolimits_{(3')} &= \sum_{l=j}^{L_q}\sum_{m=l}^{L_q} p([\sigma',j]|l=s_{qk},\cancel{m=e_{qk}})\, p(l=s_{qk},m=e_{qk}|p^s_{qk},p^e_{qk}) \\ &= \sum_{l=j}^{L_q} p([\sigma',j]|l=s_{qk}) \sum_{m=l}^{L_q} p(l=s_{qk},m=e_{qk}|p^s_{qk},p^e_{qk}) \\ &= \sum_{l=j}^{L_q} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l)\end{aligned} \tag{B.23}$$

Calculation of $f_{\sigma|\sigma'}(i-j) := p([\sigma,i]|[\sigma',j])$ by dynamic programming

Note that $f_{\sigma,\sigma'}(i-j) := p([\sigma,i]|[\sigma',j])$ depends only on the difference of the position indices, $i-j$. For $i \ge 0$, the function $f_{\sigma|\sigma'}(i) = p([\sigma,i]|[\sigma',0])$ can be calculated by dynamic programming by summing over all possible states $[\sigma'',j]_{\nu-1}$ that precede the last state $[\sigma,i]_\nu$,

$$\boxed{\begin{aligned} f_{\sigma|\sigma'}(0) &= I(\sigma=\sigma') \\ f_{\sigma|\sigma'}(i) &= \sum_{\sigma''\in \mathrm{pred}(\sigma)} \sum_{j=0}^{i-1} f_{\sigma''|\sigma'}(j)\, p_{\sigma|\sigma''}(i-j) \end{aligned}} \tag{B.24}$$

$\mathrm{pred}(\sigma)$ is the set of possible predecessor states of σ. As long as we do not consider inserted domains, we have $\mathrm{pred}(s) = \{e\}$ and $\mathrm{pred}(e) = \{s\}$.

We calculate $p([\sigma',j]|[\sigma,i])$ for $j \le i$ by Bayes' theorem:

$$f_{\sigma'|\sigma}(j-i) := p([\sigma',j]|[\sigma,i]) = p([\sigma,i]|[\sigma',j])\frac{\cancel{p([\sigma',j])}}{\cancel{p([\sigma,i])}} = f_{\sigma|\sigma'}(i-j) \tag{B.25}$$

We assume approximately the same priors for domain start and end sites, so the ratio before the sum cancels out.

Calculation of $R_{qk}(\sigma,i|\sigma',j)$ - final

Putting equations (B.10), (B.13), (B.15), (B.19)-(B.23) together, we obtain

$$R_{qk}(\sigma,i|\sigma',j) = \frac{1}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}\frac{\sum_{(1)}+\sum_{(2)}+\sum_{(3)}}{\sum_{(1')}+\sum_{(2')}+\sum_{(3')}}$$

B.2.3 Sequence weighting

We can estimate weights to account for redundancy among the alignments the following way:

$$\boxed{w_{qk}^s = \left[\sum_{k'=1}^{K}\sum_{i=1}^{L_q}\sqrt{p_{qk}^s(i)\,p_{qk'}^s(i)}\right]^{-1}} \tag{B.26}$$

$p_{qk}^s(i)$ are normalized vectors. This expression is a sum of K scalar products of normal vectors. Each scalar product is between 0 and 1. Since the term for $k' = k$ is equal to 1, we know that $1 \leq w_{qk}^{-1} \leq K$ and therefore

$$\frac{1}{K} \leq w_{qk} \leq 1 \tag{B.27}$$

Without redundancy $w_{qk} = 1$ and with complete redundancy $w_{qk} = 1/K$.

We can generalize the above weights in the following way:

$$\boxed{w_{qk}^s = \left[\sum_{k'=1}^{K}\frac{\sum_{i=1}^{L_q} p_{qk}^s(i)^{\alpha_w}\, p_{qk}^s{}'(i)^{\alpha_w}}{\sqrt{\sum_i p_{qk}^s(i)^{2\alpha_w}}\,\sqrt{\sum_i p_{qk}^s{}'(i)^{2\alpha_w}}}\right]^{-1}} \tag{B.28}$$

Again, we have the desired limits in the case of full and no redundancy, with $1/K \leq w_{qk} \leq 1$. For $\alpha_w = 0.5$ we get the previous formula. For smaller α_w, the weights tend to get smaller. α_w can be optimized.

B.3 Domain Prediction with Inserted Domains

B.3.1 Calculation of $g_q(\sigma, i|\sigma', j) := p([\sigma, i]_\nu|[\sigma', j]_{\nu-1})$

The calculation proceeds as in section B.2.2. We generalize equations (B.5) and (B.6),

$$p([\sigma, i]_\nu|[\sigma', j]_{\nu-1}) = p([\sigma, \cdot]_\nu|[\sigma', \cdot]_{\nu-1})\, p([\cdot, i]_\nu|[\sigma, \cdot]_\nu, [\sigma', j]_{\nu-1}) \tag{B.29}$$

which can be abbreviated by defining

$$\boxed{g_q(\sigma, i|\sigma', j) := p([\sigma, i]_\nu|[\sigma', j]_{\nu-1}) = p(\sigma_\nu|\sigma'_{\nu-1})\, p_{\sigma|\sigma'}(i - j)} \tag{B.30}$$

The transition probabilities $p(\sigma_\nu|\sigma'_{\nu-1})$ can be taken from figure 2.7b:

$$\boxed{\begin{aligned} p(s^0|e^0) &= 1 \\ p(e^0|s^0) &= 1 \\ p(s^{\lambda+1}|e^{\lambda+1}) &= p_{\text{rep}} \\ p(e^\lambda|s^\lambda) &= 1 - p_{\text{ins}} \\ p(s^{\lambda+1}|s^\lambda) &= p_{\text{ins}} \\ p(e^\lambda|e^{\lambda+1}) &= 1 - p_{\text{rep}} \end{aligned}}$$

Calculation of $R_{qk}(\sigma, i|\sigma', j)$

To calculate $R_{qk}(\sigma, i|\sigma', j)$ for any $\sigma, \sigma' \in \{s^0, e^0, s^1, e^1, s^2, e^2, \ldots\}$, we proceed as in section B.2.2.

As in equations (B.13) and (B.26), we calculate R as

$$R_{qk}(\sigma, i|\sigma', j) = \frac{p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})}{p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk})} = \frac{1}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \frac{\sum_{(1)} + \sum_{(2)} + \sum_{(3)}}{\sum_{(1')} + \sum_{(2')} + \sum_{(3')}}$$

$l = s_{qk}$ refers to any of the start states s^λ and $m = e_{qk}$ to any of the end states e^λ in the six sums (B.15), (B.19), (B.20), (B.21), (B.22) and (B.23). In the following, we describe the changes to the six partial sums.

$$
\begin{aligned}
\sum\nolimits_{(1)} &= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{m=1}^{j} p([\sigma',j]_{\nu-1}|m=e_{qk})\, p^e_{qk}(m) \\
\sum\nolimits_{(2)} &= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l) \sum_{m=i}^{L_q} \frac{p(m=e_{qk}|[\sigma,i])}{p(m=e_{qk}|l=s_{qk})}\, p^e_{qk}(m) \\
\sum\nolimits_{(3)} &= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) \frac{p([\sigma',j])}{p([\sigma,i])} \sum_{l=i}^{L_q} p([\sigma,i]|l=s_{qk})\, p^s_{qk}(l) \\
\sum\nolimits_{(1')} &= \sum_{m=1}^{j} p([\sigma',j]|m=e_{qk})\, p^e_{qk}(m) \\
\sum\nolimits_{(2')} &= \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l) \sum_{m=j}^{L_q} \frac{p(m=e_{qk}|[\sigma',j])}{p(m=e_{qk}|l=s_{qk})}\, p^e_{qk}(m) \\
\sum\nolimits_{(3')} &= \sum_{l=j}^{L_q} p([\sigma',j]|l=s_{qk})\, p^s_{qk}(l)
\end{aligned}
\tag{B.31}
$$

We transform the fraction

$$\frac{p(m=e_{qk}|[\sigma,i])}{p(m=e_{qk}|l=s_{qk})}$$

in $\sum_{(2)}$ and $\sum_{(2')}$, defining the union event $[e,m] = ([e^0,m] \vee [e^1,m] \vee \ldots)$:

$$
\begin{aligned}
\frac{p(m=e_{qk}|[\sigma,i])}{p(m=e_{qk}|l=s_{qk})} &\approx \frac{\cancel{p(m=e_{qk}|[e,m])}\, p([e,m]|[\sigma,i])}{\sum_{\lambda=0}^{\infty} \cancel{p(m=e_{qk}|[e^\lambda,m])}\, p([e^\lambda,m]|[s^\lambda,l])\, p([s^\lambda,l]|l=s_{qk})} \\
&= \frac{\sum_{\lambda=0}^{\infty} p([e^\lambda,m]|[\sigma,i])}{\sum_{\lambda=0}^{\infty} p([e^\lambda,m]|[s^\lambda,l])\, p([s^\lambda,l]|l=s_{qk})}
\end{aligned}
\tag{B.32}
$$

Estimation of $p([e^\lambda,m]|m=e_{qk})$

$p([e^\lambda,m]|m=e_{qk})$ can be estimated by deriving the steady-state populations $p(\sigma)$ from the state diagram in figure 2.7b. Note that $p(e^\lambda) = p(s^\lambda) =: x_\lambda$ because for each domain start there exists exactly one domain end. At equilibrium, the flux into e^λ must equal the flux out of it, hence

$$p_{\text{ins}} x_{\lambda-1} + p_{\text{rep}} x_\lambda = (1-p_{\text{ins}}) x_\lambda + p_{\text{ins}} x_\lambda \tag{B.33}$$

Solving for x_λ gives

$$x_\lambda = \frac{p_{\text{ins}}}{1-p_{\text{rep}}} x_{\lambda-1} = \left(\frac{p_{\text{ins}}}{1-p_{\text{rep}}}\right)^\lambda x_0 = a^\lambda x_0 \tag{B.34}$$

with

$$a := \frac{p_{\text{ins}}}{1 - p_{\text{rep}}} \tag{B.35}$$

$(m = e_{qk})$ implies $[e^\lambda, m]$ for a $\lambda \geq 0$. Therefore, normalization over the e^λ states yields

$$p([e^\lambda, m] | m = e_{qk}) = a^\lambda (1 - a) \tag{B.36}$$

Calculation of $f_{\sigma|\sigma'}(i - j) := p([\sigma, i]|[\sigma', j])$

The calculation of $f_{\sigma|\sigma'}(i - j)$ can be done for the new states exactly as formulated in equation (B.24).

Calculation of $f_{\sigma|e}(i - m) := p([\sigma, i]|m = e_{qk})$ and $f_{\sigma|s}(i - l) := p([\sigma, i]|l = s_{qk})$

$p([\sigma, i]|m = e_{qk})$ can be calculated as follows:

$$\begin{aligned} f_{\sigma|e}(i - m) :=& p([\sigma, i]|m = e_{qk}) && \text{(B.37)} \\ =& \sum_{\lambda=0}^{\infty} p([\sigma, i], [e^\lambda, m]|m = e_{qk}) \\ =& \sum_{\lambda=0}^{\infty} p([\sigma, i]|[e^\lambda, m])\, p([e^\lambda, m]|m = e_{qk}) \\ =& \sum_{\lambda=0}^{\infty} f_{\sigma|e^\lambda}(i - m)\, p([e^\lambda, m]|m = e_{qk}) && \text{(B.38)} \end{aligned}$$

With equation (B.36):

$$\boxed{f_{\sigma|e}(i - m) := p([\sigma, i]|m = e_{qk}) = (1 - a) \sum_{\lambda=0}^{\infty} f_{\sigma|e^\lambda}(i - m)\, a^\lambda} \tag{B.39}$$

Analogously:

$$\boxed{f_{\sigma|s}(i - l) := p([\sigma, i]|l = s_{qk}) = (1 - a) \sum_{\lambda=0}^{\infty} f_{\sigma|s^\lambda}(i - l)\, a^\lambda} \tag{B.40}$$

Calculation of $f_{e|\sigma}(m-i) := \sum_{\lambda=0}^{\infty} p([e^\lambda, m]|[\sigma, i])$ and $f_{e|s}(m-l) := \sum_{\lambda=0}^{\infty} p([e^\lambda, m]|[s^\lambda, l])\, p([s^\lambda, l]|l = s_{qk}))$

For the nominator and denominator in equation (B.32) we get:

$$\boxed{\begin{aligned} f_{e|\sigma}(m-i) &:= \sum_{\lambda=0}^{\infty} p([e^\lambda, m]|[\sigma, i]) = \sum_{\lambda=0}^{\infty} f_{e^\lambda|\sigma}(m-i) \\ f_{e|s}(m-l) &:= \sum_{\lambda=0}^{\infty} p([e^\lambda, m]|[s^\lambda, l])\, p([s^\lambda, l]|l = s_{qk})) = (1-a) \sum_{\lambda=0}^{\infty} f_{e^\lambda|s^\lambda}(m-l)\, a^\lambda \end{aligned}} \tag{B.41}$$

Calculation of $\frac{p([\sigma', j])}{p([\sigma, i])}$

The ratio of priors in equation (B.31) does not cancel out anymore when taking inserted domains into account. The steady state population is proportional to $a^\lambda(1-a)$ for insertion level λ. Therefore,

$$\boxed{\frac{p([\sigma', j])}{p([\sigma, i])} = \frac{a^{\lambda'}(1-a)}{a^\lambda(1-a)} = a^{\lambda'-\lambda}} \tag{B.42}$$

λ and λ' are the insertion levels of the states $[\sigma, i]$ and $[\sigma', j]$, respectively.

B.4 Consistency Iterations

B.4.1 Calculation of $g_q(\sigma, i|\sigma', j) := p([\sigma, i]_\nu|[\sigma', j]_{\nu-1}, \mathcal{A}_q, p^s_{1\ldots K}, p^e_{1\ldots K})$

For the consistency iterations we use almost the same procedure to compute the next iteration of $p^\sigma_q(i)$, including the forward-backward algorithm. But instead of computing $g_q(\sigma, i|\sigma', j)$ as defined in equation (B.30), we now need to compute $g_q(\sigma, i|\sigma', j) := p([\sigma, i]_\nu|[\sigma', j]_{\nu-1}, \mathcal{A}_q, p^s_{1\ldots K}, p^e_{1\ldots K})$

which differs from equation (B.30) only by conditioning also on the the domain start and end probability densities $p^s_k(j)$ and $p^e_k(j)$ from the previous iteration (see equation (2.6)). In the following, the information in $p^s_k(j)$, and $p^e_k(j)$ is included through ${p^s_{qk}}'(i)$, ${p^e_{qk}}'(i)$, $r^s_{qk}(i)$, $r^e_{qk}(i)$ and $r^*_{qk}(i)$.

Calculation of $R_{qk}(\sigma, i|\sigma', j)$

With the conditioning on $p^s_k(j)$ and $p^e_k(j)$ equation (B.10) becomes:

$$R_{qk}(\sigma, i|\sigma', j) = \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})} \tag{B.43}$$

The numerator becomes:

$$p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk}) = \frac{p([\sigma,i]_\nu, [\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk})}{p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk})} \tag{B.44}$$

In the following we use the approximations:

$$p(l = s_{qk}|p^s_{qk}) \approx {p^s_{qk}}'(l)$$
$$p(m = e_{qk}|p^e_{qk}) \approx {p^e_{qk}}'(m)$$

Calculation of $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk})$ for $l < m \leq j < i$

The first partial sum (equation (B.15)) becomes

$$\frac{\sum_{(1)}}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_k, p^e_k, A_{qk})} = \sum_{m=1}^{j} p([\sigma', j]||[e, m], \cancel{p^s_k, p^e_k, A_{qk}})\, p(m = e_{qk}|p^e_{qk}, p^e_k, A_{qk}) \tag{B.45}$$

To simplify this expression, note

- $p([\sigma', j]||[e, m], p^s_k, p^e_k, A_{qk}) = p([\sigma', j]||[e, m])$, since from $m \leq j$ it follows that the alignment which ends at m does not overlap j.
- $p(m = e_{qk}|p^e_{qk}, p^e_k, A_{qk}) = p(m = e_{qk}|{p^e_{qk}}')$, since the information from the domain end probabilities of t_k is transferred to the domain end probabilities of the query q by replacing p^e_{qk} with ${p^e_{qk}}'$ defined in equation (2.15)

Equation (B.15) therefore becomes

$$\frac{\sum_{(1)}}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})} = \sum_{m=1}^{j} p([\sigma',j]|[e,m])\, {p_{qk}^e}'(m) \tag{B.46}$$

Calculation of $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_{qk}^s, p_{qk}^e, p_k^s, p_k^e, A_{qk})$ **for** $l \le j < i \le m$

We modify the second partial sum (equation (B.15)) in a similar way by conditioning on p_k^s, p_k^e, A_{qk} and simplifying the resulting expressions:

$$\begin{aligned}
\frac{\sum_{(2)}}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})} &= \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk}, p_k^s, p_k^e, A_{qk})\, p(l=s_{qk}|p_{qk}^s, p_k^s, p_k^e, A_{qk}) \\
&\quad \times \sum_{m=i}^{L_q} \frac{p(m=e_{qk}|[\sigma,i]_\nu, p_k^s, p_k^e, A_{qk})}{p(m=e_{qk}|l=s_{qk}, p_k^s, p_k^e, A_{qk})}\, p(m=e_{qk}|p_{qk}^e, p_k^s, p_k^e, A_{qk}) \\
&= p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk}) \sum_{l=1}^{j} p([\sigma',j]|l=s_{qk}, r_{qk}^{\lfloor\sigma\rfloor})\, {p_{qk}^s}'(l) \\
&\quad \times \sum_{m=i}^{L_q} \frac{p(m=e_{qk}|[\sigma,i]_\nu, r_{qk}^e)}{p(m=e_{qk}|l=s_{qk}, r_{qk}^e)}\, {p_{qk}^e}'(m)
\end{aligned} \tag{B.47}$$

In the second step we used the fact that $l \le j < i \le m$ and hence i and j both overlap with the alignment A_{qk}.

Using the definition of $f_{\sigma|\sigma'}(i|j, r_{qk})$, the second partial sum can be written

$$\begin{aligned}
&\frac{\sum_{(2)}}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})} = \\
&\qquad \sum_{l=1}^{j} f_{\sigma'|s}(j|l, r_{qk})\, p(l=s_{qk}|{p_{qk}^s}') \sum_{m=i}^{L_q} \frac{f_{e|\sigma}(m|i, r_{qk})}{f_{e|s}(m|l, r_{qk})}\, p(m=e_{qk}|{p_{qk}^e}')
\end{aligned} \tag{B.48}$$

Calculation of $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_{qk}^s, p_{qk}^e, p_k^s, p_k^e, A_{qk})$ **for** $j < i \le l < m$

In the third partial sum the alignment does not overlap with i and j. We obtain

$$\frac{\sum_{(3)}}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})} = q^{\lambda'-\lambda} \sum_{l=i}^{L_q} p([\sigma,i]|l=s_{qk})\, {p_{qk}^s}'(l) \tag{B.49}$$

Calculation of $p([\sigma',j]_{\nu-1}|p^s_{qk}, p^e_{qk}, p^s_k, p^e_k, A_{qk})$

The partial sums in the denominator (see equation (B.44)) can be derived with similar modifications:

$$\sum\nolimits_{(1')} = \sum_{m=1}^{j} p([\sigma', j]|m = e_{qk})\, p^e_{qk}{}'(m) \tag{B.50}$$

$$\sum\nolimits_{(2')} = \sum_{l=1}^{j} p([\sigma', j]|l = s_{qk}, r_{qk})\, p^s_{qk}{}'(l) \sum_{m=j}^{L_q} \frac{p(m = e_{qk}|[\sigma', j], r_{qk})}{p(m = e_{qk}|l = s_{qk}, r_{qk})}\, p^e_{qk}{}'(m) \tag{B.51}$$

$$\sum\nolimits_{(3')} = \sum_{l=j}^{L_q} p([\sigma', j]|l = s_{qk})\, p^s_{qk}{}'(l) \tag{B.52}$$

Calculation of $f_{\sigma|\sigma'}(i|j, r_{qk}) := p([\sigma, i]|[\sigma', j], r^{\lfloor\sigma\rfloor}_{qk})$

The generalized version of the $f_{\sigma|\sigma'}(i - l)$

$$f_{\sigma|\sigma'}(i|j, r_{qk}) := p([\sigma, i]|[\sigma', j], r^{\lfloor\sigma\rfloor}_{qk}) \tag{B.53}$$

can be computed iteratively:

$$\boxed{\begin{aligned} f_{\sigma|\sigma'}(i|i, r_{qk}) &= I(\sigma = \sigma') \\ f_{\sigma|\sigma'}(i|j, r_{qk}) &= \sum_{\sigma'' \in \mathrm{pred}(\sigma)} \sum_{l=\max\{i-L_{\max},0\}} p([\sigma,i]_\nu|[\sigma'',j]_{\nu-1}, r^{\lfloor\sigma\rfloor}_{qk})\, f_{\sigma''|\sigma'}(l|j, r_{qk}) \end{aligned}} \tag{B.54}$$

Calculation of $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, r^{\lfloor\sigma\rfloor}_{qk})$

We make the modeling assumption that $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, r^{\lfloor\sigma\rfloor}_{qk}) \propto p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}) r^{\lfloor\sigma\rfloor}_{qk}(i)$. Since this probability must sum to 1 over all $i \in \{i : j < i \leq L_Q + 1\}$ and all $\sigma \in \mathrm{succ}(\sigma')$, we obtain

$$\boxed{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, r^{\lfloor\sigma\rfloor}_{qk}) \approx \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})\, r^{\lfloor\sigma\rfloor}_{qk}(i)}{\sum_{\sigma'' \in \mathrm{succ}(\sigma')} \sum_{i'=j+1}^{L_q+1} p([\sigma'', i']_\nu|[\sigma',j]_{\nu-1})\, r^{\lfloor\sigma''\rfloor}_{qk}(i')}} \tag{B.55}$$

Calculation of $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_k, p^e_k, A_{qk})$

We can expand $p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p^s_k, p^e_k, A_{qk})$ by the sum rule for the case in which the alignment A_{qk} covers i, that is to say $s_{qk} \leq i \leq e_{qk}$, and the case in which it does not cover i:

$$
\begin{aligned}
&p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk}) \\
&\quad = p([\sigma,i]_\nu|s_{qk}\leq i\leq e_{qk}, [\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})\, p(s_{qk}\leq i\leq e_{qk}|A_{qk}) \\
&\qquad + p([\sigma,i]_\nu|\neg(s_{qk}\leq i\leq e_{qk}), [\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})\, (1-p(s_{qk}\leq i\leq e_{qk}|A_{qk})) \\
&\quad = p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, r_{qk}^{\lfloor\sigma\rfloor})\, r_{qk}^*(i) + p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})(1-r_{qk}^*(i))
\end{aligned}
\tag{B.56}
$$

Calculation of $R_{qk}(\sigma, i|\sigma', j)$ - final

$$
\boxed{R_{qk}(\sigma, i|\sigma', j) = S_{qk}(\sigma, i|\sigma', j)\, \frac{\sum_{(1)}+\sum_{(2)}+\sum_{(3)}}{\sum_{(1')}+\sum_{(2')}+\sum_{(3')}}}
\tag{B.57}
$$

with equation (B.56):

$$
\boxed{S_{qk}(\sigma, i|\sigma', j) := \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, p_k^s, p_k^e, A_{qk})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1},)} = \frac{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1}, r_{qk}^{\lfloor\sigma\rfloor})}{p([\sigma,i]_\nu|[\sigma',j]_{\nu-1})}\, r_{qk}^*(i) + 1 - r_{qk}^*(i)}
\tag{B.58}
$$

B.4.2 Sequence weighting during consistency iterations

Analogous to section B.2.3, the redundancy within the set of profiles $r_{q,1\ldots K}^\sigma$ can be corrected by weights

$$
\boxed{w_{qk}'^\sigma = \left[\sum_{k'=1}^{K} \frac{\sum_{i=1}^{L_q}(r_{qk}^\sigma\, r_{qk'}^\sigma)^{\frac{1}{2}}}{(\sum_i r_{qk}^\sigma)^{\frac{1}{2}}\,(\sum_i r_{qk'}^\sigma)^{\frac{1}{2}}}\right]^{-1}}
\tag{B.59}
$$

Contrary to the case in section B.2.3, the normalization is necessary, since the r_{qk}^σ are *not* normalized over i. Again, $1 \leq (w_{qk}'^\sigma)^{-1} \leq K$ and therefore $\frac{1}{K} \leq w_{qk}'^\sigma \leq 1$. We can again generalize this using the parameter α_w as before:

$$
\boxed{w_{qk}'^\sigma = \left[\sum_{k'=1}^{K} \frac{\sum_{i=1}^{L_q}(r_{qk}^\sigma\, r_{qk'}^\sigma)^{\alpha_w}}{(\sum_i (r_{qk}^\sigma)^{2\alpha_w})^{\frac{1}{2}}\,(\sum_i (r_{qk'}^\sigma)^{2\alpha_w})^{\frac{1}{2}}}\right]^{-1}}\,.
\tag{B.60}
$$

$w_{qk}'^\sigma$ replaces w_{qk}^σ in the calculation of $g_q(\sigma, i|\sigma', j)$ for the consistency iterations.

Bibliography

[1] Celery. *http://www.celeryproject.org/*.

[2] Django. *https://www.djangoproject.com/*.

[3] Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen. *https://www.gwdg.de/*.

[4] HHsuite at GitHub. *https://github.com/soedinglab/hh-suite*.

[5] HHsuite userguide. *https://github.com/soedinglab/hh-suite/blob/master/hhsuite-userguide.pdf*.

[6] MySQL. *https://www.mysql.de/*.

[7] Redis. *http://redis.io/*.

[8] Travis CI. *https://travis-ci.org/*.

[9] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

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

[11] Alex Bateman, Maria Jesus Martin, Claire O'Donovan, Michele Magrane, Rolf Apweiler, Emanuele Alpi, Ricardo Antunes, Joanna Arganiska, Benoit Bely, Mark Bingley, Carlos Bonilla, Ramona Britto, Borisas Bursteinas, Gayatri Chavali, Elena Cibrian-Uhalte, Alan Da Silva, Maurizio De Giorgi, Tunca Dogan, Francesco Fazzini, Paul Gane, Leyla Garcia Castro, Penelope Garmiri, Emma Hatton-Ellis, Reija Hieta, Rachael Huntley, Duncan Legge, Wudong Liu, Jie Luo, Alistair Macdougall, Prudence Mutowo, Andrew Nightingale, Sandra Orchard, Klemens Pichler, Diego Poggioli, Sangya Pundir, Luis Pureza, Guoying Qi, Steven Rosanoff, Rabie Saidi, Tony Sawford, Aleksandra Shypitsyna, Edward Turner, Vladimir Volynkin, Tony Wardell, Xavier Watkins, Hermann Zellner, Andrew Cowley, Luis Figueira, Weizhong Li, Hamish McWilliam, Rodrigo Lopez, Ioannis Xenarios, Lydie Bougueleret, Alan Bridge, Sylvain Poux, Nicole Redaschi, Lucila Aimo, Ghislaine Argoud-Puy, Andrea Auchincloss,

Kristian Axelsen, Parit Bansal, Delphine Baratin, Marie Claude Blatter, Brigitte Boeckmann, Jerven Bolleman, Emmanuel Boutet, Lionel Breuza, Cristina Casal-Casas, Edouard De Castro, Elisabeth Coudert, Beatrice Cuche, Mikael Doche, Dolnide Dornevil, Severine Duvaud, Anne Estreicher, Livia Famiglietti, Marc Feuermann, Elisabeth Gasteiger, Sebastien Gehant, Vivienne Gerritsen, Arnaud Gos, Nadine Gruaz-Gumowski, Ursula Hinz, Chantal Hulo, Florence Jungo, Guillaume Keller, Vicente Lara, Philippe Lemercier, Damien Lieberherr, Thierry Lombardot, Xavier Martin, Patrick Masson, Anne Morgat, Teresa Neto, Nevila Nouspikel, Salvo Paesano, Ivo Pedruzzi, Sandrine Pilbout, Monica Pozzato, Manuela Pruess, Catherine Rivoire, Bernd Roechert, Michel Schneider, Christian Sigrist, Karin Sonesson, Sylvie Staehli, Andre Stutz, Shyamala Sundaram, Michael Tognolli, Laure Verbregue, Anne Lise Veuthey, Cathy H. Wu, Cecilia N. Arighi, Leslie Arminski, Chuming Chen, Yongxing Chen, John S. Garavelli, Hongzhan Huang, Kati Laiho, Peter McGarvey, Darren A. Natale, Baris E. Suzek, C. R. Vinayaka, Qinghua Wang, Yuqi Wang, Lai Su Yeh, Meher Shruti Yerramalla, and Jian Zhang. UniProt: A hub for protein information. *Nucleic Acids Research*, 43(D1):204–212, 2015.

[12] M J Bennett, M P Schlunegger, and D Eisenberg. 3D domain swapping: a mechanism for oligomer assembly. *Protein Science*, 4(12):2455–2468, 1995.

[13] Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. GenBank. *Nucleic Acids Research*, 41(D1):36–42, 2013.

[14] H M Berman, J Westbrook, Z Feng, G Gilliland, T N Bhat, H Weissig, I N Shindyalov, and P E Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[15] Peer Bork. Shuffled domains in extracellular proteins. *FEBS Lett.*, 286(1-2):47–54, 1991.

[16] S E Brenner. Target selection for structural genomics. *Nature structural & molecular biology*, 7:967–969, 2000.

[17] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[18] I. D. Campbell and A. Kristina Downing. Building protein structure and function from modular units. *Trends in Biotechnology*, 12(5):168–172, 1994.

[19] Lusheng Chen, Wei Wang, Shaoping Ling, Caiyan Jia, and Fei Wang. KemaDom: A web server for domain prediction using kernel machine with local context. *Nucleic Acids Research*, 34(Web Server issue):158–163, 2006.

[20] Dylan Chivian, David E. Kim, Lars Malmstrom, Philip Bradley, Timothy Robertson, Paul Murphy, Charles E M Strauss, Richard Bonneau, Carol A. Rohl, and David Baker. Automated Prediction of CASP-5 Structures Using the Robetta Server. *Proteins: Structure, Function and Genetics*, 53:524–533, 2003.

[21] O Dekel, S Shalev-Shwartz, and Y Singer. Smooth epsilon-insensitive regression by loss symmetrization. *Journal of Machine Learning Research*, 6:711–741, 2005.

[22] RF Doolittle. The multiplicity of domains in proteins. *Annual review of biochemistry*, 64:287–314, 1995.

[23] Michel Dumontier, Rong Yao, Howard J. Feldman, and Christopher W V Hogue. Armadillo: Domain boundary prediction by amino acid composition. *Journal of Molecular Biology*, 350(5):1061–1073, 2005.

[24] Teppei Ebina, Hiroyuki Toh, and Yutaka Kuroda. Loop-length-dependent SVM prediction of domain linkers for high-throughput structural proteomics. *Biopolymers*, 92(1):1–8, 2009.

[25] Jesse Eickholt, Xin Deng, and Jianlin Cheng. DoBo: Protein domain boundary prediction by integrating evolutionary signals and machine learning. *BMC bioinformatics*, 12(1), 2011.

[26] Robert D Finn, Jaina Mistry, John Tate, Penny Coggill, Andreas Heger, Joanne E Pollington, O Luke Gavin, Prasad Gunasekaran, Goran Ceric, Kristoffer Forslund, Liisa Holm, Erik L L Sonnhammer, Sean R Eddy, and Alex Bateman. The Pfam protein families database. *Nucleic Acids Research*, 2010.

[27] Naomi K. Fox, Steven E. Brenner, and John Marc Chandonia. SCOPe: Structural Classification of Proteins - Extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):304–309, 2014.

[28] Richard a George and Jaap Heringa. SnapDRAGON: a method to delineate protein structural domains from sequence data. *Journal of Molecular Biology*, 316:839–851, 2002.

[29] R. Guerois and L. Serrano. Protein design based on folding models. *Current Opinion in Structural Biology*, 11(1):101–106, 2001.

[30] Andreas Hauser. ffindex at GitHub. *https://github.com/soedinglab/ffindex_soedinglab*.

[31] M Hauser, C E Mayer, and J Söding. kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinformatics*, 14:248, 2013.

[32] Maria Hauser. *MMseqs : ultra fast and sensitive clustering and search of large protein sequence databases*. PhD thesis, 2014.

[33] Maria Hauser, Martin Steinegger, and Johannes Söding. MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330, 2016.

[34] Andreas Heger and Liisa Holm. Exhaustive Enumeration of Protein Domain Families. *Journal of Molecular Biology*, 328(3):749–767, 2003.

[35] Jaap Heringa and William R. Taylor. Three-dimensional domain duplication, swapping and stealing. *Current Opinion in Structural Biology*, 7(3):416–421, 1997.

[36] Andrea Hildebrand, Michael Remmert, Andreas Biegert, and Johannes Söding. Fast and accurate automatic structure prediction with HHpred. *Proteins: Structure, Function and Bioinformatics*, 77(SUPPL. 9):128–132, 2009.

[37] Timothy A. Holland, Stella Veretnik, Ilya N. Shindyalov, and Philip E. Bourne. Partitioning Protein Structures into Domains: Why Is it so Difficult? *Journal of Molecular Biology*, 361(3):562–590, 2006.

[38] Takayuki Hondoh, Atsushi Kato, Shigeyuki Yokoyama, and Yutaka Kuroda. Computer-aided NMR assay for detecting natively folded structural domains. *Protein Science*, 15(4):871–883, 2006.

[39] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden Markov model speed heuristic and iterative HMM search procedure. *BMC bioinformatics*, 11:431, 2010.

[40] Scott D. Kahn. On the Future of Genomic Data. *Science*, 331(6018):728 –729, 2011.

[41] David E. Kim, Dylan Chivian, Lars Malmstrom, and David Baker. Automated prediction of domain boundaries in CASP6 targets using Ginzu and RosettaDOM. *Proteins: Structure, Function and Vioinformatics*, 61(S7):193–200, 2005.

[42] Sarah K Kummerfeld and Sarah A. Teichmann. Relative rates of gene fusion and fission in multi-domain proteins. *Trends in Genetics*, 21(1):25–30, 2005.

[43] Jinfeng Liu and Burkhard Rost. Domains, motifs and clusters in the protein universe. *Current Opinion in Chemical Biology*, 7:5–11, 2003.

[44] Alexey G. Murzin, Steven E. Brenner, Tim Hubbard, and Cyrus Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[45] Peter K. Nielsen and Yoshihiko Yamada. Identification of Cell-binding Sites on the Laminin a5 N-terminal Domain by Site-directed Mutagenesis. *Journal of Biological Chemistry*, 276(14):10906–10912, 2001.

[46] Chin I. Pang, Kuang Lin, Merridee A. Wouters, Jaap Heringa, and Richard A. George. Identifying foldable regions in protein sequence from the hydrophobic signal. *Nucleic Acids Research*, 36(2):578–588, 2008.

[47] Sophie Pasek, Jean Loup Risler, and Pierre Brézellec. Gene fusion/fission is a major contributor to evolution of multi-domain bacterial proteins. *Bioinformatics*, 22(12):1418–1423, 2006.

[48] Jian Peng and Jinbo Xu. Low-homology protein threading. *Bioinformatics*, 26(12):294–300, 2010.

[49] N. Perdigao, J. Heinrich, C. Stolte, K. S. Sabir, M. J. Buckley, B. Tabor, B. Signal, B. S. Gloss, C. J. Hammang, B. Rost, A. Schafferhans, and S. I. O'Donoghue. Unexpected features of the dark proteome. *Proceedings of the National Academy of Sciences*, 112(52):15898–15903, 2015.

[50] Elon Portugaly, Amir Harel, Nathan Linial, and Michal Linial. EVEREST: automatic identification and classification of protein domains in all protein sequences. *BMC bioinformatics*, 7(1), 2006.

[51] Bhanu Rekapalli, Kristin Wuichet, Gregory D Peterson, and Igor B Zhulin. Dynamics of domain coverage of the protein sequence universe. *BMC genomics*, 13(1):634, 2012.

[52] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2):173–175, 2011.

[53] Torbjørn Rognes. Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. *BMC bioinformatics*, 12(1), 2011.

[54] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, 2008.

[55] I. Sillitoe, T. E. Lewis, A. Cuff, S. Das, P. Ashford, N. L. Dawson, N. Furnham, R. A. Laskowski, D. Lee, J. G. Lees, S. Lehtinen, R. A. Studer, J. Thornton, and C. A. Orengo. CATH: comprehensive structural and functional annotations for genome sequences. *Nucleic Acids Research*, 43(D1):376–381, 2015.

[56] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[57] Johannes Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[58] Johannes Söding and Michael Remmert. Protein sequence comparison and fold recognition: Progress and good-practice benchmarking. *Current Opinion in Structural Biology*, 21(3):404–411, 2011.

[59] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[60] D Vitkup, E Melamud, J Moult, and C Sander. Completeness in structural genomics. *Nature Structural & Molecular Biology*, 8(6):559–566, 2001.

[61] Christine Vogel, Sarah A. Teichmann, and Jose Pereira-Leal. The relationship between domain duplication and recombination. *Journal of Molecular Biology*, 346(1):355–365, 2005.

[62] Benjamin Webb and Andrej Sali. *Comparative Protein Structure Modeling Using MODELLER*. 2014.

[63] Donald B. Wetlaufer. Nucleation, rapid folding, and globular intrachain regions in proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 70(3):697–701, 1973.

[64] Yinghao Wu, Athanasios D Dousis, Mingzhi Chen, Jialin Li, and Jainpeng Ma. OPUS-Dom: Applying the Folding-Based Method VECFOLD to Determine Protein Domain Boundaries. *Journal of Molecular Biology*, 385(4):1314–1329, 2009.

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

[66] Y Xu, D Xu, and H N Gabow. Protein domain decomposition using a graph-theoretic approach. *Bioinformatics*, 16(12):1091–1104, 2000.

[67] Corin A Yeats and Christine A Orengo. Evolution of Protein Domains. *Encyclopedia of Life Sciences*, 2007.

[68] Christine Zardecki, Shuchismita Dutta, David S. Goodsell, Maria Voigt, and Stephen K. Burley. RCSB Protein Data Bank: A Resource for Chemical, Biochemical, and Structural Explorations of Large and Small Biomolecules. *Journal of Chemical Education*, 93:569–575, 2016.

[69] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function and Genetics*, 57(4):702–710, 2004.

Acknowledgment

I would like to express my special appreciation and thanks to my advisor Dr. Johannes Söding, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on research has been invaluable. A special thanks to my family. Words can not express how grateful I am to my mother, my father and my sisters for all the sacrifices that you have made on my behalf.