THE UNIVERSITY OF CHICAGO

COMPUTATIONAL ANALYSIS OF PROTEIN MODULAR DOMAIN ARCHITECTURES

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
GÜLRİZ AYTEKİN-KURBAN

CHICAGO, ILLINOIS
JUNE, 2011
ACKNOWLEDGEMENTS
TABLE OF CONTENTS

ACKNOWLEDGEMENTS .............................................. ii
LIST OF ILLUSTRATIONS ........................................... v
LIST OF TABLES .................................................. vi

CHAPTER

1. INTRODUCTION ............................................... 1
   1.1 Overview ................................................. 1
   1.2 Thesis Statement ....................................... 8
   1.3 Summary of the Approach ............................... 8
   1.4 Overview of Chapters ................................. 9

2. BACKGROUND IN PREDICTING PROTEIN DOMAINS ........... 11
   2.1 Protein Domain Families ............................... 11
      2.1.1 Structural domain families ..................... 11
      2.1.2 Sequence domain families ....................... 12
   2.2 Predicting The Modular Domain Architecture (MDA) of a Protein 15

3. PREDICTING THE MDA OF A PROTEIN ......................... 18
   3.1 Overview ................................................. 18
   3.2 Examples of Local Alignments Predicting MDA ........... 19
      3.2.1 PTF A_ECOLI: Two Functional and One Structural Domain 20
      3.2.2 1b70: A protein with seven structural domains .... 25
   3.3 Building a Probabilistic Model for a Protein’s MDA .... 30
      3.3.1 Formal description of the alignment data ........ 30
      3.3.2 Domain Signature Model(DSM) ..................... 30
   3.4 Computing an Optimal DSM for a Protein ............... 33
      3.4.1 Maximum Likelihood Estimation(MLE) ............. 34
      3.4.2 Deterministic annealing applied to clustering .... 40
   3.5 The Results for DSM Approach on the Examples ........ 46
      3.5.1 Example 1: Protein PTF A_ECOLI .................. 46
      3.5.2 Example 2: Protein 1b70 Chain B ............... 51
   3.6 Large Scale Tests Using PDB Proteins .................. 54
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The structure drawings of Phenylalanyl-tRNA synthetase</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>The structure drawings of Phenylalanyl-tRNA synthetase (continued)</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Domains on PTF_A_ECOLI</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>PTF_A_ECOLI sequence segments</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>A detail from the PTF_A_ECOLI alignments</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>The SVM analysis for PTF_A_ECOLI</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>Domains in the protein Phenylalanine tRNA Synthetase</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>The PFAM domains in 1B70_B</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>1B70 Chain B sequence segments in each local alignment</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>The EM algorithm</td>
<td>39</td>
</tr>
<tr>
<td>11</td>
<td>The DAEM algorithm</td>
<td>45</td>
</tr>
<tr>
<td>12</td>
<td>PTF_A_ECOLI: $\beta = 0.01$ with two classes</td>
<td>48</td>
</tr>
<tr>
<td>13</td>
<td>PTF_A_ECOLI: $\beta = 0.011$, three classes</td>
<td>49</td>
</tr>
<tr>
<td>14</td>
<td>PTF_A_ECOLI: $\beta = 1$</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>1B70 B: The DSM has five classes</td>
<td>52</td>
</tr>
<tr>
<td>16</td>
<td>1B70 B: The DSM has six classes</td>
<td>53</td>
</tr>
<tr>
<td>17</td>
<td>The results for two-domain proteins</td>
<td>57</td>
</tr>
<tr>
<td>18</td>
<td>The results for three-domain proteins</td>
<td>58</td>
</tr>
<tr>
<td>19</td>
<td>The results for four-domain proteins</td>
<td>59</td>
</tr>
<tr>
<td>20</td>
<td>Example protein 1KHB_A</td>
<td>60</td>
</tr>
<tr>
<td>21</td>
<td>Example protein 1KKH_A</td>
<td>61</td>
</tr>
<tr>
<td>22</td>
<td>Example protein 1WOO_A</td>
<td>63</td>
</tr>
<tr>
<td>23</td>
<td>Example protein 1UAS_A</td>
<td>64</td>
</tr>
<tr>
<td>24</td>
<td>Example protein 1LNX_B</td>
<td>65</td>
</tr>
<tr>
<td>25</td>
<td>The domain linker sequence for 1B70_B</td>
<td>70</td>
</tr>
<tr>
<td>26</td>
<td>The DA-HMM</td>
<td>76</td>
</tr>
<tr>
<td>27</td>
<td>Posterior probabilities for 1KHB_A</td>
<td>81</td>
</tr>
<tr>
<td>28</td>
<td>Posterior probabilities for 1KKH_A</td>
<td>82</td>
</tr>
<tr>
<td>29</td>
<td>Posterior probabilities for 1WOO_A</td>
<td>83</td>
</tr>
<tr>
<td>30</td>
<td>Posterior probabilities for 1UAS_A</td>
<td>84</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
</tbody>
</table>

LIST OF TABLES
CHAPTER 1
INTRODUCTION

1.1 Overview

This thesis contributes a set of methods for the preliminary analysis of a protein’s domain architecture. My methods predict protein domains by exploiting protein amino acid sequence similarity. Identifying domains in a protein involves bringing together structural, functional, and evolutionary information, which may require expert knowledge \[11\]. Fortunately, multiple properties of domains make computational approaches possible. Using domain segments in multi-domain proteins, rather than the whole proteins, can strengthen various experimental and computational protein analysis endeavors, such as selecting targets for structural genomics projects, annotating proteins with similar evolutionary origins, and improving the prediction of protein structure algorithms, such as threading.

Nomenclature

A protein is a complex molecule formed by one or more chains of amino acids. Proteins are the building blocks of cellular structure. They perform the majority of cellular functions, and make up most of a cell’s dry mass. Proteins acting as enzymes catalyze metabolic reactions. Proteins also transport other molecules (e.g., hemoglobin), regulate cellular processes (e.g., insulin), and take part in translating the DNA code. Interactions among the atoms of a protein and its environment force it to fold into a particular three-dimensional(3D) structure, which ultimately determine the protein’s function.\(^1\)

\(^1\)Protein folding is a result of intermolecular forces such as hydrogen bonds, ionic and hydrophobic interactions.
A protein domain was originally defined as a compact substructure of a protein that can have a stable 3D form independently from the rest of the protein [41]. See, for example, Figure 1 which shows an abstract structure drawing of a protein Phenylalanyl-tRNA synthetase from the organism Thermus aquaticus [40]. This protein consists of multiple chains of two kinds. One of these chains, Figure 1(b), has six compact substructures, namely domains, that fold independently.

Studies of structural domains have shown that useful structural properties such as stability, compactness and independence are likely to have transformed the domains into evolutionary units [32]. Throughout the protein universe, they can be observed as recurring modules of conserved amino acid sequence, distinct structure and function [6,8]. The recurrence of domains with the same structure and function strengthens the domain definition. For instance, Ponting et al. [38] mention the protein lysozyme that was initially classified as having two structural domains. Later, when more lysozyme structures were determined, all were considered to be single domain proteins which were variations from an ancestral 3D fold.

A domain family is a set of recurring domains with the same structure and function. On a 3D protein structure, it is possible to compute whether a substructure is compact and independent as well as whether similar substructures recur in other proteins. In the absence of structural information, we can establish domain families based on the recurrence of conserved amino acid sequence segments that are thought to possess common ancestry, i.e., homologs, across the protein space. The sequence domains usually correspond to structural domains with some exceptions.3

The modular domain architecture (MDA) of a protein refers to its domain composition, i.e., the number, type and positions of the domains. A protein may

---

2 Some researchers reserve the use of the term “domain” only for structural units while they use the term “module” for recurring units of conserved protein sequence.

3 Multiple congruent structural domains that often recur together could be predicted as a single sequence domain because it appears as a single evolutionary unit.
contain a single or multiple distinct or repeating domains. Many domains in a protein have a specific biochemical function, and working together, they contribute to the protein’s complex cellular function. Some domains do not have functions but are rather structural units that provide a scaffold to fold around. It is also possible that a functional site in a protein lies in the interface of two structural domains and the two domains act together to perform a single function.

The identification of a protein’s MDA is a prerequisite to experimental analysis such as determining the protein’s three-dimensional structure and further computational analysis such as threading to predict the structure. I will review the practical applications of my methods for MDA prediction in Chapter 6. Here, I will quote Ponting & Russell

In recent decades the concepts of domains and domain families have risen to greater prominence within science. This has been due to an increasing realization that division of protein’s structure and sequence into domains often precedes reliable and accurate predictions of molecular function. A view of a multidomain protein’s as the sum of its constituent parts is obviously simplistic, as it ignores possible interdomain interactions and cooperative effects. Nevertheless this view does provide a “first-approximation” that is amenable to investigation and subsequence refinement using experimental approaches.

The MDA of an example protein. To give a clear picture of multi-domain protein architecture, let us look more closely at the protein Phenylalanyl-tRNA example. Phenylalanyl-tRNA synthetase attaches an amino acid Phenylalanine to a transfer RNA. The synthetase should bring three substrates together: an amino acid Phenylalanine, a tRNA specific for the Phenylalanine, and an ATP molecule for the energy to drive the synthesis. Figure 2 (a) shows a domain whose function is to catalyze the enzymatic action, i.e., the binding of Phenylalanine to the

---

4 There are also fused proteins where each domain has a related function but are not parts of a single complex function.

5 Transfer RNAs carry amino acids used in protein synthesis to the ribosomes. Each type of tRNA molecule carries a particular amino acid. “Transfer RNAs are a set of small RNA molecules used in protein synthesis as an interface (adapter) between messenger RNA and amino acids. Each type of tRNA molecule is covalently linked to a particular amino acid.” [1]
tRNA (with the use of an ATP). Two other individual domains are shown in (c) and (d). The domain in (c) binds to an anticodon for Phenylalanine on the tRNA to identify it correctly, the one in (d) is another tRNA binding domain whose function is thought to further guide the tRNA. Other domains in the protein do not have any known functional sites and are predicted to provide a structural scaffold.

**Amino acid sequences and homology.** The protein amino acid sequence analysis is probably the oldest computational approach to predicting protein structure and function. Because DNA sequencing technology is well advanced and it is easy to obtain a protein’s amino acid sequence by translating the gene that encodes it, on the DNA, the sequence analysis has stayed as the most popular method of preliminary analysis. Each genome project brings out an almost complete DNA sequence of an organism, which enables a large subset of protein coding genes to be easily predicted and translated to the amino acid sequences of proteins. The next step is to predict the function and structure of these proteins.

The computational structure and function prediction usually starts with amino acid sequence. The main assumption underlying sequence analysis is that if two proteins or domains share significant sequence similarity, they must have evolved from a common ancestor, i.e., they are homologs. It is highly unlikely for nonhomologous proteins to converge to similar sequences through evolution. As homologous proteins evolve away from each other by random mutations of their amino acid sequences, the function and structure are usually preserved. Homology lets us conclude which proteins have similar structure and function and to group them into families. Consequently, using sequence similarity, which implies homology by the above assumption, we can cluster proteins into groups that help us predict the structure and the function of the group members. Such clustering becomes complicated when proteins have multiple domains with distinct ancestry, because the homology between two multi-domain proteins can involve only some

---

6 Each amino acid has a specific tRNA.
Figure 1: The structure drawings of Phenylalanyl-tRNA synthetase. The ribbon representation drawings are obtained from PDB database website. The top drawing shows a naturally assumed complex that consists of four disconnected chains of amino acids, i.e., it is a tetramer. There are two symmetrical substructures, each contains two chains $\alpha$ and $\beta$. The bottom drawing is one of the substructures, its $\alpha$ and $\beta$ chains are further shown in Figure 2.
Figure 2: The Structure drawings of Phenylalanyl-tRNA synthetase (continued). The drawings are obtained from CATH database website. (a) chain $\alpha$ is a single catalytic domain, (b) chain $\beta$ contains six structurally identified domains some of which also has a specific function, (c) anticodon binding domain, (d) tRNA binding domain. This example will further serve as a demonstration in Chapter 3.
of their domains. In our work, we exploit the distinct sets of homologs for each domain to predict an MDA for a protein.

Multi-domain proteins are believed to have evolved through gene duplication, gene fusion, exon shuffling and deletion, resulting in the novel combinations of existing domains. For instance, two multi-domain proteins may have a single domain from the same family. Enzymes involved in oxidation reactions, e.g., alcohol dehydrogenase, use NAD+ (Nicotinamide Adenine Dinucleotide) coenzyme. These proteins all have common NAD+ binding domains beside distinct catalytic domains specific to each enzymatic function.

We observe that the evolutionary process that created multi-domain proteins tends to respect the integrity of domains. If this had not been the case, the resulting protein in all likelihood would not have been functional and not have survived. Our expectation is that the process of evolution tend to conserve domains as atomic units albeit subject to random mutations that change the amino acid sequence. The domains themselves are recombined in descendant proteins, which makes it possible to isolate domains using local sequence similarity. The methods presented in this thesis particularly make use of the multiple distinct recombinations that can be observed in local sequence alignments of a protein against a larger set of database proteins.

However, computational approaches based on sequence similarity, including the methods presented in my thesis, rely on the expectation that we can detect and model sequence similarity among homologous proteins. Consequently, they run into trouble when sequence similarity between proteins is no longer detectable, i.e., deemed no more significant than the similarity between two random proteins. Moreover, since the evolution tends to preserve amino acid sequence of the domain core and less likely do so around domain boundaries, the local alignments tends to get fuzzy around boundaries. This presents a major technical challenge that we needed to overcome.

\footnote{Domain recombination was a major source for creating new proteins of extracellular and membrane functions in multicellular organisms \cite{38}.}
1.2 Thesis Statement

This thesis presents a novel computational method to identify domains in multi-domain protein sequences. I demonstrate that a protein’s modular domain architecture can be derived from the occurrence pattern that each domain’s homologs display. Such a pattern is present in the protein’s local sequence alignments to other similar proteins. The probabilistic modeling approach introduced here predicts a modular domain architecture as a randomized partition of a protein’s amino acid sequence positions from its local alignments. My approach makes no attempt to build any sequence conservation models for individual domains found in a protein. Rather, domain architecture shows us the modular structure of a protein. A preliminary determination of domain positions helps the experimental and computational structure determination. The end product of my research is a tool that can be used to select targets for structural genomics projects, annotate proteins for each individual domain, and divide the protein sequence into domains so that protein structure prediction algorithms such as threading can have better success using domain segments rather than whole sequences.

1.3 Summary of the Approach

The instances of the same domain in multiple combinations can be observed as locally conserved regions in multi-domain proteins. Two distinct domains in a protein are likely to occur in combinations in two distinct sets of proteins. We will call the set of proteins that has an instance of a domain occur as the signature of the domain. As the initial step, we use a multi-domain protein as a query to a protein sequence database using a search tool and collect all proteins that share sequence similarity, namely hits. Given a multi-domain protein and its local alignments to a set of database proteins, the amino acid sequence positions of the protein participate in a subset of the local alignments depending on which domain the positions belong. For every position, we will also call such subset of
local alignments the *signature* of the position. From a set of local alignments, we can extract signatures for individual sequence positions. A pair of positions that belong to the same domain should have similar signatures and any two positions that belong to distinct domains should have different signatures.

The set of position signatures extracted from the local alignments are our observations. We will number the hit proteins from the database from 1 to $N$. Let us represent each signature as a 0/1 vector of size $N$ such that if a position is in a local alignment with the $n^{th}$ protein, the vector has 1 in the $n^{th}$ position, 0 otherwise. Each signature vector is a point in N-dimensional space where the signatures of positions that belong to the same domain cluster together. I present a probabilistic modeling approach to predict a modular domain architecture (MDA) for a multi-domain protein by finding the clusters of signatures. The approach computes the MDA as a randomized partition of a protein’s amino acid positions.

The approach makes no attempt to build any sequence conservation models for individual domains found in a protein. Rather, domain architecture shows us the modular structure of a protein. In two applications, automated function annotation and experimental structure determination, quick computational methods are needed to predict possible domain positions. Moreover, building meaningful conservation models requires a well-chosen set of family instances and careful manual examination of alignments; the applications here only need to find domain positions to facilitate subsequent analysis.

## 1.4 Overview of Chapters

This thesis addresses the problem of predicting the modular domain architecture of a single multi-domain protein sequence. The examples presented throughout the thesis are all real world examples. Although, this may make the text a little harder to read by a computer scientist, in bioinformatics and computational biology where biological results matter, it is essential.
Chapter 2 provides a review of the protein domain databases and methods to predict domains in a protein.

Chapter 3 starts with the description of Modular Domain Architecture (MDA) prediction. First, it introduces examples to have a clear picture of the problem. It continues with the description of a probabilistic model employed. Then, it presents the results obtained for the examples in detail. The chapter concludes with the results of the application of the method to a set of proteins whose modular domain architecture is already known.

Chapter 4 describes further improvements over the main method presented in Chapter 3. We developed a new model that is a product of the model developed in Chapter 3 and another model developed by Bae et al. [5]. The results show that new model is able to overcome one of problems we faced in the DSM method: assigning short segments to switching domains. It improves the prediction accuracy.

Chapter 5 presents a few application areas of MDA prediction in detail.

Chapter 6 is a conclusion about the overall contribution of the thesis.
CHAPTER 2

BACKGROUND IN PREDICTING PROTEIN DOMAINS

This chapter provides a review of the protein domain databases we examined and the existing computational methods to predict protein domains and domain boundaries using amino acids sequences of proteins. I begin with a brief overview of two structural domain databases that we used to measure the success of our methods—SCOP, a database comprising manually curated structural domain families and hierarchy and CATH, a database of structural domain families predicted by computational methods. I then examine protein domain families whose members are protein amino acid sequences or domain sequences, and conclude with a review of existing computational approaches to finding domains and domain boundaries, based on sequence conservation and amino acid composition along the sequence.

2.1 Protein Domain Families

2.1.1 Structural domain families

The Protein Data Bank (PDB) is an archive that collects experimentally-determined three-dimensional (3D) structures of proteins. The two databases that I will review briefly below, use the 3D protein structures stored in PDB to identify and predict structural domains.

1http://www.rcsb.org/pdb/home/home.do
2PDB also stores the structures for other macromolecules, such as RNA.
SCOP. The SCOP database \cite{11} contains protein domains constructed through expert manual decomposition of 3D protein structures into separate domains. The underlying heuristic used in SCOP to identify domains assumes that 3D topologies repeated in more than one protein represent general protein structure folds, or domains. The curated structural domains are then classified into structural domain families whose members have similar 3D structures. The families are further clustered into major fold groups according to the similarity of their folds.

CATH. Protein structure domains can be identified computationally using methods that trace the topology of the structure and partition the 3D structure according to computable topological properties. The CATH \cite{27} database is also a classification of structural domains in the PDB proteins. However, this classification uses both manual and computational methods. The computational part applies three topological algorithms–DETECTIVE \cite{47}, PUU \cite{25}, and DOMAK \cite{44}–to subdivide multi-domain proteins into separate domains. If the three algorithms all agree on similar domain boundaries then CATH accepts the domains for classification into families of domains with similar structural topology.

Because SCOP domains are identified manually by experts, they are considered to be more reliable than CATH or any other automated method, and used as a test set for the automated methods in domain prediction. We used the SCOP database to evaluate our results.

2.1.2 Sequence domain families

In this section, I describe protein domain families whose members are either protein amino acid sequences or domain sequences. These families are built using sequence conservation for a large set of known proteins whose 3D structure are not usually known.

The methods that construct protein sequence families using sequence information are divided into two broad categories–classification and clustering. Pro-
tein sequence classification methods start with a known subset of protein domain family members. A model is then built using common sequence properties, and used to predict additional members. Protein clustering methods attempt to organize a set of proteins into an arbitrary number of families based on sequence similarity.

Protein domain sequence families

Early approaches to building protein family databases (for example COGS, PROTOMAP and SYSTERS \cite{33, 48, 52}) employed clustering methods using pairwise sequence similarity. More recent approaches build clusters by computing a position-specific family profile. For instance, the PRODOM and BIOSPACE databases \cite{21, 51} use PSIBLAST as a fast and statistically sound tool to build a family profile (PSSM). The PICASSO \cite{22} database builds the profiles from the multiple alignments generated for each computed cluster.

Both the BIOSPACE and PICASSO databases capture distant homology and structural similarity among proteins through the hierarchical clustering of the cluster profiles.

Clustering protein sequences into domain families, rather than whole sequence families, increases the accuracy of clusters. Many proteins consist of multiple domains that are both functional and structural units. Each structural domain in a multi-domain protein has a specific 3D fold and belongs to a separate fold family. But, for any two given multi-domain proteins, only a subset of their domains may be shared. If this is the case, the similarities between their structures and between their sequences are local to the shared domains. A clustering method that employs sequence similarity in order to build protein families must delineate segments of a multi-domain protein sequence, where each segment belongs to a separate domain family. Whole sequence clustering places a multi-domain protein into a single family.
In the COGS, PROTOMAP, SYSTERS, and BIOSPACE databases, each protein is placed into a single cluster; hence, a cluster of sequences does not necessarily represent a single family if the members are multi-domain proteins. The approaches that cluster whole sequences [48, 52] acknowledge that there is a need to divide sequences into domains in order to derive a more accurate set of protein families. For instance, the method described in Yona et al. [51, 52] starts with SCOP structural domain profiles as seeds to build Type I clusters. The profiles of Type I clusters are restricted to a single structural domain, even though sequences collected in a Type I cluster may have more domains. Each Type II cluster is built by starting with a single random sequence among remaining unclustered proteins, with no guarantee that it is indeed a single domain. As a result, the families obtained from the unification of Type I and Type II clusters based on profile-profile similarity may contain multi-domain protein sequences. The COGS database [48] integrates domain subsequences by ad hoc computation and manual examination. There are only two databases–PRODOM and PICASSO–where the computed clusters represent predicted domain families.

The PRODOM database [21] contains domain families built by applying a single PSIBLAST database search for a query protein sequence or fragment. For a set of input protein sequences, PRODOM partitions the sequences iteratively by choosing the shortest sequence in the input set as a PSIBLAST query at each iteration. All sequences that align to the query above a cutoff score are removed from the input set and partitioned into fragments along alignment boundaries. Fragments that align with the query sequence are assigned to a domain family. Each remaining unaligned fragment is returned to the input set. Iteration continues until no more sequences in the input set align with each other. Each of the resulting sequence domain families contain conserved sequence fragments of minimal length, based on a single query sequence. PRODOM assumes that the shortest sequence in the database is a complete domain. However, some of the unaligned fragments can be artifacts resulting from low sequence conservation. As the number of unaligned fragments returned to the input set increases, the probability of
the shortest sequence being a complete domain decreases. This can result in the fragmentation of complete domains.

The PICASSO database [22] contains domains built by separating the identification of clusters of sequences and the identification of domains into two sequential steps. Clusters are built hierarchically by comparing the multiple sequence alignment of the sequences in one cluster with the profile of another cluster. After the hierarchical clustering step, each cluster is further processed to detect if it contains more than one domain as follows. A cluster is defined to be “maximal” if all of its sequences are not contained in another cluster. Multi-domain proteins may be assigned to more than one maximal cluster resulting in overlapping clusters. The domain identification step in PICASSO searches for pairs of overlapping maximal clusters. For every pair of overlapping maximal clusters, the method computes the set of column positions for sequence segments that are unique each cluster and the set of column positions for those that are common to both clusters. If the two sets are disjoint, we infer that the segments are distinct domains.

Our goal here is to compute domain boundaries for proteins with no known structure, such that the boundaries are likely to correspond to structural domain boundaries. Each protein sequence can be divided into segments at the boundaries. The resulting segments can then be clustered into families. However, we left the computation of sequence families as future work, it is not included in this thesis.

2.2 Predicting The Modular Domain Architecture (MDA) of a Protein

In this section, we will look at early and current research on predicting the MDA of a single protein. In contrast with the methods described above, the approaches presented in this section do not build families of domains.
George et al. [18] presents an algorithm that computes a scoring function of endpoints with a sliding window across the protein sequence positions. They employ statistical methods for setting parameter values. More importantly, they modify the protein sequence similarity search tool PSI-BLAST such that, after each cycle of search, domain boundaries are identified, and the sequence is cut into its domain segments; the next cycle runs a PSI-BLAST search for each segment separately. While this sort of “cut proteins as you search” approach is similar to PRODOM, the authors note that their approach yields many segments that are too short to be a domain.

Recently, Rigden [42] argued that amino acids of proteins in close proximity within domains should have correlated mutation due to space constraints. The approach clusters the covariance matrix of multiple alignment positions to predict groups of such amino acids, including discontinuous domains. The limitation of this approach is that it is best suited for small proteins with two domains.

Another approach, called SnapDRAGON [19], tries a limited ab initio modeling of a protein in an attempt to find domain linkers and compact regions. It can also predict discontinuous domains, but with a lower success rate than for continuous domains.

Finally, [36], Nagarajan et al. present an expert system whose inputs are a dozen of properties about each position in a protein derived from multiple local alignments, e.g., consistency of alignments, entropy and covariance, and whose output is the most likely domain architecture for a protein. Their results are best achieved thus far for continuous domains. However, this method performs a local computation around each position, and it is not built to handle discontinuous domains.

\[3\] For specific numbers on success and coverage, please refer to the original papers. There are many combinations of tests and measures giving too many numbers to report here due to space limitations.
All of the recent approaches described above—with the exception of modified PSIBLAST—do not compute models from which we can predict functions for domains. They are specifically built to find domain boundaries.
CHAPTER 3
PREDICTING THE MDA OF A PROTEIN

3.1 Overview

Previously, I described domains as the functional, evolutionary, and structural units that make up proteins. This chapter presents a method for computing a sensible domain architecture for any given protein from a set of pairwise alignments between the protein’s sequence and a large database of protein sequences. Here, I will give a simple definition of a protein domain sequence that is more useful for computational analysis. In the most general case, a protein domain sequence consists of one or more disconnected segments with each segment comprising a set of contiguous amino acid positions.\(^1\) Let us define protein’s modular domain architecture (MDA) as a partition of its amino acid positions, where each partition element comprises a domain’s set of positions. In the following pages, I will define a formal model for MDA and a method to compute the model parameters. The model will show us the number and positions of domains within a protein.

Sequence similarity is one of the main tools to predict protein homology, i.e., two proteins evolve from the same ancestral protein. However, multi-domain proteins have evolved by combining existing domains with distinct evolutionary histories. Homologous domains occur in different combinations in other proteins. Let us call the input protein as a query protein. If the query protein has multiple domains, it is quite likely that the homologs of its domains will occur in different combinations in the database proteins. Consequently, the query protein will only

\(^1\)All of the previous approaches to finding domain architecture using sequence similarity work only on continuous domains comprising a single segment, excluding the discontinuous domains. Here we attempt to include discontinuous domains to the extent that sequence similarity can be found for the individual segments. Therefore, we start with a general definition.
show local sequence similarity to any database protein that has domains homologous to a subset of query domains. Since we consider a domain as an atomic unit, we assume that a domain is either in alignment or not. The database protein sequences align to the query protein in positions that comprise a union of domains. If the database is large enough, it will contain a diverse set of protein sequences of different domain combinations whose alignments to the query will separate out each individual domain.

In this research, we used pairwise local alignments that were generated by PSI-BLAST since it is a fast search algorithm that can extract similar sequences from a database in a short time. However, any type of multiple alignments where the columns are reduced to the positions of the query can also be used. I provide examples in the following section of multi-domain proteins and their local alignments, which show distinct occurrence signatures among database proteins for each domain.

3.2 Examples of Local Alignments Predicting MDA

The examples presented in this section show how local alignments contain enough information to predict the domain architecture of a multi-domain protein. We collect local alignments using a sequence similarity search tool that takes the multi-domain protein as a query and searches a large database of protein sequences. The search returns a set of local alignments between the query protein sequence and database protein sequences that are similar to the query protein. If the query protein has multiple domains, then database proteins sharing each domain with the query will have sequence similarity represented by a local alignment covering that domain. Moreover, if there is a domain in the query that does not

\[\text{Sequence alignment algorithms mostly likely produce alignments shorter or longer than the actual domain. Missing or extra segments in alignments will be treated as noise in the data.}\]

\[\text{PSI-BLAST computes a position specific scoring matrix (PSSM) for each iteration based on all database sequences matching the query in that iteration. PSSM is used to score the set of alignments generated in the next iteration.}\]
have a homolog on a database protein, then no alignment between the query and the database protein will exist to cover that domain. Therefore, the local alignments of the query sequence with the database sequences show a pattern that can be used to predict the domain architecture of the query.

3.2.1 PTFA_ECOLI: Two Functional and One Structural Domain

An Escherichia coli protein with SWISSPROT ID PTFA_ECOLI is a multi-domain protein that is 376 amino acids long. It is a fructose specific enzyme that takes part in the phosphotransferase system (PTS) of E. coli. Figure 3 shows that there are two functional PFAM domains in the protein. One domain in the PTS_EIIA family is at the N-terminal end; it occupies amino acid positions from 2 to 142. The other domain belongs to the PTS_HPr family; it is at the C-terminal end, from positions 285 to 371. There is a third domain in the middle between amino acid positions 143 and 275. The third domain does not have any functional sites; however, it belongs to a family in the PFAM-B database.

We can make an educated guess that the middle domain is a structural domain because its homologs occur in many other proteins and it does not seem to have a specific function.

To generate the alignment data for analysis, we first run a database sequence search tool, PSI-BLAST, using the protein PTFA_ECOLI as a query sequence against the nonredundant protein sequence database nr. PSI-BLAST returned a large set of alignments including those that aligned to the entire query sequence. We removed such global alignments since they do not contain any information about domains using our method. We required all alignments in the set to have at

---

4 This example protein is taken from a paper by Storm and Sonnhammer where the authors describe a system called NIFAS to perform a visual analysis of domain evolution.

5 The phosphotransferase system is used by anaerobic bacteria for the uptake and phosphorylation of specific carbohydrates, such as glucose and fructose, from the extracellular environment.

6 PFAM-B database contains domain families found by automated methods of PRODOM which passed the scrutiny of manual check. Yet, they are kept separately from PFAM since the models in PFAM are built by manual selection of individual sequences.
SwissPfam entry for PTFA_ECOLI

Description from Swissprot for PTFA_ECOLI:

pts system, fructose-specific iia/fpr component (eiiia-fru) (fructose-permease iia/fpr component) (phosphotransferase enzyme ii, a/fprcomponent) (ec 2.7.1.69) (phosphotransferase fpr protein) (pseudo-hpr) (eiii-fru) (fructose pts diphosphoryl transfer prot

[376 residues]

**PTS_EIIA_2** 2-142

**PTS-HPr** 285-371

Key

signal peptide: > pfamA: > pfamB: > low complexity: > transmembrane: > coiled coil: > smart: >

Domain Order: Change the domain order using the drop down menus and click the 'Change order' button.

**pfamA**

low complexity pfamB

**Pfam Domains**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Start</th>
<th>End</th>
<th>Type</th>
<th>Source</th>
<th>Start</th>
<th>End</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTS_EIIA_2</td>
<td>2</td>
<td>142</td>
<td>low complexity seg</td>
<td>246</td>
<td>284</td>
<td>2.7800</td>
<td></td>
</tr>
<tr>
<td>Pfam-B_26782</td>
<td>143</td>
<td>275</td>
<td>low complexity seq</td>
<td>246</td>
<td>284</td>
<td>2.7800</td>
<td></td>
</tr>
<tr>
<td>PTS-HPr</td>
<td>285</td>
<td>371</td>
<td>low complexity seg</td>
<td>246</td>
<td>284</td>
<td>2.7800</td>
<td></td>
</tr>
</tbody>
</table>

Comments or questions on the site? Send a mail to pfam@sanger.ac.uk
least one segment with 40 amino acids missing from the query sequence positions in the alignment. The resulting set included 4,804 alignments.

The dot plot in Figure 4 shows the segments of the query protein sequence that are aligned to a database protein. The x-axis depicts the amino acid positions of the query sequence. The y-axis depicts the database sequences, numbers from 1 to 4,804. For each database sequence, the amino acid positions in the query protein that are in the alignment are plotted as dots in a line on those positions. An (x,y) pair is plotted as a dot if the amino acid position x in the query sequence aligned to the database sequence y. Therefore, each line in the plot displays one or more aligned segments of the query protein (not a database protein). If there are more than one local alignments to a single database protein, they are displayed together on the same line. To have an easily recognizable visual pattern, the aligned segments are displayed as sorted by their beginning positions. The figure shows that there are three regions where alignments pile up.

We can clearly see from the figure that alignments partition the query into four regions: A, B, C, and D. Each is covered by a different set of local alignments. For segments A, B, and D, we have local alignments covering only those segments and others covering multiple segments. Segment C is a low-complexity region filtered in the PSIBLAST output. The only alignments covering C are those that cover one or more of the other segments.

Next, a quick application of Singular Value Decomposition (SVD) shows that it is possible to differentiate between amino acid sequence positions that belong to distinct domains using the aligned target sequences to that position. Suppose each query position is assigned a vector of zeros and ones, where each element represents a database protein. If a database protein is aligned to a position that element has value 1; otherwise it has value 0. So, we can convert a set of local alignments for a query protein into a 0/1 matrix, $M$, where each row represents a position of the query protein, or an equivalent column in the alignments. Moreover, these vectors are clustered points in Euclidean space, with each cluster
Figure 4: PTF_A_ECOLI sequence segments in each local alignment. The top figure is the full set of alignments; the bottom figure is the detail of the boxed area. The protein sequence is divided into segments A, B, C, and D.
Figure 5: A detail from the PTF_A ECOLI profile alignment, positions from 100 to 250.
roughly representing the segments A, B, C, and D. To be able to see this in 3D, we can reduce the dimensions of the matrix $M$ into three using SVD.

Figure 6: The SVM analysis for PTFA_ECOLI. (a) Eigenvalues of the column matrix $M$; (b) 0/1 column vectors are displayed in 3D. (Note: I might redraw this to replace colors with different shape so that it is clear in black and white print out.)

Figure 6(a) shows the eigenvalues. The first three eigenvalues corresponding to regions A, D, and B, are significantly greater than the rest. In Figure 6(b), regions A, B, and D on PTFA_ECOLI are colored differently in order to see if they truly cluster within each domain. We see that domain colors and vector clusters do coincide. The SVD result is suggestive of three domains, but not sufficient to assign each position to the correct domain since there are many points in the plot that are either far away from the cluster to which they belong or closer to another cluster.

3.2.2 1b70: A protein with seven structural domains

Our second example is a more complex protein, Phenylalanyl tRNA Synthetase, whose 3D structure, shown in Figure 7(a), is solved in bacterium Thermus
aquaticus. It is an important enzyme in protein synthesis, and therefore, occurs in almost all organisms. The enzyme attaches the amino acid Phenylalanine to a tRNA. Its PDB ID is 1B70, which is the designation I will use to refer to this protein throughout this thesis. Overall, the protein contains seven structural domains and consists of two types of disconnected chains of amino acids. Figure (b) shows that Chain A is a single domain molecule whereas Chain B in Figure (c) has six domains. Let us use Chain B, with its six domains, as an example.

Except the first domain, all other five structural domains Figures (e)-(i), we can easily recognize five of the six structural domains the Chain B structure depicted in Figure (c). The first domain (Figure (d)) is behind the large third domain at the top.

Figure 8 displays PFAM functional domains. Although structural domains and functional domains for the most part overlap, there are few significant differences that need further comment. Three PFAM functional domains are embedded in structural domains. First, the tRNA binding functional domain between amino acid positions 45 and 145 is inside the second structural domain which runs from 38 to 153. Second, the PFAM domain B5 between 401 and 469 is inside the fourth structural domain, from 401 to 474. Third, the FDX_ACB domain between 688 and 780 covers the sixth structural domain, from 690 to 775. Moreover, the PFAM domain Pfam-B_281, from 470 to 675, overlaps with structural domain 5. However, the PFAM domain B3_4, from 147 to 291, contains the second part of the first structural domain and a half of the third structural domain. The remaining half of the third structural domain has no functional assignments although the whole protein segment from 187 to 400 is certainly a single structural module con-

---

7 T. aquaticus lives in high temperatures, such as the hot springs of Yellowstone National Park. Its DNA polymerase is used for high-temperature PCR reactions.

8 The bondage of the appropriate amino acid to a tRNA is catalyzed by a specific aminoacyl-tRNA synthetase. The energy of this bond later initiates the peptide bonds between adjacent amino acids in a growing polypeptide chain. [1]

9 In this case, two chains have separate sets of domains. It is possible, however, that more than one chain participate in a single structural domain.

10 Additional information on 1B70 Chain B maybe obtained from PDB database.
Figure 7: Domains in the protein Phenylalanine tRNA Synthetase. The structural domain positions and the graphical display of their 3D folds are taken from 1b70 obtained from CATH and PDB databases. (a) 3D structure of 1b70, (b) chain A, (c) chain B, and six structural domains in 1b70 chain B, (d) D1: 1-37, 154-186 (70 aa), (e) D2: 38-153 (116 aa), (f) D3: 187-400 (214 aa), (g) D4: 401-474 (74 aa), (h) D5: 489-679 (191 aa), (i) D6: 690-775 (86 aa)
**Protein families database of alignments and HMMs**

SwissPfam entry for P27002

**Description from Swissprot for SYFB_THETH:**

Phenylalanyl-tRNA synthetase beta chain (EC 6.1.1.20) (phenylalanine-tRNA ligase beta chain) (phers)

**Domain Diagram:**

- **tRNA_bind** 45-145
- **B3_4** 147-291
- **B5** 401-469
- **FDX-ACB** 688-780

**Key:**
- Signal peptide: 🟢
- PfamA: 🟢
- Context: 🟣
- SMART: 🔴
- Transmembrane: 🔵
- Low complexity: 🔴
- Coiled coil: 🟢
- PfamB: 🟢

**Domain Order:** Change the domain order using the drop down menus and click the ‘Change order’ button.

**PfamA**

- **Low complexity** PfamB

**Pfam Domains**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Start</th>
<th>End</th>
<th>Type</th>
<th>Other Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tRNA_bind</td>
<td>45</td>
<td>145</td>
<td>Low complexity</td>
<td>Seg 12-23, Score 2.1200</td>
</tr>
<tr>
<td>B3_4</td>
<td>147</td>
<td>291</td>
<td>Low complexity</td>
<td>Seg 122-134, Score 2.0800</td>
</tr>
<tr>
<td>B5</td>
<td>401</td>
<td>469</td>
<td>Low complexity</td>
<td>Seg 167-181, Score 2.2400</td>
</tr>
<tr>
<td>Pfam-B_281</td>
<td>470</td>
<td>675</td>
<td>Low complexity</td>
<td>Seg 183-202, Score 2.4900</td>
</tr>
<tr>
<td>FDX-ACB</td>
<td>688</td>
<td>780</td>
<td>Low complexity</td>
<td>Seg 372-387, Score 2.3500</td>
</tr>
</tbody>
</table>

**Low complexity Segments:**
- Seg 757-775, Score 2.5400
- Seg 521-48, Score 2.7700
- Seg 473-483, Score 1.9400
- Seg 372-387, Score 2.3500
- Seg 167-181, Score 2.2400
- Seg 122-134, Score 2.0800
- Seg 12-23, Score 2.1200

**Comments or questions on the site? Send a mail to Pfam-B_281 470 675 pfam@sanger.ac.uk**

**Figure 8:** Sanger Center web site display of the PFAM domains in 1B70_B.
nected with loop regions to domains 2 and 4. Indeed, the PFAM domain B3.4 is a functional site at the "interface" of two structural domains. This example particularly illustrates that the sequence conservation-based domain analysis cannot follow structural domain decomposition if sequence conservation does not overlap with the cores of structural domains.

Similarly, the sequence of 1B70 Chain B is used as a query against the \( \text{nr} \) database. The resulting aligned segments are displayed in Figure 9.\(^{11}\)

---

\(^{11}\)Vertical white stripes are due to size/resolution of the figure generated by the program gnu-plot. A different size could get rid of the white stripes.
3.3 Building a Probabilistic Model for a Protein’s MDA

3.3.1 Formal description of the alignment data

Let $q = \langle q_i | i \in [1..m] \rangle$ be the query protein sequence for which we want to compute a modular domain architecture (MDA). The notation indicates that $q$ is a sequence of $m$ amino acids, with $q_i$ as the amino acid at the $i^{th}$ position of $q$. Let us number the set of database sequences in alignment with the query sequence from 1 through $n$ and let $B = \{b_j | j \in [1..n]\}$ be the set of database protein sequences. The term $b_j$ denotes the database protein sequence with index $j$.

A set of local alignments of $q$ to $B$, as shown in the above examples, can be represented as a match relation $\alpha \subset [1..m] \times [1..n] \times [1..\infty)$ where $(i, j, k) \in \alpha$ if position $k$ of $b_j$ is aligned to position $i$ of $q$. The above definition is general enough to include the following cases. A single query position can be in more than one alignment that involve different positions in a database protein if the domain homologs repeat in the database protein. Similarly, if there is more than one instance of the same domain in the query protein, those instances may align to the same homolog domain positions in a database protein. Since we are interested in the structure of $q$, we will focus on the observed match relation $\gamma = \pi_{1,2} \alpha$, i.e., the set of pairs $(i, j)$ in $[1..m] \times [1..n]$ where some position in $b_j$ aligns to the $i^{th}$ position of $q$. The inserted gaps in $b_j$ can be treated either as a part of an alignment or not depending on the degree of the continuity of alignments we would like to stress. Our expectation is that $\gamma$ contains enough information to construct a sensible domain architecture model for $q$.

3.3.2 Domain Signature Model (DSM)

For the sake of clarity, we have given a formal definition for a query-based multiple alignment—the observed match relation $\gamma$. Since we are building a probabilistic model, we will recast $\gamma$ as a set of observations. The match relation
$\gamma$ can be viewed as a set of column vectors, with one vector per query protein sequence position. We will call the column vector for position $i$ signature vector $s_i$.

The signature vector $s_i$ is a binary-valued vector of size $N$, the size of the database. In the signature $s_i$, $j^{th}$ element is 1 if there exists an alignment between the query and the database protein $j$ covering position $i$. Otherwise it is 0. Formally,

$$s_i(j) = \begin{cases} 
1, & \text{if } (i, j) \in \gamma; \\
0, & \text{otherwise.}
\end{cases}$$

(3.1)

An alignment is, then, an observed sequence of signature vectors $s_i$ for the query protein sequence positions, $i = 1, \cdots, M$. Let $S$ be an observed variable whose values come from the signature set $S = \{s_i | i \in [1..M]\}$. We expect that the vectors $s_i$ will partition into clusters where each cluster corresponds to the set of sequence positions on the query protein that are in a natural 1-to-1 correspondence with an individual domain. We also expect that there is a strong locality in the clustering even if our model does not assume so. Consequently, clustering will allow us to partition query protein sequence positions into domains.

In order to find a cluster structure in the set $S$ of signature vectors, we build a probabilistic latent class\textsuperscript{12} model with a set of domain classes representing clusters of signature vectors. We call it the Domain Signature Model (DSM). The probabilistic model allows us to compute soft clusters where membership is defined by a probability distribution over the signature vectors. Each domain class

\textsuperscript{12}Latent Variable Analysis (LVA) with discrete and continuous variables is a method for guessing missing data, i.e., the values of unobserved variables for each observation, using observed data such that the missing data reveals a structure. The Probabilistic Latent Class (PLC) analysis is an instance of LVA where the latent variables are categorical, i.e., classes, and the relationship between variables is probabilistic.

For an introduction to PLCA and LVA, see [50] [23] [24] [15].

In PLC analysis, one assumes that each observation belongs to one of $K$ classes whose size and number one does not know (a priori). All observations belonging to the same class come from the same joint probability mass distribution. In a PLC model a class has a conditional probability mass distribution for each observed categorical variable. The joint distribution for a class models a cluster of observations where each observation contains a value generated by the class conditional probability for each observed variable.
has a conditional probability distribution over the vectors. In this sense, our model is similar to mixture models. We will introduce a discrete latent variable $D$ whose values, aka states, $d_1, \cdots, d_K$ denote distinct domains. Each domain $d_k$ has an associated probability vector, $v_k$, where the position $j$ stores the probability of database protein $b_j$ carrying the domain $d_k$. In other words, $v_k(j)$ is the probability of seeing 1 in $j^{th}$ position of a column signature $s_i$ if the query sequence position $i$ belongs the domain $d_k$. The domain classes of the DSM model represent clusters of signature vectors whose centers are defined by each $v_k$. Below, we explain how to compute the conditional probability $P(s_i|d_k)$.

**Generative description of the model.** Given that position $i$ belongs to domain $d_k$, the probability of generating signature $s_i$ by our model is obtained as a product of individual probabilities for each vector element in $s_i$. We can describe the generation of $s_i$ as follows. Let us imagine a binary tree with $N$ many levels, one level for each $b_j$. A signature is generated by assigning 1 to the $j^{th}$ element with probability $v_k(j)$ and 0 with probability $(1 - v_k(j))$ at each level. Therefore, the probability of generating $s_i$ given that position $i$ belongs to $d_k$ is

$$P(s_i|d_k) = \prod_{j \in [1..N]} v_k(j)^{s_i(j)}(1 - v_k(j))^{1 - s_i(j)}.$$  

**Definition 3.3.1** Given a query protein sequence $q$ of length $M$, and a set of signature vectors $S = \{s_i| i \in [1..M]\}$ where each signature $s_i$ is a 0/1 vector of size $N$, the Domain Signature Model (DSM) of $q$, $\mathcal{M}^q$, is a probabilistic latent variable model where

- the observed variable is $S \in \mathcal{S}$;
- the latent class variable is $D \in \{d_k | k \in [1..K]\}$; and
- each class $d_k$ has an associated vector $v_k$ of size $N$, where $v_k[j]$ is the probability that the database protein $b_j$ has domain $d_k$. The class probabilities are $P(d_k), k =$
For each class $d_k$, the conditional probability mass distribution over $s_i$ is determined by $v_k$ as

$$P(s_i|d_k) = \prod_{j \in [1..N]} v_k(j)^{s_i(j)}(1 - v_k(j))^{1-s_i(j)}.$$  \hspace{1cm} (3.2)

Let $\theta$ be the set of DSM model parameters,

$$\theta = \{v_1, \ldots, v_k\} \cup \{P(d_1), \ldots, P(d_k)\}.$$  \hspace{1cm} (3.3)

After computing a DSM model for a protein sequence, its modular domain architecture (MDA) is determined by the posterior probability $P(d_k|s_i; \theta)$ given the signature vector $s_i$ of the protein sequence position $i$. We compute $P(d_k|s_i; \theta)$ using the Bayesian formula

$$P(d_k|s_i; \theta) = \frac{P(s_i|d_k; \theta)P(d_k; \theta)}{P(s_i; \theta)}.$$  \hspace{1cm} (3.3)

The posterior probabilities $P(D|S; \theta)$ give us a randomized partition of protein sequence positions.

Each protein sequence position belongs to a single domain. Consequently, the posterior probabilities must have their masses around a single domain class. In the following section, we will employ the expectation minimization algorithm which gives us such a distribution.

### 3.4 Computing an Optimal DSM for a Protein

In this section, I explain how to compute optimal parameters for the DSM model for the signature vector set $S$ using an iterative algorithm that maximizes the likelihood of generating $S$. 

3.4.1 Maximum Likelihood Estimation (MLE)

The maximum likelihood estimation (MLE) is a common statistical method to fit a model to observed data. In MLE, the optimal parameter values are those that maximize the likelihood of the data over all possible models. The DSM model has a set of probability vectors $v_k$ and class probabilities $P(d_k)$ as parameters, i.e., $\theta = \{v_1, \cdots, v_K\} \cup \{P(d_1), \cdots, P(d_K)\}$. The ML estimates are the parameters that maximize the data likelihood of the observed signature vectors $S$. As each protein position belongs to one domain, in the DSM model each signature vector is generated by one latent domain class. Given the model has $K$ domain classes $d_1, \cdots, d_K$, the probability of a column signature vector $s_i$ is

$$P(s_i; \theta) = \sum_{k=1}^{K} P(d_k; \theta) P(s_i|d_k; \theta)$$ (3.4)

The DSM model first chooses one of the domain classes $d_k$ as the value of the latent variable $D$ with probability $P(d_k; \theta)$, and then generates $s_i$ using $v_k$ with probability $P(s_i|d_k; \theta)$ using the Equation 3.2. The probability of the DSM model with parameters $\theta$ generating $S$ is

$$P(S; \theta) = \prod_{i=1}^{M} P(s_i; \theta)$$

$$= \prod_{i=1}^{M} \sum_{k=1}^{K} P(d_k; \theta) P(s_i|d_k; \theta).$$ (3.5)

However, $P(S; \theta)$ cannot be maximized directly due to the sum over all $k$ inside the logarithm. Since we need to consider all possibilities for $P(d_k; \theta)$ and $P(s_i|d_k; \theta)$, the computation is intractable. Instead, we compute the ML estimates for the model parameters by an iterative algorithm using expectation maximization. We also use log-space to work with sums instead of products.
The Expectation Maximization (EM) algorithm is an iterative algorithm that tries to find a maximal solution through gradient ascent on the likelihood function. We view data as a set of incomplete observations. Suppose our observations come from a set of pairs, a signature vector and a domain, \((S,D)\); we then have a pair of random variables. We observe the variable \(S\), but we cannot observe the domain variable \(D\). Let \(\mathcal{S}^c\) be the set of complete observations\(^{13}\). The likelihood of \(\mathcal{S}^c\) is

\[
P(\mathcal{S}^c; \theta) = \prod_{i=1}^{M} P(s_i, D; \theta)
\]

where \(D\) is one of \(d_1, \cdots, d_K\). Let \(\mathcal{L}(\mathcal{S}^c; \theta)\) denote the log likelihood, i.e.,

\[
\mathcal{L}(\mathcal{S}^c; \theta) = \sum_{i=1}^{M} \log P(s_i, D; \theta).
\]

**Expectation Step (E-step)**  
Although we could not observe the domain class variable \(D\), we could calculate the expected value of \(\mathcal{L}(\mathcal{S}^c; \theta)\) with respect to the posterior probability \(P(D|S; \theta)\) using the current model parameters. Let \(\theta^t\) be the set of DSM model parameters computed at iteration \(t\), \(\theta^t = \{v_k^t, k = 1, \cdots, K\} \cup \{P(d_1), \cdots, P(d_k)\}\). At the E-step of the iteration \(t + 1\), we calculate the expected

\(^{13}\)There is a many-to-one mapping from the complete sample space to the space from which \(S\) is drawn.
value of log likelihood function with respect to the posterior probability distribution of D given S under the estimates of $\theta^t$ as follows.

$$E_{P(D|S;\theta^t)}[\mathcal{L}(S^c; \theta^{t+1})] = \sum_{i=1}^{M} \log P(s_i, D; \theta^{t+1})$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta^t) \log P(s_i, d_k; \theta^{t+1})$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta^t) \log (P(d_k; \theta^{t+1}) P(s_i|d_k; \theta^{t+1}))$$

$$= \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta^t) \log P(d_k; \theta^{t+1})$$

$$+ \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta^t) \log P(s_i|d_k; \theta^{t+1}) \tag{3.6}$$

where according to the DSM definition

$$P(s_i|d_k; \theta^{t+1}) = \sum_{j=1}^{N} s_i(j) \log u_k^{t+1}(j) + (1 - s_i(j)) \log(1 - u_k^{t+1}(j)). \tag{3.7}$$

The posterior probability $P(D|S; \theta^t)$ is computed by the Bayesian formula

$$P(d_k|s_i; \theta^t) = \frac{P(d_k; \theta^t) P(s_i|d_k; \theta^t)}{P(s_i; \theta^t)} = \frac{P(d_k; \theta^t) P(s_i|d_k; \theta^t)}{\sum_{i=1}^{K} P(s_i, d_i; \theta^t)}$$

$$= \frac{P(d_k; \theta^t) P(d_k|s_i; \theta^t)}{\sum_{i=1}^{K} P(d_i; \theta^t) P(d_k|s_i; \theta^t)} \tag{3.8}$$

**Maximization Step (M-step)** In the M-step, we compute the set of parameters $\theta$ for the next iteration that maximize the expected log likelihood $E_{P(D|S;\theta^t)}[\mathcal{L}(S^c; \theta^{t+1})]$. 
To find vector values $v_{k}^{t+1}$ that maximizes the expected log likelihood (Equation 3.6), we take its derivative w.r.t each $v_{k}^{t+1}(j)$ and find the root.

$$\frac{\partial E_{P(D|S; \theta^{t})}[\mathcal{L}(S^{c}; \theta^{t+1})]}{\partial v_{k}^{t+1}(j)} = \sum_{i=1}^{M} P(d_{k}|s_{i}; \theta^{t})(s_{i}(j) - s_{i}(j)v_{k}^{t+1}(j) - v_{k}^{t+1}(j))$$

$$= \sum_{i=1}^{M} P(d_{k}|s_{i}; \theta^{t})s_{i}(j) - s_{i}(j)v_{k}^{t+1}(j) + s_{i}(j)v_{k}^{t+1}(j)$$

$$= 0$$

Setting the derivative to zero gives us the following update equations for class vectors:

$$v_{k}^{t+1}(j) = \frac{\sum_{i=1}^{M} P(d_{k}|s_{i}; \theta^{t})s_{i}(j)}{\sum_{i=1}^{M} P(d_{k}|s_{i}; \theta^{t})} \tag{3.9}$$

We will assume that $v_{k}$ and $P(d_{k}; \theta)$ are unrelated. Therefore we can maximize the expected value independently for each type of parameter [13]. To obtain $P(d_{k}; \theta^{t+1})$ we maximize $E_{P(D|S; \theta^{t})}[\mathcal{L}(S^{c}; \theta^{t+1})]$ subject to the constraint that $P(d_{k}; \theta^{t+1})$ is a probability measure, i.e.,

$$\sum_{l=1}^{K} P(d_{l}; \theta^{t+1}) = 1.$$ 

The resulting maximization with constraints problem is solved using the method of Lagrange multipliers. Let $\mathcal{L}^{Lm}$ denote the new formula resulting from the addition of multiplier $\lambda$ and the constraint,

$$\mathcal{L}^{Lm} = E_{P(D|S; \theta^{t})}[\mathcal{L}(S^{c}; \theta^{t+1})] + \lambda(\sum_{l=1}^{K} P(d_{l}; \theta^{t+1}) - 1).$$
\[
\frac{\partial \mathcal{L}_{im}}{\partial P(d_k; \theta^{t+1})} = \sum_{i=1}^{M} \frac{P(d_k|s_i; \theta^t)}{P(d_k; \theta^{t+1})} - \lambda = 0
\]

\[
P(d_k; \theta^{t+1}) = \sum_{i=1}^{M} \frac{P(d_k|s_i; \theta^t)}{\lambda}
\]

\[
\sum_{t=1}^{K} P(d_l; \theta^{t+1}) = \sum_{t=1}^{K} \sum_{i=1}^{M} \frac{P(d_l|s_i; \theta^t)}{\lambda} = 1
\]

\[
\lambda = \sum_{t=1}^{K} \sum_{i=1}^{M} P(d_l|s_i; \theta^t)
\]

\[
P(d_k; \theta^{t+1}) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)}{\sum_{i=1}^{K} \sum_{i=1}^{M} P(d_l|s_i; \theta^t)}
\]

Since \(\sum_{t=1}^{K} \sum_{i=1}^{M} P(d_l|s_i; \theta^t) = \sum_{i=1}^{M} (\sum_{t=1}^{K} P(d_l|s_i; \theta^t))\) and \(\sum_{i=1}^{K} P(d_l|s_i; \theta^t) = 1\), we can simplify the M-step update equation for \(P(d_k; \theta^{t+1})\) to

\[
P(d_k; \theta^{t+1}) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)}{M}.
\]

The EM algorithm starts with the random initialization of all probabilities and alternates between E-step and M-step until convergence, i.e., the improvement on the model is below a small threshold.

In the next section, we discuss further improvements to the EM algorithm based on my work. The EM algorithm converges quickly but does not guarantee finding the global maximum. Because, in the E-step, to obtain class posterior probabilities using the Bayesian formula (Equation 28), one uses the class conditional probabilities computed in the M-step of the previous iteration. Yet, in early iterations, these probabilities are not dependable due to the random initialization. This could lead to an early commitment to one among the many local maxima of the data log-likelihood function. So, the EM algorithm can get stuck in a local maximum when the values of parameters converge to one early on. No further improvements on the model could be achieved from the local maximum since the
Initialize $\theta^0$ randomly
Repeat E-step and M-step until convergence

**E - STEP:**

$$P(s_i|d_k; \theta^t) = \prod_{j=1}^{N} v_k^t(j)^{s_i(j)}(1 - v_k^t(j))^{(s_i(j) - 1)}$$

$$P(s_i; \theta^t) = \sum_{l=1}^{K} P(s_i|d_l; \theta^t) P(d_l; \theta^t)$$

$$P(d_k|s_i; \theta^t) = \frac{P(d_k; \theta^t) P(s_i|d_k; \theta^t)}{P(s_i; \theta^t)}$$

**M - STEP:**

$$v_k^{t+1}(j) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t) s_i(j)}{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)}$$

$$P(d_k; \theta^{t+1}) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)}{M}$$

Figure 10: The EM algorithm, iteration $t + 1$. 
expectation limits the search space to the neighborhood of the local maximum. In practice, EM can find the global maximum if several trials are conducted from different random starting points. However, the best strategy is to delay the commitment to any type of solution for as long as possible by retaining randomness. There is a theoretically sound way to aim better at the global maximum by preventing the EM algorithm from moving greedily to a local one. I describe this process below.

### 3.4.2 Deterministic annealing applied to clustering

Annealing algorithms are the methods that simulate to the physical annealing of compound materials to solve optimization problems. In physical annealing, the temperature is slowly lowered to get the physical system into its most stable minimum energy state. Finding the minimum of a physical energy function is an optimization problem and the states of the physical system are all possible solutions to the problem. The temperature determines the level of randomness in the system; as the temperature is lowered, randomness decreases. The higher the degree of randomness is, the larger the number of possible states the physical system can be in, and the higher the number of solutions to the problem exists. The stable state reached at zero temperature is the minimum energy state which is the optimum solution to the energy function. However, the simulations of such physical systems are based on stochastic annealing where the search for the next stable state at a given level of temperature is done randomly. Unfortunately, random search is time consuming, as the temperature must be decreased very slowly; for many optimization problems, it may be too slow.

In his thesis work, Kenneth Rose (1991) developed the Deterministic Annealing (DA) method within the framework of information theory \[43\]. The DA method associates a distortion cost\[14\] for a solution of an optimization problem

\[d(x, \hat{x})\]

\[\text{distortion d}(x, \hat{x})\] is defined as “a measure of the cost of representing symbol x by symbol \(\hat{x}\)”.\[14\]
such as clustering (unsupervised learning) and classification (supervised learning). It is an annealing procedure that searches for the minimum expected global distortion induced by a solution to the problem at decreasing levels of randomness; however, it speeds up the annealing schedule by replacing the random search with expectation.

In [49], Ueda and Nakano, inspired by Rose’s DA work, present a deterministic annealing approach to compute the most random class posterior probabilities for the E-step of the EM algorithm. The approach, called DAEM, chooses the distribution $P(X|Y)$ with the highest entropy at a constant level of expected data log-likelihood. The choice of the most random solution is called Jaynes’ principle of maximum entropy (Edwin Jaynes, 1957). Since the distribution with a maximum entropy has the maximum uncertainty, this choice effectively leads to the least commitment in the EM algorithm. The expected data log-likelihood, maximized in the M-step of previous iteration with respect to parameters $\theta$, enters the E-step as a constant. Here, I apply DAEM to the parameter estimation of the DSM model. Like the EM approach DAEM requires calculation of specific update equations for an application.

Let $H(D|S)$ represent the entropy of the assignment probability distribution $P(D|S)$,

$$H(P(D|S)) = -\sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i) \log P(d_k|s_i), \quad (3.10)$$

$H(P(D|S))$ is a measure of how randomized the assignment of positions to the classes are.

In the E-step of the DAEM algorithm, we obtain a new set of assignment probabilities $P(D|S; \theta)$ by maximizing its entropy $H(D|S; \theta)$ subject to a constraint of constant expected loglikelihood whose current maximum was obtained in the
previous M-step. Using Lagrange multipliers, we calculate \( P(D|S; \theta) \) that maximizes the following function.

\[
\mathcal{H} = H(P(D|S; \theta)) + \beta(E_{P(D|S)}[\mathcal{L}(S^c; \theta)] - \text{constant}) + \sum_{i=1}^{M} \lambda_i \left( \sum_{k=1}^{K} P(d_k|s_i; \theta) - 1 \right)
\]

\[
= - \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta) \log P(d_k|s_i; \theta)
\]

\[
+ \beta \sum_{i=1}^{M} \sum_{k=1}^{K} P(d_k|s_i; \theta) \log P(d_k, s_i; \theta)
\]

\[
+ \sum_{i=1}^{M} \lambda_i \left( \sum_{k=1}^{K} P(d_k|s_i; \theta) - 1 \right)
\]

\[
\frac{\partial \mathcal{H}}{\partial P(d_k|s_i; \theta)} = -\left( \log P(d_k|s_i; \theta) + 1 \right) + \beta \log P(s_i, d_k; \theta) + \lambda_i = 0
\]

\[
P(d_k|s_i; \theta) = e^{(\lambda_i - 1)} e^{\beta \log P(s_i, d_k; \theta)}
\]

\[
\sum_{l=1}^{K} P(d_l|s_i; \theta) = 1
\]

\[
\sum_{l=1}^{K} e^{(\lambda_i - 1) + \beta \log P(s_i, d_l; \theta) - 1} = 1
\]

\[
e^{(\lambda_i - 1)} = \frac{1}{\sum_{l=1}^{K} P(s_i, d_l; \theta)^\beta}
\]

\[
P(d_k|s_i) = \frac{P(s_i, d_k; \theta)^\beta}{\sum_{l=1}^{K} P(s_i, d_l; \theta)^\beta} \quad (3.11)
\]

\[
(3.12)
\]

The maximization of \( H(P(D|S; \theta)) \) subject to a constant expected log-likelihood introduces parameter \( \beta \in (0, c] \), where \( c \) is a small constant\(^{15}\). The DAEM algorithm is a procedure that repeats the EM algorithm at increasing levels of \( \beta \). At

\(^{15}\beta \) is inversely proportional to temperature.
each level, the EM iterations start with the solution found in the previous level. When the DAEM algorithm starts with small $\beta$ values, the entropy $H(P(d_k|s_i; \theta))$ is the dominant term in the formula. The algorithm initially computes random solutions that maximize $H$. As $\beta$ gets larger, the expectation term $E_{P(D|S)}[\log P(S^c; \theta)]$ becomes dominant. While the effect of $H(P(D|S; \theta))$ disappears, the less and less random solutions can be found. Hence, $\beta$ is a parameter for a tradeoff between the randomness of a solution and the maximum likelihood. When $\beta$ is 1, Equation 3.11 becomes the Bayesian formula in Equation 3.8.

The effective number of clusters and varying DSM size

The DSM described in Definition 3.3.1 is a clustering model whose size is the number of classes, $K$. The observations $s_i$ are clustered into groups modeled by each class, $d_k$. Although we choose the number of classes, the DAEM algorithm will compute only those clusters distinguishable at a given level of randomness, $\beta$. So, if $\beta$ is very small, the algorithm will put all the observations into a single cluster at the end of EM iterations in that $\beta$ cycle. All the $K$ classes will converge together to model the same cluster mass. In case the converged classes become identical, random perturbations can be introduced at the start of each $\beta$ cycle since they must be nonidentical to move away from each other in the subsequent $\beta$ cycles. As the randomness decreases, more clusters will become distinguishable. Indeed, at every level $\beta$, there is an effective number of distinguishable clusters. Starting with a single cluster, as $\beta$ increases, observations will split into more clusters, and a subset of $K$ classes will converge to model each cluster.

In essence, to obtain the effective number of clusters, we can throw in as many classes as we like. However, this is not wise since it will increase the computation time. It is sufficient to keep twice as many classes as the number of distinguishable clusters in the previous $\beta$ cycle considering that all clusters may split into two at the same time. To achieve the sufficient number of classes, the DA
method of K. Rose includes a splitting and culling procedure [43]. At the beginning of a $\beta$ cycle, every class is split into two classes with each class randomly perturbed. Splitting also divides equally the class posterior probability of a split class between new classes. At the end of each $\beta$ cycle, each set of converged classes is reduced to one class.

To implement a similar split-and-cull procedure, we need to determine whether two classes are nearly identical. The DSM classes are represented by the vectors of probabilities. If two class vectors are identical, then they model the same cluster. If they differ a very small amount, we can accept them as identical for practical purposes. We compute the Euclidean distance between pairs of class vectors to determine if they have converged to represent the same cluster.

Since individual protein domains may occur in many different combinations in database proteins, we have many overlapping clusters where overlapping area contains those database proteins that contain more than one domain. However, there has to be enough difference between two class probability vectors to represent two overlapping classes. If the overlap area contains most of the database proteins in which two or more domains occur, then they should be considered as a single cluster and only one class should be computed. Moreover, two nearly identical class vectors cannot represent two clusters with different shapes.
DAEM($K_{\text{max}}, \beta_{\text{start}}, \beta_{\text{end}}, \beta_{\text{update}}$):

$\beta = \beta_{\text{start}}$

$K = 1$

While ($\beta \leq \beta_{\text{end}}$ and $K \leq K_{\text{max}}$) repeat:

- Split all classes into two, apply random perturbations, $K = K \times 2$
- Run EM($\beta, K$)
- remove reductant classes
- $\beta = \beta \times \beta_{\text{update}}$

Run EM(1, $K$)

EM($\beta$):

$t = 0$

Initialize $\theta^{0}$ randomly

Repeat until the change in $\theta$ is smaller than $\epsilon$:

- E-step
  \[ \forall i, k : P(d_k|s_i; \theta^t) = \frac{P(s_i, d_k; \theta^t)^\beta}{\sum_{i=1}^{K} P(s_i, d_l; \theta^t)^\beta} \]

- M-Step
  \[ \forall j, k : v_k^{t+1}(j) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)s_i(j)}{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)} \] (3.13)

  \[ \forall k : P(d_k; \theta^{t+1}) = \frac{\sum_{i=1}^{M} P(d_k|s_i; \theta^t)}{M} \]

- $t = t + 1$

Figure 11: The DAEM algorithm for computing the optimal DSM.
3.5 The Results for DSM Approach on the Examples

By computing a DSM model, we can predict a reasonable modular domain architecture from a large set of local alignments. We employed two heuristics that helped to constrain the number and the size of classes. An average size for a domain is accepted to be around 150 amino acids. We used this value to calculate an approximate upper bound for the number of domains. Moreover, the analysis of the domains in the SCOP database showed that the percentage of domains that were shorter than 50 amino acids was very small (1.3%). Therefore, we applied the minimum size of 50 amino acids as a constraint on the computed domain classes. Whenever the number of classes exceeded the maximum number of domains, we applied a culling procedure to remove any domain classes that were deemed too short to be a real domain. If the remaining number of classes still exceeded the maximum, we stopped; otherwise, the DA continued at the next β cycle.

To demonstrate how the method works, the DSM models were computed for the examples given in Section 3.2. This section will go through the intermediate steps and results obtained for each example and discuss them in detail.

3.5.1 Example 1: Protein PDFA_ECOLI

The protein PDFA_ECOLI is a medium size protein with three domains. Two domains at the terminal ends are classified as functional domains in PFAM. There is no such functional classification for the middle domain, although it is in PFAM_B. However, we can conclude that it is a domain since it has homologs indicated by the middle cluster of alignments in the profile. In general, bacterial proteins are simpler in terms of domain architecture; they contain a smaller number of domains compared to the proteins in eukaryotes. For instance, many two-domain bacterial proteins are formed by the simple fusion of two proteins. In Figure 4, the aligned segments display a pattern of three easily distinguishable clusters.
In the test run, the parameter $\beta$ starts from 0.01; the multiplier is 0.1. In Figure 12, we see that at $\beta = 0.01$ two classes were computed, the first represents a single cluster while the second one represents a mixture of two clusters. In the next DA cycle, $\beta = 0.011$, each class is split into two. The split classes are randomly perturbed to move away from each other. The EM computes parameters for a DSM with four classes. However, at the end of the DA cycle, the culling procedure finds that two of the classes have converged, removing one resulting in three classes (Figure 13). At the end of the algorithm, we run EM again for $\beta = 1$. The computed DSM is identical to the one in the previous cycle(Figure 13).
Figure 12: PTFA_ECOLI: At $\beta = 0.01$, the DSM has only two classes.
Figure 13: PTFA\_ECOLI: $\beta = 0.011$, three classes emerge.
Since the maximum number of classes is set to 3, the algorithm ends after running EM at $\beta = 1$. 

Figure 14: PTFA_ECOLI: Since the maximum number of classes is set to 3, the algorithm ends after running EM at $\beta = 1$. 

(a) 

(b)
3.5.2 Example 2: Protein 1b70 Chain B

This protein is more challenging because it has six structural domains, one of which has two segments separated by another domain in the middle; in addition, unlike the first example, the alignments do not have sufficiently clear patterns to distinguish some of the domains. As Figure 15 shows, the algorithm computed a DSM with five classes. Four of the classes extract the segments that belong to the domains. The fourth class corresponds to a collection of segments that cluster together because they do not participate in a majority of alignments. Although the first two segments are part of the same domain, the last segment is a distinct domain. We ran another cycle of DA at the next $\beta$ level to see if these two domains could be differentiated. The five classes were split into ten, and after running EM, the culling step returned six classes. Figure ?? shows the resulting six partitions of the protein sequence. Although the fourth class was separated into two classes (classes 4 and 6) and each domain was correctly assigned to its own class, the assignments of other classes became less accurate.
Figure 15: 1B70_B: The DSM has five classes.
Figure 16: 1B70_B: The DSM has six classes.
3.6 Large Scale Tests Using PDB Proteins

To demonstrate the success of the method presented in this chapter, we applied it to a set of proteins whose modular domain structures were already known. The method presented in this thesis is designed to be applied to individual protein sequences where one can change the maximum number and minimum size of domains to see if a better sequence partitioning can be found. However, when demonstrating the success of the method on a large number of proteins, we do not have such an option. When interpreting the results, it should be considered that predictions can be improved if the method is applied individually to each protein and the user can input the number of domains.

We obtained the domain architecture information from the SCOP structural domain database\(^\text{16}\). The SCOP database is built by manual analysis of domains on the proteins. As such, it is considered an accurate source. As our test data, we collected a set of representative sequences from the PDB protein structure database\(^\text{17}\) such that their modular domain architectures are classified in the SCOP database. PDB provides the clustering of proteins’ amino acid sequences at identity levels of 50%, 70%, and 90%. In order to avoid bias in the test set that might result from the overrepresentation of any part of protein space, we selected representative proteins from clusters with 50% identity. In each cluster, we chose the protein first in rank as the best representative protein sequence for that cluster. At 50% identity the protein sequences in a cluster usually have identical modular domain architecture.

We narrowed down the set to those protein architectures where all domains were continuous in sequence and there were no repeating domains. The resulting data set contained 915 protein sequences. Of these 783 proteins contained two SCOP domains, 117 proteins had three SCOP domains, 14 proteins had four SCOP domains, and one protein had five SCOP domains.

\(^{16}\)See [http://scop.mrc-lmb.cam.ac.uk/scop/](http://scop.mrc-lmb.cam.ac.uk/scop/)

\(^{17}\)See [http://www.pdb.org/](http://www.pdb.org/)
As mentioned above we used protein sequences obtained from PDB. However, because SCOP often assigns numbers to protein structure positions with an offset, the numbers do not always correspond to sequence positions. To find the correct domain positions on protein sequences, we searched for the substrings of the SCOP domain sequences provided by the ASTRAL compendium.\textsuperscript{18} Because we were not able to determine the correct positions for some proteins—most likely due to missing amino acids in the middle of the domain sequence—at the end of this process, our dataset size decreased to a total of 850 protein sequences.

For each protein sequence in the test set, we generated a set of local sequence alignments using the sequence database search tool PSI-BLAST.\textsuperscript{19} As the sequence database, we used the nonredundant database \textit{nr} of protein sequences obtained from NCBI. The database \textit{nr} is the union of six sequence databases compiled together and updated weekly. The local alignments are converted to binary vectors that we supplied as input to our method. The method predicted a modular domain architecture for each protein in the test set from the binary vectors. The next section discusses the resulting predictions.

3.6.1 Results

In order to obtain a measure of success, we employ a scoring algorithm that computes a many-to-one mapping from predicted domains to the maximum overlapping SCOP domains in each protein. Here, it is possible to assign more than one predicted domain to a SCOP domain. The split-and-cull algorithm we use usually results in more domains than SCOP assigns to a protein. Currently, there is no theoretical way to compute the accurate number of domains in a protein. Instead, we guess the maximum number of domains by dividing the protein length by the average domain size accepted by structural biologist—150 amino acids. In the current version of SCOP, we computed the average domain size as 172 amino acids.

\textsuperscript{18}See \url{http://astral.berkeley.edu}
\textsuperscript{19}See \url{http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/psil.html}
However, the SCOP average is likely to be biased, considering that it contains only a small portion of protein space. Further research is needed to use the domain size distribution on the varying protein lengths in SCOP database, rather than a single value, to determine the number of domains in a protein.

It is possible that one large domain is predicted as two smaller domains that together overlap with the large domain. The scoring algorithm is intended to give high scores to predictions in which the beginning position of the first small domain and the end of the second domain are close to the boundaries of the large domain. However, if boundaries are not close, the score will be low since a predicted domain can only be assigned to a single SCOP domain.

A random partitioning of positions to $K$ predicted domains will score $\frac{1}{K}$. Figure 17 has the plots for predicted domains for two domain proteins in SCOP. The number of predicted proteins ranges from 2 to 7. On each curve, the scores that are higher than $\frac{1}{K}$ correspond to the predictions better than random. For proteins with two SCOP domains where we also predicted two domains\(^\text{20}\) the overlap is better than 80% more than half the time. As we will see in the individual examples presented in the next section, these predictions are very helpful when there needs to be a preliminary analysis for the modular domain architecture of a protein.

Figures 18 and 19 show the similar plots for three- and four-domain proteins, respectively.

\(^{20}\)These are the cases where SCOP domains are of average size.
Figure 17: The percentage of the predicted domain positions overlapping with the SCOP domain positions for proteins with two SCOP domains.
Figure 18: The percentage of the predicted domain positions overlapping with the SCOP domain positions for proteins with three SCOP domains.
Figure 19: The percentage of the predicted domain positions overlapping with the SCOP domain positions for proteins with four SCOP domains.
3.7 Randomly-selected Examples from the Test Set

In this section, we discuss individual cases. In order to see all block structures in the alignments, we have sorted the alignments in the figures are according to their beginning positions. We start with an example 1KHB_A,\(^{21}\) where the number of predicted domains are higher than the actual number of domains and the success rate is only 60\%. The reason for the low success rate is twofold. First, the sequence length is 625, which with our method sets the maximum number of domains in the protein to four. In contrast, there are only two SCOP domains, one

\(^{21}\)See http://www.pdb.org/pdb/explore/explore.do?structureId=1KHB
of them is very large (360 amino acids). Second, the alignments show three block structures, which do not correlate well with the SCOP domain boundaries. This is an interesting example because CATH structural domain database shows three domains in this protein two of which are discontinuous. The CATH domains correlate better with the alignment blocks. However, CATH uses a set of computational methods to analyze the structure of proteins, rather than manual analysis. Such examples need to be studied further.

Figure 21: SCOP domain positions match 70% of the predicted domain positions in the protein 1KKH_A.

---

22Domain 1: [10-63, 90-258], Domain 2: [66-85, 332-413], Domain 3: [260-330, 416-620]. See [http://www.cathdb.info/chain/1khhA](http://www.cathdb.info/chain/1khhA)
Figure 21 shows an example where the overlap is 70%. The number of domains are guessed correctly. However, the alignments indicate that an end segment in the query protein with significant length shows no similarity to nearly all of the aligned proteins. Since domain homology occurrence is the only information to exploit in our approach, our method predicts a domain at the end of the query protein which does not exists in SCOP. On the other hand, there is a SCOP domain covering the the initial segment of 100 amino acids and some of the database proteins are missing this segment. Yet, in the alignments for this domain the evidence is weaker than for the predicted domain at the end. If we increase the maximum number of domains to three, our method would predict the first SCOP domain. At this point, we should leave to the user the decision about whether the query segment with missing alignments is a novel domain that has not yet been classified.

The remaining three figures present examples that have 80, 90 ,and 99% overlap with the SCOP domains. The figures show that above 80% overlap, our predictions will be very helpful to molecular biologists who need to find out the domain positions as a preliminary step in their experimental analysis. As we mentioned earlier in the explanation of Figure 17 for proteins with two SCOP domains where we also predicted two domains, the overlap exceeds 80% more than half of the time. However, in these examples, there are also short segments assigned to a wrong domain class. These false assignments come about because our initial method do not take into account the locality of the positions and the likelihood of neighboring positions to belong to the same domain. Fortunately, we solve this problem by constructing a simple HMM model that enforces continuity of the predicted domains as the last step.
Figure 22: SCOP domain positions match 80% of the predicted domain positions in the protein 1WOO_A.
Figure 23:

SCOP domain positions match 90% of the predicted domain positions in the protein 1UAS_A.
Figure 24: SCOP domain positions match 99% of the predicted domain positions in protein 1LNZ_B.
3.8 Discussion

The DSM model clusters amino acid positions in a protein sequence based on its alignments to a set of protein sequences in a database. Although it is a simple and clear model, it is weak in the face of the complexity of deriving domain knowledge from alignments only. It has multiple limitations.

First, because the model relies heavily on sequence alignments, the quality and number of alignments have a large effect on the prediction success. Unfortunately, in a database search for sequence alignments, both the quality and the number can vary to a great degree. In such a case, the accurate prediction of domain boundary positions becomes particularly vulnerable. Moreover, sequence conservation among homologous domains is high inside the domain cores, but decreases near domain boundaries. Consequently, the computed alignments can be consistently shorter than the domain length. Short alignments could lead to predicted domain boundaries that are significantly distant from the actual positions, or to multiple predicted domain boundaries around a single real boundary.

Second, the MDA model assumes that two positions will appear in the same alignment if they are in the same domain. This assumption is helpful to distinguish among different domains, but the MDA model cannot distinguish domains occurring in almost same sets of proteins. Such domains may be part of a single functional unit, although they are structurally independent. We consider such structural domains as a single evolutionary unit; however, it would be beneficial to determine the boundaries (An evolutionary domain is divided into structural domains.) Although, the first limitation can be eliminated with noise-free data, the second limitation can only be overcome if we use additional sources of information.

Third, the protein sequence positions are clustered independently without any constraints in the neighboring positions. In the real world, the neighboring positions are likely to belong to the same domain except those at the domain boundaries. Switching between domains in short intervals is unlikely. Fortunately,
protein sequence searches return long continuous alignments. Yet, the alignments around domain boundaries are noisy and the alignment profiles may arbitrarily match one cluster or another, causing short lengths of protein sequence to be assigned to multiple clusters. For instance, in the example of 1B70 (sections 3.2.2 and 3.5.2), we had a few intermittent positions with the wrong cluster.
CHAPTER 4
FURTHER IMPROVEMENTS ON THE
PROBABILISTIC MODEL

4.1 Overview

In this chapter, I introduce a new model, Domain Architecture Hidden Markov Model (DA-HMM) in order to integrate two distinct sources of observations: the alignment column signature of a sequence position (as in the Domain Signature Model) and the linker propensity of the amino acid in that position. The combined DA-HMM approach increases success in predicting protein domains by accounting for both the high likelihood of domain continuity across adjacent positions and the high likelihood of domain switching around sequence segments that connect two domains.

Addressing Limitations in the Domain Signature Model (DSM) As demonstrated in the previous chapter, the DSM has multiple limitations. One limitation is that the alignment column signatures of sequence positions are clustered independently with no regard to neighboring positions. The DSM is a plain mixture model and does not allow us to take into account the dependency among neighboring positions to be in the same domain. In the real world, it is much less likely that two adjacent positions belong to two different domains than for both positions to be in the same domain. The total independence of adjacent positions causes the DSM to assign short segments to switching domains. A heuristic to solve this problem is to smooth posterior probabilities using a window large enough to remove segments that are considered too short. However, this is not a principled solution.

In order to incorporate the dependence of adjacent sequence positions, we expand the DSM to a Hidden Markov Model (HMM). The domain classes of
The DSM become states in a HMM and the probability of the transition from one domain state to another is conditioned on the current domain state. Consequently, the domain assignment of the current position becomes one of the determinants for the domain assignment of the next position.

Another limitation of the DSM approach is that it relies solely on local sequence alignments, thereby making its prediction accuracy very sensitive to how the local alignments reflect the modular domain architecture pattern, i.e., the quality and number of alignments. In the previous chapter, we presented examples of the local alignments that differed significantly from the SCOP domain assignments. This may be due either to an insufficient number of domain instances in the database and weak conservation among the domain instances or to the fact that the evolutionary domains do not match the structural domains. Fortunately, we can always improve on this limitation by taking into account other sources of observations that contain any information on the domains besides sequence conservation. For instance, Nagarajan et al. [37] train a neural network to recognize domain boundaries using multiple amino acid residue properties of protein sequence such as hydrophobicity and molecular weight. In the new model, we add the prediction of sequence segments that connect two domains, called linkers. Previous research has shown that amino acid frequencies in the linkers are distinct from amino acid frequencies inside the domains, referred as amino acid linker propensities. The distinction can be used to predict the linker regions, [5,17,20]. We integrate the previous work on linker prediction and DSM prediction in order to overcome noise in the local alignments—due to insufficient number and quality—to improve the accuracy of prediction.

4.2 Building a Hidden Markov Model (HMM) for MDA

Expanding from the Domain Signature Model (DSM) to a Hidden Markov Model (HMM) enables us to take into account the domain assignment of the cur-
rent protein sequence position in choosing a domain for the next position. The new model I propose here, Domain Architecture-Hidden Markov Model (DA-HMM), has a state for each domain class in the original DSM and a new linker state that represents the segments of a protein sequence linking domains. Conditioning on the current domain state is implemented by transitions within a domain state and between a domain and the new linker state. By building an HMM, we introduce a state path that corresponds to a domain-linker sequence in the protein. The state path mimics the most common form of protein domain architecture—a series of continuous domains connected with linker regions. Figure 25 shows the domain linker structure of the example protein 1B70_B analyzed in the previous chapter (Figure 8). In other words, conditioning imposes a graphical structure on the domain architecture such that it allows us to capture the continuity of individual domains by discouraging frequent switches between domains.

Before describing the new model, I review previous work on amino acid linker propensities and provide a brief introduction to HMMs.

### 4.2.1 Amino acid propensity indices

The homology-based prediction of protein domain architecture is not applicable when a protein has sequence similarity to a few or no other proteins. A significant number of hypothetical proteins from sequenced genomes bear little sequence similarity to other proteins. Although improved sequence-based prediction tools and methods can find more homologs, with the ever-growing universe of genomic proteins, there is always a subset with little sequence similarity to oth-
ers. The continued existence of such a subset of proteins has created a renewed interest in predicting the protein domain architecture based solely on a protein’s amino acid sequence in the absence of homologous proteins that can be detected by similarity searches.

In the absence of homology, the domain architecture prediction becomes the prediction of linker regions. The linker regions which are usually unstructured regions of varying length link compact domains together. Long loops without structure are thought to be linkers. For instance, Liu and Rost\cite{34,35} use secondary structure prediction, i.e., helical and beta-sheet substructures, to identify linker regions. Yet, many domains also have long looping regions out stick out of the compact hydrophobic core structure. However, these loops are not linkers. Moreover, recent research suggest that linker regions are characteristically different than long loops. Gokhale and Khosla\cite{20} show that linkers are a functional part of the processes of the domains they link. Amino acid frequencies inside linkers are statistically different from loop regions. Although linkers are known to have low sequence conservation, a certain set of amino acids is known to occur frequently. For instance, Galzitskaya and Melnik\cite{17} argue that domain boundaries have high Ala and Gly content resulting in flexibility of the backbone, and Pro residues in these regions induce hinge formations for dependent domain orientations.

To capture the information in amino acid propensities, Suyama and Ohara\cite{46} define the \textit{linker index}, $y_l$, of an amino acid as

$$y_l = -\ln\left(\frac{f_l^{\text{linker}}}{f_l^{\text{domain}}}\right),$$

where $f_l^{\text{linker}}$ and $f_l^{\text{domain}}$ are the relative frequencies of amino acid $l$ in the linker and core domain regions, respectively. They use the minima of the smoothed index function over the amino acid sequence of a protein to predict linker regions. This is a simple prediction algorithm with no modeling. In a later paper by Bae et al.\cite{5},
the authors train an HMM using the PFAM domain database that predicts linker regions from the smoothed index over the protein amino acid sequence.

The HMM has two states: linker and non-linker. In each state, the smoothed linker index has a distinct probability distribution. These distributions are assumed to be normal; their mean and standard deviation are computed using a training set of proteins whose domain-linker architectures are taken from the PFAM domain database. Once trained, the HMM gives us the posterior probability of being in a linker or non-linker region on observing the smoothed linker index of each position. In order to obtain hard partitioning of the protein sequence into domain-linker architecture, one needs a cutoff value on the values of posterior probabilities, as we had in the DSM approach. The authors test multiple cutoff values to find the one that has the optimal tradeoff between sensitivity (i.e., $TP/(TP + FN)$) and specificity (i.e. $TP/(TP + FP)$). At the optimal cutoff value of 0.75, the authors report the sensitivity of 63% and specificity of 93% on the test set.

In DA-HMM, we incorporate the transition and emission probabilities obtained by Bae et al. for the two-state HMM described above. However, since the domains in the PFAM database used in training are based on sequence conservation and the sequence conservation-based domains do not necessarily match the structural domains, we expect a lower accuracy on our test set obtained from the SCOP structural domains.

4.2.2 A brief introduction to Hidden Markov Models (HMM)

Hidden Markov Models are used for statistical modeling of “linear” problems such as sequences. An HMM is a finite model that describes a probability distribution over an infinite number of possible sequences. An early paper by Rabiner gives an overview for the theory of HMMs. In our problem, since the

---

1 Sensitivity is the percentage of actual linker residues that are predicted to be in a linker; specificity is the percentage of predicted linker residues that are truly in a linker.
2 They have been extensively used in speech recognition and time series problems.
sequences are made of finite number of discrete values (the averages of the fixed number of linker index values and vectors of zero and one), the number of possible sequences is finite albeit exponentially large.

We can view an HMM as either “generating an output sequence” or “accepting a sequence of observation.” Here, I will use the “generative” terminology. An HMM consists of a set of states, each of which corresponds to a position in the sequence. Each state emits an output according to output emission probabilities; the states are connected by transitions. Starting from the initial state, a path of states is generated by moving from state to state according to the state transition probabilities until the end state is reached. Each state in the path emits an output according to that state’s emission probability distribution, creating an observable sequence of outputs.

In an HMM, the choice of the next state in the path is dependent only on the identity of the current state, i.e, the state path is a Markov chain. However, this state path cannot be observed; it is hidden. We can only observe the output sequence generated by the hidden states. The most likely state path must be inferred from the HMM output emission and state transition probabilities and the observed sequence.

**Definition 4.2.1** An HMM is a finite state machine defined by a quintuple \((Q, \Sigma, A, B, \Pi)\) where \(Q\) is a finite set of states, \(Q = \{q_1, \ldots, q_K\}\), each of which is associated with an emission probability distribution over the set of output symbols \(\Sigma\); \(A\) is the set of transition probabilities between states, \(A = \{a_{kl}\}\); \(B\) is the set of output emission probabilities of observed variables in each state, \(B = \{b_k(j)\}\); \(\Pi\) is the initial state probabilities. Let \(\sigma_i\) be the random variable which holds the \(i^{th}\) observation, \(\sigma_i \in \Sigma\), the sequence of observations is \(\sigma_1, \ldots, \sigma_M\). Let \(\pi_i\) be the random variable which holds the state of the HMM generating the output \(\sigma_i\), \(\pi_i \in Q\), the sequence of states is \(\pi_1, \ldots, \pi_M\). Then,

- \(a_{kl} = P(\pi_{i+1} = q_l | \pi_i = q_k)\),
- \(b_k(j) = P(\sigma_i = o_j | \pi_i = q_k), o_j \in \Sigma.\)
Usually the set of states $Q$ and the set of output symbols $\Sigma$ are understood, so an HMM is said to be a triple $(A, B, \Pi)$.

**Assumptions of HMMs.** The HMM model is underlined by two assumptions. First, pairwise (or higher order) correlations between sequence positions are ignored. In our case, any two protein sequence positions are assumed not to be correlated. This may not be true in the real world, i.e., two amino acids in a protein might be functionally related, but for domain architecture determination it is satisfactory. Second, HMMs assume that sequences are generated independently by the model. The second assumption is not relevant in our case since we are not training on multiple observations.
4.2.3 The new model DA-HMM

We expand our domain signature model to an HMM to introduce a state path that corresponds to a domain-linker sequence in the protein. In the DSM, every class represents a domain, which means that each sequence position is assumed to be in one domain or another. However, the linkers may be long and the amino acid positions in the linkers are not necessarily classified as a part of a domain. It is better to have a non-domain class to which we can assign the positions between domains. Moreover, we want to exploit the different statistics of amino acid composition in linkers and domains for which Bae et al. developed a model as mentioned above. Hence, the DA-HMM has a state for each domain and an additional state for linkers.

The DA-HMM has transitions between states that represent the likelihood of switching from a domain to a linker region, and vice versa. When we are in a domain and the linker index of the next position is high\(^3\), it is more likely that the next position is in the same domain as the current position. The DA-HMM should make a transition from a domain state to the linker state, only after encountering low linker index values and the significant change in the alignment column signatures. The same is true for the linker region. When the linker index stays low around the current position and alignment signature of the position shows little similarity to any domain signature, it is likely that we are still in a linker region. When the linker index increases and the alignment signature observation becomes more likely to come from a domain distribution, the DA-HMM should move from the linker state to a domain state.

The HMM architecture shown in Figure 26 has \( K + 1 \) hidden states, \( Q = \{c_0, c_1, \ldots, c_K\} \), where \( c_0 \) is the linker state, \( c_k \) is the state representing the \( k^{th} \) domain, and \( K \) is the number of domains in the protein. We have one state for each domain and an additional state that represents all linker regions. The output sequence consists of pairs for each position of the protein sequence. The first element

\(^3\)Linker regions have low index values by definition
of the pair is the smoothed linker index of amino acid in that sequence position. The second element is the alignment column signature of the position obtained by searching a large protein database. We use the same alignment column signatures as in the DSM. Let $y_i$ be a smoothed linker index value and $s_i$ be an alignment column signature, the output generated by the model is a sequence $\sigma$ of $M$ pairs $\sigma_i = <y_i, s_i>$ where $M$ is the length of the protein sequence. From now on, we will drop the word “smoothed” for the linker index values, assuming that a smoothing function is employed.

Let $\pi_i$ be the random variable which takes the value of the model’s state, $\pi_i \in Q$, at protein sequence position $i$ and the state path traveled to generate the data be $\pi = \pi_1 \cdots \pi_M$. Let $P(s_i|\pi_i = c)$ denote the probability of emitting the alignment signature $s_i$ in state $c$ and $P(y_i|\pi_i = c)$ denote the probability of emitting the linker index value $y_i$ in state $c$, $c \in Q$.

**Signature emission probabilities** $P(s_i|\pi_i = c)$. As in the DSM, each domain state $c_k$ has an associated alignment signature probability vector $v_k$. The $j^{th}$ ele-
ment of \( v_k \) is the probability of seeing 1 in the \( j^{th} \) position in the alignment column signature of a sequence position in domain \( k \). Each individual probability \( v_k(j) \) is computed empirically from the input alignment column signatures and initial MDA (i.e., hard partitioning of the sequence positions) as the frequency of ones in all alignment signatures of positions in domain \( k \). Let \( f : \{c_1, \cdots, c_k\} \rightarrow \mathcal{P}\{1, \cdots, M\} \) be hard partitioning of the alignment column signatures,

\[
v_k(j) = \frac{\psi + \sum_{i \in f(c_k)} s_i(j)}{\psi + \|f(c_k)\|}, \quad j \in [1, \cdots, N].
\]

(4.1)

where \( \psi \) is a pseudocount. In our work, we obtain the hard partitioning from the posteriors of the DSM as follows

\[
f(c_k) = \{i | P(s_i | c_k) \geq 0.5 \} \text{ for } k > 0.
\]

The emission probability \( P(s_i | \pi_i = c_k) \) is the product of vector elements

\[
P(s_i | \pi_i = c_k) = \prod_{j \in [1..N]} v_k(j)^{s_i(j)}(1-v_k(j))^{(1-s_i(j))} \text{ for } k > 0.
\]

(4.2)

Since the linker state has no particular alignment column signature, we choose to use the uniform distribution over all column signatures in the alignments. To put it differently, its signature vector has equal probability for seeing 1 or 0 for each position \( j \),

\[
v_0(j) = 0.5, \quad j \in [1, \cdots, N].
\]

**Linker index emission probabilities** \( P(y_i | \pi_i = c) \). In Section 4.2.1. we discussed the two-state HMM developed by Bae et al. \[5\] that generated linker index values in linker and non-linker states. The model was trained on the set of sequences whose domain architectures were taken from the PFAM database. The authors computed two different normal probability distributions for the linker index value emission probability in the two states. We incorporate linker and non-linker prob-
ability distributions in our work as the emission probabilities in linker and domain states, respectively. All domain states have the same non-linker probability distribution. We took the estimates of the means and the standard deviations for the normal distributions directly from the table given by the authors.

**Joint emission probabilities** $b_k(i)$. We assume that the linker index values and alignment signatures are conditionally independent. Theoretically, the output emission probability in state $c_k$ is the joint probability

$$b_k(i) = P(y_i|\pi_i = c_k)P(s_i|\pi_i = c_k).$$

However, the magnitudes of these two probabilities differ substantially. More importantly, the probabilities of two signature vectors can differ significantly even if the vectors differ only in a few positions. This difference is due to the fact that probability $P(s_i|\pi_i = c_k)$ is a product of $N$ individual probabilities in the probability vector $v_k$. Such difference in magnitude makes the signature probabilities dominate in determining the optimal state path. In order to overcome this, we took $N^{th}$ root of the signature probabilities and renormalized them. In this scheme, the new $P(s_i|\pi_i = c_k)$ are no longer estimates, but rather numbers scaled to function within the probabilistic framework.

**Transition probabilities.** The DA-HMM in Figure 26 has a star-shaped structure where transitions are only between the linker state and the domain states, and within the same state. Our model is a natural product of two models: the K-state DSM (discussed in Chapter 3) and the two-state\(^4\) model developed by Bae et al. \[^5\]. Integrating them into a single model accounts for multiple domains with different signature characteristics. We directly used the transition probabilities in the two-state model since optimizing the joint model further seemed unlikely to make a significant difference. Let $p_0$ be the probability of looping the linker state and

\[^4\text{linker and nonlinker states}\]

\[^5\text{linker and nonlinker states}\]
be the probability of looping in the non-linker state of the two-state model. We set the transition probabilities in the DA-HMM as follows. The probability of transition from domain state $c_k$ to another domain state $c_l$ is zero,

$$a_{kl} = 0 \text{ for } k \neq l, k, l \in [0, \ldots, K],$$

$$a_{00} = p_0$$

and for $k \in [0, \ldots, K]$

$$a_{kk} = p_1$$

$$a_{k0} = (1 - p_0)/K$$

$$a_{0k} = 1 - p_1.$$  

**Start states.** The DA-HMM does not have a separate start state. The state path can start in any of the domain states with uniform probability, $\Pi(c_k) = 1/K, k > 0$.

**Initialization.** We initialize the HMM by first computing a DSM for the protein. The class posteriors obtained from the DSM, $P(d_k|s_i)$, are used to assign each sequence position and its column signature into a domain. Then, we estimate each domain’s signature probability vector as the frequency of seeing 1 among assigned column signatures using Equation 4.7. As in the DSM, the estimated signature probability vectors are used to compute alignment signature emission probabilities for each state, $P(s|\pi_i = c)$ (see Equation 4.2). As mentioned above, the linker and non-linker probability distributions for the linker index are taken from Bae at al. [5].

### 4.3 Examples Using DA-HMM

In this section, we review the individual cases that analyzed under the DSM in the previous chapter and compare the results when using DA-HMM. We
will compare posterior probabilities, \( P(d_k|s_i) \) of the DSM and \( P(\pi_i = c_k| < y_i, s_i >) \) of the DA-HMM. As the Figures 27-30 show, the HMM is able to impose the continuity of domains while the previous model DSM assigns short segments to alternating domains. The posterior probability \( P(\pi_i = c_k| < y_i, s_i >) \) is smoother than \( P(\pi_i = c_k|\sigma) \) and the adjacent positions are assigned to the same domain. The HMM posterior graphs also show the linker state posteriors (red line), which stay low along the domain cores and increase around the linkers. The protein 1KKH.A is particularly interesting. The alignments for this protein do not correlate with domains. We observe that a long segment of about 50 amino acids at the end of the protein is missing in most of the alignments, but it is not a domain. Because of the low correlation between local alignments and domain boundaries, neither model can predict accurate domain positions. In such cases, clearly alignments are of little help and linker index has a very weak signal, we need other sources of information such as secondary structure of the protein.
Figure 27: Posterior probabilities for protein 1KHB_A in the DSM (alignments are sorted by beginning positions) and the DA-HMM (unsorted).
Figure 28: Posterior probabilities for protein 1KKH_A in the DSM (alignments are sorted by beginning positions) and the DA-HMM (unsorted).
Figure 29: Posterior probabilities for protein 1WOO_A in the DSM (alignments are sorted by beginning positions) and the DA-HMM (unsorted).
Figure 30: Posterior probabilities for protein 1UAS_A in the DSM (alignments are sorted by beginning positions) and the DA-HMM (unsorted).
CHAPTER 5
APPLICATIONS

Throughout the thesis I have mentioned a number of applications where prede-
termination of domain positions, which can be identified by our approach, in a
protein is a prerequisite. In this chapter, I will present these applications in detail.

5.1 Domain-Specific Function Annotation

Protein function can be defined in multiple levels ranging from low level
chemical reactions to high level cellular functions. Building an ontology for protein
and other cellular functions is an ongoing process as we learn more about cell
biology; there are many current projects such as Gene Ontology Consortium [9,10]
and Metacyc [28–31]. In multi-domain proteins, each domain can have a separate,
simple function while together accomplishing a complex function. For instance,
membrane-bound signal-transduction proteins (e.g., protein kinases) usually have
a domain outside the cell to receive a chemical signal, another domain inside the
cell membrane, and additional domains inside the cell to pass the signal down in
the signal pathway. On one hand, protein sequence databases like SWISSPROT [4]
usually annotate a protein in multiple levels, especially if its high-level function
is not known. On the other hand, protein domain model databases like PFAM [7]
associate a domain-specific function to each model.

DNA sequencing technology has made it possible to collect the entire
DNA sequence of the genomes of various organisms. Meanwhile, scientists de-
veloped powerful statistical methods to predict a large number of protein-coding
genes in these genomic sequences, i.e., proteome. Every time a genome is se-
quenced, partially or fully, one can easily find a large number of protein sequences
coded by the genome. Biologists are then faced with the problem of associating a function with these hypothetical proteins. Certainly, laboratory experiments are the most accurate way to determine a protein’s function. However, biological experiments that target individual proteins may take a long time to conduct. On the other hand, protein function can be derived faster through a computational process called function annotation. Indeed, a common way to bring together a first-hand knowledge of an organism’s proteome is to hold a “jamboree” session among few dozen researchers to annotate by computational means all hypothetical proteins in its newly sequenced genome.

Automated function annotation relies on homologous proteins whose functions are already known or assigned through similar computational means [16]. In its simplest form, protein annotation transfers a function to a hypothetical protein from one or more of its homologous proteins. Protein homology is usually established through sequence similarity. In a typical case, the hypothetical protein’s sequence is used as a query to search a database for similar proteins, i.e., targets. If there is more than one target protein, a consensus of all functions derives the protein’s function. However, there are better ways to identify a function from sequence. For instance, scientists build databases of sequence conservation models using homologous proteins or domains in many known functional families (i.e., PFAM, etc.). The model of a family predicts whether a protein or a domain in a protein belongs to the family. If so, the protein is annotated with the function of the family. But preparing a database of such models takes time, and it is hard to keep up with the fast growing size of protein sequence databases and new functional domains. For this reason, predicting function by directly mining the ever-growing

\[1\] In recent years, microarray technology has made large experiments possible including all genes on a single genome to be conducted in a little time and space. These experiments currently take on the relationship between (protein coding) genes within a genome at their expression levels. Genes that are expressed as correlated in time or space are assumed to have related functions. It provides an indirect attack on the genes with unknown function by relating them to the genes of known function or cell environment (cancer) and inferring the functional pathways that a gene’s product(proteins) might take part.
protein sequence databases with search tools for homologs is still popular among biologist.

The automated function annotation that transfers the known or predicted functions of a matching target set of proteins to a hypothetical query protein is an error-prone process. Some types of errors are due to the existence of multi-domain proteins. This is because a multi-domain protein can have sequence similarity to another protein along a single domain, or any segment that involves one or more domains but not the domains of the transferred function. I can identify two possibilities where a query sequence can be annotated incorrectly.

First, if a query sequence used in searching a database of protein sequences is a multi-domain protein, then the set of target sequences found in the search can be partially similar to the query sharing some or no similarity to each other. (For instance, if the query has two domains, A and B, and one subset of the matching target proteins has only domain A and another subset has only domain B, then the consensus would be taken only within a subset of targets aligned to each domain. Knowing the domain architecture helps to sort out the target proteins for domains so that the annotation is more meaningful, and less error prone. The probabilistic model presented in this thesis not only partitions the query positions into domains, but also computes the conditional probability of a target protein for each domain. Therefore, a distribution of target proteins for each domain comes out of the model naturally. A separate consensus function for each distribution can be computed. Moreover, if the target protein is similarly analyzed, the joint posteriors of the domains can be used to decide whether to transfer the function or not.

Second, proteins in a database contain multiple levels of function annotations; sometimes only partial annotations exist for individual domains, and annotations are propagated through chains of sequence matches. Consequently, transferring a function of a multi-domain target protein to a query protein can lead to incorrect annotation if the transferred function does not belong to the shared domain. For example, suppose for the query protein of two domains AB, a target
protein found in a database contains only domain B and another domain C that is not in the query protein and only the function of domain C is known. In this case, the annotation of query sequence AB should not include C’s function in the consensus.

**Position-Specific Annotation** The problem of sequence similarity corresponding to regions of proteins that are not involved in the transferred function has been recognized before. In a 1998 paper, Miguel Andrade proposes a position-specific annotation as a solution [3]. His approach tries to sort out the target proteins’ functions with a method that might be considered an investigation of possible domains. At the beginning, the protein of interest is used to query a database of proteins using the BLAST [2] sequence similarity search tool. Next, the annotated functions of the proteins found in the database are preprocessed to obtain multiword-units that are meaningful function descriptions. The method then builds a matrix whereby each entry \((i,j)\) indicates whether the \(i^{th}\) target protein has the \(j^{th}\) word-unit. Andrade has a well-thought out scoring function for each query protein sequence position. The scoring function uses the (target protein \(\times\) word-unit) matrix to compute a score for the word-unit assignment to a given position in the query sequence. Because, the range of scoring function is in \([0, 1]\), so it works as likelihood. But, it is not true likelihood since there is no underlying probabilistic model. The function is carefully constructed so that its values display the desired behavior for given values of parameters. Andrade does not explicitly aim to predict the domain architecture of a protein sequence; however, it comes out as a byproduct of the scoring function when we think of each word-unit groups of proteins as representing one or more domains.

**Comparison.** In contrast, both the DSM and DA-HMM approaches presented in this thesis specifically aims to solve the modular domain architecture of a protein. There is an underlying probabilistic model of domain signatures. Moreover, our methods use database proteins themselves rather than their function annotations.
Some of the sequences in the target database may not have any annotations yet, but they can still contribute to the determination of domains. This is important if we are to find novel domains. Although a similar processing of words in annotations is necessary at the time of building consensus function for each domain, the word processing is done after domain architecture is determined. This means that the prediction of domain architecture does not depend on the outcome of word processing. On the other hand, if we want to make use of word-units instead of individual target proteins, we could easily modify the probabilistic model so that the word-unit counts serve to weigh the observations. Instead, target proteins are weighted equally.

5.2 Experimental Structure Determination

Nuclear Magnetic Resonance (NMR) Spectroscopy and X-ray crystallography are two techniques to determine the structure of a protein. Crystallography gives the most accurate atomic-scale structure data, but it can only analyze proteins that form crystals. NMR allows us to examine proteins in their dissolved form, more similar to their natural state in the living cell. However, can handle only small size proteins. One approach for enabling NMR to work on a large multi-domain protein is to cut the protein into structural domain fragments. Since structural domains are parts of a protein that can fold independently from the rest of the protein–and are stable by themselves–one can determine the structure of each domain independently and combine the results if possible. X-ray crystallography can handle proteins with two or three domains. Large number of domains, however, can give a protein more flexibility which could fail its crystallization. Having advanced knowledge of the number domains is helpful so that crystallographers know what type of protein they are dealing with.

Computational methods that predict domain architecture from sequence similarity information can certainly help to decide how to cut the large proteins.
The main assumption behind the usefulness of computational methods is that domains are connected by loop regions. If a protein is cut inside a connecting loop region, the lack of few amino acids does not affect the structure of resulting domains. There are cases where this assumption fails. For instance, a very short connecting region exists between two domains, a missing amino acid can prevent a structural domain from folding properly.

In such cases, limited proteolysis\(^2\) might be helpful. However, this is an experimental technique that requires expressing and purifying sufficient quantities of the protein. It is not suitable for quick analysis.

\(^2\)In limited proteolysis, specific enzymes cleave the domain linking segments on the surface of a protein that are exposed. But, the surface amino acids are not necessarily loop regions nor all loop regions are at the domain boundaries. The preliminary computational prediction of approximate domain positions is even helpful for limited proteolysis.
CHAPTER 6

CONCLUSION AND FUTURE WORK

Predicting the modular domain architecture (MDA) of a multi-domain protein is a prerequisite in exploring its function in the cell and its working structural mechanism. This thesis has presented an approach to predict the MDA of a protein from the local alignments between its domains and their homologs in other proteins. Clustering homolog signatures of individual sequence positions obtained from alignment columns is a novel approach to predict the modular domain architecture of a protein distinct from the previous approaches. The approach is founded upon the assumption that domains are the modular elements with which evolution has constructed large and complex proteins by recombination. The homologous domains in different proteins and in different organisms usually have similar structure and function. Although the amino acid sequences of homologs have evolved a way by mutation, the similarities can still be detected; the corresponding amino acids of two sequences can be aligned to each other to see where mutations have occurred. It is true that not all domains with similar structure and/or function are homologs and not all homologous domains retained their structural, functional or detectable sequence similarity. Yet, the validity of assumption holds in majority of the cases.

The local alignments that partially cover the query protein sequence positions provide a pattern from which we can build a probabilistic model of the domains. The DSM model has a class for each domain, a class represents a cluster of proteins, i.e., a domain signature, where the homologous domains occur in every element. The examples and large scale test results presented above demonstrated that the DSM model was a successful predictor for the positions of domains in the

\(^1\)in the order of decreasing possibility
amino acid sequence of a protein when the database search finds a sufficient number of local alignments that overlap with one or more domains on the protein. The large scale test on the PDB protein was conducted as a batch operation where the number of domains was determined by the DAEM algorithm and start and stop values of the loop variable $\beta$. These values are parameters to the DAEM algorithm and can be set by the user for each individual protein to produce the expected number of domains. Yet, in the test runs we used the same parameter values for all proteins resulting in over-fragmentation of large domains. In an individual application, the user can easily recognize the over-fragmentation and stop the DAEM algorithm at an earlier stage resulting in a more accurate prediction.

The DSM model has a shortcoming since it does not account for the likelihood of the neighboring positions to be in the same domain. The next model, DA-HMM, takes into account the neighboring positions. By using an HMM, we introduced a state path that corresponds to the partitioning of the sequence into domains, we incorporated additional information about the amino acid propensity of being in a linker. The results showed that the DA-HMM corrects much of the fragmentation of the sequence. However, when the alignment data carries poor information about the domain architecture, the prediction of the DA-HMM also suffers significantly. The model could be further developed to take into account more characteristics of protein sequences and alignments that may help to improve predictions such as secondary structure.

**Future work.** When we designed an HMM that combined homolog signatures with the amino acid linker propensities, we had two expectations.

The first expectation was that the graphical structure of the HMM would impose enough locality information to smooth the posteriors and help us to overcome the over-fragmentation due to noise in the data. This expectation was realized.

The second expectation was that by incorporating amino acid linker propensities, it would be possible to predict the domain boundaries more accurately. In
this, we were disappointed. Our tests showed that including linker propensities did not substantially improve the accuracy of our domain boundary predictions as compared with reference domain decompositions, particularly for proteins with a large number of domains. Currently, we do not achieve high accuracy in resolving domain boundaries, even with HMM based models. It seems reasonable to ask whether or not the notion of a precise domain boundary makes biological sense, but if we assume that it does, we’re more optimistic about approaches (extending ours in spirit, if not detail) that rely on cross-genome analyses (e.g., attempts to reconstruct domain philogeny) than we are about approaches that emphasize the near boundary amino acid sequence of the query protein alone.

Although we applied the HMM-based models only to proteins with continuous domains, they can also be used for proteins with domains that consist of multiple discontinuous segments, albeit with modifications of the underlying graph of the HMM. I consider the compilation and testing of a set of such proteins as a part of future work.

Moreover, there are domains with no stable 3D structure in their native forms. These are functional intrinsically disordered domains. I plan to analyze the MDAs of proteins containing disordered domains, such as the nuclear-receptor co-activator-binding domain and the nuclear-receptor-interaction domain discussed in [14]. Since no stable structure exists for these proteins, sequence-based domain architecture information can be great help to understand their functions.
REFERENCES


