2012

Prediction and analysis of protein-protein interaction types using short, linear motifs

Manish Kumer Pandit

PREDICTION AND ANALYSIS OF PROTEIN-PROTEIN INTERACTION TYPES USING SHORT, LINEAR MOTIFS

by
Manish Kumer Pandit

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2012

PREDICTION AND ANALYSIS OF PROTEIN-PROTEIN INTERACTION TYPES USING SHORT, LINEAR MOTIFS

by

Manish Kumer Pandit

APPROVED BY:

Dr. William Crosby, External Reader
Biological Sciences

Dr. Ziad Kobti, Internal Reader
Computer Science

Dr. Luis Rueda, Advisor
Computer Science

Dr. Jianguo Lu, Chair of Defense
Computer Science

September, 2012

Declaration of Authorship

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Protein-protein interactions (PPIs) play a key role in many biological processes and functions in living cells. Hence, identification, prediction, and analysis of PPIs are important problems in molecular biology. Traditional solutions (laboratory based experiments) to this problem are labor intensive and time consuming. As a result, the demand of a computational model to solve this problem is increasing day by day. In this thesis, I propose a computational model to predict biological PPI types using short, linear motifs (SLiMs). The information contained in a protein sequence is retrieved using the profiles of SLiMs. I use sequence information as a distinguishing property between interactions types, mainly obligate and non-obligate. I also propose another model to predict PPIs using desolvation and electrostatic energies. These computational models use the information contained in the sequence, and desolvation and electrostatic energies of the protein complex as properties. After computing all the properties, the well-known classifiers, k-nearest neighbor (k-NN), support vector machine (SVM) and linear dimensionality reduction (LDR) have been implemented. Results on two well-known datasets confirm the accuracy of the models, which is above 99%. Analysis and comparison of the results show that the information contained in the sequence is very important for prediction and analysis of protein-protein interactions.

Dedication

I would like to dedicate this thesis to my parents and elder brothers in my back home. My first elder brother was always dreaming about my higher education and his dream was my inspiration. Thanks to my elder for his constant motivation and my Mom for all her prayers. It is because of their faith on me that I am able to complete this degree. I am also grateful to my second elder brother for his support and inspiration.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to Dr. Luis Rueda, my supervisor, for his steady encouragement, patient guidance and enlightening discussions throughout my graduate studies. Without his help, the work presented here could not have been possible.

I also wish to express my appreciation to Dr. Ziad Kobti, School of Computer Science and Dr. William Crosby, Department of Biological Sciences for being in the committee and sharing their valuable time.

I would like to thank Dr. Alioune Ngom for his valuable input in this thesis, Dr. Nick Chepurniy and the SharcNet team for helping in using the facility and deploying the installation.

Finally, I would also like to thank Gokul Vasudev, Navid Shakiba, Mina Maleki, Manoj Gajja and all my friends on and off campus for their consistent moral support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Computational Molecular Biology

Molecular Biology is the branch of biology in which all biological processes and functions are viewed and analyzed in terms of molecules and molecular chemistry. This branch is able to provide all the explanations of different types of interactions between cell systems such as different types of DNA, RNA, and protein complexes. Understanding and analyzing these interactions is very important. This branch of biology studies all details of such interactions. Molecular biology revealed the original convergence of geneticists, physicists and structural biochemists on a common problem; the structure and function of a biological complex. Key concepts of molecular biology include mechanisms, information and genes. The history of molecular biology provides the importance of the discovery of macromolecular mechanisms [29].

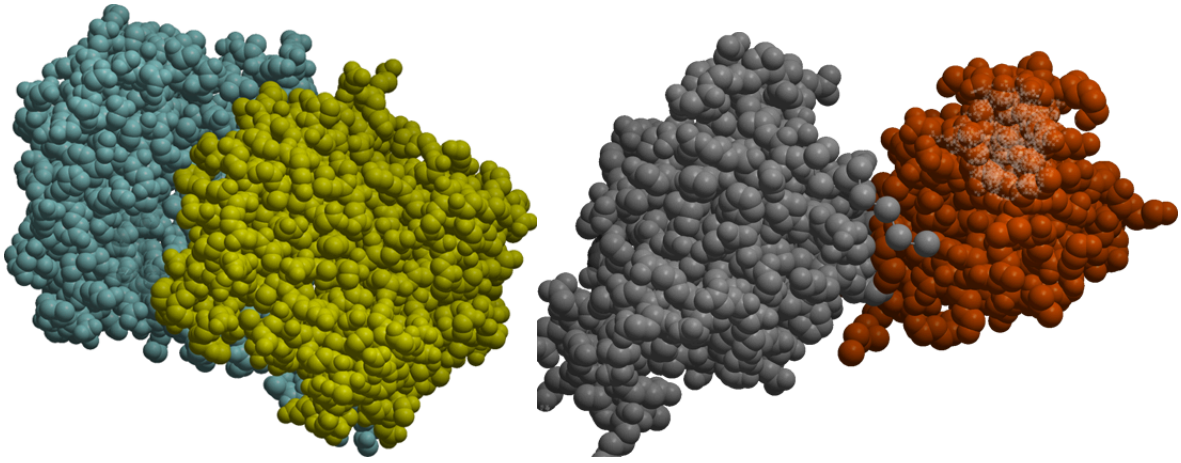
Computational molecular biology can be seen as an application of molecular biology and computer science along with statistics. The main idea is to accomplish molecular biological experiments and analysis using different computer science techniques [51]. These

days, computational molecular biology is very important in genomics. Data about different biological experiments and analysis are growing very rapidly. This rapid development in biological science is generating huge amounts of data every day. To manage and manipulate all the information and data efficiently there is no alternative of computational models. At present, analyzing DNA, protein sequences, protein complexes, and structure analysis of DNA and protein need powerful computational models. Besides all these, different machine learning, data mining, pattern recognition techniques are also being used to make the computational models more efficient, intelligent and accurate for different biological experiments. For example, the development and evolution of drugs needs information about biological processes, and hence this area of research is very important for health science.

1.2 Protein-protein Interaction

Polypeptide chains of amino acids that have a specific tertiary structure that perform a specific biological function are known as proteins. Proteins have different types and four levels of structure (details in Chapter 2). Based on their structure and function, different levels of classification are possible. In this thesis, I focus mainly on prediction of obligate and non-obligate types and their interactions. According to the authors of [39], precise collision in different proteins often leads to the final complex which is known as *obligate complex*, and when this collision forms an encounter complex that may finally lead to another different one is known as *non-obligate complex*. Figure 1.1a shows the graphical orientation of two chains (A:B) of *Aminoimidazole Ribonucleotide Synthetase (PurM)*, which is an obligate complex(1cli). Figure 1.1b shows two chains (A:E) of *RhoA.GDP - RhoGDI*, which is a non-obligate complex (1cc0).

Figure 1.1 shows a primary comparison between an obligate and non-obligate complex.



(a) Obligate complex - 1cli (A:B) (b) Non-obligate complex - 1cc0 (A:E)

Figure 1.1: Obligate vs. non-obligate complex. These figures were produced using ICM Browser [1] from the PDB [6] structure files of the complexes.

Obligate complex 1cli has a larger interface area between the chains (A:B), which indicates that this complex will have more interacting atoms and interactions. On the other hand, non-obligate complex 1cc0 has a smaller interface area and less interacting atoms between the chains (A:E). The primary focus of this thesis is to predict types of protein complexes using their structural and interaction data using a computational approach.

Biological cellular processes such as signal transduction, immune response, regulation of gene expression and many biological processes need oligomerization involved in PPIs. These interactions can be attractive or repulsive and depend on protein surface, environment conditions. A lot of efforts have been made to identify and understand different types of properties and factors for different types of interactions [40]:

1. β-factor
2. Solvent Accessibility
3. Geometric features

4. Evolutionary features

5. Physicochemical features

Different levels of information in different biological processes can be derived from different types of PPIs [39]. Obligate and non-obligate types of interactions come with information about the interaction lifetime and stability among others. Non-obligate interactions are usually less stable which makes predictions a difficult task [34, 54].

1.3 Feature Generation and Classification

To predict class types, every prediction method needs observed properties of the known class samples called *features*. Features are generally nominal or numeric values, and the process of calculating the features for each sample from a dataset is called *feature generation*. To reduce the size of the generated features from the input, I use feature extraction methods. Feature extraction is a popular pattern recognition method [27]. It is a special form of dimensionality reduction. When the length of the feature vector is very large, we may need to apply feature extraction methods to find a lower dimensional representation of that original feature vector. The transformation of high dimensional data to lower dimensional data is called *feature extraction* [27]. There are many feature extraction methods and for this thesis I use three linear dimensionality reduction (LDR) methods [43] (details in Chapter 3) which are:

- Fisher's discriminant analysis (FDA) [16, 21]
- Heteroscedastic discriminant analysis (HDA) [17]
- Chernoff discriminant analysis (CDA) [43]

Lower dimensional data obtained by these three LDR methods are then passed through quadratic Bayesian (QB) and linear Bayesian (LB) classifiers for final prediction [16]. For each classifier, prediction accuracy and time taken for that prediction are very important.

1.4 Motivation and Objectives

To understand different biological functions and processes based on protein sequence and structure many researchers have been working for stability. They are accomplishing either the traditional labor intensive biological experiments or adopting the computational approaches. For the computational approaches, they are gathering information based on the interactions and different properties of protein complexes. After each interaction, the shape of the complex and the functionality of the complex may change. Thus, by analyzing and understanding PPIs, a plausible mechanism for complex information can be delivered and also the explanation of different biological processes (e.g. signal transduction) is possible. This information about proteins helps researchers understand different diseases, which can lead to effective drug development and uncover a new arena of treatment. Different information from protein composition, structure, stability and interaction duration are key factors to differentiate two specific types of protein groups, one is obligate and another one is non-obligate which can help the research work greatly. Thus, studying these two types of PPIs helps researchers gather more information, select the best describing information for understanding and explaining different biological processes and mechanisms.

Although there are different types of PPIs, this thesis is mainly focused on obligate and non-obligate types of PPIs [34, 54]. During the interaction, the protein chains interact with each other even within themselves changing the geometric shape or other complexes to perform a specific biological task. Co-Immunoprecipitation or affinity chromatography [7,

23, 24] is often used to determine the whether an interaction is of obligate or non-obligate type. Such types of experiments are highly labor expensive and suffer from different types of system errors. Thus, efficient and intelligent computational models are necessary to solve such a problem successfully. In this thesis, the main focus is to determine the types of interactions (obligate or non-obligate) based on the different properties of the protein complexes (e.g. information contained in the sequences, desolvation or electrostatic energy and others).

Different properties, features and prediction methods can be used to solve this specific problem. Information contained in protein sequence is very useful to predict PPI sites [41] and also to predict PPI types [8, 44]. In this thesis, information contained in the sequence of a protein complex has been used to determine the types of the PPIs. Using the information contained in the sequence, I show that it is better than recently used properties such a NOXClass features (interface area, interface area ratio, amino acid composition of the interface, correlation between amino acid compositions of interface and protein surface, gap volume index and conservation score of the interface) [53], and desolvation energies [9].

The Protein Data Bank (PDB) [6] is the main source of the molecular structures of protein complexes. The PDB provides the structural information in different formats; text is one of the mostly used formats. This text format of the structure file is not perfect. It usually contains a lot of spacing errors, numbering errors, alignment issues. Besides these, in many cases the PDB does not provide complete structural information of protein complexes. Processing all the information of a protein complex at the atomic level is also time consuming and computationally expensive. Thus, a model based on structural information from the PDB can not be accurate and perfect and must be time consuming and computationally

expensive. Besides, a small number (approximately 80,000) of structures are known compared to millions number of sequences. As a result, models based on the protein structures are not compatible for a large number of proteins. One model that can replace the use of structural information with sequence information is an important subject to investigate, which is also a motivation to propose this model. On the other hand, sequence information is calculated from the SLiMs, which represent the conserved regions of protein complexes. The conserved region property for different types of complexes is different and can be used to predict PPIs, as shown presently.

1.5 Problem Definition

The main focus of this thesis is to classify a protein complex whether it belongs to obligate or non-obligate class. The aim is also to use information contained in sequences to classify obligate and non-obligate complexes. Hence, the usage of structural information can be avoided as using structural information needs more computing and in many cases, it is not available.

1.6 Contributions

The main focus of this thesis is to predict obligate/non-obligate PPIs using short, linear motifs. To solve the classification problem, this thesis includes:

- A new model (**PPI-SLiM-Seq**) to predict PPIs which uses sequence information as features and different machine learning techniques.
- Another model (**PPI-SLiM-DEEE**) to predict PPIs using desolvation and electro-

static energies.

- For SLiM identification, the parallel version of the Multiple EM for Motif Elicitation (MEME) tool has been configured and deployed.
- Information contained in the sequence has been calculated using position specific probability matrices (PSPMs) which are provided by SLiMs.
- Comparison between the performance and accuracy of the proposed models and other existing models.
- Visual and graphical analysis.
- Different programming techniques with implementation (using Python) to refine the PDB structure files.

In summary, I propose a new model which uses sequence information as distinguishing features. It calculates the information contained in sequence using a profile. In this way, it avoids the use of structural information. In many cases, the annotation of protein structure is not 100% accurate or complete yet. Thus, using such information may lead to results which are not accurate or reliable. Once the features are ready, I implement different machine learning and pattern recognition techniques to train the model which performs the classification. I also compare the results of the proposed model with other existing models with some post analysis.

1.7 Thesis Organization

This thesis has a total of eight chapters. Chapter 2 provides details of protein structures, PPIs and a brief survey on obligate and non-obligate PPIs. Chapter 3 describes differ-

ent feature extraction methods that can be used for prediction. It also includes different classifiers used for classification. Chapter 4 describes short, linear motifs (SLiMs), their identification methods and a short survey on SLiM discovery. Chapter 5 describes different SLiM identification tools briefly. It also focuses on the usefulness and configuration, parameter optimization of a SLiM identification tool - Multiple EM for Motif Elicitation (MEME). The proposed models are discussed in Chapter 6, while Chapter 7 describes the results. Finally, Chapter 8 draws the conclusions of the thesis along with the identification of some new problems and future work.

Chapter 2

Proteins and PPI Prediction

2.1 Proteins

Amino acids are the small organic molecules and the basic building blocks for polypeptides. Amino acids combine with each other and form a longer chain which is known as *polypeptide*. In the combination, amino acids bind to each other by means of peptide bonds (between carboxyl and amino groups). When these polypeptides fold into a specific three-dimensional structure and performs a specific biological function, they are called *proteins* [47]. Proteins were first named by Jakob Berzelius and described by Dutch chemist Gerhardus Johannes Mulder as an organic compound.

The most common partner of a folded polypeptide is another identical or different folded polypeptide. If the partners of a protein are identical to each other, that protein is known as *homo* protein complex. On the other hand, if the partners are not identical they are known as *hetero* protein complex [47].

2.1.1 Domains

Protein domains can be sequence or structural. Sequential domains are the sub-sequences of a chain that are present in different proteins, in different combinations. These sub-sequences must be biologically significant and functional. There should be a clear functional difference between a domain sequence and the corresponding sequence of raw amino acids. The functional level of a protein domain must be higher than the functional level of same raw amino acid sequence [46].

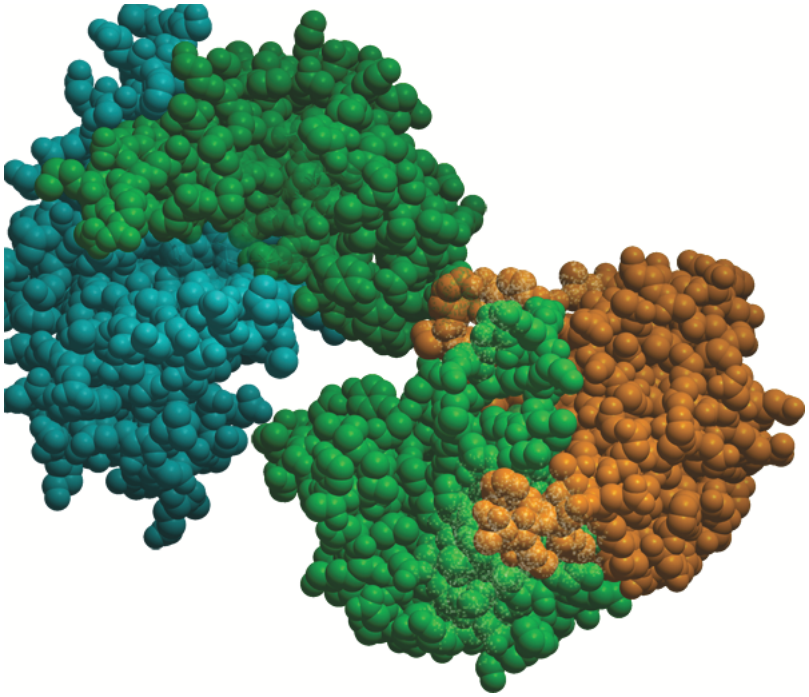


Figure 2.1: Domain α/β *C-terminal* found on *Cytoplasmic Malate Dehydrogenase* (4mdh A:B). The green space-filled area is the domain in Chains A and B. This figure was produced using ICM Browser [1] from the PDB [6] structure file of the complex.

Figure 2.1 shows the domain α/β *C-terminal*. The green space-filled area represents the domain found in Chain A (gold colored) of *Cytoplasmic Malate Dehydrogenase* (4mdh). On the other part, the green space-filled area represents the same domain found in Chain B

(light blue) of the same protein complex, 4mdh (A:B). This is a 149-residue long domain, and in both chains, it starts at position 4.

2.1.2 Motifs and SLiMs

A protein motif can be defined as a pattern of a set of protein chains. Motifs are widespread and have or might have biological significance. In proteomics, there are *sequence* motifs and *structural* motifs. Usually each motif contains a sequence pattern of 3-20 amino acids.

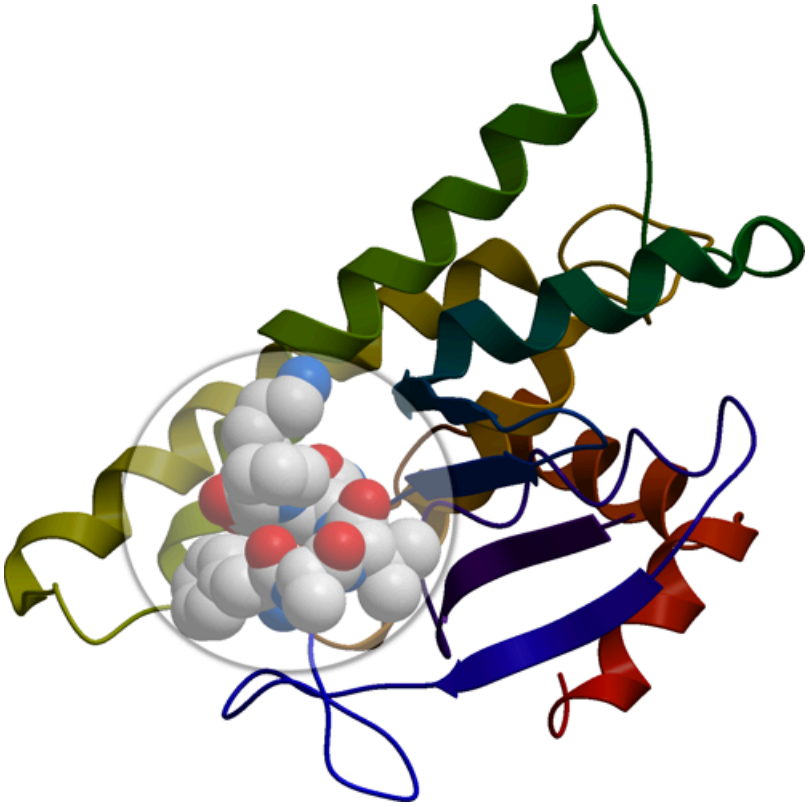


Figure 2.2: Motif (SLiM) found in *E.Coli glutathione S-transferase* (1a0f), Chain B at position 42. The SLiM was identified using MEME and the image was produced using ICM Browser [1] from the PDB [6] structure file of the complex.

Motifs of length of 2-10 amino acids are considered as *short, linear motifs* (SLiMs) or

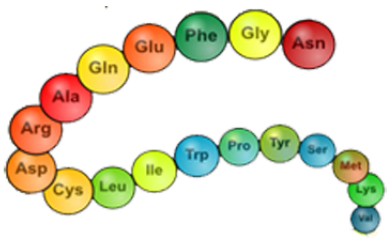
minimotifs [12]. There are some special properties that distinguish SLiMs from motifs. One of them is that SLiMs should have the capacity to encode a functional interaction interface in a short sequence, enrichment in intrinsically disordered regions of protein sequence. SLiMs should also be able to function independently of their tertiary structure context and have a tendency to evolve convergently [12]. Figure 2.2 shows a SLiM found in *E.Coli glutathione S-transferase* (1a0f), Chain B, at position 42. The space-filled part of the chain (inside the circle) is the SLiM and the regular expression is *F[AE][IEV]N[PT]K*. The SLiM was identified using MEME.

2.2 Protein Structures

Proteins or folded, biologically functional polypeptides are mainly composed of 20 different amino acids and have four different levels of structure [42]:

- Primary Structure
- Secondary Structure
- Tertiary Structure and
- Quaternary Structure

Figure 2.3a shows the the *primary structure* of a protein. These are simply the sequences of different amino acids which are similar to the polypeptide chain representations. When some parts of these polypeptide chains fold into a specific three dimensional structure, such as α-helix (Figure 2.3c) or β-sheet (Figure 2.3b), it is called *secondary structure*. Figure: 2.3d shows the secondary structure of a protein where different α-helices and β-sheets are



(a) Primary structure

(b) Secondary structure (β-sheet)

(c) Secondary structure (α-helix)

(d) Secondary structure (β-sheet and α-helix)

Figure 2.3: Primary and secondary structures of a protein.

bonded together in order to perform a biological task. These α-helices and β-sheets are held together by hydrogen bonds.

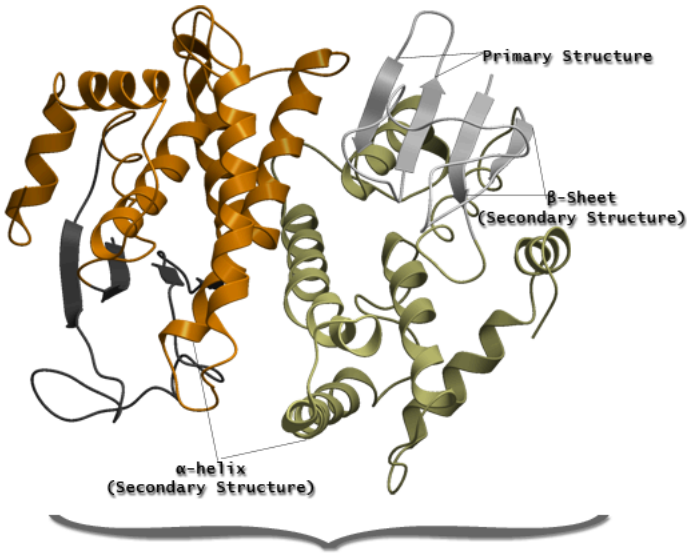


Figure 2.4: Tertiary structure of *E. Coli*. This figure was produced using ICM Browser [1] from the PDB [6] structure file of the complex 1a0f.

When primary structures combine with the secondary structure elements (α-helix, β-sheet and coils) and have a stable three dimensional shape, it is called the *tertiary structure* (Figure 2.4) of a protein. The tertiary structures of a protein complex interacting are called the *quaternary structure* (Figure 2.5). In this thesis, primary (sequence), tertiary and quaternary structures are used. The primary structure information is used to find the SLiMs, calculating the information contained in protein sequences.

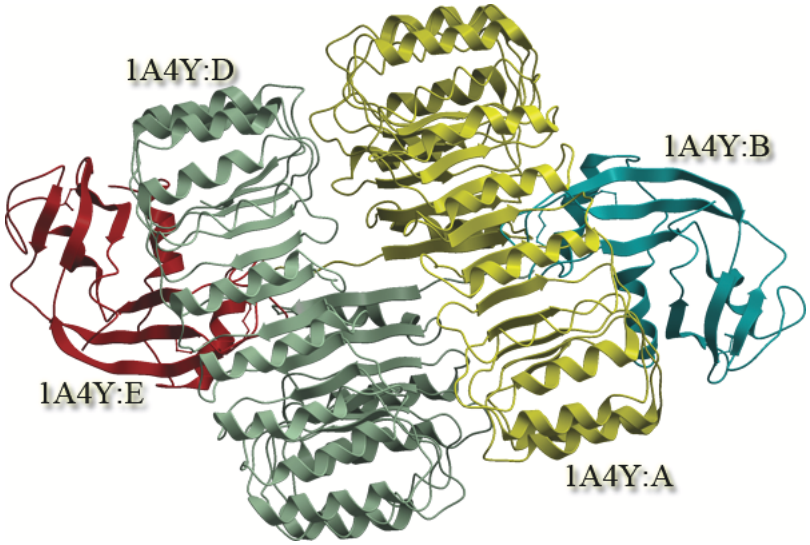


Figure 2.5: Quaternary structure of *human placental RNase inhibitor (hRI)* protein (1a4y). This figure was produced using ICM Browser [1] from the PDB [6] structure file of the complex.

The other structure information that has been used is to determine whether a SLiM is on the interface or not. If the SLiM is on the interface, is it interacting with other SLiMs on another chain or any other part of the chain? In case of the interaction, the desolvation and electrostatic energies are calculated from the structural information of the protein complexes and used for prediction.

2.3 Protein-protein Interactions (PPIs)

If two or more than two proteins or protein chains are close enough and meet some criterion, they bind with each other and may perform a new type of biological function. This process is known as *protein-protein interaction (PPI)* [39]. PPIs involve:

- **Direct Molecular Contact**: If the molecule (one or more than one) of amino acids

within a protein is close enough to the molecules (one or more than one) of the amino acids within another protein, these two proteins may interact with each other. According to the authors of [9], if the molecules from two chains are within 7Å distance, it is assumed that there will be a direct molecular contact and hence and interactions between the protein chains.

- **Long Ranged Interactions**: There is also a possibility of interactions even after if the molecules are not within 7Å distance as the environment around the molecules and also molecules themselves create an electro-magnetic forces which mediates the PPI processes. This type of interaction is known as *long ranged* interactions [31]. Long ranged interactions can take place up to 10Å distance between two molecules or even more.

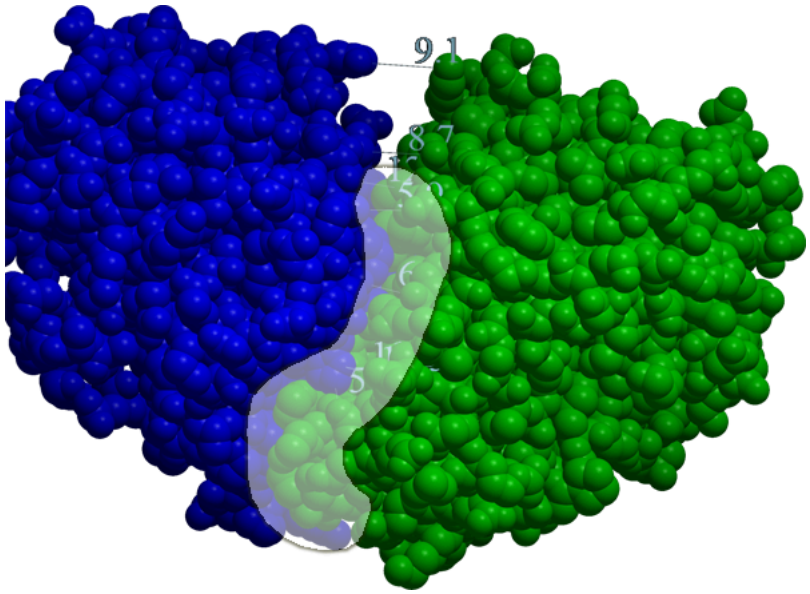


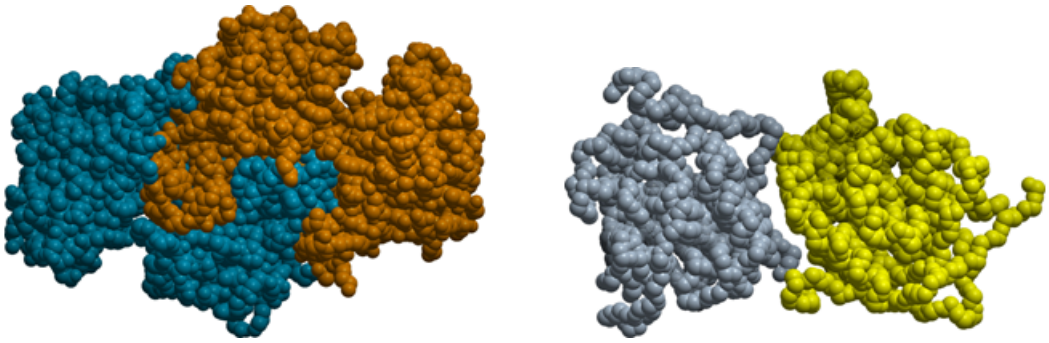
Figure 2.6: PPI between two protein chains of *Inositol Monophosphatase*, 2hmm (A:B). This figure was generated using ICM Browser [1] from the PDB [6] structure files of the complex.

Figure 2.6 shows PPIs between two chains of *Inositol Monophosphatase*, 2hhm (A:B). The molecules in the highlighted area are within a 7Å distance and they are in direct atomic contact. This region is also called the *interface* of the PPI.

PPIs are very important in many biological functions, including cell growth, nutrient uptake, motility, inter-cellular communication and apoptosis [25]. PPIs play a vital role in cell regulation, inter-cell communication, mutation, where cell regulation, inter-cell communication, mutation are very important in diseases and drug analysis, new drug discovery, among others.

2.3.1 Protein-protein Interaction Types

PPIs depend on the type of protein chains, between which the interactions take place. PPIs also depend on the family of the interacting proteins and their geometric structures. There are different types of PPIs. Based on the protein types, there are mainly three mutual nonexclusive types of PPI [39]:



(a) Obligate PPI in 2nac (A:B) (b) Non-obligate PPI in 2pcb (A:C)

Figure 2.7: Obligate and non-obligate PPIs. These figure were generated using ICM Browser [1] from the PDB [6] structure files of the complexes.

1. **Homo and hetero-oligomeric**: The interactions between two structurally symmetrical protein chains are called *homo-oligomeric* PPIs. On the other hand, if the in-

teracting protein chains are not structurally symmetrical, that interaction is called *hetero-oligomeric* PPI.

2. **Transient and permanent**: In terms of life time of a protein complex, there are transient and permanent PPIs. PPIs, which output relatively stable complexes after an interaction, are known as permanent PPIs. On the other hand, PPIs delivering relatively less stable complexes that last for a short period of time are known as *transient* PPIs.

3. **Non-obligate and obligate**: There are two types of PPIs based on the composition: obligate and non-obligate. Obligate interactions are usually permanent in nature and the structural units of obligate complexes do not exist as stable. Non-obligate interactions are less stable compared to the obligate complexes and in non-obligate complexes, the structures are more stable. The study of non-obligate and obligate PPIs are much more difficult as the life time of the non-obligate PPIs are usually very short and unstable in nature [30].

Figure 2.7a shows an obligate PPI, found between the chains of the *Holo and Apo formate Dehydrogenase* protein, 2nac (A:B). The figure shows that these types of interactions have larger interface area. A larger interface area indicates much more atomic contacts between the chains within a less distance range. In such situation, the probability of relatively strong and permanent interactions are very high. Figure 2.7b shows a non-obligate PPI, found between chains of the *Cytochrome C Peroxidase*, 2pcb (A:C). From the figure it is clear, that the interface area is smaller and hence less number of atomic contact between the chains, which will result in weak or less stable interactions. In addition, the surrounding environment of the complexes and presence of other molecules also mediate the mechanisms of PPIs.

2.3.2 Protein-protein Interaction Prediction

To predict PPIs, many researchers have been working on it from different perspectives. To predict PPIs, there are mainly two major approaches:

1. Biological experimental approaches
2. Computational approaches

In biological experimental approaches, the detection of PPI is limited to laboratory-based experimental techniques such as co-immunoprecipitation or affinity chromatography [7, 23, 24]. These types of experimental prediction are quite labor-intensive and the results of such approaches contain systematic errors as there is no backtracking mechanism. In addition, the amount of data for prediction is growing and incorporating all the available data, becoming almost impossible for the biological experimental approaches. That is one of the main reasons that computational models are in demand. In this thesis, to predict obligate and non-obligate I propose two computational models, which combine biological information and computational techniques for the prediction.

2.3.3 Protein-protein Interaction Prediction Strategy

Different classifiers such as the support vector machine (SVM), k-nearest neighbor (k-NN), linear dimensionality reduction (LDR), random forest (RF), decision trees, logistic regression can be used for predicting obligate and non-obligate PPIs. These classifiers use different properties of the protein complexes as features. Using desovation energy, amino acid composition, conservation scores, electrostatic energy, and hydrophobicity as features, the accuracy for the classifiers were about 70% [16, 21, 43]. In this thesis, different classifiers

such as SVM, *k*-NN, LDR and sequence information, desolvation and electrostatic energies are used as features.

According to the authors of [41], PPIs largely depend on the contact between residues, and the compositions of residues in protein complexes are unique. This unique composition yields unique sequence information, which can be used to distinguish between different types of interactions. Also, the authors of [8] showed a model which combines SVM and primary structure (sequence), achieving an accuracy of 80% in predicting PPIs. The authors of [44] also described another computational model that combines SVM and sequence information to predict PPIs along with PPI networks.

In this thesis, I calculate the sequence information in terms of sub-sequences. First of all, I have generated a database of protein sequences. Then I have identified SLiMs (discussed in Chapter 4 and 5) from that database of different lengths. I have used the position specific probability matrices (PSPMs) provided by the SLiMs as profiles (discussed in Chapter 3) to compute the sub-sequence information. On the other hand, I split the protein sequences into overlapping sub-sequences to match the length of the SLiMs. For example, if a sequence is 180 amino acid long, there will be a total of $\big((180-10)+1\big) = 171$ sub-sequences of length 10. Then, I use the PSPMs of the SLiMs of length 10 to calculate (discussed in Chapter 3) the information contained in that sub-sequence. For each protein, I take the top-twenty sub-sequence information for simplicity. I use sub-sequence information to fill the vector space for that protein and in this way, I generate feature vectors for classification.

Chapter 3

Feature Extraction and Classification

3.1 Features

Classification can be performed using different machine learning and pattern recognition techniques. But the main question is, what should be the input for those machine learning and pattern recognition techniques to perform the classification accurately? The properties or information of any object used to train a system are known as *features*. Not all the properties of an object are considered as features, but those which are distinguishable from object to object. In the field of machine learning and pattern recognition, one of the major challenges is to select and use the right features for successful classification or prediction. Usually, these are numerical values, but other types such as strings or graphs can also be used as features.

3.1.1 Features Used to Predict PPIs

There are many features which can be used to predict PPIs. The features are based on different properties of the protein complexes. Some of them are [40]:

β-factor : The flexibility of a protein complex at the time time of the interaction is known as the β-factor.

Solvent Accessibility : The exposed surface of a protein complex that affects contact of atoms during the interaction is known as *solvent accessibility* of that protein complex.

Geometric features : The shape index, planarity and curvedness of the interacting complexes are known as *geometric features* of that protein complex.

Evolutionary features : The conservation scores or sequence profiling information of a protein complex are known as *evolutionary features*.

Physicochemical features : This includes physiochemical properties such as hydrophobicity, electrostatic potential and desolvation energy.

In this thesis, sequence information as evolutionary features have been used to predict the PPI types (obligate and non-obligate). Sequence information as features have been proved to be very good in predicting protein structure and interacting sites within protein complexes [8, 41, 44, 50].

3.1.2 Desolvation Energy

Desolvation energy can be defined as the knowledge-based contact potential that accounts for hypdophobic interactions, self-energy change upon desolvation of charged and polar

atom groups and side-chain entropy loss [9]. The desolvation energy of an atom pair i and j of a complex can be approximated as follows [9]:

$$\Delta E_{des} = g(r) \times \sum\sum e_{ij} \tag{3.1}$$

where, e_{ij} is the atomic contact potential (ACP) [33, 52] between the i^{th} and j^{th} atoms and $g(r)$ is the smooth function score, based on their distance. For simplicity, I consider the smooth function to be linear for a distance within 5$\mathring{A}$-7$\mathring{A}$. I also consider the criteria that for a successful interaction, atoms should be within 7$\mathring{A}$ [9]. Within 5$\mathring{A}$-7$\mathring{A}$, the value of $g(r)$ varies from 1 to 0 which is equivalent to $100\% - 0\%$. According to the authors of [33, 52], desolvation energy for a particular atom-atom type pair is predetermined (approximated) which is calculated using the ACP matrix.

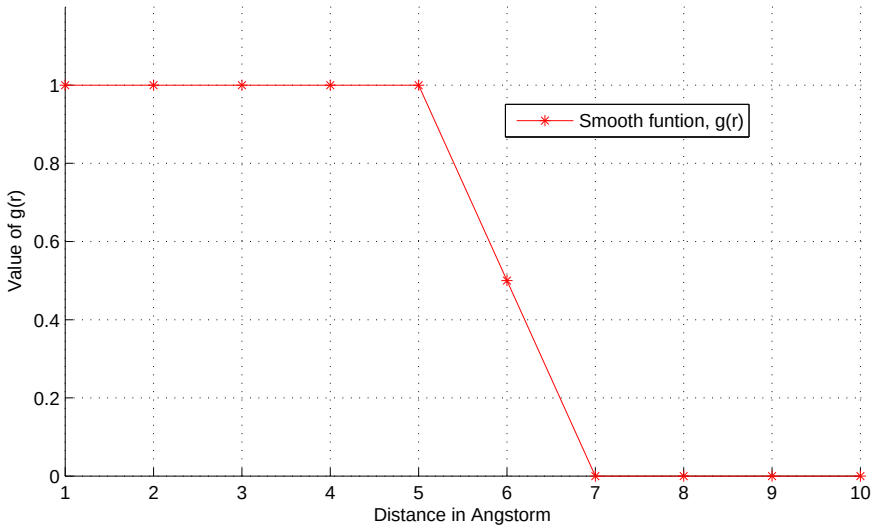


Figure 3.1: Plot of the smooth function, $g(r)$.

Thus, during the interaction the actual energy depends on the inter-atom distances. If the atoms are within 5$\mathring{A}$, then their actual energy will also be closer to that fixed value for

those particular atom pairs in the ACP matrix in [52]. If the distance is within 5$\mathring{A}$-7$\mathring{A}$, then I use Equation (3.2) to calculate the smooth function value:

$$g(r) = \frac{7-\delta}{2} \tag{3.2}$$

where δ is the distance between an atom pair in Angstroms and the value of the smooth function for that pair is $g(r)$. The behavior of the smooth function is shown in Figure 3.1.

3.1.3 Electrostatic Energy

Electrostatic energy is another important property in molecular biology as it has a high impact in biochemical reactions [5]. It also plays vital role in NMR, X-Ray and Cryo-Electron microscopy techniques. As a result, there is a steady increase in the size and number of biochemical and molecular complexes for which coordinates are available. The most widely used model to calculate the electrostatic energy is the Poisson-Boltzmann equation (PBE) [5], which is given by Equation (3.3):

$$-\bigtriangledown \cdot \varepsilon(x) \bigtriangledown \phi(x) + k^{-2} sinh\ \phi(x) = f(x) \tag{3.3}$$

Equation (3.3) is a second-order nonlinear elliptic partial differential equation that relates the electrostatic potential ϕ to the dielectric properties of the solute and solvent ε, the ionic strength of the solution and the accessibility of ions to the solute interior k^{-2}, and the distribution of solute atomic partial charges f. To expedite the solution of the equation, this nonlinear PBE is often approximated by the linearized PBE (LPBE) by assuming $sinh\ \phi(x) \approx \phi(x)$.

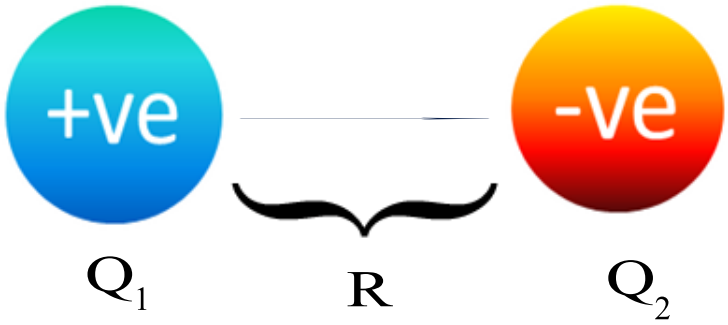


Figure 3.2: Two oppositely charged $(Q1, Q2)$ atoms within distance R.

Figure 3.2 shows two oppositely charged atoms within distance R. From Equation (3.3) a simple formula can be derived to calculate the electrostatic energy:

$$\left.\begin{aligned} \Delta E_{ele} &\propto \frac{Q_1 \times Q_2}{R} \\ \Rightarrow \Delta E_{ele} &= \frac{Q_1 \times Q_2}{4\pi\varepsilon \times R} \end{aligned}\right\} \tag{3.4}$$

Here, $4\pi\varepsilon$ is a constant, and so it is possible to calculate the electrostatic energy of an atom pair if the charges Q_1 and Q_2 are known. In this thesis, I have used electrostatic energy between two atoms as properties to evaluate long ranged PPIs. I have used APBS [5] and PDB2PQR [14] to calculate the electrostatic energies between two atoms from two chains of a complex. I consider all the atom pairs within 10Å distance.

3.2 Proposed Features

According to the authors of [8, 41, 44, 50], the information contained in the sequence has a significant impact in prediction of PPIs. There are different methods to calculate the information contained in a sequence [48]:

1. Symbol frequency scores

2. Symbol entropy scores

3. Stereochemical scores

In this thesis, I use an enhanced model of symbol entropy scores. I use position specific probability matrices (PSPMs) as background (profile) to calculate the information contained in the sites. The posterior probability of a site a of length ℓ, for a given profile X, can be defined as:

$$P(a \mid X) = \prod_{i=1}^{\ell} P(a_i) \tag{3.5}$$

As $P(a_i) \leq 1$, $\prod P(a_i)$ will be smaller for larger sites. Taking $-\log$ on both side of Equation (3.5):

$$\left.\begin{aligned} -\log\Big(P(a \mid X)\Big) &= -\left[\log\left(\prod_{i=1}^{\ell} P(a_i)\right)\right] \\ &= -\left[\sum_{i=1}^{\ell} \log\Big(P(a_i)\Big)\right] \end{aligned}\right\} \tag{3.6}$$

The information amount contained in the site is equivalent to the product of $P(a_i)$ and $\log P(a_i)$ [19, 35]. From Equation (3.6), the amount of information contained in a sequence a of length ℓ can be defined as:

$$\left.\begin{aligned} I(a \mid X) &= -\log\Big(P(a \mid X)\Big) \times P(a_i) \\ &= -\left[\sum_{i=1}^{\ell} P(a_i) \times \log(P(a_i))\right] \end{aligned}\right\} \tag{3.7}$$

Equation (3.7) implies that the larger sites is, the larger the information amount is. Thus, I divide the total information amount by the length of the site, ℓ. Finally, the information contained in a site a of length ℓ can be defined by Equation (3.8) as:

$$\hat{I}(a \mid X) = -\frac{1}{\ell} \times \sum_{i=1}^{\ell} P(a_i) \times \log(P(a_i)) \tag{3.8}$$

In this equation, X is the profile, $P(a_i)$ is the probability (of i^{th} residue of a) from the profile, and $I(a \mid X)$ represents the information contained by a. Again, it is known that $\log(1) = 0$.

$$\log(P(a_i)) = \begin{cases} \log(0.99) & \text{if } P(a_i) = 1 \\ \log(P(a_i)) & \text{otherwise} \end{cases} \tag{3.9}$$

Thus, for any $P(a_i) = 1$, a fraction (say 0.01) from $P(a_i)$ is subtracted as per the Equation (3.9).

3.3 Classifiers

Generally speaking, classification consists of determining the class of an unknown object (which class the object belongs to). There are mainly two different types of classification, supervised and unsupervised classification. In both types, classification is performed on the basis of the provided training instances. For supervised classification, the training set is provided with labels and in case of unsupervised classification, the training set is left without any labels. With the provided training set, the classifiers acquire the knowledge to perform the classification and can classify an unknown sample. As the training dataset of

an unsupervised classifier does not have label information, it attempts to find the inherent patterns in the data that can hopefully determine the correct class for the unknown sample. There are different types of classifiers available among others [20]:

1. Linear classifiers
2. Quadratic classifiers
3. Support vector machine
4. Nearest neighbor
5. Decision tree
6. Neural networks
7. Hidden Markov models

In this thesis, SVM, LDR and k-NN have been used to predict the type of complex; obligate or non-obligate.

3.3.1 Linear Dimensionality Reduction

Representing an object of dimension n as a lower-dimensional vector of dimension d is known as linear dimensionality reduction (LDR), and this is achieved by performing a linear transformation.

Suppose there are two classes, obligate or ω_1 and non-obligate or ω_2, represented by two normally distributed random vectors $\mathbf{x}_1 \sim N(\mathbf{m}_1, \mathbf{S}_1)$ and $\mathbf{x}_2 \sim N(\mathbf{m}_2, \mathbf{S}_2)$, respectively, with p_1 and p_2 the *a priori* probabilities. After LDR is applied, two new random vectors $\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1$ and $\mathbf{y}_2 = \mathbf{A}\mathbf{x}_2$, where $\mathbf{y}_1 \sim N(\mathbf{A}\mathbf{m}_1;\ \mathbf{A}\mathbf{S}_1\mathbf{A}^t)$ and $\mathbf{y}_2 \sim N(\mathbf{A}\mathbf{m}_2;\ \mathbf{A}\mathbf{S}_2\mathbf{A}^t)$ with $\mathbf{m}_i$

and $\mathbf{S}_i$ being the mean vectors and covariance matrices in the original space, respectively. The aim of LDR is to find a linear transformation matrix $\mathbf{A}$ in such a way that the new classes ($\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$) are as separable as possible. Let $\mathbf{S}_W = p_1\mathbf{S}_1 + p_2\mathbf{S}_2$ and $\mathbf{S}_E = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t$ be the within-class and between-class scatter matrices respectively. Various criteria have been proposed to measure the separability between the underlying classes [43]. I focus on the following three LDR criteria:

Fisher's discriminant analysis (FDA) [15, 22] : Its optimization criterion is as follows:

$$J_{FDA}(\mathbf{A}) = tr\left\{(\mathbf{A}\mathbf{S}_W\mathbf{A}^t)^{-1}(\mathbf{A}\mathbf{S}_E\mathbf{A}^t)\right\} \tag{3.10}$$

The matrix $\mathbf{A}$ is found by considering the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}_{FDA} = \mathbf{S}_W^{-1}\mathbf{S}_E$.

Heteroscedastic discriminant analysis (HDA) [17] : It aims to obtain the matrix $\mathbf{A}$ that maximizes the function:

$$\begin{aligned} J_{HDA}(\mathbf{A}) = tr\Big\{&(\mathbf{A}\mathbf{S}_W\mathbf{A}^t)^{-1}\big[\mathbf{A}\mathbf{S}_E\mathbf{A}^t \\ &-\mathbf{A}\mathbf{S}_W^{\frac{1}{2}}\tfrac{p_1\log(\mathbf{S}_W^{-\frac{1}{2}}\mathbf{S}_1\mathbf{S}_W^{-\frac{1}{2}})+p_2\log(\mathbf{S}_W^{-\frac{1}{2}}\mathbf{S}_2\mathbf{S}_W^{-\frac{1}{2}})}{p_1p_2}\mathbf{S}_W^{\frac{1}{2}}\mathbf{A}^t\big]\Big\} \end{aligned} \tag{3.11}$$

This criterion is maximized by obtaining the eigenvectors, corresponding to the largest eigenvalues, of the matrix:

$$\begin{aligned} \mathbf{S}_{HDA} = &\mathbf{S}_W^{-1} \\ &\left[\mathbf{S}_E - \mathbf{S}_W^{\frac{1}{2}}\tfrac{p_1\log(\mathbf{S}_W^{-\frac{1}{2}}\mathbf{S}_1\mathbf{S}_W^{-\frac{1}{2}})+p_2\log(\mathbf{S}_W^{-\frac{1}{2}}\mathbf{S}_2\mathbf{S}_W^{-\frac{1}{2}})}{p_1p_2}\mathbf{S}_W^{\frac{1}{2}}\right] \end{aligned} \tag{3.12}$$

Chernoff discriminant analysis (CDA) [43] : It aims to maximize the following function:

$$J_{CDA}(\mathbf{A}) = tr\{p_1 p_2 \mathbf{A}\mathbf{S}_E \mathbf{A}^t (\mathbf{A}\mathbf{S}_W \mathbf{A}^t)^{-1} + \log(\mathbf{A}\mathbf{S}_W \mathbf{A}^t) - p_1 \log(\mathbf{A}\mathbf{S}_1 \mathbf{A}^t) - p_2 \log(\mathbf{A}\mathbf{S}_2 \mathbf{A}^t)\} \tag{3.13}$$

In this thesis, I have used the gradient-based algorithm proposed in [43], which maximizes the function in an iterative way. For this gradient algorithm, a learning rate, α_k, needs to be computed. In order to ensure that the gradient algorithm converges, α_k is maximized by using the secant method. One of the keys in this algorithm is the random initialization of the matrix **A**, and in this work, different initializations were performed and then chosen the solution for **A** that gives the maximum Chernoff distance.

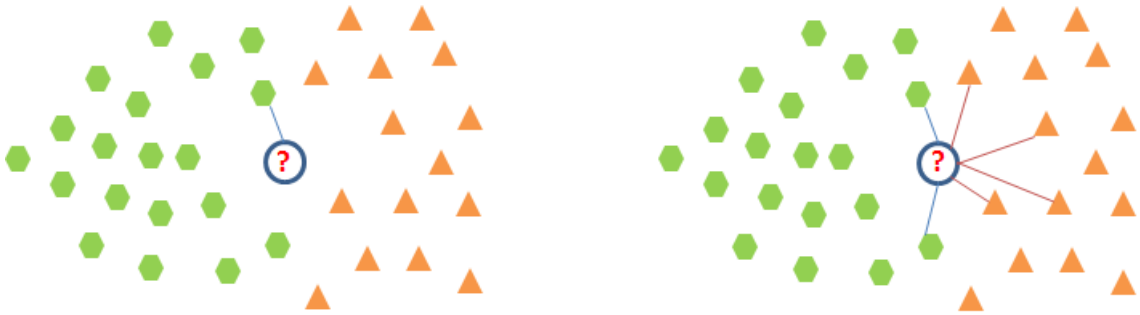
3.3.2 Support Vector Machine

I have used SVM, which is widely used in bioinformatics for classification. It takes the set of input vectors and predicts the possible classes of output based on the support vectors [28]. The kernel trick allows to map the original vectors onto a much higher dimensional space that hopefully makes class separation easier. I use linear kernel with default parameter setup which does not need any parameter optimization.

The way SVM works depends on the training data. Basically, there are two kinds of margins. The first one is called hard margin and the second one is called soft margin. In the hard margin SVM, the training data is linearly separable in the input space. On the other hand, if the training data are not linearly separable they can be mapped onto a higher dimension feature space to increase the separability.

3.3.3 *k*-Nearest Neighbor

For classification and data mining, *k*-nearest neighbor is one of the top 10 because of its simplicity. The *k*-NN method uses the well-known principle of *Cicero pares cum paribus facillime congregantur* (birds of a feather flock together) [36]. It classifies an unknown sample based on the known sample of its neighbor. It calculates the distance between the unknown sample and training samples. The distance with the smallest value corresponds to the sample in the training set closest to the unknown sample, and the unknown sample is classified as the class of that training sample.



(a) 1-NN Rule (b) 6-NN Rule

Figure 3.3: Schematic view of *k*-NN Classification for $k = 1$ and 6.

If only one instance from the training samples is used for the classification, the 1-NN rule is applied. Figure 3.3a shows a schematic view of the 1-NN rule. In the 1-NN rule, the unknown sample is compared (or distance calculated) only once. In the same way, if two instances are used, the 2-NN rule is applied. Figure 3.3b shows the schematic view of the 6-NN rule. In this 6-NN rule, the unknown sample is compared with the 6 training sample and the class of the largest number of matching sample is chosen.

In *k*-NN classification, two major aspects need to be taken care of, one being the distance function and the other being the value of *k* [36]. As *k*-NN considers that, the smaller

the distance is, the greater the likelihood the sample belongs to that class, this distance function should designed in such way that it is not biased. Another major consideration is the value of k, which represents the number of nearest neighbors for the unknown sample. If the value of k is too large, classes with the large number of classified samples can overwhelm smaller ones and the results could be biased. Again, if the value of k is too small, the advantage of using many samples in the training set is not exploited. Usually, the value of k is optimized by trials on the training and validation sets.

Algorithm 1 k-NN Algorithm

```
for all unknown samples, US_i do
    for all known samples, KS_j do
        ComputeDistance(US_i, KS_j)
    end for
    find the k smallest distances
    locate the corresponding samples KS_j1, ··· , KS_jk
    assign US_i to the class that appears more frequently
end for
```

The k-NN method does not generate an actual classifier using the training samples, but it rather uses the training samples every time it classifies an unknown sample. That is why it is called *lazy classification* method, which is much easier but computationally expensive. Algorithm 1 shows the pseudocode for the k-NN method. In the pseudocode, for each unknown sample, all possible distances between that sample and the known samples are calculated, which makes it computationally expensive. In this thesis, I use the k-NN method for the classification along with other classifiers, LDR and SVM. For the k-NN, I use the Eucledian distance and k= $1, 5, 10, 15, 20, 25, 30, 35$.

3.4 Prediction Evaluation

To validate a classification or prediction, the result needs to be validated. There are different methods available for validation. In this thesis, I have used cross validation, leave-one-out, and cross dataset validation methods to validate the results.

3.4.1 Cross Validation

Cross validation is a technique for assessing how the results of statistical analysis will generalize to independent datasets. It plays an important role on checking the prediction results. It divides the whole dataset into subsets (mainly two). One subset is called training set and the other is called test set or validation set. The most common approach used is the K-fold cross validation. Figure 3.4 explains 4-fold cross validation graphically. The whole dataset is divided into four parts, one part (orange) is used for test and the other three parts (blue) are used for training purposes. After each iteration, the test and one of the training datasets swaps and in this way it cover all the datasets.

The total values for TP, TN, FP and FN are calculated to compute the accuracy as stated in Equation (3.14). To compute the accuracy for each classifier, the following equation is used:

$$Accuracy = \left(\frac{TP + TN}{TP + FP + TN + FN} \right) \times 100\% \tag{3.14}$$

where,

- ***TP*** - when positive is classified as positive

- ***TN*** - when negative is classified as negative
- ***FP*** - when negative is classified as positive
- ***FN*** - when positive is classified as negative

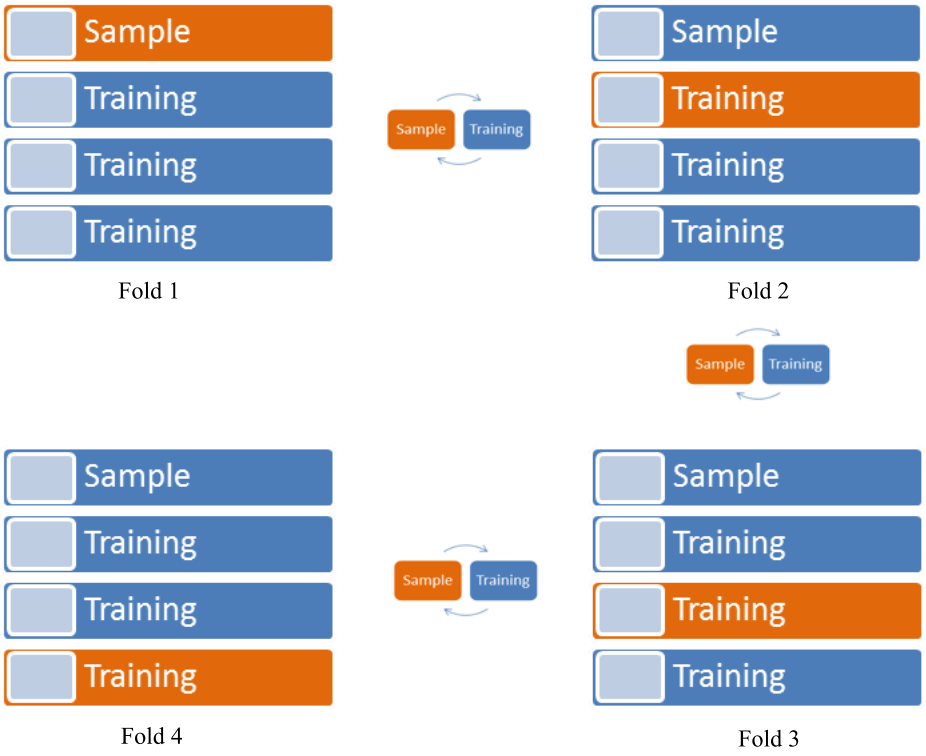


Figure 3.4: Schematic view of 4-fold cross validation.

TP is the number of correctly identified obligate complexes and *TN* is the number of correctly identified non-obligate complexes. *FP* and *FN* are number of incorrectly classified obligate and non-obligate complexes respectively.

3.4.2 Leave-One-Out

In cross validation, the dataset is divided into subsets (sample and training) in each iteration. For a dataset of size n and a validation of k folds, the iteration process runs for $\frac{n}{k}$ times. When $n = k$, the validation is called *leave-one-out* [16].

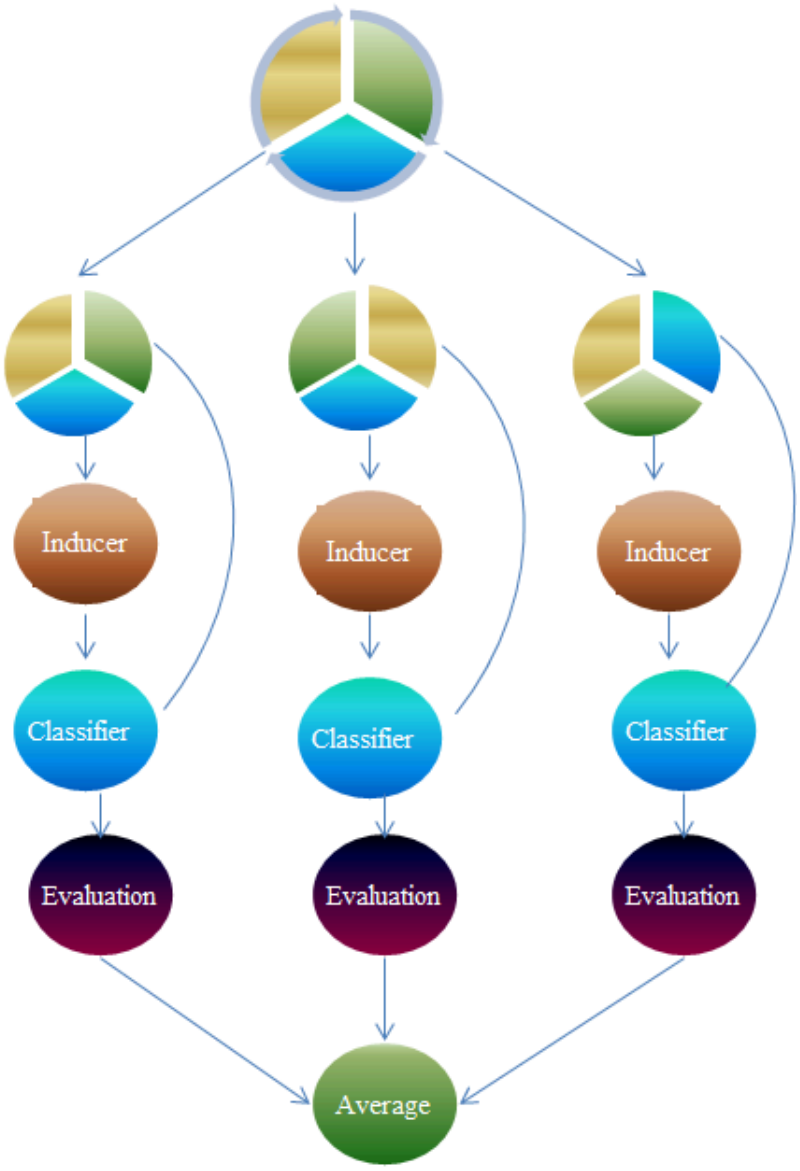


Figure 3.5: Schematic view of leave-one-out validation.

Figure 3.5 shows a schematic view of the leave-one-out method. In each iteration, one sample is kept out and all the others are used to train the classifiers to classify that sample which was kept out while training the classifier. As it is an iterative process, it covers all the samples. This is why leave-one-out is computationally expensive. In this thesis, I have used this method along with 10-fold cross validation.

3.4.3 Cross Dataset Validation

Using information from one dataset to train the classifier which will classify another dataset is known as *cross dataset validation*. A dataset *A* is used to train the classifier which will classify dataset *B* and dataset *B* is used for dataset *A*. In this thesis, I also implemented cross dataset validation. I have used two different datasets, one is the Zhu *et al.* [54] dataset and the other is the Mintseris *et al.* [34] dataset.

Chapter 4

Short, Linear Motif - SLiM

4.1 What is a SLiM?

A protein motif can be defined as a pattern of a set of protein chains. Motifs are widespread over a group of related protein sequences and have or might have biological significance. In proteomics, there are *sequence* motifs and *structural* motifs. Usually each motif contains a sequence pattern of 3-20 amino acids.

The motifs of 3-10 amino acids are considered as short, linear motifs (SLiMs) or mini-motifs [12]. There are some special properties which distinguish SLiMs from motifs. One of them is that SLiMs should have the capacity of encode a functional interaction in a short sequence, and enrichment in intrinsically disordered regions of protein sequence. SLiMs should also be able to function independently of their tertiary structure context and their tendency to evolve convergently [12]. Figure 4.1 shows a SLiM found in *E.Coli glutathione S-transferase* (1a0f), Chain B, at position 42. The space-filled area of the chain (inside the circle) is the SLiM. The SLiM was identified using MEME.

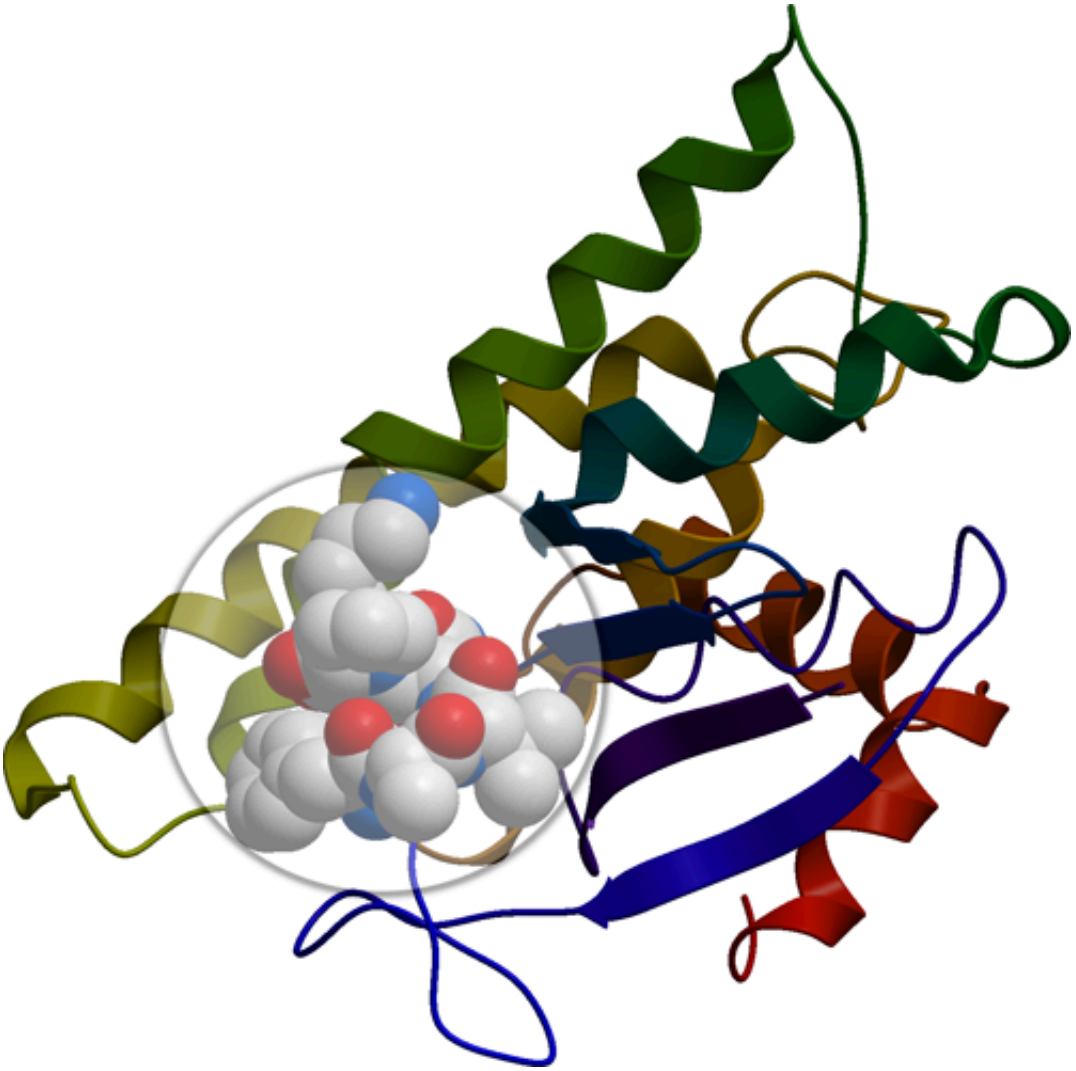


Figure 4.1: A short, linear motif found in *E.Coli glutathione S-transferase* (1a0f), Chain B at position 42. The regular expression of this SLiM is F[AE][IEV]N[PT]K. The SLiM was identified using MEME and the image was produced using ICM Browser [1] from the PDB [6] structure file of the complex.

SLiMs play a vital role in PPIs; they highly mediate the interactions between protein chains. Some recent studies show the importance of SLiMs when outsider viruses attack cell regulations which may cause different diseases such as cancer or tumors.

4.2 SLiMs on Interfaces

From the definition, it is known that SLiMs are shorter amino acid sequences which are present in protein complexes with specific biological properties. How can it be defined whether a SLiM is on the interface or not? In this thesis, I have considered a SLiM to be on the interface if at least one amino acid of a SLiM is on the interface. This answer leads to another question; how it is determined if an amino acid is on the interface? If at least one atom of the amino acid is on the interface, I have considered that amino acid to be on the interface.

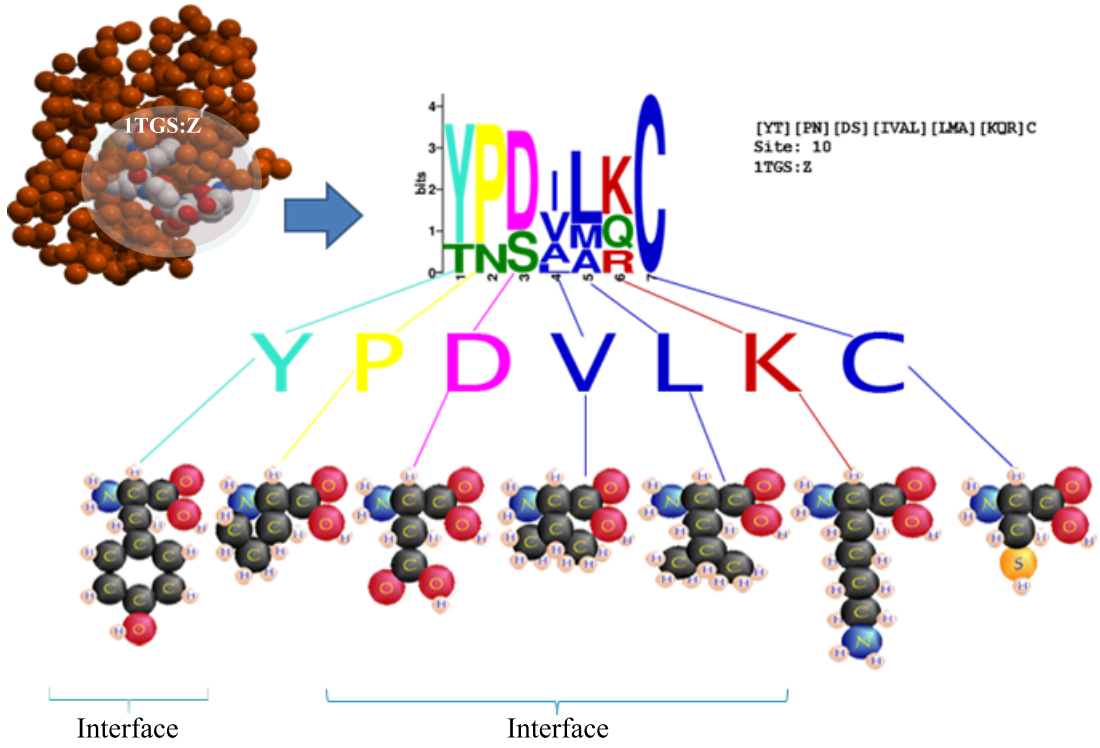


Figure 4.2: A SLiM found on the interface of 1tgs (Chain Z).

Figure 4.2 shows a SLiM which is on the interface. This SLiM was found in Chain Z

of *Pancreatic Secretory Trypsin Inhibitor* (Kazal type), PDB identifier 1tgs. The regular expression of the SLiM is [YT][PN][DS][IVAL][LMA][KQR]C. This SLiM is 7 amino acids long, four of them (Y, D, V and L) being on the interface. Thus, I consider this SLiM to be on the interface. In this thesis, I have used one amino acid as threshold to check if a SLiM is on the interface and one atom threshold to check if an amino acid is on the interface. This threshold can be changed for further experiments.

4.3 SLiM Representation

SLiMs or motifs can be represented in different ways. The most used ways to represent a SLiM are:

- Regular expression
- Sequence logo
- Position specific probability matrix (PSPM)
- Block representation

A regular expression is the simplest way, while the sequence logo is the graphical representation of a SLiM.

4.3.1 Regular Expression

A regular expression is the simplest and widely used mean to represent a SLiM. It is the consecutive appearance of the amio acid letters of the SLiM. The occurrence of multiple letters for a single position is shown inside square brackets [35]. For example, the regular

expression of one SLiM found in *E.Coli glutathione S-transferase* (1a0f), Chain B, at position 42 is **F [AE] [IEV]N[PT]K**. This SLiM is six amino acids long. The first position of the SLiM is filled by **F**, the second position is filled by either **A**or **E** and similarly for the other positions. The regular expression however, does not provide the significance, frequency, information contained in a letter of a position. It simply shows the occurrence of the letters for that position.

4.3.2 Sequence Logo

A sequence logo is a graphical representation of a motif. It shows how well are amino acids conserved at each position; the higher the number of amino acids, the higher the letter will be as the conservation at that position is higher. Different amino acids at the same position are scaled according to the information content of that amio acid [35].

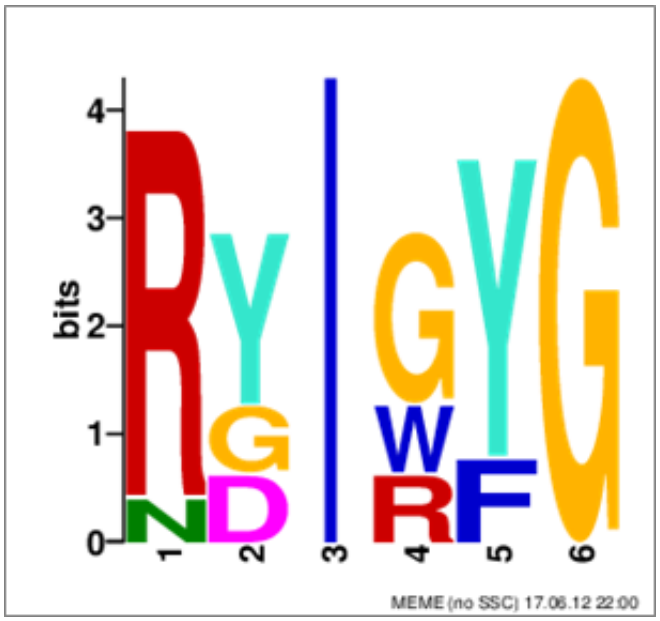


Figure 4.3: SLiM found in both chains of *Cytoplasmic Malate Dehydrogenase* (4mdh) at position 233. This sequence logo was generated using MEME.

Figure 4.3 shows the sequence logo for the SLiM found on the chains of *Cytoplasmic Malate Dehydrogenase* (4mdh) at position 233. The regular expression of the SLiM is **[RN][YDG]I[GRW][YF]G**.

The x-axis of the sequence logo represents the positions in the SLiM and the y-axis represents the information contained in a specific position. The information content of y-axis is given by Equation (4.1):

$$R_i = \log_2(20) - (H_i + e_n) \tag{4.1}$$

where H_i is the uncertainty of a position i and H_i is defined as:

$$H_i = -\sum f_{a,i} \times \log_2(f_{a,i}) \tag{4.2}$$

Here, $f_{a,i}$ is the relative frequency of amino acid a at position i, e_n is the small-sample correction for an alignment of n letters. The height of letter a in column i is given by Equation (4.3):

$$e_n = \frac{(s-1)}{2 \times ln(2) \times n} \tag{4.3}$$

For proteins, the value of s is 20 and n is the length of the SLiM.

4.3.3 Position Specific Probability Matrix

A position specific probability matrix (PSPM) is the matrix $\{m_{i,j}\}$ representation of a SLiM where the value of each element represents the probability of the j^{th} letter occuring at the i^{th} position. Each column in the matrix $\{m_{i,j}\}$ represents each of the 20 amino acids and the rows represent each potion of SLiM. Figure 4.4 shows the PSPM of a SLiM found in

E.Coli glutathione S-transferase (1a0f), Chain A, at position 310:

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
1st	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2nd	0	0	.25	0	0	0	0	0	.25	0	0	0	0	0	0	.5	0	0	0	0
3rd	0	0	0	0	0	0	0	0	0	.5	0	0	0	.25	0	.25	0	0	0	0
4th	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5th	.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.75	0	0	0	0
6th	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7th	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 4.4: PSPM representation of a SLiM, found in *E.Coli glutathione S-transferase* (1a0f), Chain A, at position 310. This figure was generated using MEME.

The regular expression of the SLiM is **C[SDK][LQS]C[SA]CT**. The value of the second column of the first row of the Matrix is 1.0, which means that the first position of the SLiM will be filled by C and its probability is 1.0. The value of the third column of the second row of the PSPM is 0.25, which means that the second position of the SLiM will be filled by D and its probability is 0.25, the probability of filling that position with K is 0.25 and this probability is 0.5.

To fill the third position, the probability of L is 0.5 and the probability of Q and S is 0.25. In this way, the PSPM represents the distribution information of the letters for a SLiM. This distribution values also represents the information contained in each position. The higher the distribution value, the higher the information content [35].

4.3.4 Block Representation

One SLiM might appear in multiple sites (on the same *chain* or different chains). The block representation shows the presence and positions of a SLiM in different chains. The block representation of the SLiMs is also very important to have an idea about the presence of a SLiM in different complex chains. Figure 4.5 shows the SLiM [LF]MQ[RS][MT][VA]E in different chains in different complexes:

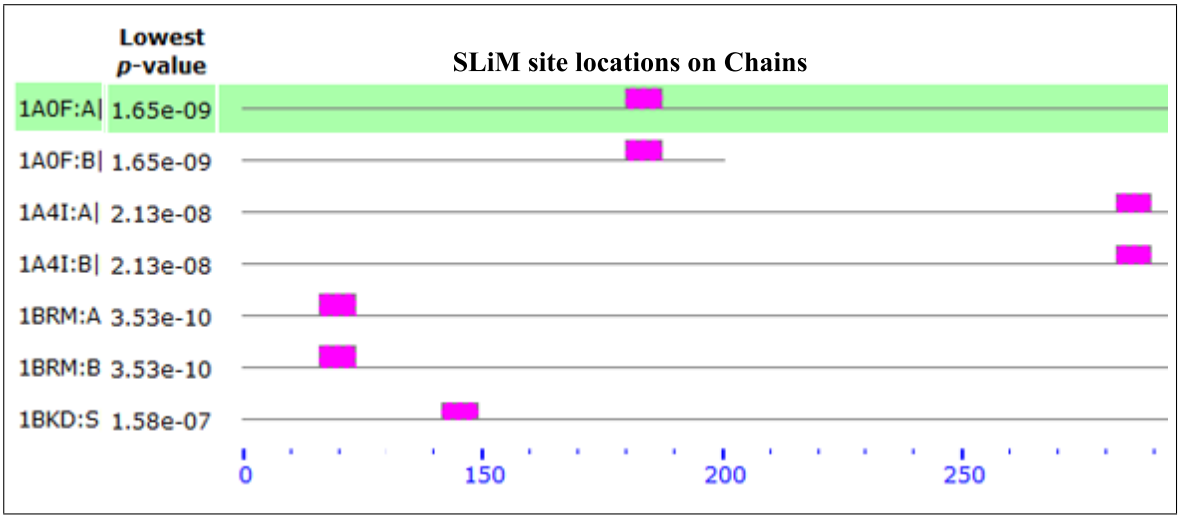


Figure 4.5: SLiM found in *E.Coli glutathione S-transferase* (1a0f), Chain A, B at position 180. This same SLiM was also found in 5 more different chains. This figure was generated using MEME.

Besides being found in 1a0f, this SLiM was found in both chains (A and B) of complex 1brm at position 16, in Chains A and B of complex 1a4i, at position 282 and in Chain S of complex 1bkd at position 142. The block diagram does provide information about the SLiMs, such as the occurrences of the SLiM in different sites.

4.4 Types of SLiMs

There are mainly three types of SLiMs and this categorization is based on their different properties. The types of SLiMs are:

- Sequence SLiMs
- Structural SLiMs
- Functional SLiMs

Sequence SLiM can be defined as an amino acid sequence pattern which is wide spread over a data set, might have biological significance and can perform some function [10]. A sequence SLiM is the simplest form of a motif. For a specific site, all the amino acids of a sequence motif will come from a single chain of a complex, while for other chains or complexes, the amino acid may vary a little. Figure 4.6 shows a sequence SLiM which was found in β-*TRYPSIN* (2ptc:E) starting from the 50^{th} amino acid up to 59^{th} amino acid covering all the consecutive amino acids.

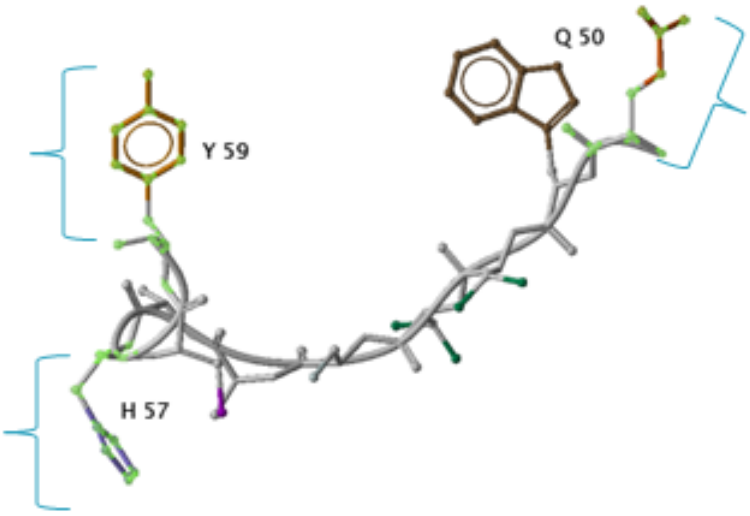


Figure 4.6: A sequence SLiM found in β-*TRYPSIN* (2ptc), Chain E and B at position 50. The SLiM was identified using MEME and the image was produced using ICM Browser [1] from the PDB [6] structure file of the complex.

On the other hand, a structural motif can be defined as a combination of amino acids from different parts of a chain or different chains, but closer enough to form a stable structure. This type of motifs are biologically more important as the structure is conserved and can perform a specific task being or not as a part of the protein complex. Figure 4.7 shows the formation of a structural motif from different parts of a single chain.

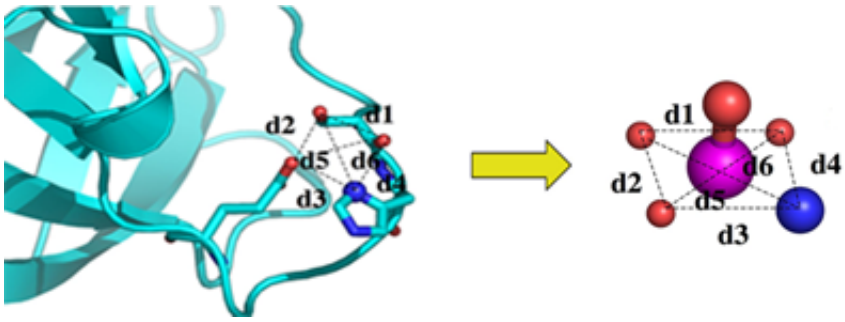


Figure 4.7: A structural SLiM formed from different part of the chains.

There is another type of SLiM, known as functional motif, which does not have either sequence similarity or structural similarity, but the sites perform the same biological task and hence they are biologically relevant.

4.5 SLiM Discovery

In 1970 Needleman and Wunsch and in 1981 Smith and Waterman introduced the Needleman-Wunsch algorithm and the Smith-Waterman algorithm respectively. These algorithms were capable only for alignment and could not detect the pattern regions from the sequences which reflect the structure and function of a protein. Also these algorithms could not deal with distantly coupled sequences. At that time pairwise alignment and visual observation was the only way to detect motifs or domains from a protein sequence which is laborious and expensive.

Neuwald *et al.* [38] proposed the Gibbs sampling algorithm that detects motif-encoding regions from protein sequences and classifies them. This strategy tries to solve the motif detection problem assuming that the motif occurrence number is provided as prior information. With the prior information, this strategy samples sites for each motif iteratively until it finds the optimal partitions of motif-encoding regions into different motifs which can also be used to classify the related motifs. Another Gibbs sampling strategy, called *column sampling*, is applied within the motif sampling algorithm for the motif length optimization.

The performance of the Gibbs sampling method compared to all other existing methods is high enough. Some methods using the Hidden Markov Model (HMM) for multiple sequence alignment can produce results closer to the Gibbs sampling. But the HMMs are EM-based, and hence suffer from the problem of becoming stuck in local optima. Also the model to deal with gaps can deal only with the closely related sequences, where on the other

hand, the Gibbs sampling method does not have all these issues even it can detect subtle block-based motifs. This strategy works fine for related protein sequences even though it does not work that much accurately for unrelated protein sequences.

Tan *et al.* [45] described a new approach with two new algorithms for finding correlated, short motifs. These types of motifs are important in PPIs and also in drug discovery, analyzing such motifs is very important. Finding such binding motifs is not easy at all by traditional biological experiments as it becomes laborious and expensive. The authors propose an *(l, d)*-motif pair finding model which is a combination of the *(l, d)*-motif model, *D-STAR* and *S-STAR* algorithms, and looks for motif pairs from interacting protein sequences, one motif from one interacting protein partner and the other motif from the other interacting protein partner. This model is based on the *(l, d)*-motif model and the *D-MOTIF* and *D-STAR* algorithms where the *D-STAR* is faster enough than the *D-MOTIF* for a larger dataset. Processing data for the algorithm is crucial, as these algorithms work on interacting data only. Once the interacting data is ready the parameters l and d need to be tuned. After successfully running the algorithms, a validation step is performed on the output.

Edwards *et al.* [18] described a probabilistic method to identify over-represented, convergently evolved, SLiMs called SLiMFinder. Typically, SLiMs consists of 3-10 residues, from the primary protein structure. From a study it has been found that 15%-40% of the interactions are mediated by SLiMs beside the large domain-domain interface. As SLiMs are short and degenerate in nature, it is hard to find new SLiMs. All the existing methods incorporate a model of convergent evolution to identify new SLiMs.

The authors introduced a novel algorithm, called *SLiMBuild*. Initially, *SLiMBuild* defines the motifs by combining residues into longer sequences before efficiently incorporating amino acid degeneracy and/or variable length wild-cards by adding variants that occur

in the desired number of unrelated proteins, and increase the total number of unrelated proteins in which the ambiguous motif occurs.

As the short motifs are expected to occur in multiple unrelated protein sequences, identifying the recurring motifs becomes the major part of the challenge. This algorithm also considers a score that indicates how likely a given motif is compared to other motifs. When the *SLiMBuild* algorithm ends, the *SLiMChance* algorithm is started to improve the scores by making a crude but efficient adjustment of motif probabilities.

The authors claim that this method was implemented on 750 random datasets and the false-positive rates for this method were found very similar at any given significant threshold for any random dataset. The differences between different types of random data are minimal. The most significant motifs returned by the *SLiMFinder* from each dataset are not strongly dependent on the size although *SLiMFinder* is relatively biased.

Narlikar *et al.* [37] described a way to use the nucleosome occupancy information to discover the transcription factor bindings or motifs. Study and identification of transcription factor bindings or moitfs are required in a large scale to understand the transcriptional regulatory process in living cells. The motifs are short, degenerate sequences which occur frequently by chance and this is the main reason that finding motifs is still difficult. Noise generated by the non-functional sites over the functional sites make this process even more complicated.

The authors use the Gibbs sampling method with nucleosome occupancy information as prior information and position-specific scoring matrix (PSSM) as a motif model. To find the motifs, they adopted a sequence model and objective function. Having a sequence model and objective function, a strategy to optimize the model parameter and scoring scheme is selected. Gibbs sampling is in the heart of the model, with prior information. Here the

prior information is based on the nucleosome occupancy from both bound and unbound sequences. They also use another type of prior called *discriminative prior* with the Gibbs sampling model. Using all the prior information, they derive a prior set which is called positional prior which is used to find the starting point of a motif in a sequence. If no prior is provided, it will assume all the positions having same probability as a starting point.

The authors claim that the method is implemented using four different prior models PRI-U, PRI-N, PRI-D and PRI-DN, where PRI-DN is the combination of the priori from nucleosome occupancy and discriminative prior. In an experiment, a set of 156 sequences is used. While PRI-U finds 46 correct motifs, PRI-DN finds 69 correct motifs which implies an improvement of 50%. The nucleosome occupancy controls the binding activity and plays an important role in finding the motifs. The authors also claim that they are the first in introducing this feature for finding motifs with high accuracy.

The methods/algorithms/approaches discussed here are the most well-known for motif or SLiM discovery. Only these methods proposed new, unique algorithms and all other methods are any how fully or partially based on one of these methods. For example, we can state that BLAST and Pfam are based on multiple sequence alignment and the Hidden Markov model (HMM) respectively. So far, the Gibbs sampling algorithm still achieves the better performance than any other algorithm or method.

Chapter 5

SLiM Finding Tools

5.1 Introduction

In proteomics, to understand the PPIs, the structure and function of a protein the study of pairwise alignment, local/global alignment, multiple alignment, SLiMs, domains are very important as they play a vital role on these. A lot of research has been conducted for more than two decades on the protein sequences, motifs and domains, and as a result there are many online databases and tools to facilitate the protein sequence research. These online tools allow the user to find motifs or domains for a specific dataset although motif/domain identification uses the same kind of knowledge-base and sometimes identifications are highly biased. Beside identification, searching motifs/domains from a specific data set is also provided by these facilities. These tools and databases are mainly web-based (installation versions are also available) and open for all researchers.

Gutman *et al.* [26] described QuasiMotiFinder which is a new, complementary method for motif search against the original database. In this method, the original query protein is replaced by multiple aligned, homologous proteins. Highly conserved signatures are con-

sidered to indicate the correct protein function. The degree of evolutionary conservation of the signature within the protein family is estimated, and this estimation is used to calculate the likelihood of the motif.

According to the authors, the input for QuasiMotiFinder should be multiply aligned homologous query proteins in MSA format. If FASTA format is provided instead of the MSA, QuasiMotiFinder uses CLUSTALW to find the MSA file for that query protein sequence. For each position of the MSA, evolutionary conservation scores are calculated using the likelihood-based algorithm Rate4Site. A negative score indicates high conservation and a positive score indicates a variable residue. A physiochemical similarity using the physiochemical similarity matrix with the PROSITE signatures is performed to assign the scores. After the total score, the statistical significance of a putative signature is estimated based on its pre-calculated distribution. 181 proteins sequences were used for the testing and 65 were found in the true-positive subgroup, 65 in the false-positive sub-group and 51 in the false-negative subgroup using both strict and quasi-PROSITE. Some random experiments with some sequences (i.e. FURIN BOVIN) was performed to make a comparison between the QuasiMotiFinder and PROSITE. It was always found that QuasiMotiFinder detects at least one more motif than PROSITE.

Bailey *et al.* [4] described the online tool, Multiple EM for Motif Elicitation (MEME) which can identify motifs without prior knowledge or database. As motifs are shorter in nature, it is not easy to find a motif from a given data set. That is why motif discovery can be compared with finding a needle in a haystack problem. On the other hand, comparatively longer motifs are easy to find than shorter motifs. Again, finding TFBS motifs from DNA sequences is quite difficult, which have a tendency of having binding sites to be short and degenerate. Another issue in this case is that, it is often difficult to identify precisely the

promoter regions. This problem becomes more severe in Eukaryotes than in Prokaryotes and yeast as TFBS in eukaryotes tend to become shorter and variable.

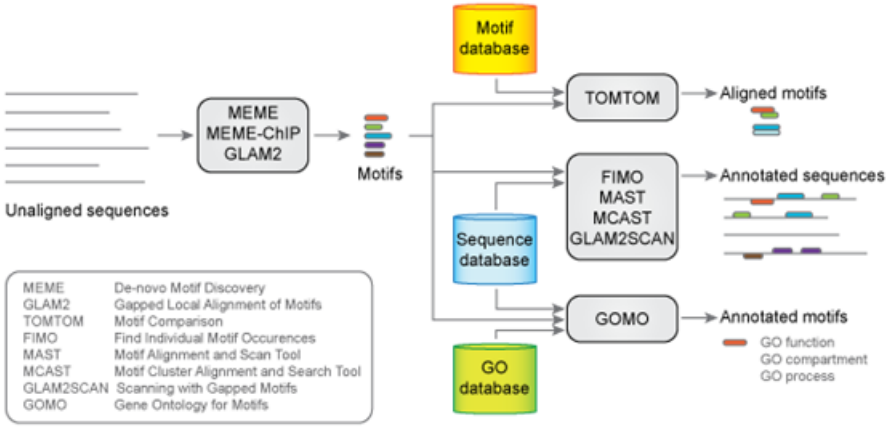


Figure 5.1: Schematic diagram of the MEME software suite. The diagram was taken from www.meme.sdsc.edu/meme/intro.html [4].

According to the authors, the user need to input a set of protein or DNA sequences in FASTA format in order to find motifs using MEME. The user can simply copy-paste the sequence set or upload a sequence file. For the online version, the user needs to provide a valid email address. In the online version, there are a few parameters, which the user can leave as default or can set the parameters. Among the parameters are: width of the motif, number of motifs, number of sites, and model (ZOOPS, OOPS, ANR) being used by MEME are available. Although it is easy to use, it has a limitation on the size of the input set of sequences. It can not take an input set of more than 60,000 characters.

MEME delivers an output in three different formats, HTML, TEXT and XML, where HTML represents the motif information in the most interactive way. It shows the local multiple alignments along with block diagrams. The block diagram shows a schematic view of the motifs in a sequence. This HTML file also shows the sequence logo representation

of a motif along with PSPM information. Besides, there are also some buttons available to submit the motif information for further processing, such as the user can submit the motif information to the MAST web server to search the motif into different sequence server. On the other hand, TEXT and XML versions are mostly used for further analysis and information processing. MEME's developers plan to add a facility to upload a background Markov model to increase the sensitivity of MEME searches. Also they plan to add an algorithm for removing low-complexity regions and repeated elements. Another option called MCAST will also be introduced.

Davey *et al.* [11] described the *SLiMSearch* tool for searching minimotifs for a given protein sequence set. Tools such as ELM, MnM, SIRW, ScanProsite and QuasiMotifFinder are well-known for motif search, but generally they search the motifs only from databases of known motif patterns provided by the user although they have some special features. The question is about to predict genuinely novel motifs, so that knowledge of existing motifs is less useful. Instead of using knowledge about existing motifs, using the contextual information about the input data sequences helps more in finding new, novel motifs. *SLiMSearch* receives the motif information from the powerful novel motif discovery tool, *SLiMFinder*, along with the same contextual information and enables the results to be masked or ranked based on the biological indicators. These indicate the sequence conservation and structural disorder and features which are as same as *SLiMChance* for assessing statistical over representation of SLiM occurrences, correcting for biases introduced by evolutionary relationships within the data. *SLiMSearch* was also optimized for small protein data sets such as *SLiMFinder*.

The total *SLiMSearch* method consists of three major parts, pre-formatted database, scoring and enrichment of the scores. Pre-computed databases for each proteome are used

to speed up the motif attribute calculations. There are several pre-formatted databases such as domain data from Pfam, structural data from PDB, experimentally validated data from the ELM database and SNP, and modification data from UniProt. *SLiMSearch* extends the RLC score to return a probability score, P(RLC) using the Gaussian cumulative distribution function. This P(RLC) is then converted to *p*-value as the final score for a motif. In the last enrichment score step, the scores are calculated against the reverse of the motifs and a randomly shuffled variant of the motif. The score is the ratio of the input motif count and reverse motif count.

The authors claim that for using *SLiMSearch*, a motif in a regular expression form is compulsory. The response time and complexity depending on the motif complexity and current load of the server. The output is delivered in a tabular format with annotated attributes such as conservation and disorder statistics, overlapping features, and overlapping experimentally validated motifs. Currently, *SLiMSearch* includes all the human proteome. At present, thousands of users from different countries are using this online tool for searching *de novo* motifs.

Dinkel *et al.* [13] described a well formatted database of 1,800 annotated eukaryotic linear motifs and a SLiM search mechanism for a set of input sequences. Although there some other resources for SLiMs such as ProSite, MiniMotifMiner, ScanSite, none of them has the base of manually annotated motifs while all of them are generalized for all types of SLiMs. That is why there is a need of a SLiM database which would be specialized on a certain type of SLiMs along with the search facility. Based on this, ELM started with manually annotated Eukaryotic linear motifs.

ELM is designed in a way that it consists of two major applications, one is the linear motif database and the other is the motif detection pipeline to detect putative motifs from

a query sequence. The main idea is to annotate one SLiM with its class and instance. Besides, it provides the user an interactive way to visualize the motifs, its location on the main sequence graphically, and user friendly representation of other information. The current version of the ELM resource contains 170 classes which is about 43 times more than the initial version and 1,800 instances and all the instances are linked to more than 1,500 annotations. Regular expressions are used to represent the classes and also for searching motifs in an input sequence set. When a query is performed, the output page can be divided into two parts: the upper part which contains the colors/symbols used to explain different issues, and the lower part which contains the annotated and putative ELM instances for the input sequences.

The authors also claim that the SLiMs are very important in virus-host interactions, which make the ELM resource a highly important tool for the researcher who deals with virus studies. A study showed that phosphatase 1 (PP1) docking motif in protein 7 of a transmissible gastroenteritis virus motif mediates binding to the PP1 catalytic subunit [13], a key regulator of the cellular antiviral defense mechanisms. This is also found in other viral proteomes, suggesting that it might be a recurring strategy to counteract the host defense against RNA viruses. A special focus has been attributed to the annotation of viral instances in the ELM database to increase the awareness of viral motifs.

Mi *et al.* [32] described the online tool Mini Motif Miner (MnM) and the mini motif database. There are a few online tools for finding mini motifs from protein sequences which include MEME, Eukaryotic Linear Motif (ELM), and Phospho-ELM. MEME finds the motif in an unsupervised way, and for this reason the accuracy of the motifs found by MEME is not that high. On the other hand, ELM focuses only on special types of motifs. Previous versions of the MnM did not have any filtering method to improve the accuracy

and also the motif database was not sufficiently large.

According to the authors, the current version of MnM has improved filters, which increase the accuracy of mini motif prediction. The filter removes the false positives for the prediction. They have also increased the size of the mini motif database by both manual annotation from different literature and by cooperation with other known databases such as PhosphoSite, DOMINO, MEROPS, UniProt, PepX, 3DID, PeptiDB and HPRD. The MnM database now includes a total of 294 933 mini motif definitions, consisting of 880 consensus mini motifs and 294 053 instances. All these mini motifs play a vital role in three major biological activity: trafficking, binding, and modifying.

The authors claim that the current version of MnM has 640 times more minimotifs than the first ever version and about 60 times more than the previous version. The current version contains 30 times more binding sites than the first version, where it has 7 times more trafficking information. In the previous version of the MnM model, each minimotif was described using only 22 attributes, while the current version uses 28 attributes. This motif describing model includes a protein sequence definition, describing the chemistry of the motif and a functional definition. All these modifications made in the newer version include an extensible expanded definition of the covalent chemistry of the protein that contains that minimotif.

5.2 Why MEME?

There are few SLiM finding tools available to find SLiMs such as SLiMFinder, SLiMDisc, SLiMSearch, QuasiMotiFinder, MnM and MEME. But none of them finds SLiMs in a unsupervised fashion. Most of them use predefined knowledge (SLiM/Motif database) to find SLiMs which are not of interest. As I wanted to find SLiMs in a specified data set,

with predefined parameters, in an unsupervised fashion, MEME resulted in the best option. Only MEME allows controlling the parameters and also assures motif identification in an unsupervised manner. The MEME suite currently offers the following facilities [4]:

- Motif Discovery: MEME, DREME (DNA only), GLAM2
- Sequence Search with Motif: FIMO, MCAST, GLAM2SCAN
- Compare a motif to all motifs in a database
- Analyze motif enrichment using SpaMo, CentriMo

Besides, MEME is based on the MM algorithm which incorporates the expectation maximization (EM) technique for the finite mixture model introduced by Aitkin and Rubin [4]. For a given sequence dataset, it searches for the maximum likelihood estimation of the parameters using EM. The idea of missing data (Z, which group of each sample in the data came from) is used by EM. When MM is applied to a dataset to find SLiMs in that dataset, this is done by repeatedly and probabilistically erasing all occurrences of the discovered SLiMs; it is able to find SLiMs with different numbers of occurrences in a single dataset.

The MM algorithm has two major advantages over the Gibbs sampling algorithm [3], proposed by Neuwald *et al.* [38], it is the most successful existing algorithm for discovering SLiMs in a set of protein sequences. First, the classification of input data sequences by a biologist as known to contain the motif in advance is not required, while it calculates the number of occurrences of a motif in the input dataset. Second, it is the formal probabilistic model of the whole dataset used by MM and can systematically select the parameter value of this model that maximizes the likelihood.

5.3 MM with the EM Algorithm

The MM algorithm aims to maximize the likelihood of the parameters of a finite mixture model. Let us consider Y, a dataset of protein sequences represented as:

$$Y = \{Y_1, Y_2, \ldots \ldots Y_N\} \tag{5.1}$$

where $Y_1, Y_2, \ldots \ldots Y_N$ are the N sequences in the dataset and let us consider the sequence Y_i represented as:

$$A = \{a_1, a_2, \ldots \ldots a_L\} \tag{5.2}$$

This is provided to the MM as input. The model used by MM does not use the dataset directly. Instead, it breaks the dataset into overlapping subsequences of length W. The new dataset is presented as:

$$X = \{X_1, X_2, \ldots \ldots X_n\} \tag{5.3}$$

Sequence X is divided into n overlapping subsequences. The model used by MM uses the new abstract dataset.

The new model of the new dataset consists of two components, one of them models the motifs and the other models the background. The MM algorithm assumes that each position in a subsequence of a motif model is generated by an independent random variable describing a multinomial trial with parameter $f_i = (f_{i1}, f_{i2}, \ldots . f_{iL})$. This means the probability of a letter a_j appearing at position i in the motif is f_{ij}. The parameters f_{ij} for $i = 1, 2, \ldots W$ and $i = 1, 2, \ldots L$ must be estimated from the data.

For the background model of the new dataset, each position of a subsequence which is not a part of a motif occurrence is generated independently by a multinomial random variable with a common parameter $f_0 = (f_{01}, \ldots \ldots f_{0L})$. Thus, finally the dataset used by

the MM algorithm has two models, one is the motif model (with probability λ_1) and the other one is the background model (with probability λ_2 = λ_1 - 1). Thus, the parameters of the mixture model are λ = (λ_1, λ_2), the vectors of letter frequencies for the motif model θ_1 = $(f_1, f_2, \ldots \ldots f_W)$ and a single vector of letter frequencies for the background model θ_0 = f_0.

The main goal is to find the values of the parameters for the overall model which maximize the likelihood of the data. For this purpose, EM is used which is an iterative process and finds the values for $\lambda = (\lambda_1, \lambda_2)$ and $\theta = (\theta_1, \theta_2)$. The EM algorithm uses the concept of missing data, which is the knowledge of which group each sample in the data came from. For missing data, the following notations are important:

$$Z = \{Z_1, Z_2, \ldots \ldots Z_n\}, n \text{ is the number of samples} \tag{5.4}$$

$$Z_i = \{Z_{i1}, Z_{i2}\} \tag{5.5}$$

$$Z_{ij} = \begin{cases} 1 & \text{if } X_i \text{ is from group } j \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

Z_i provides the information about the group of the sample. If $Z_{ij} = 1$ then X_i has the probability distribution of $p(X_i \mid \theta_j)$. Then, EM will try to find out the value of Z_{ij} along with the other two parameters θ and λ. The likelihood of the model parameters for the data X can be defined as:

$$L(\theta, \lambda \mid X, Z) = p(X, Z \mid \theta, \lambda) \tag{5.7}$$

and the log-likelihood is given by:

$$\log(L(\theta,\lambda \mid X,Z)) = \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}\ \log[p(X_i \mid \theta_j)\lambda_j] \tag{5.8}$$

The E-step and the M-step of the EM algorithm repeatedly maximize the log-likelihood over the conditional distribution of the missing data, Z, given the observed data X and the current estimates of the parameters θ and λ.

The E-step finds the expected value of Equation (5.8) over the values of missing data Z, given the observed data X, and the current parameter values $\theta = \theta^{(0)}$ and $\lambda = \lambda^{(0)}$,

$$E\{\log[L(\theta,\lambda \mid X,Z)]\} = \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)}\ \log[p(X_i \mid \theta_j)] + \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)} \log(\lambda_j) \tag{5.9}$$

$$Z_{ij}^{(0)} = \frac{p(X_i \mid \theta_j^{(0)})\lambda_j^{(0)}}{\sum_{k=1}^{2} p(X_i \mid \theta_k^{(0)})\lambda_k^{(0)}} \tag{5.10}$$

for $i = 1,\ldots,n$ and $j = 1,2$

The M-step maximizes Equation (5.9) over θ and λ in order to find the next estimates for them, say $\theta^{(1)}$ and $\lambda^{(1)}$. The maximization over λ involves only the second term in Equation (5.9)

$$\underbrace{argmax}_{\lambda} = \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)} \log(\lambda_j) \tag{5.11}$$

which has the solution

$$\lambda_j^{(1)} = \sum_{i=1}^{n} \frac{Z_{ij}^{(0)}}{n} \quad \text{for } j = 1, 2 \tag{5.12}$$

and θ can be maximized by separately maximizing the first term of Equation (5.9) over θ_j:

$$\theta_j^{(1)} = \underbrace{argmax}_{\theta} \sum_{i=1}^{n} Z_{ij}^{(0)} \log[p(X_i) \mid \theta_j] \quad \text{for } j = 1, 2 \tag{5.13}$$

The distributions for the motif class and background class are assumed by the MM algorithm as:

$$p(X_i \mid \theta_1) = \prod_{j=1}^{W} \prod_{k=1}^{L} f_{jk}^{I(k,X_{ij})} \tag{5.14}$$

$$p(X_i \mid \theta_2) = \prod_{j=1}^{W} \prod_{k=1}^{L} f_{0k}^{I(k,X_{ij})} \tag{5.15}$$

where X_{ij} represents the j^{th} letter of sample X_i and $I(k,a)$ indicates the function which is 1 if and only if $a = a_k$, which implies:

$$I(k,a) = \begin{cases} 1 & \text{if } a = a_k \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

for $k = 1, \ldots, L$. Now, let us consider,

$$c_{0k} = \sum_{i=1}^{n} \sum_{j=1}^{W} Z_{i2}^{(0)} \, I(k, X_{ij}) \quad \text{and} \tag{5.17}$$

$$c_{jk} = \sum_{i=1}^{n} E_i Z_{i1}^{(0)} \, I(k, X_{ij}) \tag{5.18}$$

here, c_{0k} is the estimated number of times a letter a_k occurs in a position for the background model, and c_{jk} is the estimation for the motif model for $j = 1 \ldots W$. θ can be re-estimated by replacing Equation (5.14) and (5.15) into the right side of Equation (5.13):

$$\begin{aligned} \theta^1 &= (\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_W) \\ &= \left(\underbrace{argmax}_{\theta} \right) \sum_{j=0}^{W} \sum_{k=1}^{L} c_{jk} \log(f_{jk}) \end{aligned} \tag{5.19}$$

thus finally,

$$\hat{f}_{jk} = \frac{c_{jk}}{\sum_{k=1}^{L} c_{jk}} \quad \text{for j= 0 \ldots W and k = 1 \ldots L} \tag{5.20}$$

For a smaller dataset, if the frequency $\hat{f}_{ij}$ of any letter becomes 0, this value is never changed. Thus, according to Brown *et al.* and Lawrence *et al.*, Equation (5.20) can be written as [3]:

$$\hat{f}_{ij} = \frac{c_{ij} + \beta_j}{\sum_{k=1}^{L} c_{ik} + \acute{\beta}} \tag{5.21}$$

here, $i = 0, \ldots, W$, $j = 1, \ldots, L$, $\beta = \sum_{k=1}^{L} \beta_k$ and $\acute{\beta} = (\beta_1, \ldots, \beta_L)$. This is equivalent to using Bayes estimates for the value of θ.

Chapter 6

Model Implementation

In this thesis, I have implemented two different models to predict PPIs. One of the models (PPI-SLiM-Seq) predicts PPIs using the information contained in the sequence and the other model (PPI-SLiM-DEEE) predicts PPIs using desolvation and electrostatic energies as features. For both models, I use the same two datasets. This chapter discusses both models.

6.1 Datasets

For all experiments of the models proposed in this thesis, two well-known different datasets have been used:

1. Dataset by Zhu *et al.* [54] (the ZH dataset) and

2. Dataset by Mintseris *et al.* [33] (the MW dataset)

Table 6.1 shows an overview of the datasets used in this thesis. The ZH dataset has a total of 137 binary complexes and the MW dataset has a total of 327 complexes. Although

Dataset name	No. of obligate complex	No. of non-obligate complex
Mintseris dataset	115	212
Zhu dataset	75	62

Table 6.1: Overview of the ZH and MW datasets.

most of the complexes of the MW dataset are dimers, there are also some complexes which are not multimers.

6.1.1 The ZH dataset

The ZH dataset is composed of 137 different complexes, 75 of the complexes are obligate and the remaining 62 are non-obligate. This dataset was compiled and proposed by Zhu *et al.* [54]. Each complex of this dataset has only two chains; that is why it also known as the ZH binary dataset.

Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain
1AHJ	A:B	1B34	A:B	1DCE	A:B	1EFV	A:B	1GUX	A:B
1H2A	L:S	1LUC	A:B	1PNK	A:B	1REQ	A:B	1TCO	A:B
2AAI	A:B	1A0F	A:B	1A4I	A:B	1aFW	A:B	1AJ8	A:B
1AJS	A:B	1AOM	A:B	1AQ6	A:B	1AT3	A:B	1B3A	A:B
1B5E	A:B	1B7B	A:C	1B8A	A:B	1B8J	A:B	1B9M	A:B
1BJN	A:B	1BOL	A:B	1BRM	A:B	1BYF	A:B	1BYK	A:B
1C7N	A:B	1CLI	A:B	1CMB	A:B	1CNZ	A:B	1COZ	A:B
1CP2	A:B	1DOR	A:B	1F6Y	A:B	1GPE	A:B	1HGX	A:B
1HJR	A:C	1HSS	A:B	1ISA	A:B	1JKM	A:B	1KPE	A:B
1MSP	A:B	1NSE	A:B	1ONE	A:B	1PP2	A:B	1QAE	A:B
1QAX	A:B	1QBI	A:B	1QFE	A:B	1QFH	A:B	1QOR	A:B
1QU7	A:B	1SMT	A:B	1SOX	A:B	1SPU	A:B	1TRK	A:B
1VLT	A:B	1VOK	A:B	1WGJ	A:B	1XIK	A:B	1XSO	A:B
1YPI	A:B	1YVE	I:J	2AE2	A:B	2HDH	A:B	2HHM	A:B
2NAC	A:B	2PFL	A:B	2UTG	A:B	3TMK	A:B	4MDH	A:B

Table 6.2: Obligate complexes in the ZH dataset.

Table 6.2 shows all 75 obligate complexes used for the experiments from the dataset compiled by Zhu *et al.* [54]. Each cell in the table represents one complex, the first part is the PDB ID of the complex and second part specifies the chains (separated by ":") associated with the complex.

Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain
1AVA	A:B	1AVW	A:B	1BVN	T:P	1CSE	I:E	1F34	A:B
1FSS	L:S	1GLA	F:G	1KXQ	H:A	1SMP	I:A	1TAB	I:E
1TGS	I:Z	2PTC	I:E	2SIC	I:E	4SGB	i:E	1AGR	E:A
1ATN	A:D	1B6C	A:B	1BKD	R:S	1BUH	A:B	1DOW	A:B
1EUV	A:B	1I2M	A:C	1I8L	A:C	1KAC	A:B	1PDK	A:B
1QAV	A:B	1TX4	A:B	1C0F	S:A	1ZBD	A:B	1AK4	A:D
1D09	A:B	1CQI	A:B	1FIN	A:B	1DHK	A:B	1BI7	A:B
1RRP	A:B	1CC0	A:E	1EG9	A:B	1AVZ	B:C	1FRV	A:B
3HHR	A:C	1YCS	A:B	1CVS	A:C	1ARO	L:P	1CMX	A:B
1BML	A:C	2PCB	A:B	1F60	A:B	1STF	E:I	1EMV	A:B
1UEA	A:B	1QBK	B:C	1HLU	A:P	1ITB	A:B	1ETH	A:B
1JTD	A:B	1LFD	A:B	1Dn1	A:B	1TMQ	A:B	1A4Y	A:B
1WQ1	R:G	1EAI	C:A						

Table 6.3: Non-obligate complexes in the ZH dataset.

Table 6.3 shows all 62 non-obligate complexes used for the experiments from the dataset compiled by Zhu *et al.* [54].

6.1.2 The MW dataset

The MW dataset is composed of 327 different complexes; 115 of the complexes are obligate and the remaining 212 complexes are non-obligate. This dataset was compiled and proposed by Mintseris *et al.* [33].

Table 6.4 shows all 115 obligate complexes used in the experiments from the dataset compiled by Mintseris *et al.* [33]. Table 6.5 shows all 212 non-obligate complexes used in the experiments from the dataset compiled by Mintseris *et al.* [33].

Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain
1NBW	AC:B	1DXT	A:B	1B8M	A:B	1K8K	C:G	2KAU	B:C
1H8E	A:D	1B7Y	A:E	4RUB	A:DT	1JB0	AB:C	1FFV	A:B
1K8K	C:F	1E50	A:B	1LD8	A:B	1B4U	A:B	1DKF	A:B
2Q33	A:B	1CPC	A:B	1RAF	A:BD	1VKX	A:B	1JMZ	AD:B
1FXW	A:F	1JB0	AB:E	1IR1	A:S	1H2R	L:S	1SGF	A:BY
1K8K	D:F	1JV2	A:B	1E6V	A:B	1HSA	A:B	1H2V	C:Z
1MJG	AB:M	1F3U	A:B	2MIN	A:B	1SPP	A:B	1KTD	A:B
1GKA	A:B	1IHF	A:B	1A6D	A:B	1TBG	A:B	1FFU	A:C
1K8K	B:F	1DM0	A:BCFDE	1MRO	A:C	1DTW	A:B	1FS0	E:G
1H32	A:B	1POI	A:B	1JK0	A:B	1JRO	A:BD	1YTF	BC:D
3PCE	A:M	1EFV	A:B	1JWH	A:CD	2MTA	H:L	1JKJ	A:B
1C3O	A:B	1K8K	A:E	1MRO	B:C	1GPW	A:B	1M2V	A:B
1HCN	A:B	1JB0	A:BD	2AHJ	A:B	3GTU	A:B	1KQF	A:B
1HZZ	A:BC	1HFE	L:S	1EZV	C:F	1EEX	A:G	1H4I	A:B
1L9J	C:HLM	1QLB	B:C	1CCW	A:B	1JK8	A:B	1JB7	A:B
1REQ	A:B	1QGW	B:C	1VCB	A:B	1JNR	A:B	1LUC	A:B
1JB0	C:D	1BE3	CDEGK:A	1EG9	A:B	1K28	A:D	1DII	A:C
2BS2	A:B	1EEX	A:B	1EXB	A:E	1E8O	A:B	1HXM	A:B
1E9Z	A:B	1HR6	AE:B	1DJ7	A:B	1LTI	AC:DEHFG	1JB0	C:B
1LDJ	A:B	1FCD	A:C	1K3U	A:B	1L7V	AB:C	1AUI	A:B
1GO3	G:F								

Table 6.4: Obligate complexes in the MW dataset.

6.2 PPI-SLiM-Seq

In this thesis, I propose a computational model, PPI-SLiM-Seq, to predict PPIs using information contained in the sequences. I have used leave-one-out validation to validate this model. This model has a few components:

- Sequence database compilation
- SLiM finding
- Separating sample and training sequences
- Sequence partitioning
- Calculation of the information contained in the ℓ-mers
- Select top 20 values from the calculated ℓ-mers' information values

Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain	Complex	Chain
1WQ1	G:R	1TMQ	A:B	1FBI	HL:X	1BQH	AB:GM	1AKJ	AB:DE
1LK3	A:HL	1F51	AB:E	1BVN	P:T	1G73	AB:C	1IB1	AB:E
1EER	A:B	4SGB	E:I	1G4Y	B:R	1STF	E:I	1GL1	A:I
2BTC	E:I	1FQJ	A:C	1DKG	AB:D	1DF9	B:C	1EBP	A:CD
1M10	A:B	2SIC	E:I	1GP2	A:B	1WWW	VW:X	1CS4	AB:C
1MBU	A:C	1NF5	A:B	1ICF	AB:I	1BKD	R:S	1AR1	AB:CD
1JMA	A:B	1QFU	AB:HL	1EZV	E:XY	1BDJ	A:B	1ILA	AB:CD
3G94	A:BC	1CC0	A:E	1KYO	O:W	1IBR	A:B	1DEV	A:B
1JCH	A:B	2TEC	E:I	1EZX	AB:C	1GH6	A:B	1AK4	A:D
1FQ1	A:B	1JIW	I:P	1F60	A:B	1STF	E:I	1EMV	A:B
1UEA	A:B	1QBK	B:C	1HLU	A:P	1ITB	A:B	1ETH	A:B
1JTD	A:B	1LFD	A:B	1DN1	A:B	1TMQ	A:B	1A4Y	A:B
1WQ1	R:G	1EAI	C:A	1AVZ	B:c	1D4X	A:C	1BI8	A:B
1ES7	AC:B	1HE1	A:C	1KZY	A:C	1DOA	A:B	1K5D	A:B
1GL4	A:B	1FQV	A:B	1O6S	A:B	1FBV	A:C	1T7P	A:B
1FSK	A:BC	1E6J	HL:P	1BZQ	A:L	1GC1	C:G	1CLV	A:I
1F34	A:B	1JTD	A:B	1EJA	A:B	1AY7	A:C	1DFJ	E:I
1F80	A:E	1F02	I:T	1AVX	A:B	1RLB	ABCD:E	2BTF	A:B
1BHU	A:B	1K5D	A:C	1KCG	AB:C	1YCG	A:B	1YCS	A:B
2MTA	A:HL	1N2C	AB:BF	1D2Z	A:B	1K3Z	AB:D	1H2K	A:S
1C4Z	A:D	1ZBD	A:B	1KXT	A:B	1WEJ	F:HL	1NSN	HL:S
1BJ1	HL:VW	1AVA	A:C	1DPJ	A:B	1D5X	A:C	1LPB	A:B
1DX5	A:C	1ZBD	A:B	1AVG	HL:I	1BUV	M:T	1TAB	E:I
1AZZ	A:CD	1ARO	L:P	1HX1	A:B	1QO0	A:DE	1HWG	A:BC
1EFX	ABC:D	1GLA	F:G	1B9Y	AB:C	1MAH	A:F	3YGS	C:P
1M4U	A:L	1MLE	A:B	1FNS	A:HL	1AHW	AB:C	2JEL	HL:P
1BGX	HL:T	1MR1	A:D	1K4C	AB:C	1DHK	A:B	1IM3	AB:D
1I4E	A:B	1DTD	A:B	1KK1	ABC:H	1K90	A:D	4HTC	HL:I
2PTC	E:I	1BML	A:C	1AWC	A:B	1ATN	A:D	1UGH	E:I
1FG9	AB:C	1DE4	CF:A	1CXZ	A:B	4CPA	I:O	1GCQ	B:C
1GRN	A:B	1M2O	AC:B	1O94	AB:CD				

Table 6.5: Non-obligate complexes in the MW dataset.

These components are executed one after the other. Figure 6.1 shows the work flow diagram of the PPI-SLiM-Seq model. I discuss all the components in detail in the next sub-sections.

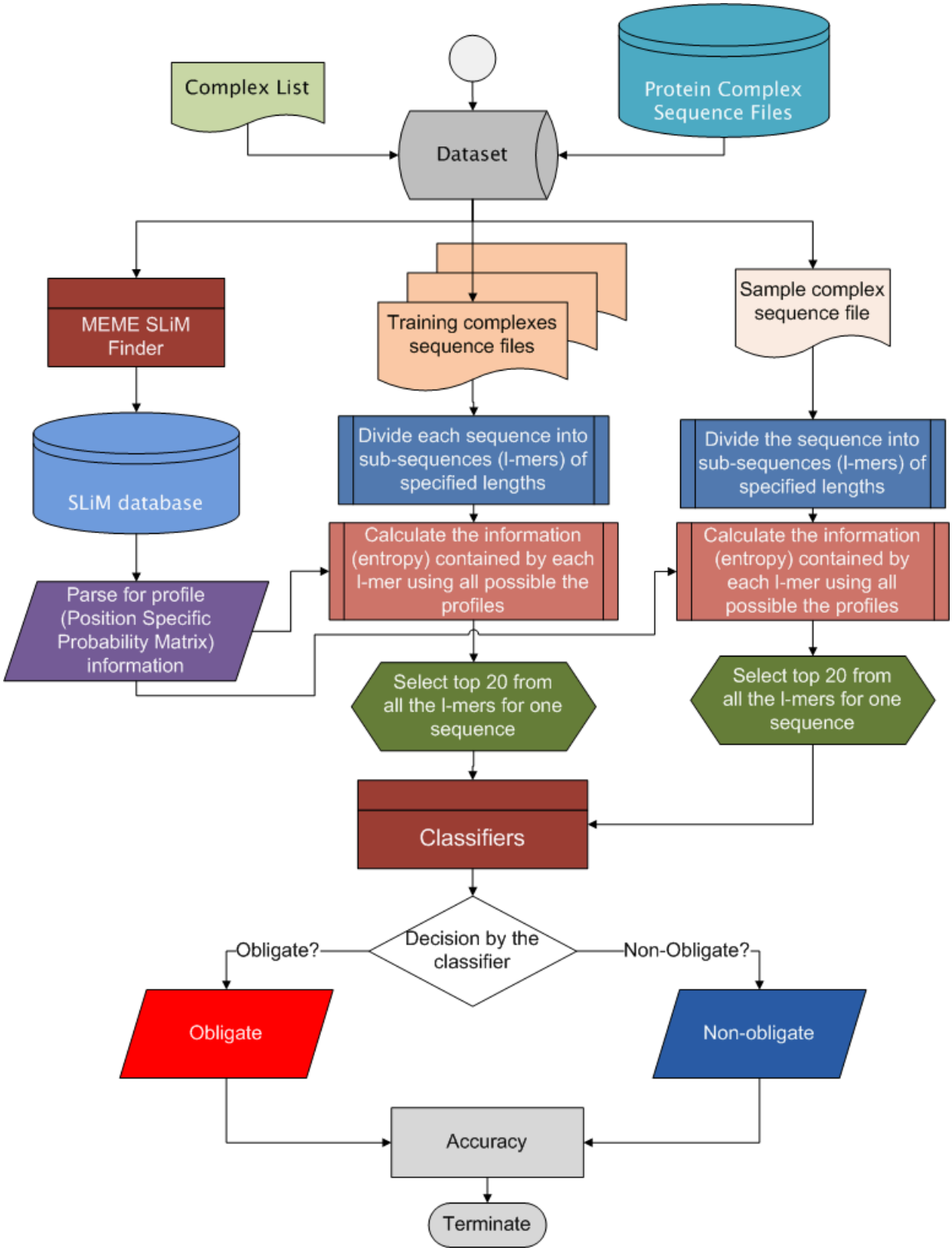


Figure 6.1: Work flow diagram of the PPI-SLiM-Seq.

6.2.1 Sequence Database Compilation

This component takes the list of all the complexes for a dataset as input. For each complex in the list, it downloads the FASTA sequence file from the PDB server. It extracts the sequence for each chain (listed in the dataset) from the file and adds it to the sequence database. Finally, the database has all the sequences for all the chains of each complex.

6.2.2 SLiM Finding

I have used MEME to find SLiMs from the collected sequence dataset. I optimized the parameters of the tools. I did run MEME to find 500 and 1,000 SLiMs from the dataset. I specified the lengths of the SLiMs as $3-10$ and $2-7$, the minimum number of sites as 8 and the maximum number of sites as 200. Details about MEME are discussed in Chapter 5 and Appendix A.

6.2.3 Separating Sample and Training Sequences

To implement the leave-one-out process, dividing the sequence database into target sample and training sample is needed. Each time, the target sample (one) and training samples (remaining) are separated. As it is an iterative process, the same procedure continues until it covers all the complexes. The components of this model take care of this.

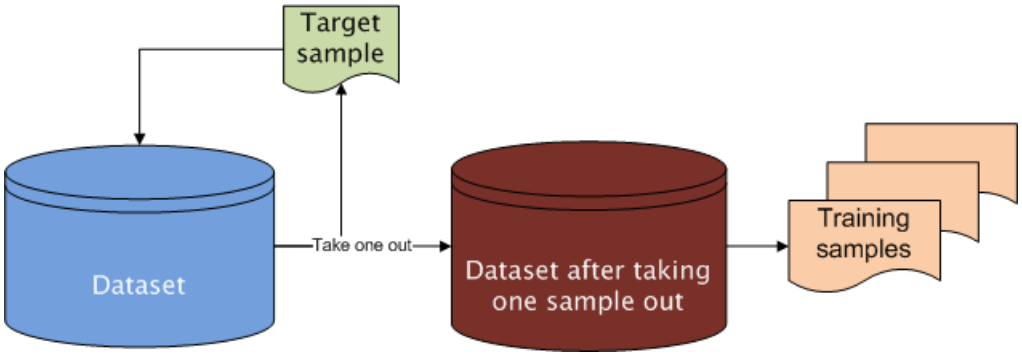


Figure 6.2: Separating sample and training sequences.

Figure 6.2 shows the working procedure of the component. Each iteration, the model takes one complex out as target sample and uses the remaining samples for training. In the next iteration, it places the previously taken out complex back and takes another complex out to cover all the complexes.

6.2.4 Sequence Partitioning

For both training and sample complexes, the sequences are divided into overlapping subsequences. The sequence partitioning component takes a sequence as input and outputs all possible subsequences (ℓ-mers). Let us consider a, a sequence of length L which is input into the sequence partitioning component. The component will divide the sequence into all possible overlapping ℓ-mers of length W and deliver a total of $\{L-(W+1)\}$ ℓ-mers.

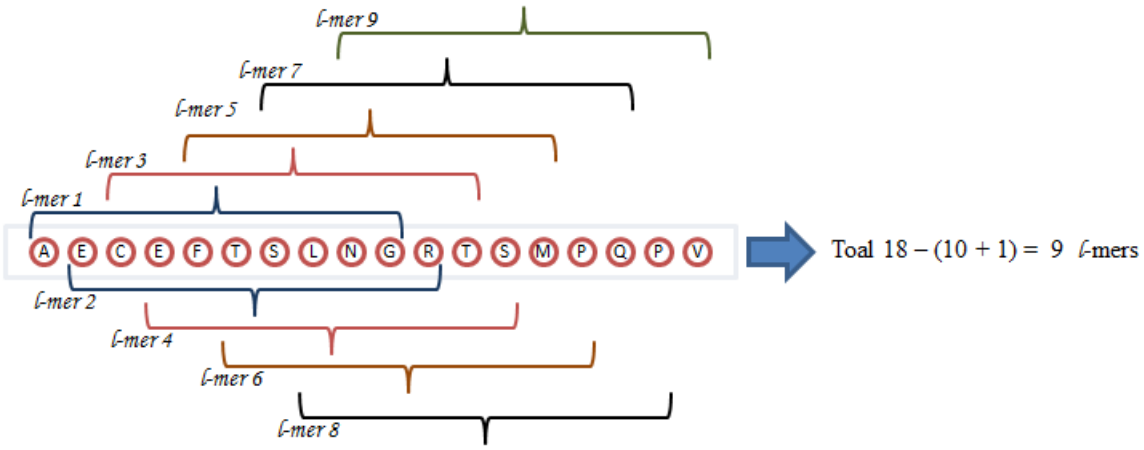


Figure 6.3: Splitting a sequence into ℓ-mers.

Figure 6.3 shows the partitioning process of a sequence of length 18 delivering a total $(18-(10+1))=9$ of ℓ-mers.

6.2.5 Calculating Information Contained in the ℓ-mers

This component calculates the information contained in each ℓ-mer for a given profile. It uses the PSPMs as profiles from the SLiM database. For profile selection, it follows mainly two rules:

1. It excludes all the profiles of all the SLiMs which were found in the current complex
2. It takes only those profiles of SLiMs which have the same length as the current ℓ-mer.

For calculating the information contained in the ℓ-mer, it follows Equation (6.1):

$$\hat{I}(a \mid X) = -\frac{1}{\ell} \times \sum_{i=1}^{\ell} P(a_i) \times \log(P(a_i)) \tag{6.1}$$

where X is the given profile (PSPM) from the SLiM, a is the ℓ-mer, $P(a_i)$ is the probability of i^{th} amino acid in the profile and $\hat{I}(a \mid X)$ is the information contained in the ℓ-mer.

6.2.6 ℓ-mer Information Selection

Each sequence is then partitioned into all possible ℓ-mers. If the sequence is of length L, then it has a total of $(L-(W+1))$ ℓ-mers of length W. For each ℓ-mer, a_i, I calculate the information $(\hat{I}_1, \cdots, \hat{I}_n)$ for all given profiles, $(X_1, \cdots, X_n)$ using Equation (6.1). From the set of information values, I select only the maximum. In this way, I cover all the ℓ-mers. Once I have information values for all the ℓ-mers, I select the top 20 ℓ-mer's information values. I use top 20 ℓ-mer information values as features for the complex.

6.3 Pseudo Code for PPI-SLiM-Seq

Algorithm 2 shows all the steps for the PPI-SLiM-Seq model. It takes a list of complexes, sequence files for all the complexes, SLiM database as inputs. It outputs the feature vector, which is then used for classification. For the ZH and MW datasets, each complex has 20 features. Thus, the ZH dataset yields feature matrices of 136×20 and 137×20 dimensions and the MW dataset yields feature matrices of 326×20 and 327×20 dimensions. The feature matrices generated by this model are then used for the classification. I use k-NN, LDR and SVM classifiers for classification.

Algorithm 2 PPI-SLiM-Seq Algorithm

$FeatureVector = [\,]$
for all the complexes SC_i **do**
  SequencePartition(SC_i)
  $LISTSC_i = [\,]$
  for all the ℓ-mers LL_j **do**
    $MaxInfo_j = 0$
    SelectProfile(SC_i, LL_j)
    for all the profiles LP_k **do**
      $LLINFO_k$ = CalculateInformation(LL_j, LP_k)
      if $LLINFO_k > MaxInfo_j$ **then**
        $MaxInfo_j = LLINFO_k$
      end if
    end for
    ADD($MaxInfo_j, LISTSC_i$)
  end for
  ADDROW($LISTSC_i, FeatureVector$)
end for

6.4 PPI-SLiM-DEEE

In this thesis, I also propose another model to predict PPIs using SLiMs. In this model, I determine if two SLiMs from two different chains of a protein complex are interacting or not and using this information, I predict the type of the complexes; obligate and non-obligate. Besides, I also check if one SLiM from one chain is interacting with the other chain (any part of the chain) of the complex. To predict these interactions I use desolvation and electrostatic energies as features. This model also contains different components:

1. Sequence dataset compilation
2. SLiM finding
3. Selecting interface SLiMs

4. Selecting interacting SLiM pairs

5. Calculating desolvation energy

6. Calculating electrostatic energy

7. Generating atom-atom type feature matrix

In PPI-SLiM-Seq, I have already described the first two components, sequence database compilation and SLiM finding. Here I describe the other components. Thus, I assume that I already have the SLiMs from the sequence dataset.

6.4.1 Selecting Interface SLiMs

This component checks if a SLiM is on the interface or not. In this experiment, I consider a SLiM is on the interface if at least one atom from any of the residues (amino acids) of the SLiM is on the interface (as discussed in Chapter 4). This component takes a SLiM as input and outputs *TRUE*, if the SLiM is on the interface otherwise it outputs *FALSE*. Figure 6.4 shows the work flow diagram of the component. Once it has a SLiM as input it has the following information about that SLiM:

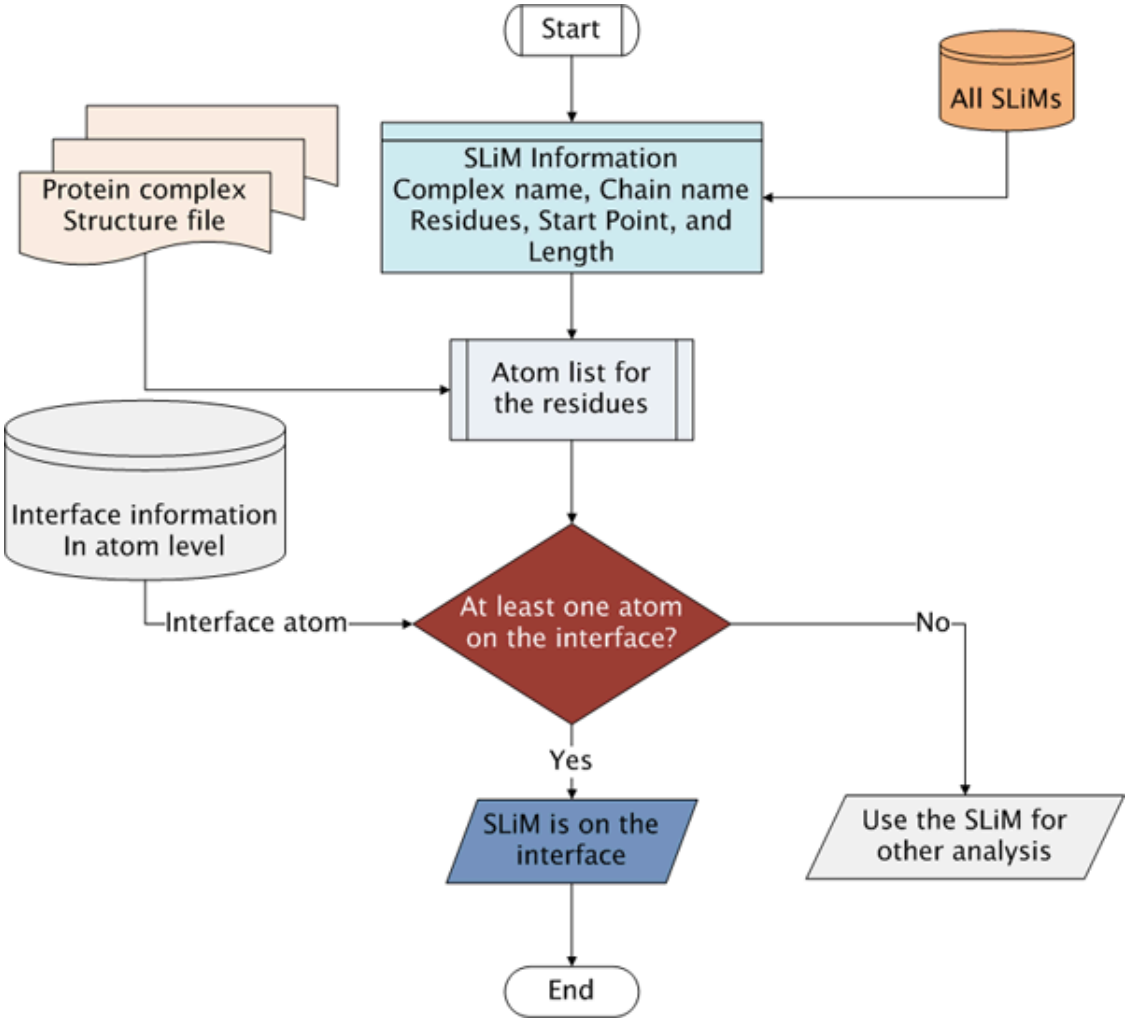


Figure 6.4: Work flow diagram to check if a SLiM is on the interface.

- Complex name
- Chain name
- Residue name
- Start point of the SLiM in the chain
- Length of the SLiM

Algorithm 3 Check if a SLiM is on the interface

```
SLiMAtomList = [ ]
FLAG = FALSE
for all the residues in SLiM SR_i do
    for all the residues in PDB PR_j do
        if SR_i = PR_j then
            Add(AtomNum_k, SLiMAtomList)
        end if
    end for
end for
for all atoms in SLiMAtomList AS_m do
    for all atoms in interface database AD_n do
        if AD_n == AS_m then
            FLAG = TRUE
            break
        end if
    end for
end for
return FLAG
```

Using this information, it generates a list of all the atoms that belong to this SLiM from the PDB structure file and interface database. Algorithm 3 shows all the steps of this component.

6.4.2 Selecting Interacting SLiM Pairs

This component of the model checks if two SLiMs M_i and M_j are interacting with each other or not. If a_i and a_j are the atoms where $a_i \in M_i$ and $a_j \in M_j$, if $\forall\ \delta(a_i, a_j)$ there $\exists\ \delta(a_i, a_j) : \delta(a_i, a_j) < 7\text{Å}$, then M_i and M_j will be interacting, where $\delta(a_i, a_j)$ represents the distance between a_i and a_j.

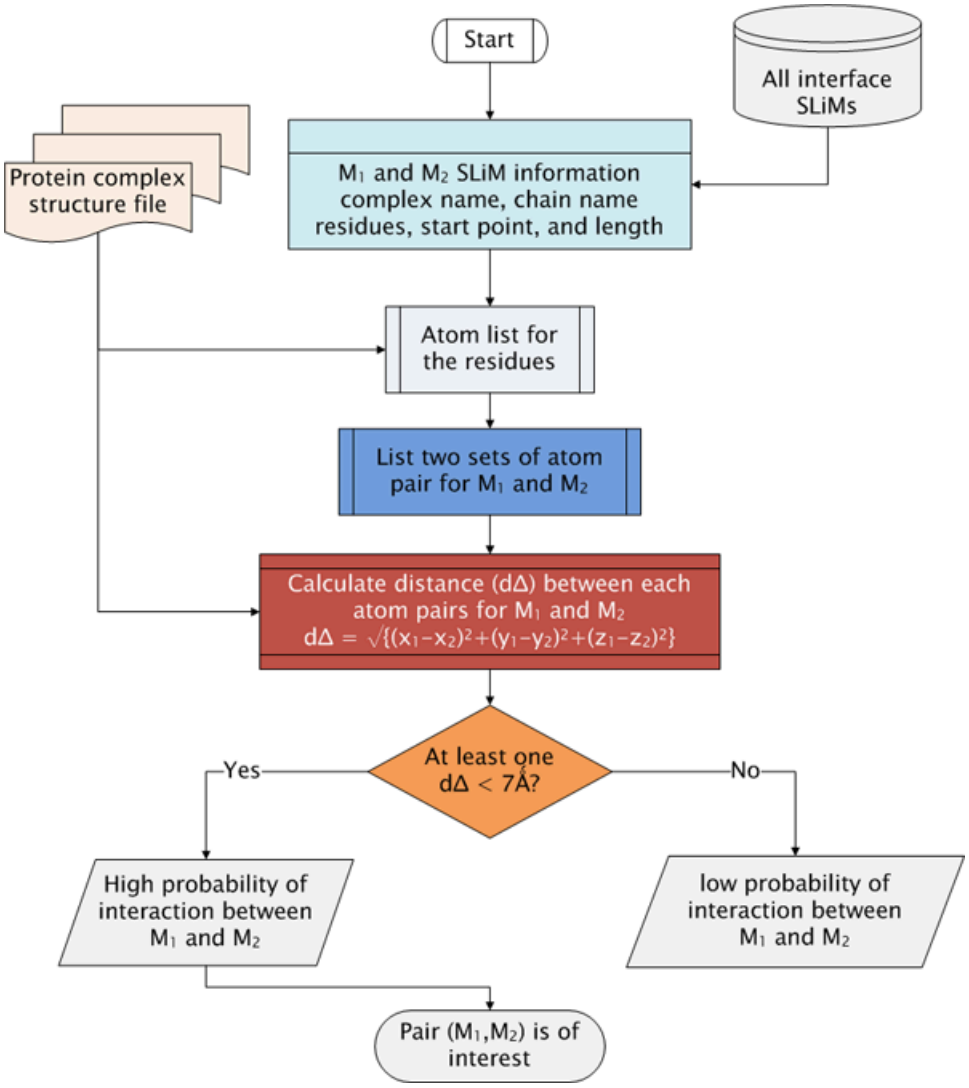


Figure 6.5: Work flow diagram to check if two SLiMs are interacting.

To check the interactions between two SLiMs, the algorithm takes two SLiMs as input and outputs *TRUE* if they are interacting, otherwise *FALSE* is returned. Figure 6.5 shows the work flow diagram of the component. It takes two SLiMs as input and has the following information about the SLiMs:

- Complex name

- Chain name
- Residue name
- Start point of the SLiM in the chain
- Length of the SLiM

Using this information with the structural information from the PDB structure file, this component determines whether the SLiMs are interacting or not. In this thesis, I have used $7\mathring{A}$ distance as threshold when using desolvation energy and $10\mathring{A}$ distance for electrostatic energy.

6.4.3 Calculating Desolvation Energies

To predict PPIs, I have formalized the desolvation energy in an atom-atom type matrix. For SLiM-SLiM interaction, I consider all the atoms of the SLiM pair ($M_1 - M_2$) that are within the threshold distance. In case of desolvation energy, I have used $7\mathring{A}$ as the threshold distance. Calculation of desolvation energy has been discussed in Chapter 3.

Algorithm 4 shows all steps to calculate the desolvation energies of all atom-atom pairs for all atoms that belongs to the SLiM pair. Again, when I consider SLiM-protein interactions, I consider all the atoms from the protein chain interacting with a SLiM (M_1) rather than considering only atoms from the other SLiM (M_2). I use the same algorithm (with some minor modifications) to compute the desolvation energy for SLiM-protein.

This component is based on Algorithm 4. The same algorithm was also used to calculate electrostatic energies. It takes two SLiMs or one SLiM as input, and delivers all the atom-atom pairs found for the SLiM-SLiM and SLiM-protein schemes along with the calculated desolvation energies for each atom-atom pair.

Algorithm 4 Calculating desolvation/electrostatic energy

$\Delta E_m = 0$
if $PDBID_1 == PDBID_2$ AND $CHAIN_1 \neq CHAIN_2$ **then**
  $AtomListM_1 = AtomSelector(PDBID_1, CHAIN_1, STPOS_1, L_1)$
  $AtomListM_2 = AtomSelector(PDBID_2, CHAIN_2, STPOS_2, L_2)$
  for all atoms in $AtomListM_1$, A_{1i} **do**
    $XA_{1i} = xCoord(A_{1i}, PDBID_1)$
    $YA_{1i} = yCoord(A_{1i}, PDBID_1)$
    $ZA_{1i} = zCoord(A_{1i}, PDBID_1)$
    for all atoms in $AtomListM_2$, A_{2j} **do**
      $XA_{2j} = xCoord(A_{2j}, PDBID_2)$
      $YA_{2j} = yCoord(A_{2j}, PDBID_2)$
      $ZA_{2j} = zCoord(A_{2j}, PDBID_2)$
      $\delta = distanceCalculator(XA_{1i}, YA_{1i}, ZA_{1i}, XA_{2j}, YA_{2j}, ZA_{2j})$
      if $\delta < 7\mathring{A}$ **then**
        $\Delta E_m = \Delta E_m + EnergyCalculator(A_{1i}, A_{2j}, \delta)$
      end if
    end for
  end for
else
  return ΔE_m
end if
return ΔE_m

6.4.4 Calculating Electrostatic Energies

To predict PPIs, I have gathered the electrostatic energies in the atom-atom type matrix. For SLiM-SLiM, I consider all the atoms of the SLiM pair that are within the threshold distance. In case of electrostatic energies, I have used $10\mathring{A}$ as the threshold distance. Calculation of electrostatic energy is discussed in Chapter 3.

I have also used Algorithm 4 to calculate the electrostatic energies. Again, when I consider SLiM-protein, I consider all the atoms from the protein chain interacting with a SLiM (M_1) rather considering only atoms from the other SLiM (M_2). I use the same algorithm (with some minor modification) to compute the electrostatic energies.

6.4.5 Generating Atom-atom Type Feature Vector

This component generates the complex vs. atom-atom type matrix. According to the authors of [52], there are mainly 18 different types of atoms and the atomic contact potential (ACP) for each atom-atom type pair is fixed [9]. Thus, for each complex, I consider all the atom-atom types, which yields a total of $(^{18}C_2) + 18 = 171$. For the ZH dataset, I obtain a feature matrix of 137×171 dimensions and the MW dataset a feature matrix of 327×171 dimensions.

This component is internally integrated with the previous components and each time the previous component outputs one atom-atom type with desolvation/electrostatic energies, it manages them to map it onto the feature matrix.

Chapter 7

Results and Analysis

Two different computational models have been designed and implemented in this thesis. To measure the performance and accuracy of those models, different experimental setups were adopted. For all the experiments and analysis, the two datasets have been used. In this chapter, results from both models for different experimental setups are discussed and analyzed. This chapter also provides a comparative view and analysis of the performance and accuracy of these models with other existing models.

7.1 Dataset Analysis

As discussed in Chapter 6, I use two different datasets; the ZH and MW dataset. The ZH dataset contains 75 obligate and 62 non-obligate complexes while the MW dataset has 212 non-obligate and 115 obligate complexes.

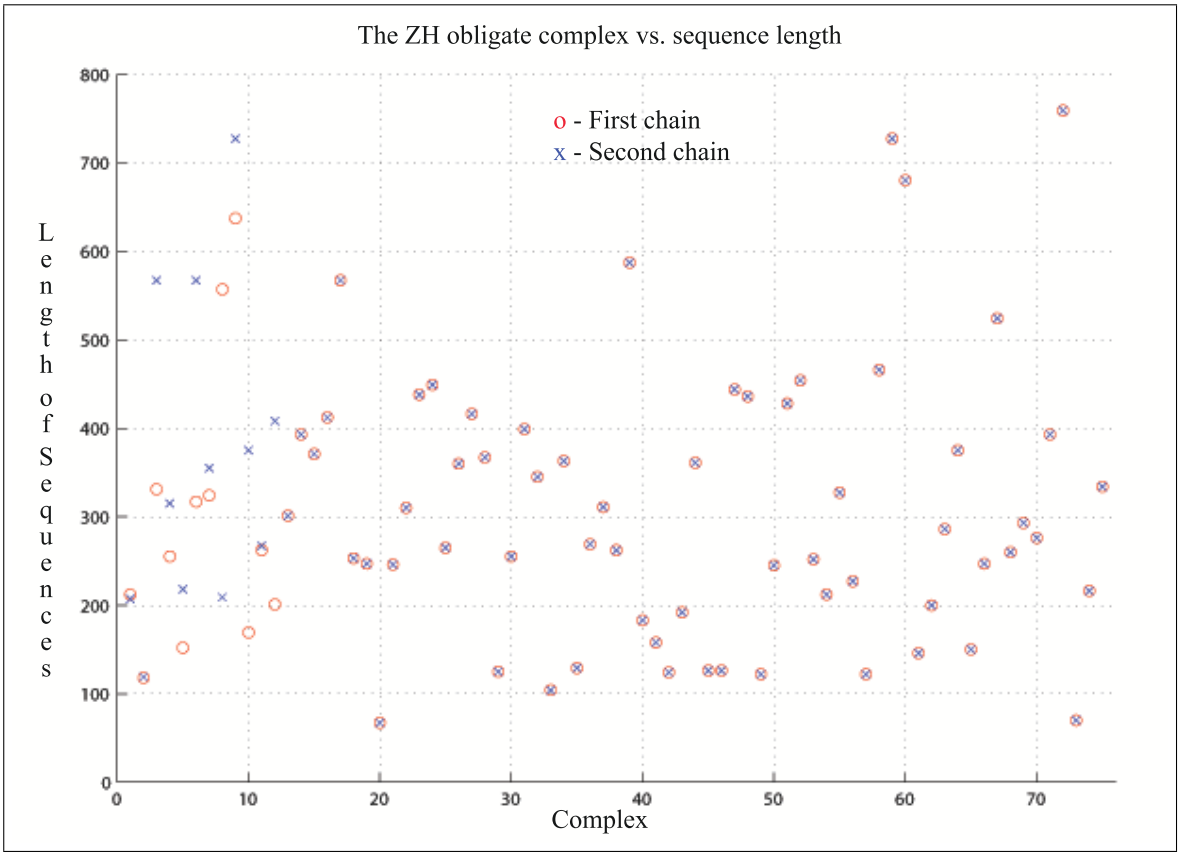


Figure 7.1: Sequence length analysis of the ZH obligate dataset.

Figure 7.1 shows the sequence length distribution of ZH obligate complexes. The *x*-axis represents all the complexes and the *y*-axis represents the lengths of the chains. In the figure, each complex is represented with two marks (red o and x). Each mark represents the length of one chain of the complexes. Complex 2pfl has the longest chain which is 759 residues long while complex 1b3a has the shortest chain of 67 residues. It also shows that in most of the complexes, both chains have the same length. Again, Figure 7.2 shows the sequence distribution of the ZH non-obligate complexes. In case of the ZH non-obligate, complex 1qbk has the longest chain of 890 residues and complex 1dow has the shortest chain of 32 residues long. But no complex has chains of the same length. This is a noticeable difference

between these two datasets. The obligate complexes are mostly *homodimers* and this is one of the main reasons behind this difference.

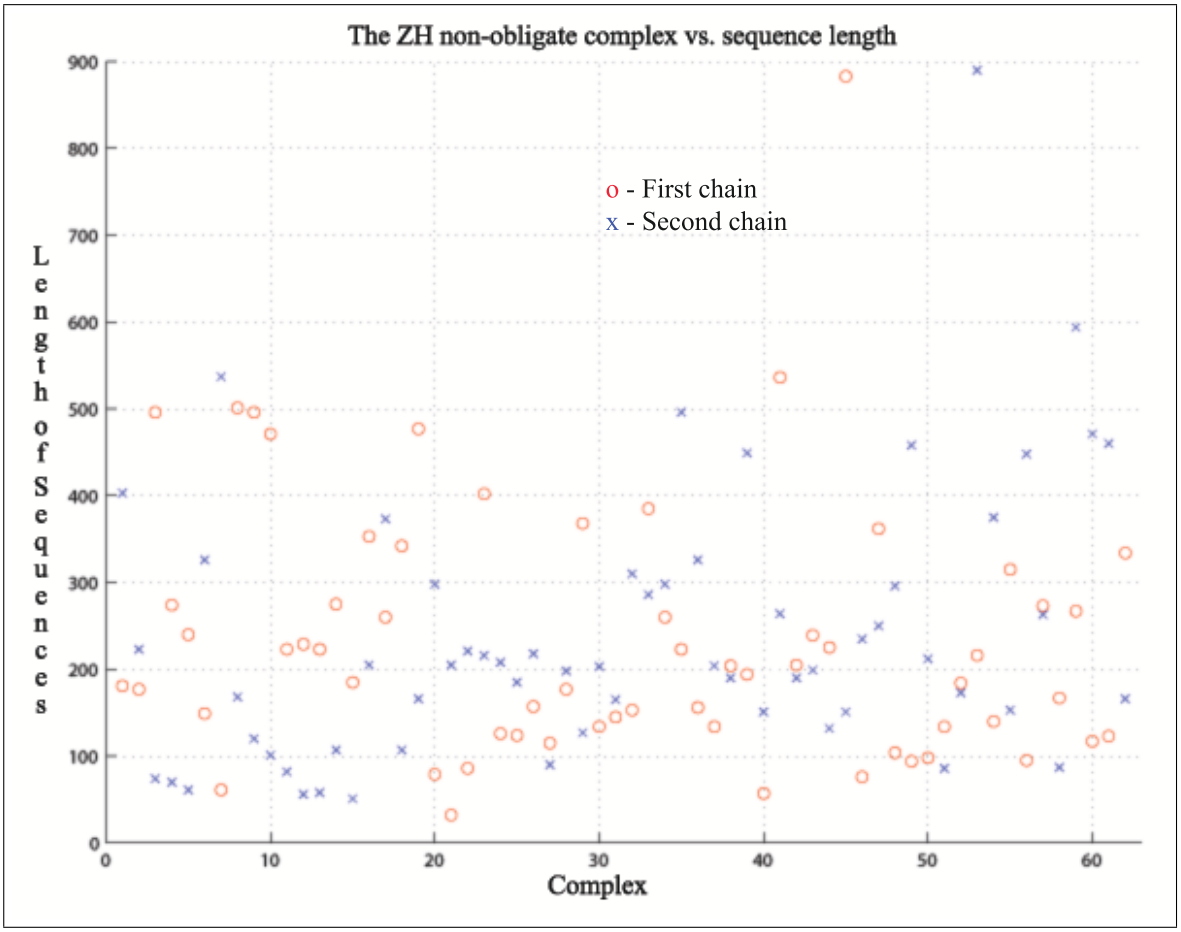


Figure 7.2: Sequence length analysis of the ZH non-obligate dataset.

Figure 7.3 shows the sequence length distribution of the MW obligate complexes. The *x*-axis represents all the complexes and the *y*-axis represents the lengths of the chains. Most of the complexes in the MW dataset have two chains, but there are some complexes which have more than two chains. For simplicity, I consider up to five chains from complexes having more than two chains.

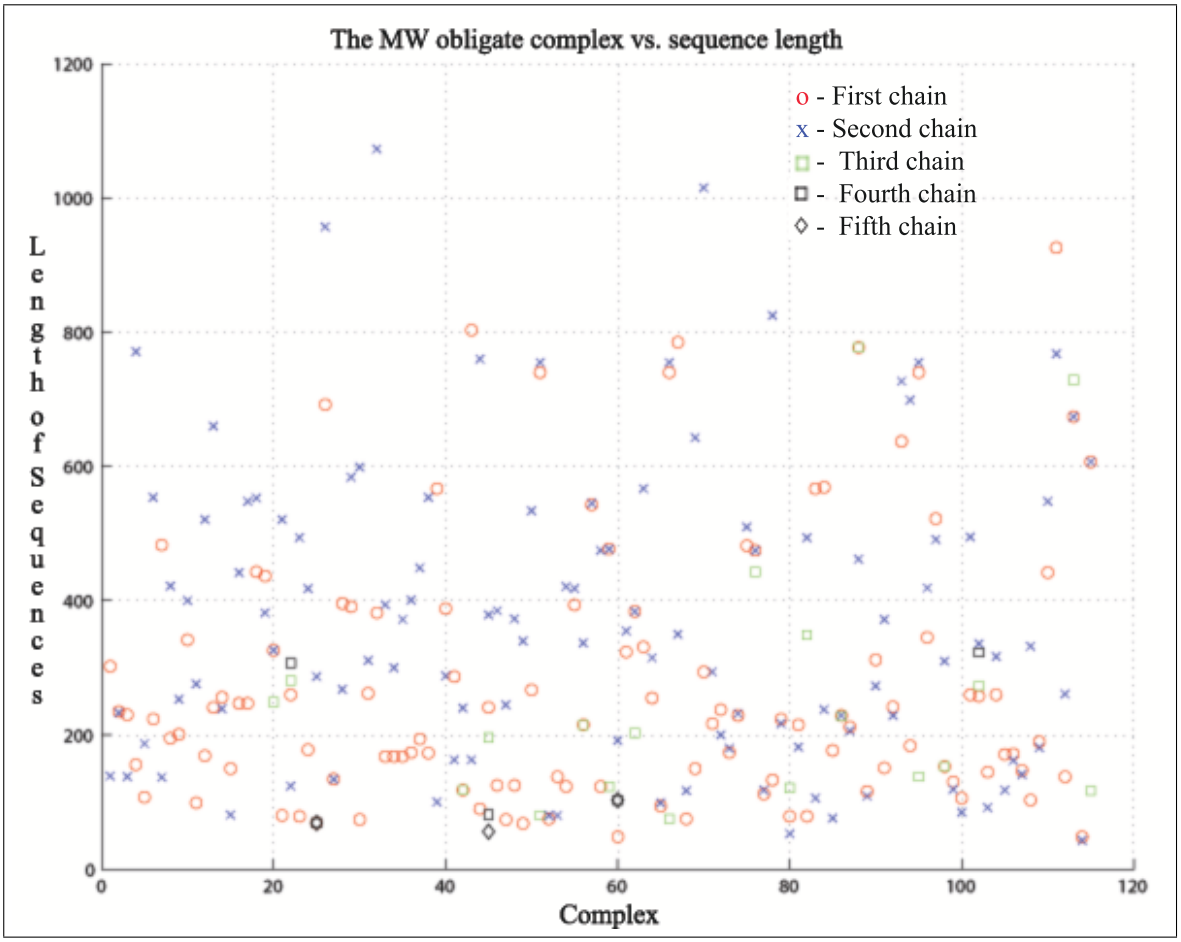


Figure 7.3: Sequence length analysis of the MW obligate dataset.

A red circle represents the first chain, a blue cross, green square, black square and black diamond represent the second, third, fourth and fifth chain, respectively. In the MW obligate dataset, complex 1c3o has the longest chain of 1,073 residues while complex 2q33 has the shortest chain of 43 residues.

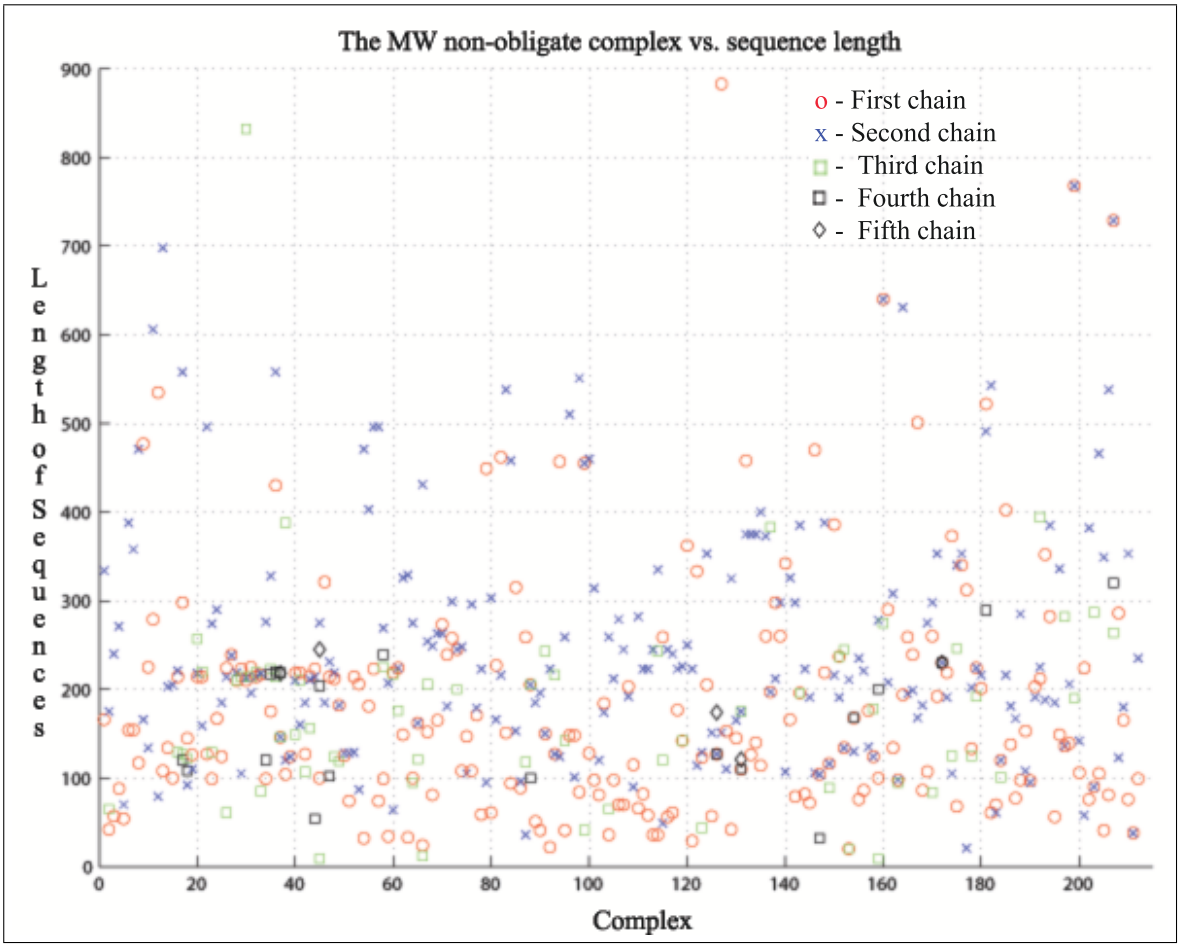


Figure 7.4: Sequence length analysis of the MW non-obligate dataset.

Figure 7.4 shows all the complexes of the MW non-obligate dataset. In this dataset, complex 1aro has the longest chain of 883 residues and 1ao7 has the shortest chain of 9 residues. The differences between the chains of obligate and non-obligate are not noticeable as in the ZH dataset.

7.2 SLiM Analysis

The two datasets, MW and ZH, have been used for all experiments. Based on the length and total number of SLiMs, different sets of SLiMs have been defined from the datasets:

- *SLiM_ZH_1000_3_10* - 1000 SLiMs, length : $3-10$, dataset: ZH
- *SLiM_ZH_1000_2_7* - 1000 SLiMs, length : $2-7$, dataset: ZH
- *SLiM_ZH_500_3_10* - 500 SLiMs, length : $3-10$, dataset: ZH
- *SLiM_ZH_500_2_7* - 500 SLiMs, length : $2-7$, dataset: ZH
- *SLiM_MW_1000_3_10* - 1000 SLiMs, length : $3-10$, dataset: MW
- *SLiM_MW_1000_2_7* - 1000 SLiMs, length : $2-7$, dataset: MW
- *SLiM_MW_500_3_10* - 500 SLiMs, length : $3-10$, dataset: MW
- *SLiM_MW_500_2_7* - 500 SLiMs, length : $2-7$, dataset: MW

7.2.1 SLiMs for the ZH Dataset

Figure 7.5 shows the SLiM distribution of *SLiM_ZH_500_3_10* over the ZH obligate dataset. SLiMs that have sites in the ZH obligate complexes are considered. The *x*-axis represents the complexes and the *y*-axis represents the SLiMs found in the complexes. Complex 1 (1ahj A:B), complex 25 (1b9m A:B), complex 50 (1qfe A:B) and complex 75 (4mdh A:B) have a total of 15, 27, 20 and 45 SLiMs respectively.

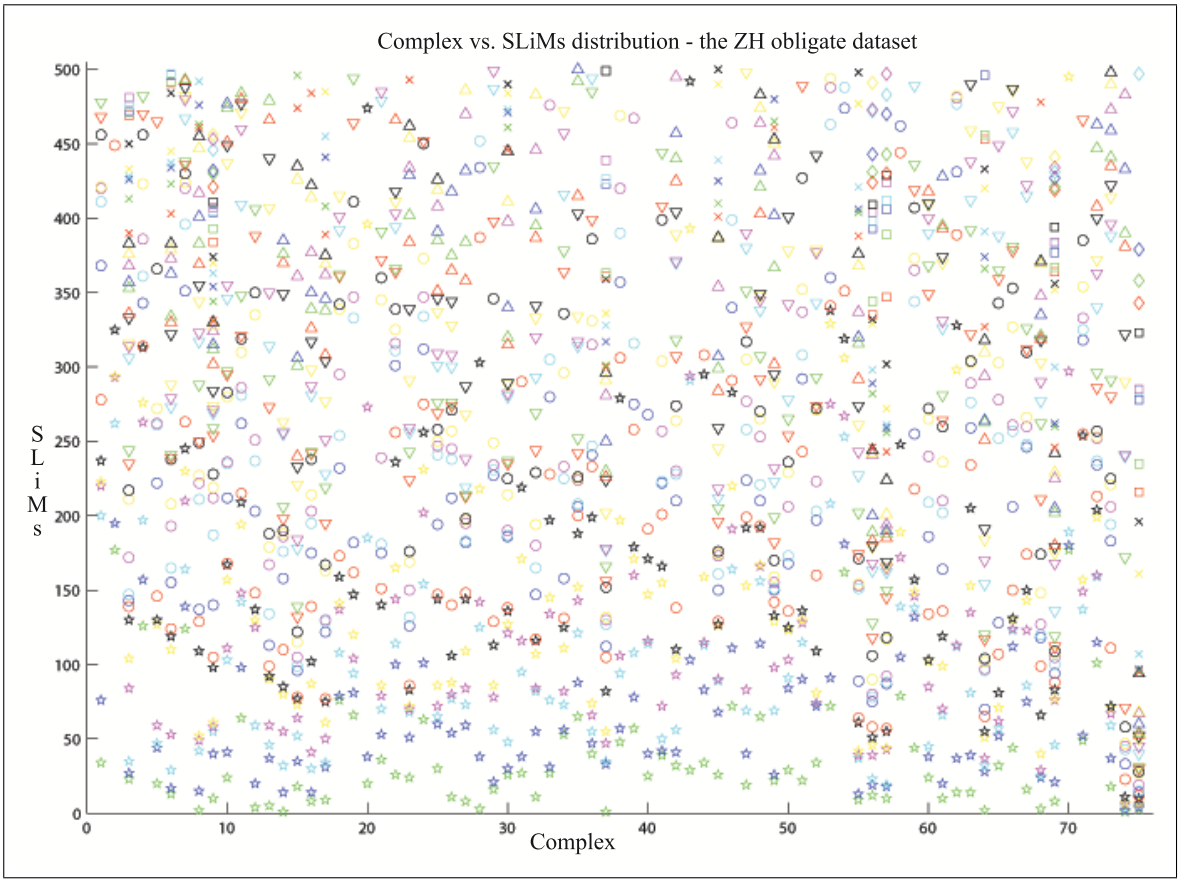


Figure 7.5: SLiM distribution in the ZH obligate dataset.

Figure 7.6 shows the SLiM distribution of *SLiM_ZH_500_3_10* over the ZH non-obligate dataset. SLiMs that have sites in the ZH non-obligate complexes are considered. In the ZH non-obligate dataset there are a total of 62 complexes. Complex 1 (1ava A:C), complex 25 (1kac A:B), complex 50 (1stf E:I) and complex 62 (1wq1 R:G) have a total of 56, 18, 23 and 26 SLiMs respectively.

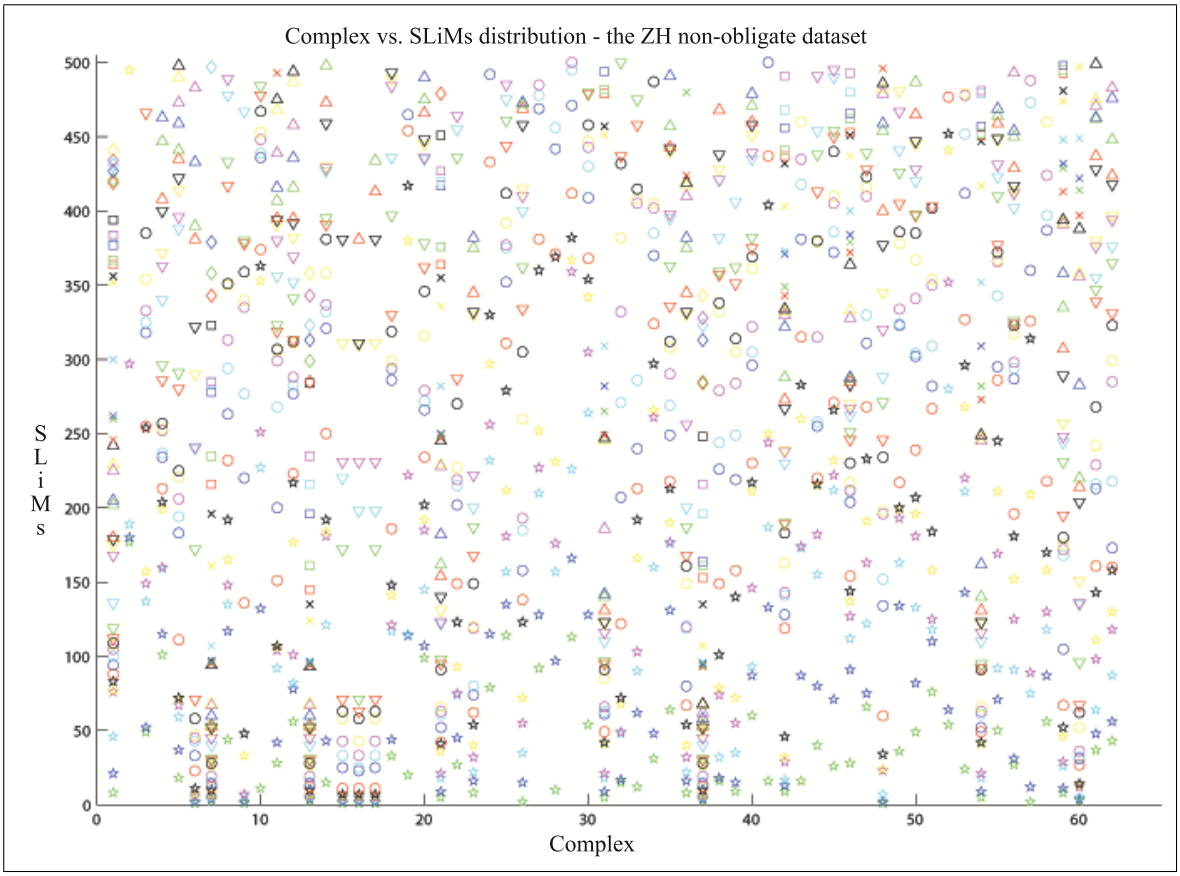


Figure 7.6: SLiM distribution in the ZH non-obligate dataset.

As the figure for the entire ZH dataset is larger, I show SLiM distribution separately, obligate and non-obligate. The full SLiM distribution of *SLiM_ZH_500_3_10* over the entire ZH dataset is included in Appendix C.

All the SLiMs of *SLiM_ZH_500_3_10* are distributed over 137 complexes (274 chains), a total of 4,286 sites. Figure 7.7 shows the number of the sites of SLiMs in *ZHSLiM3*. The *x*-axis shows the SLiMs and the *y*-axis shows the number of sites. The minimum number of sites is 8 and the maximum number of sites is 22 for *SLiM_ZH_500_3_10*. From the graph, it is clear that most SLiMs have 8 sites, and in average 8.58 sites. As the figure for all the SLiMs in *ZHSLiM3* is larger, I show only the first fifty SLiMs in this figure. The SLiM vs.

site number figure for all the SLiMs of *SLiM_ZH_500_3_10* is included in Appendix C.

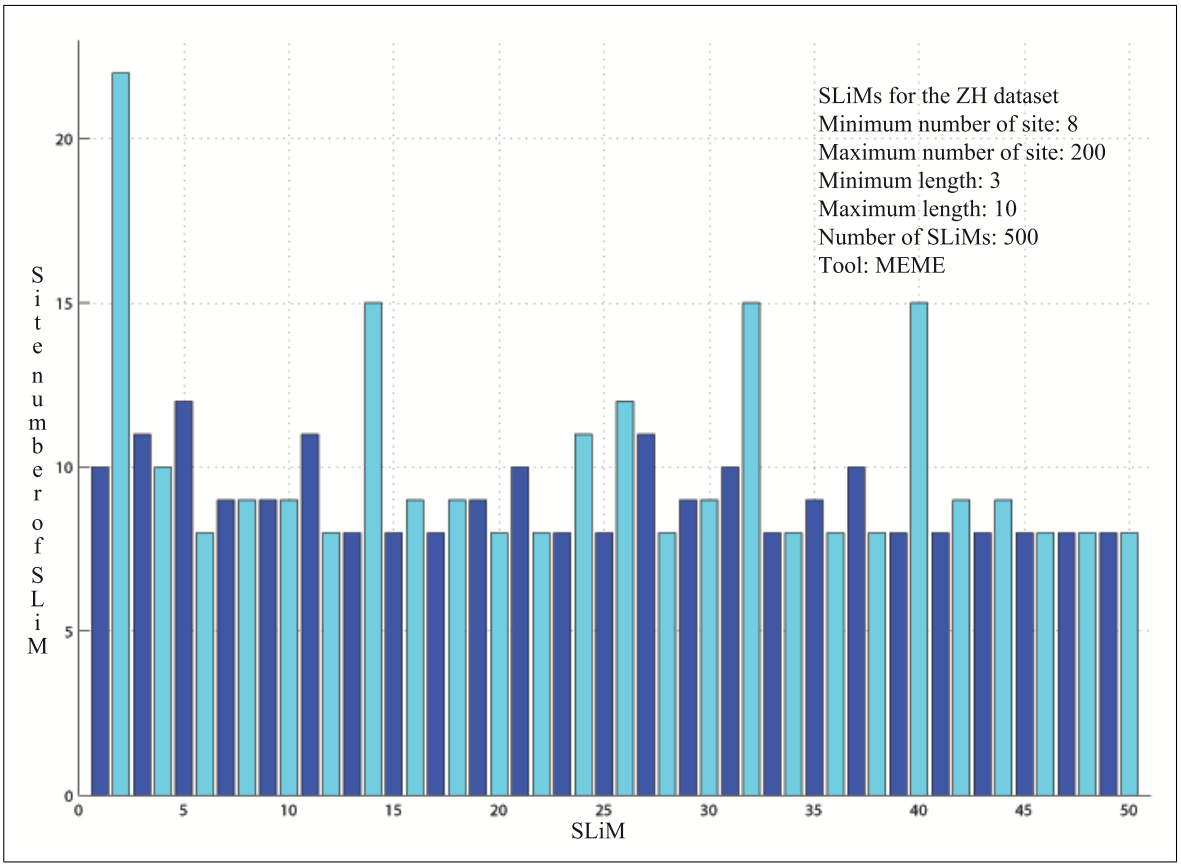


Figure 7.7: Distribution of the number of sites of SLiMs in *SLiM_ZH_500_3_10*.

In *SLiM_ZH_1000_3_10*, there are 1,000 SLiMs, a total of 8,298 sites, 5,052 of them are from the obligate complexes and 3,246 of them are from non-obligate complexes. Figure 7.8 shows the ratio of complex types holding SLiM sites of *SLiM_ZH_1000_3_10*. 61% of the sites are from obligate complexes and 39% from non-obligate. The minimum number of sites of SLiMs in *SLiM_ZH_1000_3_10* is 8 and the maximum is 22, which is similar to *SLiM_ZH_500_3_10*.

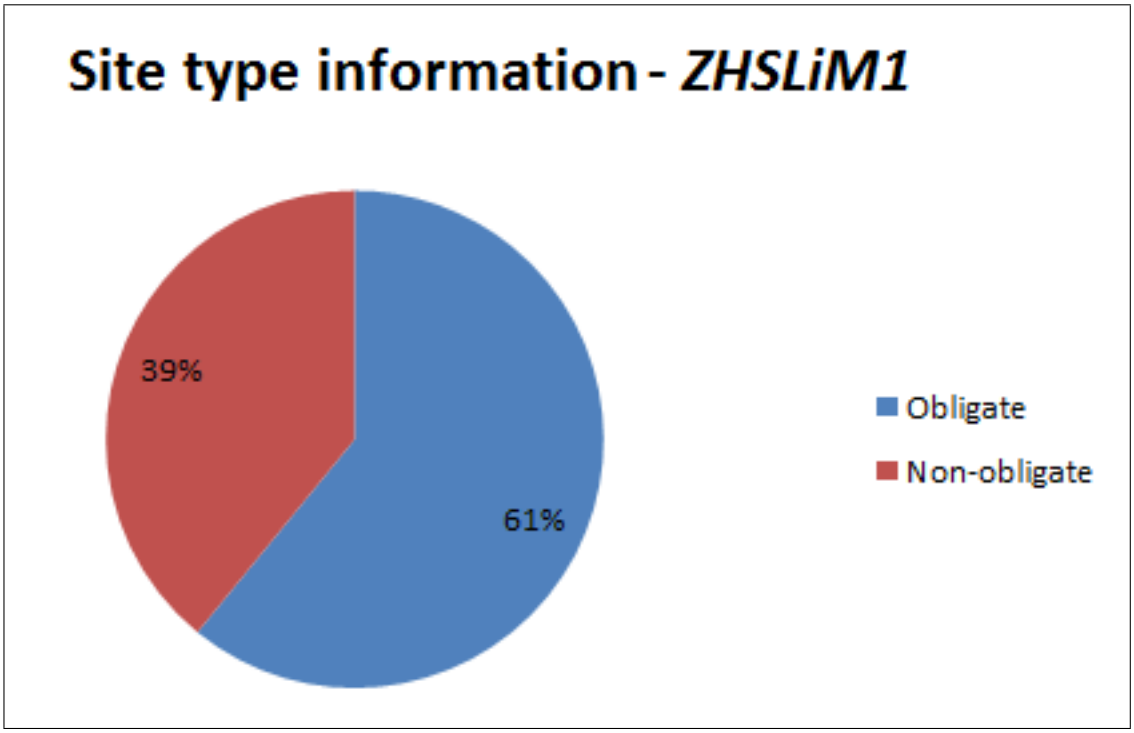


Figure 7.8: Types of the complexes holding sites of SLiMs in *SLiM_ZH_1000_3_10*.

7.2.2 SLiMs for the MW Dataset

Figure 7.9 shows the SLiM distribution of *SLiM_MW_500_3_10* over the MW obligate dataset. SLiMs that have sites in the MW obligate complexes are considered. The *x*-axis represents the complexes and the *y*-axis represents the SLiMs found in the complexes. Complex 1 (1b4u A:B) holds a total of 4 SLiMs, while complex 50 (1k8k A:B) has a total of 27 SLiMs.

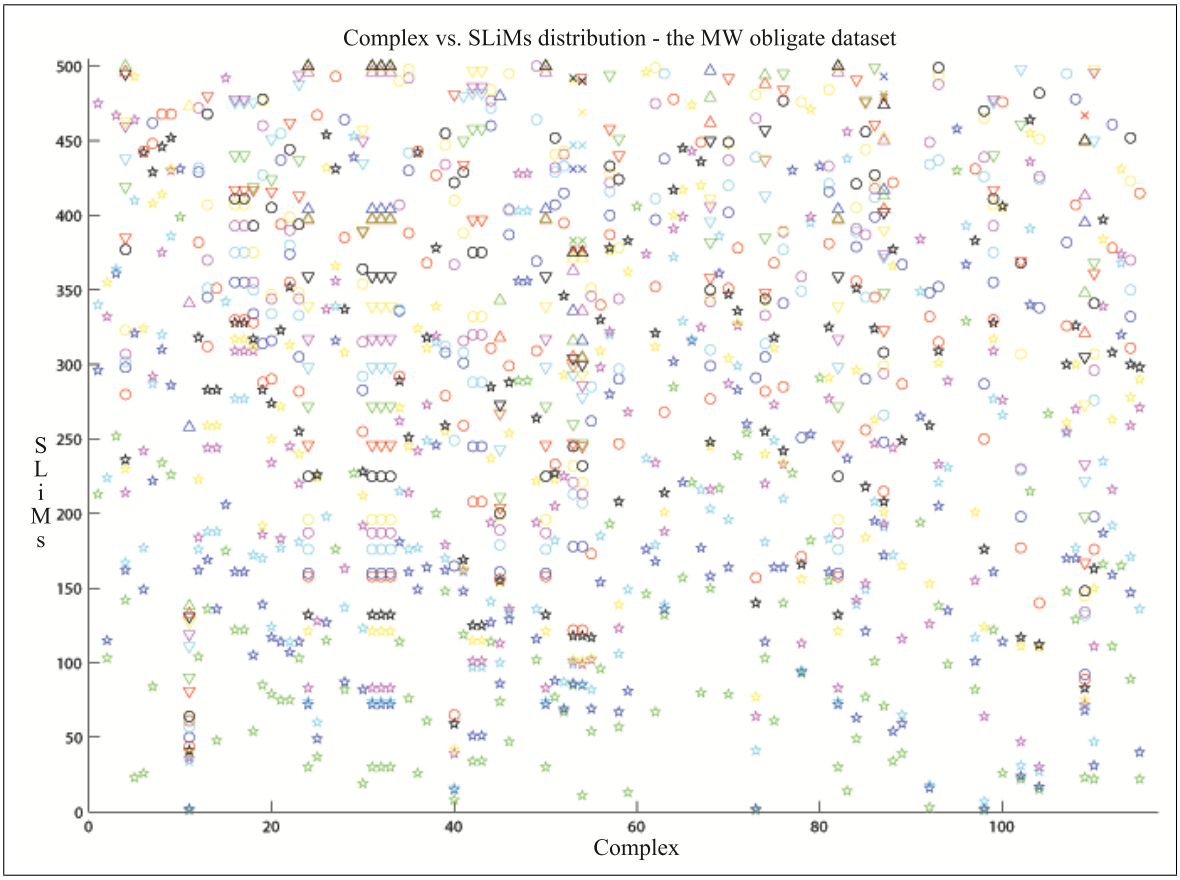


Figure 7.9: SLiM distribution in the MW obligate dataset.

Figure 7.10 shows the SLiM distribution of *SLiM_MW_500_3_10* over the MW non-obligate dataset. SLiMs that have sites in the MW non-obligate complexes are considered. In the MW non-obligate dataset there are a total of 212 complexes. Complex 1 (1wq1 G:R), complex 25 (1jma A:B), complex 50 (1bdj A:B), complex 100 (1e6e A:B), complex 200 (1mbu A:C) and complex (1mr1 A:D) have a total of 13, 32, 37, 23, 12 and 8 SLiMs respectively.

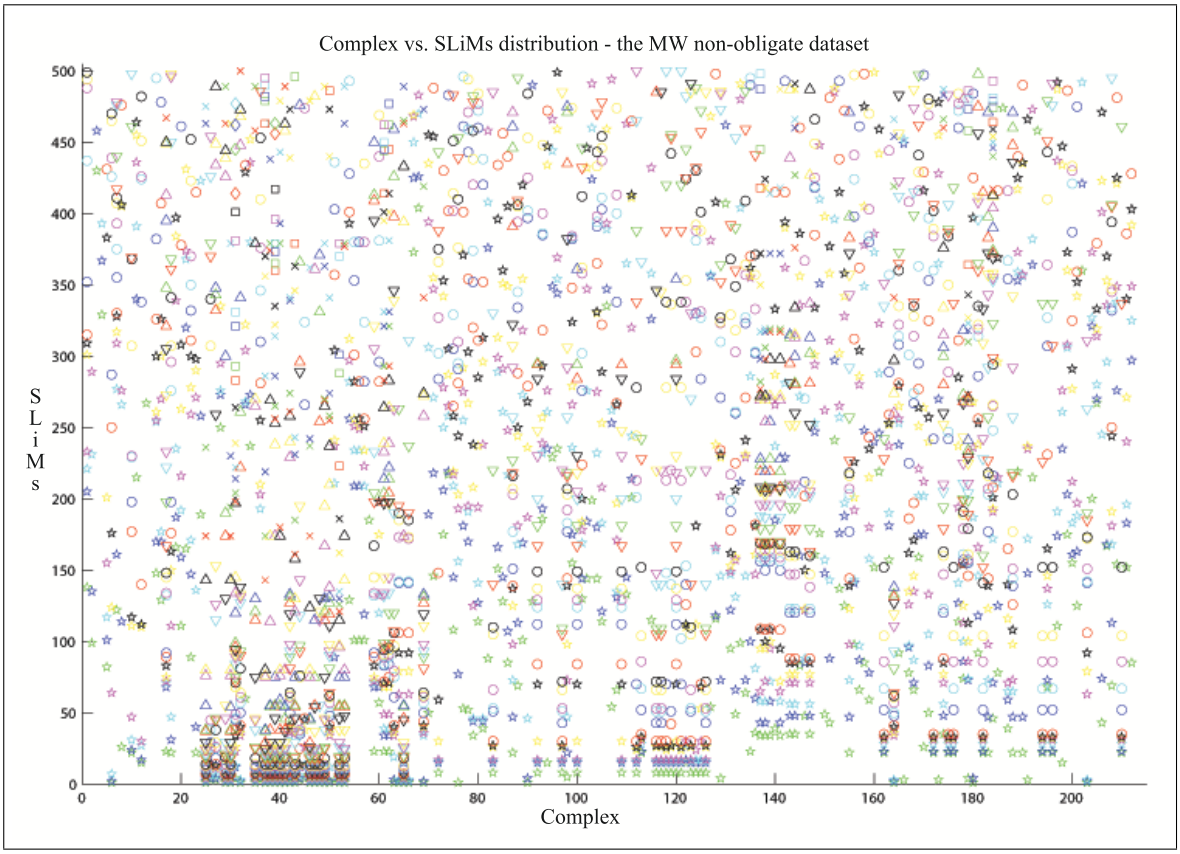


Figure 7.10: SLiM distribution in the MW non-obligate dataset.

As the figure for the entire MW dataset is larger, I show SLiM distribution separately, obligate and non-obligate. The full SLiM distribution of *SLiM_MW_500_3_10* over the entire MW dataset is included in Appendix C.

All the SLiMs in *SLiM_MW_500_3_10* are distributed over 327 complexes (more than 654 chains). Figure 7.11 shows the number of sites of SLiMs in *SLiM_MW_500_3_10*. The *x*-axis shows the SLiMs and the *y*-axis shows the site number. This set has a minimum site number of 8 and a maximum of 62. Most of the SLiMs have 8 sites and the average number of sites is 9.55. As the figure for all the SLiMs in *SLiM_MW_500_3_10* is larger, I show only the first fifty SLiMs in this figure. The SLiM vs. site number figure for all the SLiMs

of *SLiM_MW_500_3_10* is included in Appendix C.

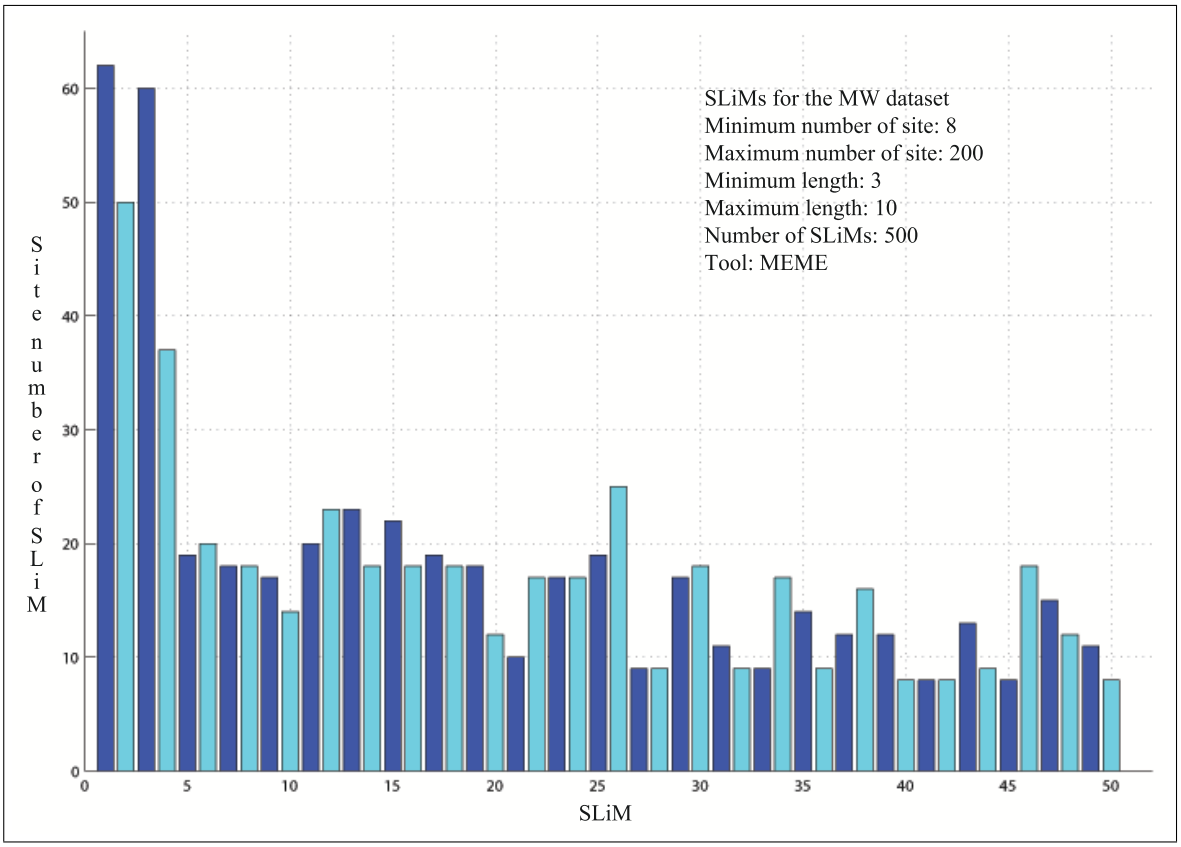


Figure 7.11: Distribution of number of sites of SLiMs in *SLiM_MW_500_3_10*.

In *SLiM_MW_1000_3_10*, there are 1,000 SLiMs and a total of 9,556 sites, 2,452 of them are obligate complexes and 7,104 of them are non-obligate. Figure 7.12 shows the ratio of complex types holding SLiM sites of *SLiM_MW_1000_3_10*. 26% of the sites are obligate complexes and 74% from non-obligate.

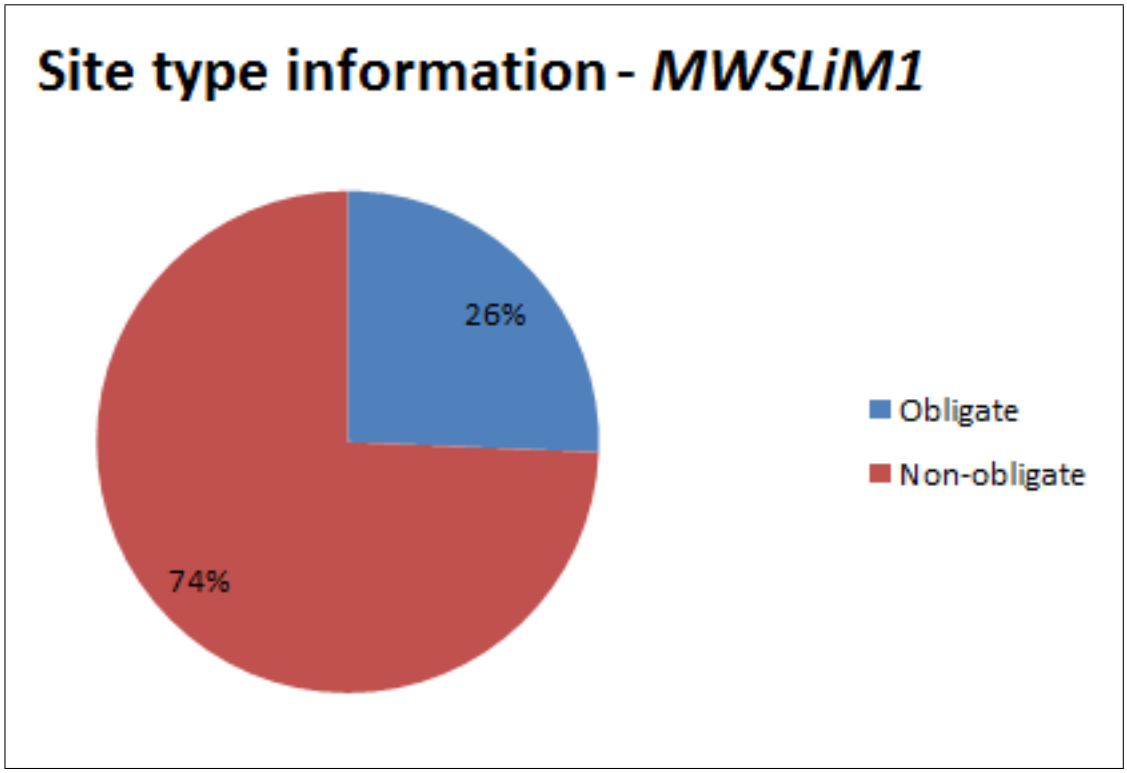


Figure 7.12: Types of the complexes holding sites of SLiMs in *SLiM_MW_1000_3_10*.

7.3 Experimental Results

In this thesis, I propose two different models and one new feature, sequence information. I also use desolvation and electrostatic energies. I use sequence information in PPI-SLiM-Seq and desolvation and electrostatic energies in PPI-SLiM-DEEE. To validate the performance of the models, I use leave-one-out with a k-NN, and cross dataset and 10-fold cross validation with LDR and SVM.

7.3.1 Results for PPI-SLiM-Seq

In PPI-SLiM-Seq, I use sequence information as features for prediction. The sequence information extraction procedure has been discussed in Chapter 3. I implement leave-one-out validation with a k-NN classifier. I also implement cross dataset validation to test the accuracy and significance of the newly proposed features. For cross dataset validation I also use LDR and SVM for classification. For SVM, I use the *linear kernel* and for LDR I use FDA, HDA and CDA with QB and LB classifiers. In a 10-fold cross validation process, reductions to dimensions $d = 1, 2, \ldots, 20$ were performed, followed by QB and LB. I use the datasets ZH and MW. For each dataset, classification accuracy is shown here.

7.3.1.1 The ZH Dataset

Table 7.1 shows the results of leave-one-out cross validation with k-NN. The first column of the table shows the SLiM set used for the results. The next column holds the SLiM length which is the same as the partition length (discussed in Chapter 6). The next columns show the classification results for different values of k. The values of $k = 1, 5, 10, 15, 20, 25, 30, 35$ have been chosen arbitrarily. I use the Euclidean distance as distance metric which is learned with the specialized large margin nearest neighborhood algorithm.

	Length	k-NN (k =)							
	(ℓ)	1	5	10	15	20	25	30	35
ZH SLiMs	10	96.35	95.62	97.08	97.08	97.08	97.08	97.81	98.54
$\ell = 3-10$	9	96.35	96.35	97.81	97.81	97.81	99.27	98.54	99.27
# of SLiMs	8	93.43	95.62	95.62	95.62	95.62	95.62	95.62	95.62
$= 1,000$	7	99.27	99.27	99.27	99.27	99.27	99.27	99.27	99.27
	6	99.27	99.27	99.27	99.27	98.54	98.54	98.54	96.35
	5	99.27	98.54	95.62	95.62	95.62	95.62	95.62	95.62
	4	99.27	97.81	98.54	98.54	98.54	99.27	99.27	99.27
	3	67.88	69.34	62.77	56.93	65.93	59.85	61.31	62.77
ZH SLiMs	7	98.35	98.35	98.35	98.35	98.35	98.35	98.35	98.35
$\ell = 2-7$	6	97.81	98.54	98.54	98.54	98.54	98.54	98.54	98.54
# of SLiMs	5	99.27	99.27	99.27	99.27	99.27	99.27	99.27	99.27
$= 1,000$	4	94.16	96.35	94.16	93.43	92.70	92.70	92.70	91.97
	3	93.43	92.70	92.70	91.97	91.97	91.97	91.24	91.24

Table 7.1: k-NN classification results for the ZH dataset using PPI-SLiM-Seq.

For $\ell = 10$, the highest accuracy is 98.54% for k= 35 and the lowest is 95.62% for k= 5. For $\ell = 9,7,6,5$, it achieves the highest accuracy of 99.27% and for $\ell = 8$ it achieves the highest accuracy of 95.62%. This table also shows that for a partition size of 3, it achieves lower accuracy. For $k = 5$, it achieves the highest accuracy of 69.34% and lowest accuracy of 56.93% is for $k = 15$. Again, for the other SLiM set, when $\ell = 3$, it achieves the lowest accuracy of 91.24%. For the ZH dataset, the highest accuracy is 99.27% for different values

of ℓ and k. For $\ell = 7$, all the values of k result in accuracy of 99.27%.

7.3.1.2 The MW Dataset

Table 7.2 shows the results of leave-one-out cross validation with a k-NN for the MW dataset. The first column of the table shows the SLiM set used for the results. The next column shows the SLiM length which is same as the partition length (discussed in Chapter 6). The next columns contains the classification results for different values of k.

	Length	k-NN ($k =$)							
	(ℓ)	1	5	10	15	20	25	30	35
MW SLiMs	10	98.15	98.77	98.77	98.46	98.46	98.46	98.46	98.46
$\ell = 3-10$	9	98.77	99.07	99.07	99.07	99.07	99.07	99.07	99.07
# of SLiMs	8	98.15	99.07	99.07	99.07	99.07	99.07	99.07	99.07
$= 1,000$	7	92.63	96.31	96.31	96.31	96.31	96.31	96.31	96.31
	6	96.62	96.62	96.31	96.01	96.01	96.01	96.01	96.01
	5	97.54	99.07	98.15	98.46	99.07	99.07	99.07	99.07
	4	96.01	97.54	96.62	96.31	95.70	95.70	95.39	95.09
	3	98.46	99.07	99.07	98.77	98.77	98.77	98.77	98.77
MW SLiMs	7	97.54	98.46	97.85	97.85	97.85	97.85	97.85	97.85
$\ell = 2-7$	6	96.31	96.31	96.62	96.62	96.62	96.62	96.62	96.62
# of SLiMs	5	98.77	98.46	98.46	98.46	98.46	98.46	98.15	98.15
$= 1,000$	4	97.54	99.07	99.07	99.07	99.07	99.07	99.07	99.07
	3	55.21	58.28	64.41	61.34	55.82	65.64	63.80	63.49

Table 7.2: k-NN classification results for the MW dataset using PPI-SLiM-Seq.

The values of $k = 1,5,10,15,20,25,30,35$ have been chosen arbitrarily. I use the Euclidean distance metric and large margin nearest neighborhood algorithm for k-NN. For $\ell = 10$, the highest accuracy is 98.77% for k= 5,10 and the lowest is 98.46% for k= 15,20,25,30,35. For $\ell = 9,8,5$, it yields the highest accuracy of 99.07% and yields the lowest accuracy of 95.70% for $\ell = 4$, k= 20. For the other set $\ell = 3$ it gives the lowest accuracy of 55.21% for k= 1.

7.3.2 Cross-dataset Validation of PPI-SLiM-Seq

To validate the results achieved by PPI-SLiM-Seq, I also implemented the cross dataset validation method. I use the two datasets, ZH and MW, for all the experiments. To perform the cross dataset validation, I use the SLiMs of the MW dataset for training with the ZH dataset for testing and vice versa. Using the cross dataset validation, PPI-SLiM-Seq gives almost the same accuracy using different classifiers.

	Length (ℓ)	SVM	LDR					
			Quadratic			Linear		
			FDA	HDA	CDA	FDA	HDA	CDA
ZH dataset	5	95.62	95.62	97.81	96.35	93.43	97.08	96.35
MW SLiMs	4	97.81	97.08	98.54	99.27	97.81	97.81	97.81
MW dataset	6	98.77	98.77	98.77	98.77	98.16	98.47	98.47
ZH SLiMs	5	98.47	99.08	99.08	99.08	98.47	98.47	98.47

Table 7.3: SVM and LDR classification results for the ZH and MW datasets with the MW and ZH SLiMs respectively.

Table 7.3 shows the results of cross dataset validation. The first column briefly describes the dataset with SLiM, length column is for the partition size or SLiM length. The SVM column holds the classification accuracy obtained by SVM and the remaining columns are for different LDR criteria. I use the ZH SLiMs for training with the MW dataset for $\ell = 5, 6$ and achieve almost the same accuracy. The MW SLiMs for training with the ZH dataset for $\ell = 4, 5$ also yields almost the same results. I chose the values of ℓ experimentally.

Cross dataset validation yields an accuracy of 99.27% and 97.81% for $\ell = 4, 5$ respectively using SVM and different LDR for the ZH dataset with the MW SLiMs for training. For the same values of ℓ, leave-one-out with a k-NN gives an accuracy of 99.27% (for k= 1) and 99.27% (for k= 1, 25, 30, 35) respectively. Again, for the MW dataset with the ZH SLiMs for training cross dataset validation gives an accuracy of 98.77% and 99.08% for $\ell = 6, 5$ respectively using SVM and different LDR, while for the same values of ℓ, leave-one-out with a k-NN yields an accuracy of 96.62% (for k= 1, 5) and 99.07% (for k= 20, 25, 30, 35) respectively.

7.3.3 PPI-SLiM-DEEE Results

For PPI-SLiM-DEEE, I use desolvation and electrostatic energies as features. Calculation of desolvstion and electrostatic energies has been described in Chapter 3. For the LDR schemes, six different classifiers were implemented and evaluated, namely the combinations of FDA, HDA and CDA with QB and LB classifiers. In a 10-fold cross validation process, reductions to dimensions $d = 1, 2, \ldots, 20$ were performed, followed by QB and LB. I use the two datasets, ZH and MW. For each dataset, classification accuracy is described here.

7.3.3.1 The ZH dataset

Tables 7.4 and 7.5 show the classification results achieved by PPI-SLiM-DEEE for the ZH dataset. I use SLiM-protein interactions (discussed in Chapter 6) to produce the atom-atom type feature vectors. The first column shows the energy used as features. The second column contains the SLiM lengths. I used two different sets of SLiM lengths, $\ell = 3-10$ and $\ell = 2-7$. Table 7.4 shows the results considering $1,000$ SLiMs and Table 7.5 shows the results considering 500 SLiMs from the same ZH dataset. The next column shows the accuracy obtained by SVM and all the remaining columns are for different LDR.

	# of SLiMs = 1000 ZH dataset	SVM	LDR					
			Quadratic			Linear		
			FDA	HDA	CDA	FDA	HDA	CDA
Desolvation	$\ell = 3-10$	67.69	59.23	62.31	60.00	59.23	61.54	<u>70.77</u>
Energy	$\ell = 2-7$	63.85	48.46	55.38	56.15	49.23	51.54	<u>68.46</u>
Electrostatic	$\ell = 3-10$	<u>77.69</u>	53.08	60.77	66.92	57.69	55.38	71.54
Energy	$\ell = 2-7$	<u>76.15</u>	46.92	54.62	67.69	49.23	51.54	72.31

Table 7.4: Classification results for the ZH dataset using desolvation and electrostatic energies with SVM and LDR. $1,000$ SLiMs for two different sets of lengths (ℓ) are considered.

Table 7.4 shows that PPI-SLiM-DEEE yields a maximum accuracy of 70.77% for $\ell = 3-10$ using desolvation energy with linear CDA. It gives a maximum accuracy of 77.69% for $\ell = 3-10$ using electrostatic energy with SVM. In both cases, $1,000$ SLiMs from the same ZH dataset were considered.

	# of SLiMs = 500 ZH dataset	SVM	LDR					
			Quadratic			Linear		
			FDA	HDA	CDA	FDA	HDA	CDA
Desolvation Energy	$\ell = 3-10$	67.69	55.38	58.46	56.15	52.31	53.08	78.46
	$\ell = 2-7$	63.08	47.69	56.15	56.92	53.08	53.08	65.38
Electrostatic Energy	$\ell = 3-10$	73.85	52.31	62.31	72.31	54.62	54.62	79.23
	$\ell = 2-7$	71.54	56.92	58.46	67.69	56.15	57.69	73.85

Table 7.5: Classification results for the ZH dataset using desolvation and electrostatic energies with SVM and LDR. 500 SLiMs for two different sets of lengths (ℓ) are considered.

Table 7.5 shows that PPI-SLiM-DEEE yields a maximum accuracy of 78.46% for $\ell = 3-10$ using desolvation energy with linear CDA. It gives a maximum accuracy of 79.23% for $\ell = 3-10$ using electrostatic energy linear CDA. In both cases, 500 SLiMs from the same ZH dataset were considered.

7.3.3.2 The MW dataset

Tables 7.6 and 7.7 show the classification results achieved by PPI-SLiM-DEEE, using the MW dataset. I use SLiM-protein interactions (discussed in Chapter 6) to produce the atom-atom type feature vectors.

	# of SLiMs = 1000 MW dataset	SVM	LDR					
			Quadratic			Linear		
			FDA	HDA	CDA	FDA	HDA	CDA
Desolvation	$\ell = 3-10$	60.20	52.17	66.22	61.54	53.51	67.56	<u>68.56</u>
Energy	$\ell = 2-7$	68.72	49.79	69.55	60.49	53.50	69.55	<u>69.14</u>
Electrostatic	$\ell = 3-10$	65.31	60.52	68.27	67.90	62.36	67.90	<u>69.37</u>
Energy	$\ell = 2-7$	66.22	51.51	66.22	<u>68.23</u>	61.54	66.89	67.22

Table 7.6: Classification results for the MW dataset using desolvation and electrostatic energies with SVM and LDR. $1,000$ SLiMs for two different sets of lengths (ℓ) are considered.

	# of SLiMs = 500 MW dataset	SVM	LDR					
			Quadratic			Linear		
			FDA	HDA	CDA	FDA	HDA	CDA
Desolvation	$\ell = 3-10$	61.87	58.19	70.23	<u>70.90</u>	58.53	68.90	70.23
Energy	$\ell = 2-7$	62.54	62.88	66.89	61.20	61.87	66.89	<u>67.89</u>
Electrostatic	$\ell = 3-10$	66.89	59.53	72.24	73.24	62.21	71.91	<u>73.91</u>
Energy	$\ell = 2-7$	66.89	64.21	72.24	68.90	65.89	<u>74.58</u>	74.58

Table 7.7: Classification results for the MW dataset using desolvation and electrostatic energies with SVM and LDR. 500 SLiMs for two different sets of lengths (ℓ) are considered.

The first column shows the type of energy used as features. The second column shows the SLiM lengths. I used two different sets of SLiM lengths, $\ell = 3-10$ and $\ell = 2-7$. Table

7.6 shows the results for considering $1,000$ SLiMs, while Table 7.7 shows the results for considering 500 SLiMs from the same MW dataset. The next column shows the accuracy obtained by SVM and all the remaining columns are for different LDR criteria.

Table 7.6 shows that PPI-SLiM-DEEE yields a maximum accuracy of 69.14% for $\ell = 2-7$ using desolvation energy with linear CDA and 69.37% for $\ell = 3-10$ using electrostatic energy with linear CDA. Again, Table 7.7 shows that PPI-SLiM-DEEE gives a maximum accuracy of 70.90% for $\ell = 3-10$ using desolvation energy with quadratic CDA and 74.58% for $\ell = 2-7$ using electrostatic energy with linear HDA.

7.4 Result Comparison

In [54], Zhu *et al.* predicted obligate and non-obligate complexes with 70.07% accuracy. They used four NOXclass features with two stage SVM to achieve that performance. Again the authors of [2] claim that they achieved maximum accuracy of 82.13% with LDR (HDA, FDA, CDA) combined with quadratic Bayesian (QB) and linear Bayesian (LB). Thus, they had over 12% improved result for this case. To do a fair comparison, they also used the same features with the prediction methods by Zhu *et al.* [54] and it achieved maximum accuracy of 77.92% which is still lower than the accuracy of their approach by 4%.

Vasudev and Rueda predicted obligate and non-obligate complexes with an accuracy of 96.17% [49] which is over 14% improved result than the results by Aziz *et al.* [2] and over 26% improved than the results by Zhu *et al.* [54]. In this thesis, I use information contained in sequence as features to achieve maximum accuracy of 99.27% using leave-one-out with a *k*-NN, which is over 17% more than the results in [2] and over 3% improved than the results by Vasudev and Rueda [49]. I use different lengths of sequences ($\ell = 10,9,8,7,6,5,4,3$) to calculate the information values. For the classification, I use Eucledian distance and

different values of k, $k = 1, 5, 10, 15, 20, 25, 30, 35$.

	NOXClass Feature + SVM	NOXClass feature + LDR	Approach by Aziz *et al.* [2]	Approach by Vasudev and Rueda [49]	Sequence Information with k-NN
Zhu *et al.* [54]	70.07	77.92	82.13	96.17	99.27
Mintseris *et al.* [34]	77.64	77.32	80.86	97.36	99.07

Table 7.8: Comparison of classification accuracy with other related works.

With the same four NOXclass features, Mintseris *et al.* [34] achieved 77.64% accuracy with an optimized SVM. The authors of [2] achieved 80.86% accuracy which is over 3% improvement from results in [34]. The authors of [2] also tested those features for [34] with their prediction method, in which they achieved 77.32% accuracy. Using sequence information I achieve a maximum accuracy of 99.07%, which is over 19% improved results than the results in [2]. Table 7.8 shows a comparison between the results of PPI-SLiM-Seq and the results obtained by Zhu *et al.* [54], Mintseris *et al.* [34], Aziz *et al.* [2] and Vasudev and Rueda [49]. For the ZH dataset, the model achieves an accuracy of 99.27% which is about 29% higher than the accuracy obtained by Zhu *et al.* [54]. Accuracy by the PPI-SLiM-Seq, for the ZH dataset is about 17% higher than the accuracy obtained by Aziz *et al.* [2] and about 3% higher than the accuracy obtained by Vasudev and Rueda [49]. Again, PPI-SLiM-Seq yields an accuracy of 99.07% for the MW dataset which is

about 22% higher than the accuracy obtained by Mintseris *et al.* [34]. For the MW dataset, accuracy achieved by the PPI-SLiM-Seq is about 19% higher than the accuracy obtained by Aziz *et al.* [2] and 2% higher than the accuracy obtained by Vasudev and Rueda [49].

For a fair comparison, I also implemented cross dataset validation with SVM and LDR. To perform the cross dataset validation, I use SLiMs from the MW dataset to calculate the sequence information of the ZH dataset and vice versa. Table 7.9 shows that the cross dataset validation with SVM yields over 15% improved accuracy than the results by Aziz *et al.* [2] and with LDR it yields over 17% improved accuracy. It gives over 1% and 3% improved accuracy than the results by Vasudev and Rueda [49] using SVM and LDR respectively.

	NOXClass Feature + SVM	NOXClass feature + LDR	Approach by Aziz *et al.* [2]	Approach by Vasudev and Rueda [49]	Cross dataset validation + SVM/LDR
Zhu *et al.* [54]	70.07	77.92	82.13	96.17	97.81/99.27
Mintseris *et al.* [34]	77.64	77.32	80.86	97.36	98.77/99.08

Table 7.9: Accuracy comparison of cross dataset validation with other related works.

Cross dataset validation yields over 18% and 19% improved accuracy than the results by Aziz *et al.* [2] with SVM and LDR respectively using the ZH SLiMs with the MW dataset. It also yields accuracies 1% and 2% higher than that of the work by Vasudev and

Rueda [49]. For cross dataset validation, I use two values of ℓ, $\ell = 5,4$ for the ZH dataset and $\ell = 6,5$ for the MW dataset. These values of ℓ were found experimentally.

7.5 PPI-SLiM-Seq vs. PPI-SLiM-DEEE

As discussed earlier, the PPI-SLiM-Seq model yields very good accuracy which is much higher than the accuracy by the other model, PPI-SLiM-DEEE. For the ZH dataset, PPI-SLiM-Seq achieves an accuracy of 99.27% using information contained in the sequences while PPI-SLiM-DEEE yields an accuracy of 79.23% using electrostatic energy. PPI-SLiM-Seq achieves an accuracy of 99.07% for the MW dataset while PPI-SLiM-DEEE yields 74.58% accuracy using electrostatic energy. In both cases, PPI-SLiM-Seq results in better accuracy.

PPI-SLiM-DEEE uses desolvation and electrostatic energies as distinguishing features. Desolvation and electrostatic energies are based on atomic distance and interface area. The distances between two atoms are calculated using three coordinates (X, Y, and Z) of the atoms. Thus, distance calculation for all possible atom-atom pairs requires long time. If a protein contains a total of n atoms, there will be $\frac{n\times(n-1)}{2}$ possible unique atom-atom pairs. Calculating distances for such a high number of atom-atom pairs runs in $O(\frac{n^2}{2})$ time which makes the model time consuming.

The Protein Data Bank (PDB) [6] is the main source of the structures of protein complexes. The text format of the structure file which is mostly used is not perfect. It usually contains a lot of spacing errors, numbering errors, alignment issues. In many cases the PDB does not provide complete structural information of protein complexes. All these issues may lead to wrong result. Also, processing all the information of a protein complex at the atomic level is also time consuming and computationally expensive.

PPI-SLiM-Seq uses information contained in the sequences. Sequence information values are calculated using the sequences and PSPMs. Calculating sequence information values is simple and much faster than calculating desolvation or electrostatic energies as tomic level information of protein atoms are not required. In this way, PPI-SLiM-Seq avoids the usage of the PDB structure files. As a result, the performance of the PPI-SLiM-Seq is not affected by the errors in the PDB structure files.

Structure files are available for approximately 80,000 protein complexes while there are millions of sequences available. The number of available structure files limits the scope of the PPI-SLiM-DEEE as it is based on the structural information of protein complexes. PPI-SLiM-Seq uses sequence information values and hence, has a wider scope than the PPI-SLiM-DEEE.

Considering all the issues, it is clear that the performance of the PPI-SLiM-Seq model is better than the performance of PPI-SLiM-DEEE. The accuracy achieved by the PPI-SLiM-Seq is also higher than the accuracy by PPI-SLiM-DEEE. PPI-SLiM-Seq can be used for millions of proteins while PPI-SLiM-DEEE is limited only to 80,000 proteins.

Chapter 8

Conclusion

8.1 Summary of Contributions

In this thesis, I present two new models to predict obligate and non-obligate protein complexes. In both models, I use SLiMs to generate the features for classification. The key contributions of the thesis can be summarized as follows:

- The PPI-SLiM-Seq model presented in this thesis results in significant improvements in the ZH and MW datasets for predicting obligate and non-obligate complexes with information contained in sequences as properties and leave-one-out with a k-NN.
- I also introduce a new feature, information contained in sequence, which can be calculated only from sequence data using PSPMs. It does not need to use the protein structure file.
- Corss dataset validation with SVM and LDR also shows a better performance, which indicates the significance of sequence information values for predicting obligate and non-obligate complexes.

- The other model, PPI-SLiM-DEEE uses desolvation and electrostatic energies as features with SVM and LDR, and achieves higher accuracy than that of Zhu *et al.* [54], Mintseris *et al.* [34] and close to the accuracy by Aziz *et al.* [2].
- A visual analysis of SLiMs shows how the regions are conserved over a sequence dataset and how they are important in PPI. Another visual analysis of site numbers of SLiMs provides a very good insight about how the sites of SLiMs are distributed in different protein chains.

8.2 Limitations

In PPI-SLiM-Seq, for the leave-one-out method, I do not use well-known classifiers such as SVM and LDR although I used SVM and LDR for cross dataset validation. Again, in PPI-SLiM-DEEE, I use SLiMs to calculate the features. SLiMs do not cover the entire interface of a complex. Thus, I exclude a potential amount of interface area which is important in PPI. However, the advantage of using sequence information only is that the method can be applied even when protein structures are unknown.

8.3 Future Work

I use the SLiMs identified by MEME where there are other SLiM identification tools available. One can use other SLiM identification tools in order to compare the results. Again, I use a maximum sequence partition of length 10, which can be extended to a partition size up to 20. While identifying SLiMs from the sequence datasets, I always use ranges of lengths, such as $3-10$, $2-7$ and ranges of site numbers such as $8-200$. Different parameters can be defined in different combinations to identify different SLiM sets. Fixed length of SLiMs

and fixed site number can be another approach in SLiM identification. Again, in PPI-SLiM-DEEE, I use only SLiMs. One can use motifs and domains in PPI-SLiM-DEEE instead of SLiMs, or combine sequence information with different physiochemical properties or other evolutionary properties for generating features. All these can be summarized as follows:

- Sets of SLiMs of different lengths, numbers and site number can be defined and used.
- Different combinations of other MEME parameters can be used to define different sets of SLiMs.
- Other SLiM identification tools can be used in order to find different sets of SLiMs.
- Extend the partition size up to 20 while varying the SLiM lengths.
- Longer motifs or shorter domains of different lengths can be used to calculate the information contained in sequences.
- Combination of sequence information with other evolutionary information can be used as properties.
- A way to use the combination of sequence information with different physiochemical properties can also be investigated.

Appendix A

Discovering SLiMs using MEME

Finding SLiMs using MEME is easy enough and straightforward. MEME offers SLiM finding services both as online web server and also via installation of software packages. The standard version can be configured as serial or parallel. The performance of the parallel version depends on the processors available and is faster than the serial version.

A.1 MEME Web Service (Online)

The online version of MEME can be used at *http://meme.sdsc.edu/meme/intro.html* or *http://meme.nbcr.net/meme/intro.html*. This version of MEME is very user friendly but limited in terms of defining different parameters as per the requirements. It only allows the user to define the maximum and minimum width of a motif, maximum number of motifs and maximum and minimum number of sites. It does not support maximum number of motifs more than 100 and can not handle a sequence dataset of more than 60,000 characters. It shows the output online and also it emails the output to the email address provided by the user.

A.2 Installing the Standard version

The installation version of MEME can be configured as per the requirements. The latest source code of this version is available at *http://meme.nbcr.net/downloads/meme_4.8.1.tar.gz*. I configured it to support a maximum number of motifs $= 2,000$, a dataset of 200,000 characters, and minimum width of a motif $= 2$. It can also be configured either as a serial or parallel process. As the serial process is time consuming, I configured the parallel one. For a comparison between serial and parallel MEME it can be said that to find 100 motifs (of 3-10 width) from a dataset of 60,000 characters, serial MEME takes about 24 hours while parallel MEME takes about 5 hours given that 8 processors run at the same time.

Currently, MEME can be installed only on Linux given that some prerequisite software and packages/libraries are already installed or configured. The prerequisite software packages are as follows:

1. Perl (v.5.10 or higher)
    - 1.1 HTML::PullParser
    - 1.2 HTML::Template
    - 1.3 LWP
    - 1.4 SOAP::Lite
    - 1.5 XML::Simple
2. Python (v2.6 up to v2.7.x)
3. Bourne compatible shell
4. C Compiler

5. GZip/gunzip utilities
6. ImageMagick
7. Libxml2
8. Libxslt
9. MPICH (for the parallel version)

A.3 Installing Serial MEME

The installation of the serial version of MEME is straightforward and can be accomplished using a few commands:

$ *tar zxf meme_VERSION.tar.gz*

$ *cd meme_VERSION*

$ *./configure --prefix=$home/meme*

$ *make*

$ *make test*

$ *make install*

As mentioned above, I configured the parallel version of MEME.

A.4 Installing Parallel MEME

A.4.1 Configuring the Source Code

The default installation of MEME outputs a maximum of 100 motifs. To configure MEME for a higher number of motifs, the MAXG parameter needs to be defined based on the requirements (in our case 2000) in the *user.h* file (at line #20) which is located at the /src directory. By default, MEME does not find the SLiMs. The default minimum width of a motif is defined as 8. Thus, to find motifs with shorter lengths (e.g. 2, 3, 5) the MIN_W parameter (at line #31) needs to be defined in the same *user.h* file, which I set to 2. Having all these changes, MEME can be used to discover SLiMs from a sequence dataset.

A.4.2 Installation of the Parallel Version

Installing the parallel version of MEME is not that much difficult. MPICH package, which supports parallelism, needs to be configured correctly. Once MPICH is configured and working properly, just an additional parameter needs to be passed. The commands to install the parallel version of MEME are as follows:

$ *tar zxf meme_VERSION.tar.gz*

$ *cd meme_VERSION*

$ *./configure −−prefix = $home/meme −−with − mpicc = /usr/bin/mpicc*

$ *make*

$ *make test*

$ *make install*

The $ *make test* command checks the installation whether or not it is correct. There are

three major tests, if it passes all the tests, then the installation is successful.

A.4.3 MEME on Sharcnet

Shared Hierarchical Academic Research Computing Network (SHARCNET) was formally established in 2001. SHARCNET is the largest high performance computing consortium in Canada, including 17 universities, colleges and research institutes across South western, Central and Northern Ontario. Currently, the grid architecture of the SHARCNET facility is serving all the researchers with more that 10,000 processors and 1 PB of storage. One user can use 32 processors at the same time (e.g., on Hound Cluster) and can use 120 GB of physical memory.

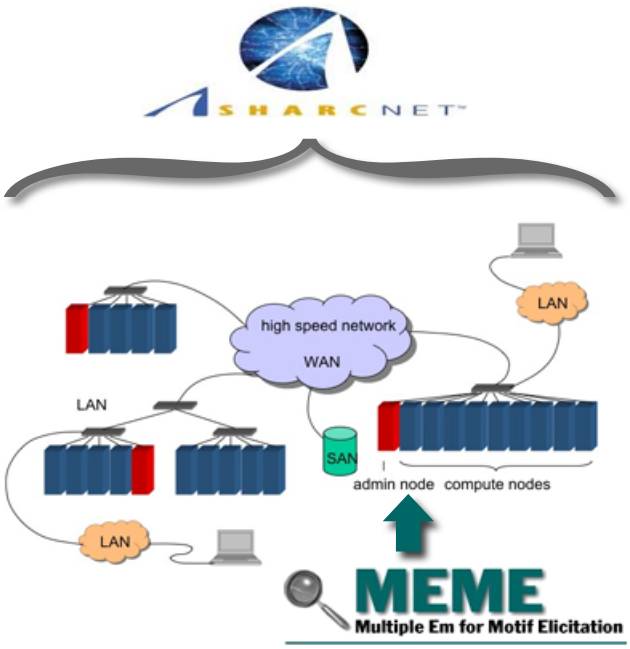


Figure A.1: Schematic diagram of MEME on SHARCNET.

As finding SLiMs from a sequence dataset using MEME is an iterative process, it needs longer time and enough computation power. It also needs large amounts of physical memory for larger sequence datasets. To meet this requirement, I deployed and configured MEME on SHARCNET. It showed a great performance by finding 2000 motifs within 48 hours using 20 processors and 40 GB of physical memory. In this way, I was able to run MEME multiple times for different datasets, with different parameter settings.

A.5 Inputs to MEME

All the inputs to MEME can be categorized into two categories; one is the input dataset and the other is the parameters.

A.5.1 FASTA Format Dataset

MEME takes a *FASTA* format sequence file as input. The input file may contain the sequence for a single protein complex or a set of sequences for different protein complexes. A sample *FASTA* sequence file for protein complex *Cytoplasmic Malate Dehydrogenase* (4mdh) is the following:

4mdh:A

XSEPIRVLVTGAAGQIAYSLLYSIGNGSVFGKDQPIILVLLDITPMMGVLDGVLMELQDCALPLLKDVIATDKEEIA

FKDLDVAILVGSMPRRDGMERKDLLKANVKIFKCQGAALDKYAKKSVKVIVVGNPANTNCLTASKSAPSIPKENFSC

LTRLDHNRAKAQIALKLGVTSDDVKNVIIWGNHSSTQYPDVNHAKVKLQAKEVGVYEAVKDDSWLKGEFITTVQQRG

AAVIKARKLSSAMSAAKAICDHVRDIWFGTPEGEFVSMGIISDGNSYGVPDDLLYSFPVTIKDKTWKIVEGLPINDF

SREKMDLTAKELAEEKETAFEFLSSA

4mdh:B

```
XSEPIRVLVTGAAGQIAYSLLYSIGNGSVFGKDQPIILVLLDITPMMGVLDGVLMELQDCALPLLKDVIATDKEEIA
FKDLDVAILVGSMPRRDGMERKDLLKANVKIFKCQGAALDKYAKKSVKVIVVGNPANTNCLTASKSAPSIPKENFSC
LTRLDHNRAKAQIALKLGVTSDDVKNVIIWGNHSSTQYPDVNHAKVKLQAKEVGVYEAVKDDSWLKGEFITTVQQRG
AAVIKARKLSSAMSAAKAICDHVRDIWFGTPEGEFVSMGIISDGNSYGVPDDLLYSFPVTIKDKTWKIVEGLPINDF
SREKMDLTAKELAEEKETAFEFLSSA
```

A.5.2 Defining the Parameters

To run MEME for the expected output, the parameters need to be defined accordingly. Among all the supported parameters, the dataset parameter is the only one required and the others are optional. MEME allows defining lots of parameters [4], but I only used the following:

- dataset - file containing sequences in FASTA format
- -h - print this message
- -o - name of directory for output files will not replace existing directory
- -oc - name of directory for output files will replace existing directory
- -dna - sequences use DNA alphabet
- -protein - sequences use protein alphabet
- -mod [oops, zoops, anr] specify any one
- -nmotifs nmotifs - maximum number of motifs to find
- -nsites nsites - number of sites for each motif

- -minsites minsites - minimum number of sites for each motif
- -maxsites maxsites - maximum number of sites for each motif
- -w w - motif width
- -minw minw - minumum motif width
- -maxw maxw - maximum motif width
- -maxiter maxiter - maximum EM iterations to run
- -psp psp - name of positional priors file
- -maxsize maxsize - maximum dataset size in characters
- -p np - use parallel version with n processors

A.6 Output from MEME

MEME delivers four different output formats for motif information; HTML, XML, TEXT and sequence logo. Among all the formats, HTML has the format for visualization along with all other information.

- **HTML**: In the HTML output, all the motifs and corresponding motif information are graphically presented in an HTML file. It contains the regular expression, start point, end point, all the sites, site containing chains and sequence logo of each motif. It also shows the block representation of all the sites on the different chains.

- **TEXT**: It shows regular expressions, start point, end point, and site information for a motif in text format. It does not incorporate the sequence logo and block diagram of the sites.

- **XML**: XML format is good for precise parsing. It shows regular expression, start point, end point, and site information for a motif in text format. It does not incorporate the sequence logo and block diagram of the sites.

- **Sequence Logo**: The sequence logo is a graphical representation of the protein motifs. It shows how well are amino acids conserved at each position. Different amino acids at the same position are scaled based on the information content of that amino acid.

Appendix B

Source Code Explanation

Most of the modules of the models are developed in Python. All the source code used in this thesis is available at *http://sanjuan.cs.uwindsor.ca/ppipaw/download/SourceCode/manish/PPI-SLiM-Seq.zip* and *http://sanjuan.cs.uwindsor.ca/ppipaw/download/SourceCode/manish/PPI-SLiM-DEEE.zip*.

Here, I explain all the source code (input, output and parameters) briefly. For classification, Matlab implementations of k-NN, SVM and LDR have been used. Different SLiM sets are available and can be downloaded at *http://sanjuan.cs.uwindsor.ca/ppipaw/download/SLiMs*.

B.1 Generic Modules

The modules under the generic module section are used in both models. Other modules are model-specific and can not be used for other models.

B.1.1 File Downloader

The main purpose of the *file downloader* module is to download a file from a specified server (given that the server permits downloading). I developed and used this module to download the PDB and sequence files from the PDB server. It needs a list of file names and a destination folder where it will save the downloaded files. The *main* function calls the *fileDownloader(listFileName, targetFolder)* function with the provided inputs. In the source code, the server address can be changed easily by changing the value of the ***lnk*** variable inside the *fileDownloader(listFileName, targetFolder)* function.

B.1.2 Sequence Dataset Compiler

This module merges protein sequences from different sequence files into a single file, which is known as sequence dataset/database in this thesis. This module takes two inputs, one is the list of complexes with chains and the other is the path to the protein sequence files. Before using this module, it should be assured that the path provided has sequence files for all the complexes in the provided list. The name of the function that needs to be called is *dbCompiler(path, listFile)*. This module outputs a sequence dataset which contains sequences for all the chains of all the complexes separated by new line. I use this compiled sequence dataset as an input to MEME to find SLiMs.

B.1.3 XML Parser

MEME outputs the SLiM information in XML, TEXT and HTML format. I use the XML format and this is why I needed to develop this module to parse the XML file and get the SLiM information. The name of the function is *parseBookXML(xmlFile, textFilePath)* which takes two inputs. One is the XML file and the other is the path in which SLiM

information is to be stored. This module is slightly modified and used to parse the PSPMs from the XML files.

B.2 PPI-SLiM-Seq Modules

These modules are developed particularly for the PPI-SLiM-Seq model and will not work with other models such as PPI-SLiM-DEEE. The source code is available to download at *http://sanjuan.cs.uwindsor.ca/ppipaw/download/SourceCode/manish/PPI-SLiM-Seq.zip*.

B.2.1 Sequence Partitioning

This module partitions a sequence into all overlapping sub-sequences of specified length. This module takes a sequence and an integer number as input. Then, it divides the sequence into sub-sequences of provided lengths (integer number) and outputs all the overlapping sub-sequences of that length.

B.2.2 Information Calculator

This module calculates the information contained in a sequence using a profile. This module take one sequence and one profile file as input and outputs the information contained in that sequence.

B.2.3 Main Module

The main module coordinates all other modules to accomplish the goal. First of all, it compiles the sequence dataset by calling *dbCompiler(path, listFile)*, then for each sequence, it calls the sequence partitioning module, *seqSplitter(STR, LEN)*. For each parti-

tion, it selects all possible PSPMs for that partitioned sequence using the *selectMotifProfileFiles(PDB, CH, LEN, filePath)* function. It calls the information calculator module, *gramScoring(SubStr, Profile)*.

For a sub-sequence, it calculates the information values using all the PSPMs and selects the best one. And for a sequence it selects the top-20 sub-sequence information values and writes them in a text file which is used as the feature matrix.

B.3 PPI-SLiM-DEEE Modules

These modules are developed particularly for the PPI-SLiM-DEEE model and do not work with other models such as PPI-SLiM-Seq. All the source code is available for downloading at *http://sanjuan.cs.uwindsor.ca/ppipaw/download/SourceCode/manish/PPI-SLiM-DEEE.zip*.

B.3.1 Chain Separator

The PDB file of a complex contains structural information of all the chains that belong to that complex. This module is used to separate the structure information of the chains from the PDB file. For example, protein complex 1a4y has two chains, A and B. The PDB file, *1a4y.pdb* has the information for both chains. This module separates the information and generate new files for the chains, *1a4y_A.pdb* and *1a4y_B.pdb*. This module, *chainSeparator(fileName, pdbFilePath)*, takes two inputs, one is the PDB file name and the other is path of the PDB file. It generates the chain-separated PDB files and saves them in the same path.

B.3.2 Distance Calculator

This module is used to calculate the distance between two chains of a complex. This module, *distanceCalculator(fileOneName, fileTwoName, distanceFile, pdbFilePath, destinationFolder, th)*, takes six inputs and outputs one distance file. First, two inputs are the chain-separated PDB files of the complex, *distanceFile* is the name of the distance file, *pdbFilePath* is the path to the PDB files, *destinationFolder* is the path in which the distance file is to be saved and *th* is the distance threshold. If the distance between two atoms from two chains is less than or equal to *th*, the atoms are of interest and the atomic information along with the distance are saved in the distance file.

B.3.3 SLiM Atom Selector

This module selects all the atoms that belong to a SLiM. The module *allAtomSelect(PDB_ID, CH_ID, ST_POS, END_POS, filePath)* takes five inputs, where *PDB_ID*, *CH_ID* and *ST_POS* come with the SLiM information. *END_POS* is determined by adding the length of the SLiM to the *ST_POS* and the *filePath* is the PDB file of the complex. It outputs a list of atoms belonging to that SLiM.

B.3.4 Interface SLiM Checker

This module checks whether a SLiM is in the interface or not. This module takes a SLiM as input and outputs *TRUE* is the SLiM is in the interface, otherwise it outputs *FALSE*. When it takes a SLiM, that SLiM has the complex and chain names, start position of the sites of the SLiM. For each site, it selects all the atoms using the *allAtomSelect(PDB_ID, CH_ID, ST_POS, END_POS, filePath)* function. Then, it checks if any of the atoms are present in the distance file and determines if the SLiM is in the interface or not.

B.3.5 Desolvation Energy Calculator

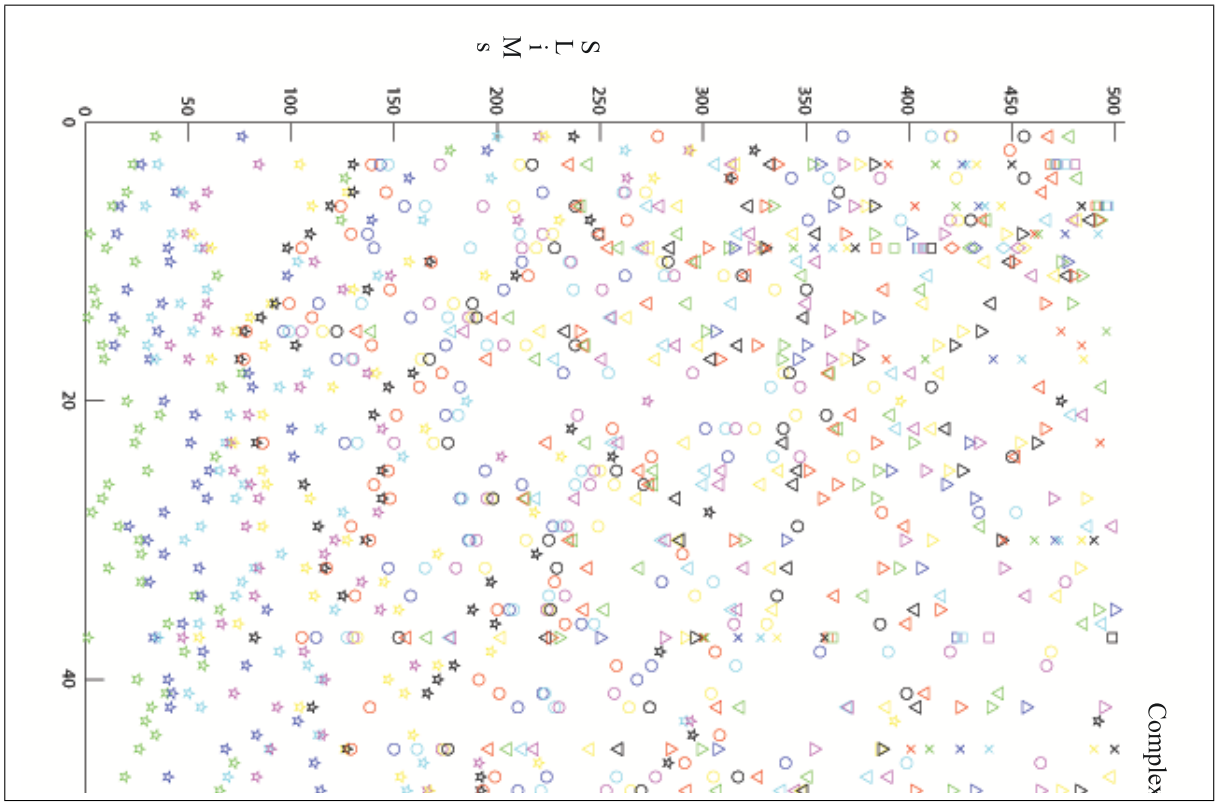
This modules calculates desolvation energies using the ACP matrix, saved in a text file. It takes the atomic distance from the distance files, generates the value of the smooth function and outputs the desolvation energy which is the product of the ACP matrix value and a smooth function.

B.3.6 Electrostatic Energy Calculator

Using two different software packages (PDB2PQR and APBS), atomic electrostatic energies are calculated and saved in a text file. This module uses those text files to obtain the electrostatic energies for the specific atoms.

Appendix C

SLiM Distribution



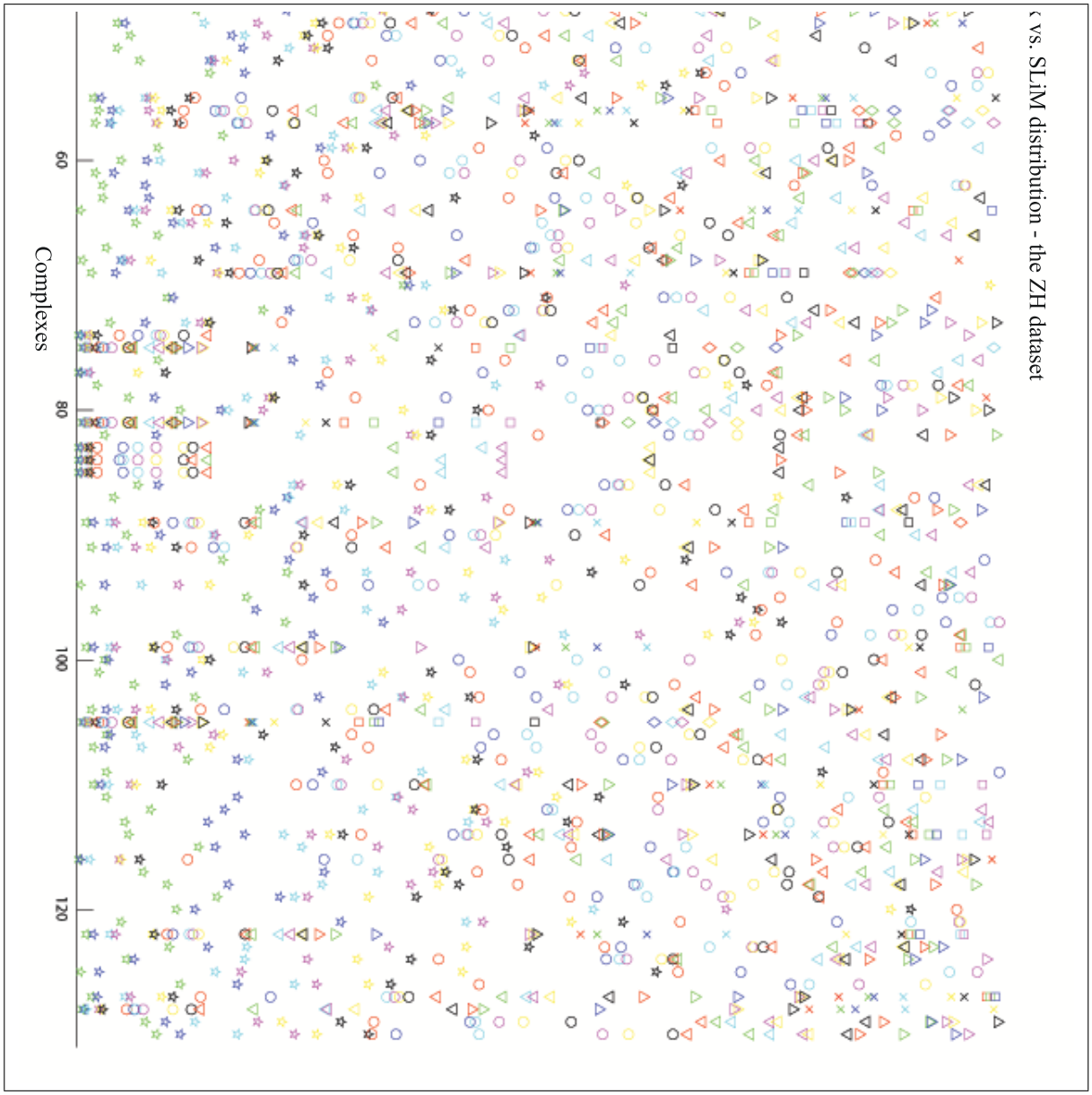
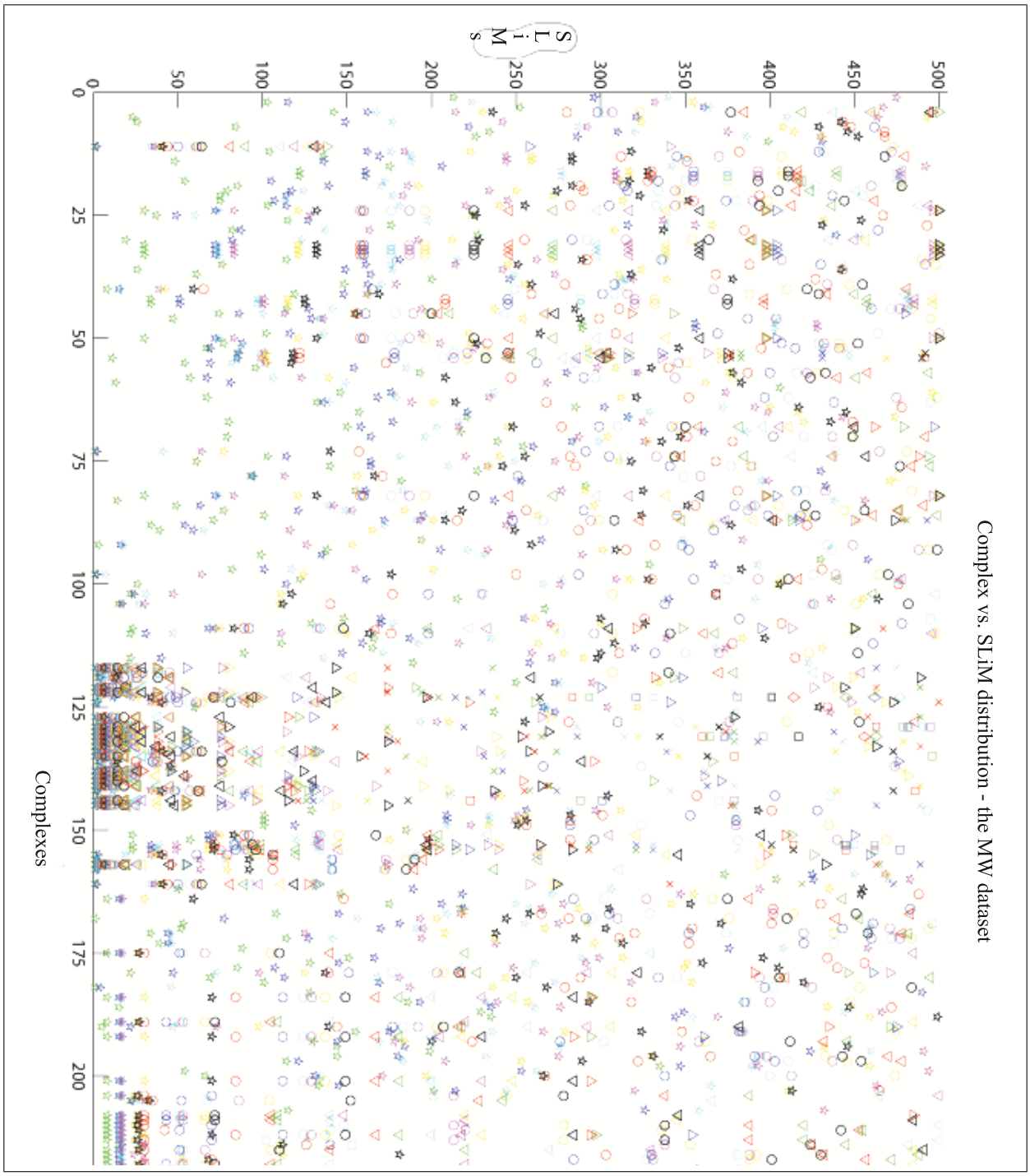


Figure C.1: SLiM distribution in the ZH dataset. 500 SLiMs of length $3-10$ were identified using MEME.


Complex vs. SLiM distribution - the MW dataset
SLiMs
Complexes
0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300
350
400
450
500

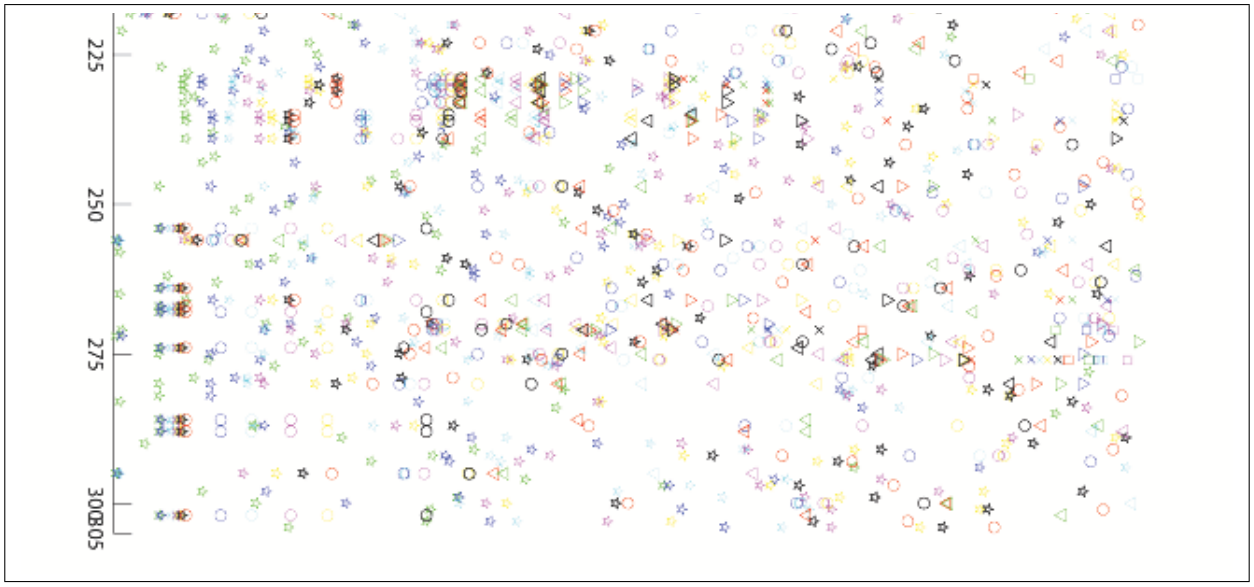
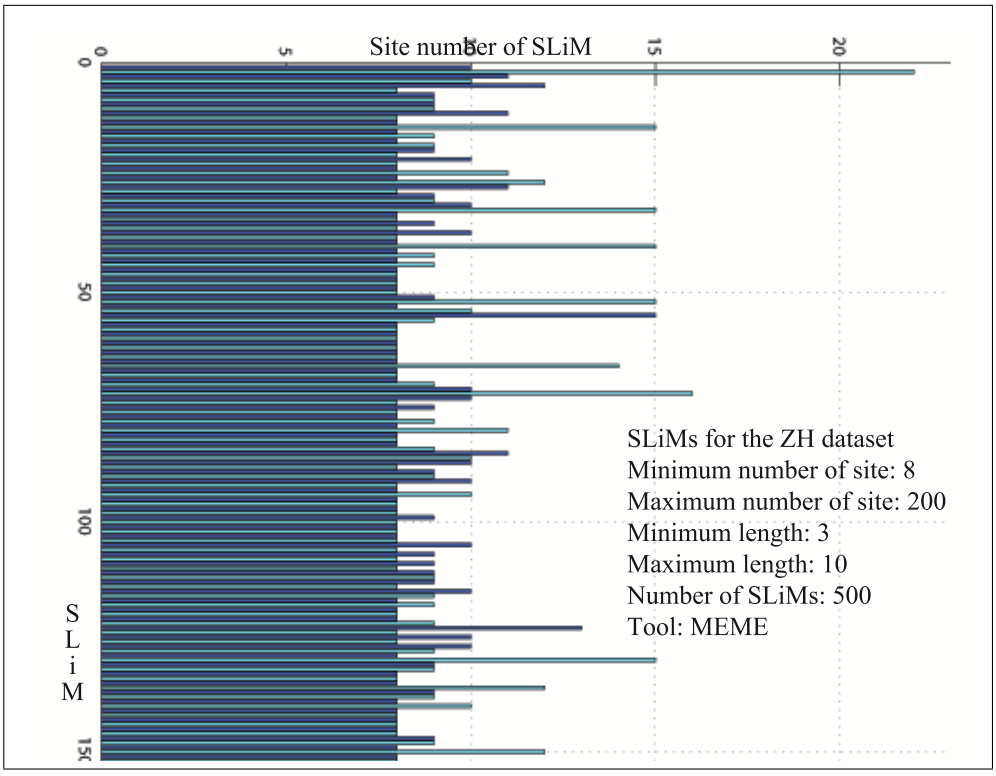


Figure C.2: SLiM distribution in the MW dataset. 500 SLiMs of length 3 – 10 were identified using MEME.



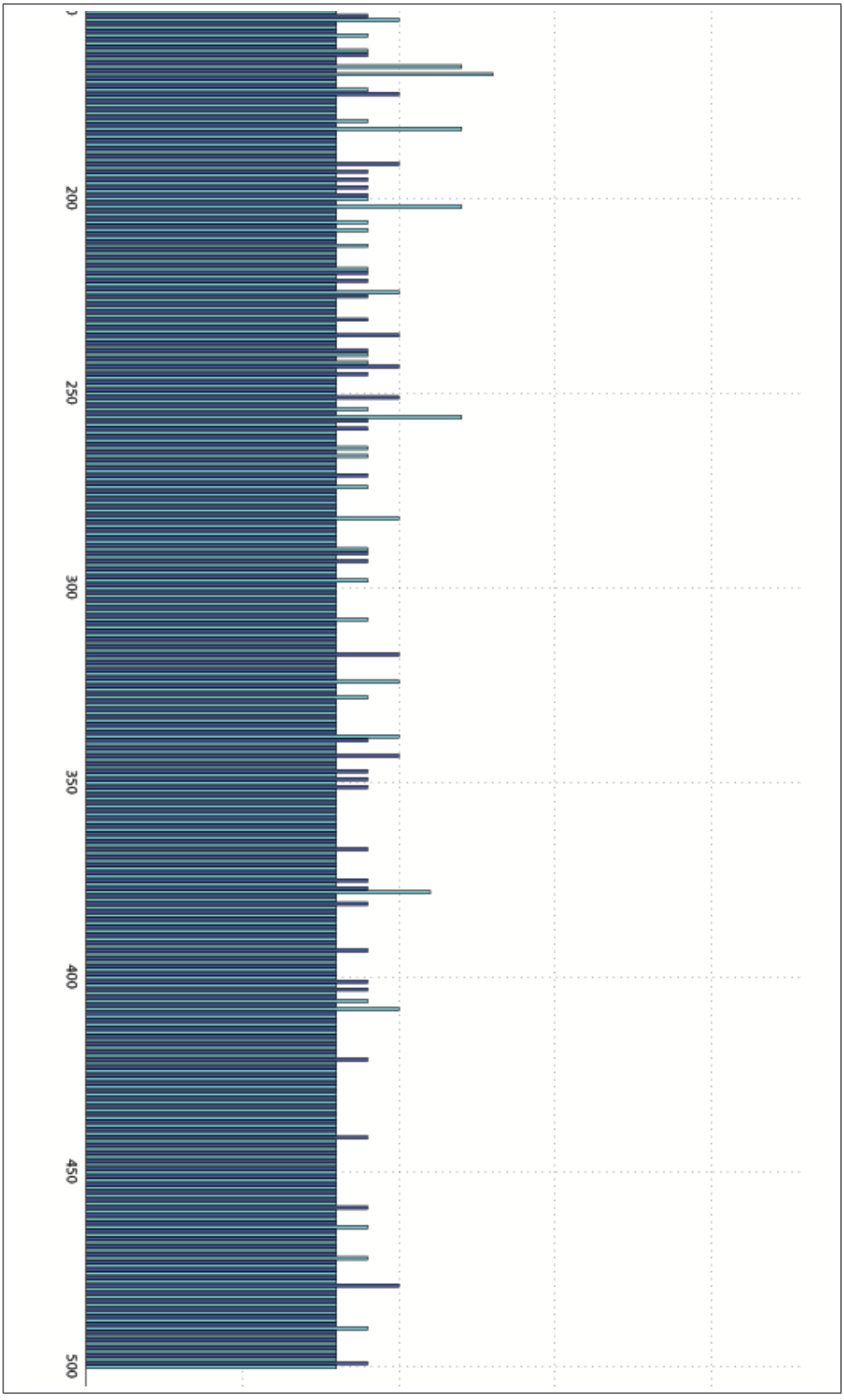
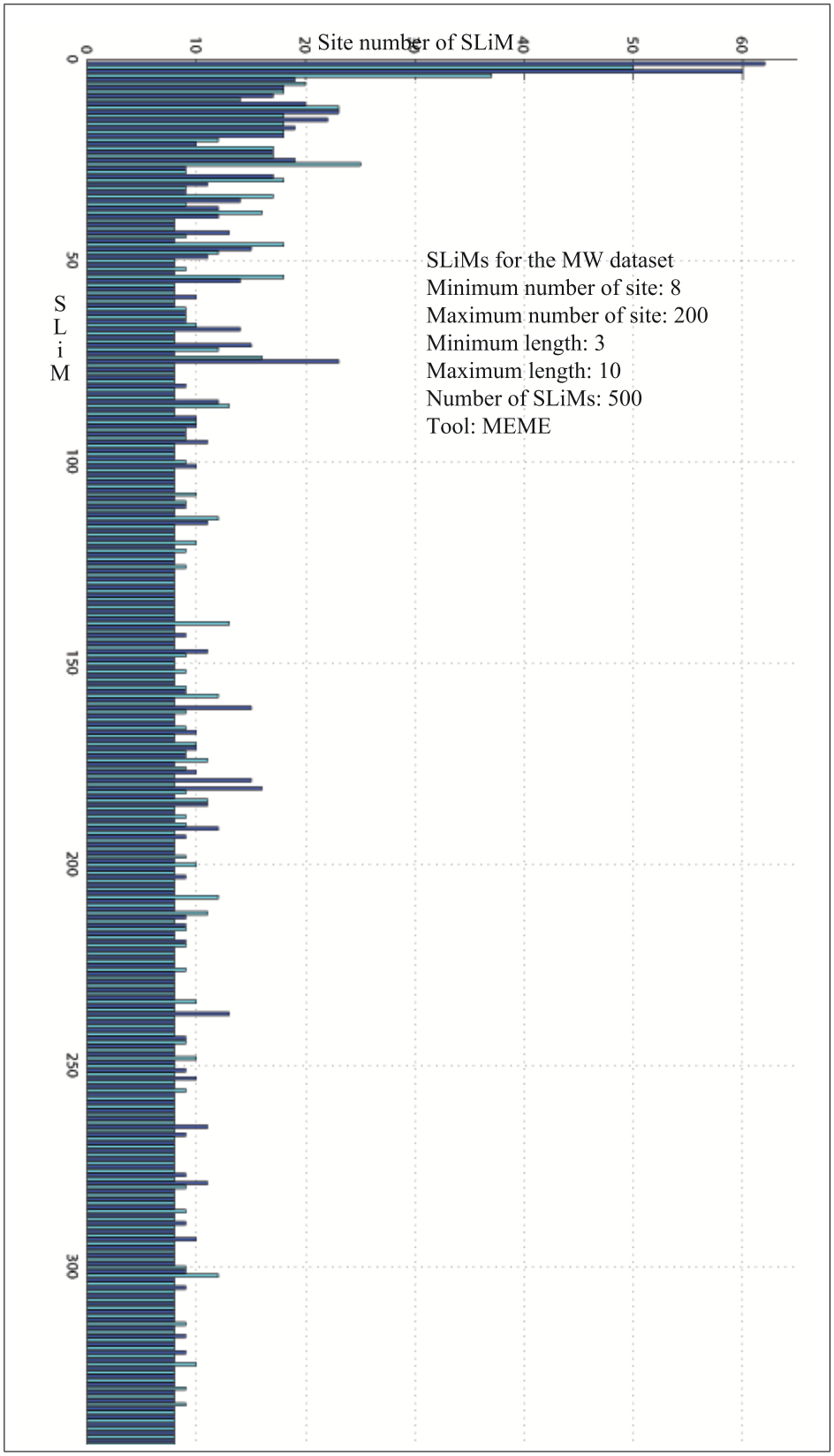


Figure C.3: Distribution of the number of sites of SLiMs in *SLiM_ZH_500_3_10*. 500 SLiMs of length $3-10$ were identified from the ZH dataset using MEME.


Site number of SLiM
SLiM
SLiMs for the MW dataset
Minimum number of site: 8
Maximum number of site: 200
Minimum length: 3
Maximum length: 10
Number of SLiMs: 500
Tool: MEME

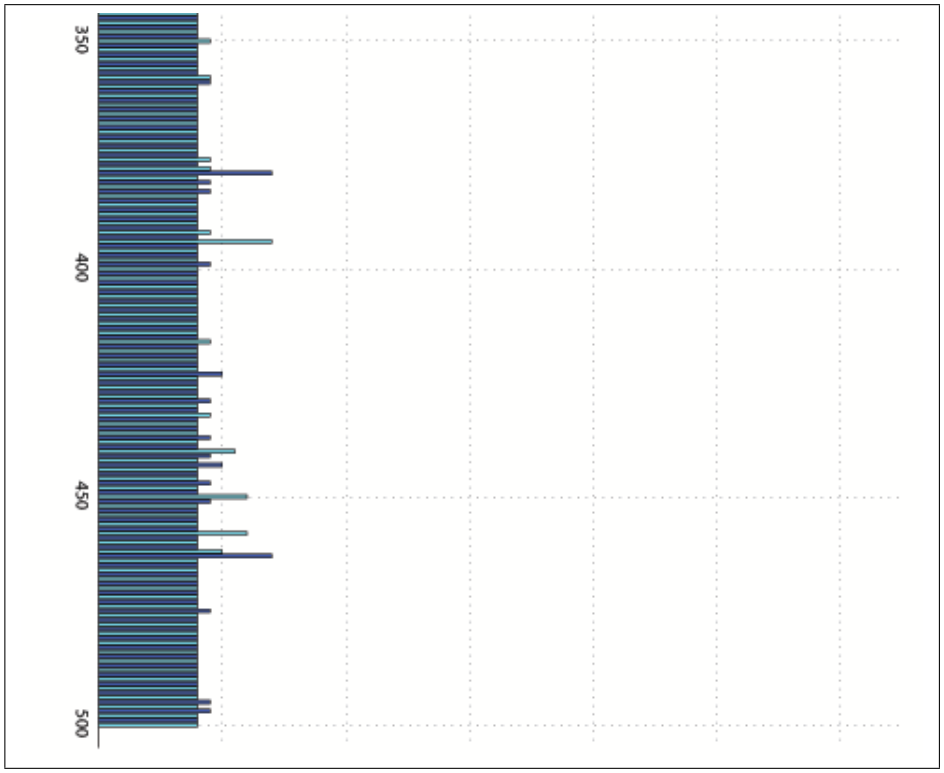


Figure C.4: Distribution of the number of sites of SLiMs in *SLiM_MW_500_3_10*. 500 SLiMs of length $3-10$ were identified from the MW dataset using MEME.

Bibliography

[1] R. Abagyan, M. Totrov, and D. Kuznetsov. Icma new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation. Journal of Computational Chemistry, 15(5):488–506, 1994.

[2] M. Aziz, M. Maleki, L. Rueda, M. Raza, and S. Banerjee. Prediction of biological protein–protein interactions using atom-type and amino acid properties. Proteomics, 2011.

[3] T. Bailey, C. Elkan, S. D. D. o. C. S. University of California, and Engineering. Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Bipolymers. Citeseer, 1994.

[4] T. Bailey, N. Williams, C. Misleh, and W. Li. Meme: discovering and analyzing dna and protein sequence motifs. Nucleic acids research, 34(suppl 2):W369–W373, 2006.

[5] N. Baker, D. Sept, S. Joseph, M. Holst, and J. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. Proceedings of the National Academy of Sciences, 98(18):10037, 2001.

[6] F. Bernstein, T. Koetzle, G. Williams, E. Meyer Jr, M. Brice, J. Rodgers, O. Ken-

nard, T. Shimanouchi, and M. Tasumi. The protein data bank. European Journal of Biochemistry, 80(2):319–324, 1977.

[7] A. Biosciences. Affinity chromatography; principles and methods. Edition AD, pages 18–1022, 2002.

[8] J. Bock and D. Gough. Predicting protein–protein interactions from primary structure. Bioinformatics, 17(5):455–460, 2001.

[9] C. Camacho and C. Zhang. Fastcontact: rapid estimate of contact and binding free energies. Bioinformatics, 21(10):2534–2536, 2005.

[10] Y. Chiang, T. Gelfand, A. Kister, and I. Gelfand. New classification of supersecondary structures of sandwich-like proteins uncovers strict patterns of strand assemblage. Proteins: Structure, Function, and Bioinformatics, 68(4):915–921, 2007.

[11] N. Davey, N. Haslam, D. Shields, and R. Edwards. Slimsearch 2.0: biological context for short linear motifs in proteins. Nucleic acids research, 39(suppl 2):W56–W60, 2011.

[12] N. Davey, G. Travé, and T. Gibson. How viruses hijack cell regulation. Trends in biochemical sciences, 36(3):159–169, 2011.

[13] H. Dinkel, S. Michael, R. Weatheritt, N. Davey, K. Van Roey, B. Altenberg, G. Toedt, B. Uyar, M. Seiler, A. Budd, et al. Elmthe database of eukaryotic linear motifs. Nucleic Acids Research, 40(D1):D242–D251, 2012.

[14] T. Dolinsky, P. Czodrowski, H. Li, J. Nielsen, J. Jensen, G. Klebe, and N. Baker. Pdb2pqr: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. Nucleic acids research, 35(suppl 2):W522–W525, 2007.

[15] R. Duda, P. Hart, and D. Stork. Pattern classification and scene analysis 2nd ed. 1995.

[16] R. Duda, P. Hart, and D. Stork. Pattern Classification. John Wiley and Sons, Inc., New York, NY, 2nd edition, 2000.

[17] R. Duin and M. Loog. Linear dimensionality reduction via a heteroscedastic extension of lda: the chernoff criterion. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(6):732–739, 2004.

[18] R. Edwards, N. Davey, and D. Shields. Slimfinder: a probabilistic method for identifying over-represented, convergently evolved, short linear motifs in proteins. PLoS One, 2(10):e967, 2007.

[19] I. Eidhammer, I. Jonassen, and W. Taylor. Protein bioinformatics: an algorithmic approach to sequence and structure analysis. Wiley Online Library, 2004.

[20] V. Estivill-Castro. Why so many clustering algorithms: a position paper. ACM SIGKDD Explorations Newsletter, 4(1):65–75, 2002.

[21] R. Fisher. The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7:179–188, 1936.

[22] R. Fisher. The use of multiple measurements in taxonomic problems. Annals of Human Genetics, 7(2):179–188, 1936.

[23] R. Free, L. Hazelwood, and D. Sibley. Identifying novel protein-protein interactions using co-immunoprecipitation and mass spectroscopy. Curr Protoc Neurosci, 2009.

[24] G. Geva and R. Sharan. Identification of protein complexes from co-immunoprecipitation data. Bioinformatics, 27(1):111–117, 2011.

[25] E. Golemis and P. Adams. Protein-protein interactions: a molecular cloning manual. Cold Spring Harbor Laboratory Pr, 2005.

[26] R. Gutman, C. Berezin, R. Wollman, Y. Rosenberg, and N. Ben-Tal. Quasimotifinder: protein annotation by searching for evolutionarily conserved motif-like patterns. Nucleic acids research, 33(suppl 2):W255–W261, 2005.

[27] M. N. Isabelle Guyon, Steve Gunn and L. Zadeh. Feature Extraction, Foundations and Applications. Springer, first edition, 2006.

[28] O. Ivanciuc. Applications of support vector machines in chemistry. Reviews in computational chemistry, pages 291–400, 2007.

[29] S. P. B. A. G. M. L. James D. Watson, Tania A. Baker and R. Losick. Molecular Biology of the Gene. Benjamin Cummings, sixth edition, 2008.

[30] S. Jones and J. Thornton. Principles of protein-protein interactions. Proceedings of the National Academy of Sciences, 93(1):13, 1996.

[31] A. Kessel and N. Ben-Tal. Introduction to Proteins: Structure, Function, and Motion. CRC Press, 2010.

[32] T. Mi, J. Merlin, S. Deverasetty, M. Gryk, T. Bill, A. Brooks, L. Lee, V. Rathnayake, C. Ross, D. Sargeant, et al. Minimotif miner 3.0: database expansion and significantly improved reduction of false-positive predictions from consensus sequences. Nucleic Acids Research, 40(D1):D252–D260, 2012.

[33] J. Mintseris and Z. Weng. Atomic contact vectors in protein-protein recognition. Proteins: Structure, Function, and Bioinformatics, 53(3):629–639, 2003.

[34] J. Mintseris and Z. Weng. Structure, function, and evolution of transient and obligate protein–protein interactions. Proceedings of the National Academy of Sciences of the United States of America, 102(31):10930, 2005.

[35] D. W. Mount. Bioinformatics: sequence and genome analysis. CSHL press, 2004.

[36] A. Mucherino, P. Papajorgji, and P. Pardalos. k-nearest neighbor classification. Data Mining in Agriculture, pages 83–106, 2009.

[37] L. Narlikar, R. Gordân, and A. Hartemink. Nucleosome occupancy information improves de novo motif discovery. In Research in Computational Molecular Biology, pages 107–121. Springer, 2007.

[38] A. Neuwald, J. Liu, and C. Lawrence. Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein science, 4(8):1618–1632, 1995.

[39] I. Nooren and J. Thornton. Diversity of protein–protein interactions. The EMBO journal, 22(14):3486–3492, 2003.

[40] Y. Ofran. Prediction of protein interaction sites. In R. Nussinov and G. Shreiber, editors, Computational Protein-Protein Interactions, chapter 9, pages 167–184. CRC Press, 2009.

[41] Y. Ofran and B. Rost. Predicted protein-protein interaction sites from local sequence information. FEBS letters, 544(1-3):236–239, 2003.

[42] G. Petsko and D. Ringe. Protein structure and function. Sinauer Associates Inc, 2004.

[43] L. Rueda and M. Herrera. Linear dimensionality reduction by maximizing the chernoff distance in the transformed space. Pattern Recognition, 41(10):3138–3152, 2008.

[44] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang. Predicting protein–protein interactions based only on sequences information. Proceedings of the National Academy of Sciences, 104(11):4337, 2007.

[45] S. Tan, W. Hugo, W. Sung, and S. Ng. A correlated motif approach for finding short linear motifs from protein interaction networks. BMC bioinformatics, 7(1):502, 2006.

[46] N. Terrapon, O. Gascuel, É. Maréchal, and L. Bréehélin. Detection of new protein domains using co-occurrence: application to plasmodium falciparum. Bioinformatics, 25(23):3077–3083, 2009.

[47] B. Tropp and D. Freifelder. Molecular biology: genes to proteins. Jones & Bartlett Learning, 2008.

[48] W. Valdar. Scoring residue conservation. Proteins: Structure, Function, and Bioinformatics, 48(2):227–241, 2002.

[49] G. Vasudev and L. Rueda. A model to predict and analyze protein-protein interaction types using electrostatic energies. BIBM, 2012.

[50] B. Wang, P. Chen, D. Huang, J. Li, T. Lok, and M. Lyu. Predicting protein interaction sites from residue spatial sequence profile and evolution rate. FEBS letters, 580(2): 380–384, 2006.

[51] L. Wong. The practical bioinformatician. World Scientific Pub Co Inc, 2004.

[52] C. Zhang, G. Vasmatzis, J. Cornette, and C. DeLisi. Determination of atomic desolvation energies from the structures of crystallized proteins1. Journal of molecular biology, 267(3):707–726, 1997.

[53] H. Zhu, F. Domingues, I. Sommer, and T. Lengauer. Noxclass: Prediction of protein-protein interaction types. BMC Bioinformatics, 7(27), 2006. doi:10.1186/1471-2105-7-27.

[54] H. Zhu, F. Domingues, I. Sommer, and T. Lengauer. Noxclass: prediction of protein-protein interaction types. BMC bioinformatics, 7(1):27, 2006.

Vita Auctoris

Manish Kumer Pandit was born in 1984 in Mymensingh, Bangladesh. He received his Bachelors degree from Shah Jalal University of Science and Technology, Sylhet, Bangladesh in Computer Science and Engineering in 2007. His research interests include machine learning algorithm, programming language, pattern recognition, SLiM Discovery, protein-protein interaction and bioinformatics.