REPRESENTATIVE BASED PROTEIN SEQUENCE CLUSTERING

by

M. Riadul Mannan Riad
B.Sc. University of Windsor, 2000

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the School

of

Computing Science

Approval

Name:	**M.Riadul Mannan Riad**
Degree:	**Master of Science**
Title of Project:	**Representative Based Protein Sequence Clustering**

Examining Committee:

Chair: **Dr. Binay Bhattacharya**
Professor

Dr. Martin Ester
Senior Supervisor
Associate Professor

Dr. Jian Pei
Supervisor
Assistant Professor

Dr. Anoop Sarkar
Examiner
Assistant Professor
School of Computing Science, SFU

Date Approved: June 17, 2005

Abstract

Over the years, many methods have been developed for clustering protein sequences based on their similarity. However, most of the methods are based on all-against-all sequence comparison that requires at least quadratic computation on the number of sequences. Furthermore, many methods do not address the issues and challenges associated with protein clustering explicitly such as finding distant relatives and detecting multi-domain proteins. Here, we develop a novel clustering technique based on representatives with successfully avoiding the pair-wise sequence comparison. We address the protein clustering issues in details and give a solution for finding distant relatives and multi-domain proteins. We also develop a new similarity measure that captures the significant similarity information embedded in a sequence such as frequent pattern and sequence length.

Acknowledgements

First, I want to thank Almighty for keeping us in good health and giving me this opportunity to complete my master's degree. Then, I would like to express my deepest gratitude to my senior supervisor Dr. Martin Ester for introducing me into the field of data mining and biological sequence mining, and providing me with tremendous guidance, great support and feedback through out the project work. I would also like to thank Dr. Ke Wang for his advice and guidance during the study of biological sequence mining.

I would also like to extend my sincere gratitude to my supervisor Dr.Jian Pei for providing me insightful advice, help and support for completing the project.

Finally, I would like thank all the members of Database and Data mining lab for their continuous help and support during my study at SFU.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The Human Genome Project and similar work on other species are producing biological sequence databases at an accelerating rate. The number of unique entries in all protein sequence databases together now exceeds more than a million. With this overwhelming growth of biological sequence databases and continuing addition of fully sequenced genomes, handling of these amounts of data has increasingly become a problem. This enormous information has created many challenges in developing novel and scalable computational techniques for searching, comparing and analyzing these databases. Large scale protein sequence analysis is now becoming an effective way to extract useful biological information from the genome sequences for better understanding the structure and function of proteins. The knowledge of biological function of a protein is important for the understanding of fundamental biochemical process for drug design and genetic engineering. The 3D structure of a protein gives the most information of its biological function. However, it is difficult and not always feasible to determine the 3D structure of protein sequences. At present, there exist only few thousands of protein structures, which is significantly low, compare to the enormous sequence databases. Therefore, sequence analysis remains the main source of information for most new proteins. As a result, a considerable effort has been made to predict functional role based only on sequences and the biochemical properties of proteins.

The key notion of sequence analysis is based on the fact that sequences which share similar primary structure are more likely to have common or similar functions. The most effective and efficient way of analyzing protein sequences is to partition them into biologically meaningful groups based on their sequence similarity. Such groups contain sequences that are evolutionary related and share common functions. This is because biological evolution lets proteins fall into groups, thus imposing a natural partitioning. In order to achieve such grouping of protein sequences, the most preferable solution is to use clustering techniques.

1.1 What is Clustering?

Clustering is the process of grouping a large set of unlabeled data into classes or clusters, based on the extent of shared object identity where objects within a cluster are highly similar to each other, but are very dissimilar to objects in other clusters. Dissimilarity measures are basically based on the attribute values describing the objects.

Clustering is not a new concept, it has been widely recognized as a powerful technique in the areas of statistics and computer science for long time, and has been studied extensively during the recent years. The clustering result can potentially reveal unknown relation between objects that may lead to a better understanding of the nature.

Clustering is a reliable and effective mechanism for reducing redundancy in the dataset, and save database search time and analysis effort. It also offers several advantages as opposed to dealing with a single object. A frequent problem in the biological domain is the identification of newly discovered sequences. This task can be performed very quickly and effectively when comparisons are made with the clusters rather than comparing every sequence in the large databases. Another important application lies in the possibility of analyzing evolutionary relationships among the sequences in a cluster. Additionally, a clustered protein database can be used for selecting candidate proteins for structural analysis.

The problem of clustering protein sequence has a long history and has been studied since early 70's. Recently, the studies of protein clustering have been exercised extensively by the researchers from various disciplines including molecular biology, biochemistry, computer science, mathematics and statistics due to the excessive rate of publicly available genome sequences and the challenges associated with it. Though it is an active research field for a long time, there is no such single method existed at this time which can accurately and efficiently partition the proteins into its desired clusters. Because of the complex nature of this problem researchers are still trying to search for *The Solution*. In this project, our goal is to develop a clustering method that can efficiently partition a database of protein sequence into biologically meaningful clusters such that similarity among the sequences within a cluster are maximal and dissimilarity between the clusters are minimal according to a given set of comparison measures.

1.2 Challenges and issues in Protein Clustering

It is generally accepted that when two sequences are highly similar they are said to be homologous. But the homologous sequence doesn't necessarily share high sequence similarity. So, sequence similarity cannot be used as transitive to detect the homology. This makes the homology detection more challenging for the methods based on sequence analysis. It is generally claimed that proteins which share 30% sequence identity are very likely to have similar functions and thus homologous. When similarity falls below 25%, it becomes very hard to detect the homologous sequences using the alignment methods and at this level chances for false positives starts to grow. There are many homologous sequences in the region of 10-25% similarity which is known as the *twilight zone* [16]. These occur during the course of evolution when amino acids substitution, addition or deletion takes place.

Another potential problem which results from the existence of multi-domain proteins in the database can distort the clustering result significantly. These problematic proteins cause the unnecessary links in the database. The Figure 1.1 depicts the problem of multiple domain proteins in the pictorial form. When search is being done using protein A as a seed sequence, the result will return both protein B and C as high similarity scores. Later, when these scores are used for the clustering it may distort the result significantly. In the graph based approach, this causes the unnecessary linking which is sometimes very hard to detect.

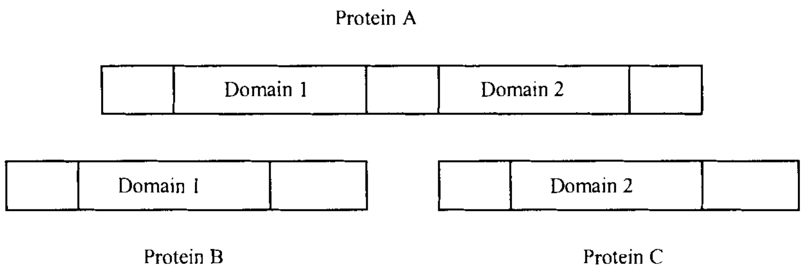

Figure 1.1: Multiple domain protein problem

The final challenge is the efficient clustering of the large protein databases. The scalability is one of the big issues as the protein databases are growing at a high rate. At present, most of the existing methods suffer from this issue as their prerequisite of clustering is based on all-against-all comparison of sequences using an alignment method. So, these methods and methods those require manual investigations become more inefficient as the database size is getting bigger. Therefore, it is highly desirable to design an algorithm that can efficiently handle the clustering problem without sacrificing much quality.

1.3 Existing Methods for Protein Clustering

The most commonly used method for detecting similarity between proteins for the purpose of clustering is to use the single sequence similarity search algorithms such as BLAST [13], FASTA [27] and Smith-Waterman [28] algorithm. These algorithms can detect homologous groups of proteins by performing pair-wise similarity search in the database. This type of all-against-all analysis forms the basis for many existing clustering algorithms such as [14, 15, 16, 17, 18, 29, 30, 31, 32]. The problem for finding distant homologue is solved by using the above similarity search algorithms. But, the issues associated with detecting multi-domain protein are explicitly addressed only in [18, 29, 31] and indirectly addressed in [14,15].

However, the pair-wise similarity based methods suffer from the following aspects, (1) their computation is very high e.g. to compute the pair-wise similarity of each sequence the complexity is at least $O(n^2 \times l^2)$ where n is number of sequences and l is the average sequence length in the database, (2) the incremental clustering would be very inefficient as well, because it will require pair-wise similarity between clustered and unclustered sequences and (3) search results can greatly vary depending on which distance functions or similarity methods are used and often it is hard to find an optimal parameter settings for better results. Besides that, search results are questionable in detecting distant pairs when the sequence similarity is absent or very low. Efficiency is also a key problem for these methods. The run-time reported in [33] for a pair-wise similarity computation of 56,563 sequences using Smith-Waterman algorithm took 70 CPU days.

Lot of other methods such as [2,5,7,9,21,22,23,24] uses multiple alignment procedures to determine the similarity relationship between proteins. These methods can often produce excellent information for the clustering basis but tend to be more CPU intensive than the single sequence based search procedures. The problem of finding distance homologue is addressed in [2,5,7,9,21,22] and multi-domain detection is explicitly used in [7,9,21,22]. However, it is not an easy task to build the multiple alignment of many sequences. The existing methods apply heuristic algorithms, which are not guaranteed to find the optimal alignment. Furthermore, the optimal alignment itself is not guaranteed to be biologically accurate.

In contrast to the previous studies, which are based on alignment methods, there are some other methods [10,34,35] which use feature selection or short-word filtering as the basis for the clustering method. Feature selection method basically treats frequent patterns as features and uses k-means portioning algorithm for clustering. For the Short-word filtering algorithm, the words are collected usually length 2-6 from a large set of non-redundant alignments. However, these methods don't consider the problem of multiple domains explicitly and also, don't take any extra care for detecting the distant homologue proteins.

1.4 Clustering Algorithms

The most well-known and commonly used clustering methods are k-means, k-medoids and their variants. We give a brief introduction to these clustering methods in the following sections.

1.4.1 K-means

The k-means [40] algorithm takes the input parameter *k* and partitions a set of *n* objects into *k* clusters such that objects of the same clusters are highly similar and objects between the clusters are dissimilar. Cluster similarity is measured with respect to the mean value of the objects in a cluster, which can be viewed as the cluster's center of gravity.

The k-means algorithm works in the following way. First, it randomly selects *k* of objects where each object initially represents a cluster center or mean. For each of the remaining objects, an object is assigned to the cluster based on the similarity between the object and the cluster mean. Typically, the similarity is defined as a distance function of the two objects. It then computes the new mean for each cluster. This process continues until the criterion function converges. The most commonly used criterion function is squared-error criterion. The run time of this algorithm is $O(nkt)$ where, n is the number of objects, k is the number of clusters and t is the number of iterations.

The drawbacks of this method are (1) users need to specify the number of clusters k in advance and (2) sensitive to noise and outliers since an object with extremely large value can substantially influence the mean value and distort the distribution of the data.

1.4.2 K-medoids

The k-medoids [37, 40] is a variant of the k-means method that tries to improve the sensitivity issue discussed previously. The basic idea of k-medoids clustering is to find k clusters in n objects by first randomly selecting a representative object (the medoid) for each cluster. Each remaining object is clustered with the medoid to which it is most similar. The strategy then iteratively replaces one of the medoid by one of the non-medoids as long as the quality of the resulting clustering is improved. This quality is estimated using a cost function that measures the average dissimilarity between an object and the medoid of its cluster. Typical k-medoids methods are PAM [37] and CLARA [37]. The run time for these methods are higher than k-means however, all these methods require users to specify k, the number of clusters.

The protein clustering problem is such that, it is not possible to determine the number of groupings (i.e. clusters) in advance without investigating the data [16]. The huge amount of protein databases also makes it impossible to manually investigate the data. Furthermore, it requires lot of expertise for this investigation. Therefore, we need automatic detection of the number of clusters that naturally partitions the protein into distinct clusters. In addition, identifying the multi-domain proteins and distant relatives are not straightforward from these methods. So, we need new methodology to solve the protein clustering problem.

1.5 Our Clustering Approach

Unlike the previously discussed methods, our approach is based on the careful selection of representatives from the database. We devise a representative selection procedure that carefully selects each representative in order to make our initial clustering as good as possible close to the final clustering. Another reason for choosing the representative set is to avoid the all-against-all sequence comparison, which is unrealistic and highly inefficient due to the presence of huge protein databases. The clustering quality highly depends on the similarity measure that is being used for comparing the sequences. We define a new similarity measure based on the frequent patterns and sequence properties such as length. This new similarity measure also helps identifying the multi-domain proteins that are problematic for the clustering.

Our clustering method works in the following way. First, we carefully select a set of representative from the database. Then, we compare each sequence with the representatives and assign them to the one, which is most similar. During this process, we also apply the multi-domain detection procedure to identify the multi-domain proteins. This forms the basis of initial clustering. We further optimize the quality of initial clustering by improving the appearance of frequent patterns for each cluster. Then, we apply the homology detection procedure to find the distantly related proteins. There are sequences that are distantly related i.e. similarity is not significantly reflected in the sequences, we use transitivity to infer these homologies. Therefore, our method helps identifying a range of distantly related sequences from high similarity to very weak similarity. The resulting final clustering is flat and overlapping.

The runtime of our clustering method is $O(n.r.t)$ where n is the number of sequences in the database, t is the number of iterations and r is the number of representatives. Our method is as efficient as K-means and also, it doesn't require specifying the number of clusters in advance. We achieve the following goals in our proposed protein clustering method: (1) no pair-wise sequence comparison hence, highly scalable for large databases, and (2) detection of distantly related proteins and finally, (3) handling the issues of multi-domain protein problem.

The rest of the document is organized as follows. Chapter 2 gives the details of background studies of existing clustering methods followed by the discussion of basics of protein sequences in chapter 3. The details description of our clustering methods is presented in chapter 4. The dataset and experimental results are presented in chapter 5. Finally, we conclude by giving some possible future studies and directions.

Chapter 2: Related Studies – An Overview

Studying protein sequence data is an active research field since early 70's and now, has become one of the popular research areas due to the availability of huge protein data. As clustering is a powerful tool for the meaningful portioning of dataset, lot of work has already been done in identifying the clusters for protein sequence databases. These studies are basically focused on three different categories: motif, pattern and domain based analysis, pair-wise sequence comparison and hidden markov modeling. Before we go into the details of these studies we discuss the source databases for protein sequence that are publicly available.

Clustering methods based on external sequence comparison tool such as BLAST, FASTA or Smith-Waterman algorithm can identify the distant relative sequences as we mentioned in the earlier section. However, not all of these clustering methods explicitly addressed the problem of identifying multi domain proteins.

2.1 Protein Databases

2.1.1 SWISS-PROT

SWISS-PROT [3] is a curated protein sequence database established in 1986 that strives to provide a high level of annotation, a minimal level of redundancy and high level of integration with other databases. The annotation includes the description of protein function, domain structure, post-translational modifications, variants, etc. It is a collaborative effort of the Swiss Institute for Bioinformatics (SIB) and the European Bioinformatics Institute (EBI). The current release is version 42.1 as of 24-Oct-2003, and contains 135,356 entries, comprising 50,223,175 amino acids.

2.1.2 TrEMBL

TrEMBL (Translated EMBL) [3] is a computer-annotated supplement to SWISS-PROT. The TrEMBL database contains the translations of all coding sequences (CDS) present in the EMBL Nucleotide Sequence Database that are not yet integrated into SWISS-PROT. TrEMBL has two main sections: (1) SP-TrEMBL (SWISS-PROT TrEMBL) contains entries that will eventually be incorporated into SWISS-PROT, but that have not yet been manually annotated; (2) REM-TrEMBL contains sequences that are not destined to be integrated in SWISS-PROT. The current release is version 25.1 (SP-TrEMBL) as of 24-Oct-2003, and contains 1,016,356 entries comprising of 315,510,213 amino acids.

2.2 Motif, Pattern and Domain Based Analysis

2.2.1 PROSITE

The **PROSITE** [1,2] is a database of protein families and domains. It is based on the fact that some regions of a protein sequence are better conserved than others during evolution. These regions referred to as motifs, are generally important for the function of a protein and/or for the maintenance of a 3-dimensional structure. Within PROSITE, motifs are encoded as regular expressions, often simply referred to as patterns. The process for deriving the patterns involves the construction of multiple alignments and manual inspections to identify the conserved regions. The patterns are then used to search the sequence database (SWISS-PROT) for finding the homologue proteins. Manual investigation of the results gives the finer tune patterns. For some families no motif or pattern can be defined since the sequences have significantly diverged. These families are described by means of a *profile* derived from a multiple alignment of related proteins.

The current release 17.38, as of 23-Feb-2003 contains 1170 documentation entries that describe 1605 different patterns, rules and profiles/matrices. This manually defined highly reliable pattern database has served as a reference classification for lot of other studies in this field. The PROSITE database is linked and cross-referenced to the entries in the SWISS-PROT database.

2.2.2 PRINTS

From the experience of sequence alignments, it is clear that most protein families are characterized by several conserved motifs. Therefore, the group of motifs makes it more flexible and more powerful than single-motif approaches. This observation motivates the construction of **PRINTS** [4] database. The PRINTS is a manually defined protein fingerprint database. A fingerprint is a group of conserved motifs that used to characterize a protein family. Fingerprinting is an iterative process that starts with manual sequence alignment and extracting the conserved motifs. The motifs are then used to search the source databases (SWISS-PROT and TrEMBL) iteratively for constructing the fingerprints. The purpose of the PRINTS database construction is not to compete with PROSITE rather than complement the PROSITE pattern/profile resources and facilitate sequence analysis. To date, PRINTS database have developed and manually annotated 1750 fingerprints, encoding 10,626 individual motifs.

2.2.3 BLOCKS

The **BLOCKS** [5] database is a collection of blocks where each block represents a conserved region of a protein family. The blocks are automatic generation of ungapped multiple alignment corresponding to the most conserved regions of a protein family. The database is constructed from the documented families of related proteins in PROSITE, PRINTS, Pfam [19] and Domo [9]. The block construction method is a two-step process: (1) first, candidate blocks are detected from a set of related proteins by identifying short motifs and then, extending these motifs in both directions until the similarity score drops below a threshold. The resulting blocks are at most 60 amino acids long. (2) The final step creates an ordered set of non-overlapping blocks called *path* that satisfies the min support threshold. Each individual block is given a score based on its length, the level of similarity and the number of occurrences. The score of a path is the sum of all block scores weighted by the number of sequences that contain the path. Finally, the best score of all the paths is selected to represent the corresponding group of related proteins.

The Blocks+ [6] is a new version of the original Blocks database that adds another documented protein family database called ProDom [7]. The direct derivation of BLOCKS database from the mentioned documented databases gives it more power for searching related proteins. For example, 50% of families encoded in PRINTS are not represented in PROSITE, so searches of BLOCKS database will be more comprehensive than searches of either database alone. To date, Blocks Database consists of 8656 blocks representing 2101 groups documented in InterPro 3.1 [25] keyed to SWISS-PROT 39.17 and TrEMBL.

2.2.4 ProDom

The **ProDom** [7] is a protein domain database consists of an automatic generation of homologous domains from the SWISS-PROT and TrEMBL sequence databases. The database is built in three steps: (1) identify all high scoring segment pairs (HSPs) using BLAST [13] all versus all sequence comparisons, (2) construct a set of homologous segment by transitive closure of HSPs if the common segments overlap above a minimum overlap threshold, and (3) detect domain boundaries and generate multiple alignments and a consensus sequence for each domain family. The current version of ProDom [8] is based on iterative PSI-BLAST searches.

2.2.5 DOMO

The **DOMO** [9] is a fully automated classification of the SWISS-PROT and PIR sequence databases that apply compositional and local similarity searches followed by multiple sequence alignments. The procedure starts by detecting global similarities from the pairwise comparison of amino acid and dipeptide composition of each protein, and grouping the most closely related sequences into clusters. Each cluster is represented by one sequence and representatives are stored in a suffix tree. This tree is self-compared to detect local sequence similarities. The local similarities are clustered and multiply aligned to detect domain boundaries and the sequences are spilt into domains accordingly. Finally, multiple alignment is generated for each set of proteins that share similar segments.

2.2.6 InterPro

It has been estimated that, the total number of protein families might be in the range 1000 to 10,000, so there is still a long way to go before any single database can be considered as complete. Thus, in building a search strategy it is a good practice to include all the resources that are available to make the search as comprehensive as possible. This motivates the integration of all documented resources for protein families, domains and sites, which is called as **InterPro**. It is an international collaborative project to integrate the database PROSITE, Pfam, PRINTS, ProDom and SMART within a unified protein family annotation resource. It aims to reduce the duplication of effort for the laborious annotating process and to facilitate communication between disparate resources. Each combined InterPro entry includes functional descriptions and literature references, and links are made back to the relevant member databases. InterPro provides a convenient means for the analysis of newly determined sequences. To date, it contains 6725 entries representing 1453 domains, 5121 families, 136 repeats, and 15 post-translational modification sites.

2.2.7 The Study by V. Guralink and G. Karypis

The study by [10,11] describes a scalable algorithm for clustering protein sequences. The key idea of this approach is to find a set of features that capture the sequential nature of the various proteins. Then, project each protein into a new space whose dimensions are these features and use a traditional vector-space k-means based clustering algorithm to find the protein clusters. The features are the amino acid subsequences that satisfy a given minimum support constraint. These frequent amino acid subsequences are often called motifs.

To evaluate the performance, three different datasets were chosen from 20 different protein families from the SWISS-PROT database. The clustering quality is measured by entropy and in comparison with k-mediod based clustering this algorithm performs better. Although, this algorithm outperforms k-mediod, it shows high degree of overlap i.e. some clusters containing sequences from different families. The limitation of this approach is the features/motifs that were used for clustering the sequences. Because, there are some sequences which do not share any pattern at all. As a result, they are reported as outliers in the database. The limitation can be overcome by using substitution matrix or similarity matrix to define equivalent classes of motifs. This method does not consider the issues associated with multi-domain and distant proteins.

2.3 Sequence Comparison Approach

2.3.1 SYSTERS

The **SYSTERS** [12] database, an iterative method for database searching to cluster proteins in the SWISS-PROT and PIR databases. The clustering procedure is fully automatic and does not require pair-wise comparisons between sequences. The procedure starts with a seed sequence and using the BLASTP [13] program, it finds and retains all the sequences that are highly significant to the seed. The lowest scoring sequence is used as a query for the next database search to explore the sequence space below the significant level for sequences possibly related to the seed. The process repeats until no new sequences above the significant level are found or the search has no sequence in common with the set of accepted hits in the first search.

The resulting clusters correspond to groups of sequences that do not share domains with other families. If a group of proteins share a domain with other families then each family is mapped to a different cluster.

The clustering quality is evaluated by an internal consistency test. The underlying idea is that, if a set of sequences indeed forms a cluster, the same cluster should be identified independently for any cluster member that is used as a seed for the search. The result shows the large degree internal consistent in a sense that the resulting clusters in most cases show little dependence on the specific query. The runtime for clustering 59,021 sequences took roughly 5 days on a workstation cluster consisting of eight SUN Ultra workstations. BLASTP runs performed for the SYSTERS searches dominate the runtime. This method indirectly addressed the distant relative problems.

2.3.2 ProtoMap

ProtoMap [14,15,16] offers a classification of the protein sequences into groups of related proteins. Several common measures of similarity between protein sequences such as Smith-Waterman algorithm, FASTA and BLAST are combined with two different scoring matrices (BLOSUM50 and BLOSUM62) to create an exhaustive list of neighboring sequences for each sequence in the database. A weighted directed graph is created using the lists of neighboring sequences whose nodes are the sequences. The weight of an edge represents the degree of similarity between two connected sequences. Thus, clusters of related proteins correspond to strongly connected components of the graph.

The analysis starts from a very conservative classification, based on highly significant similarities and later, classes are merged to account for less significant similarities. The procedure is repeated at varying confidence levels and for each step the algorithm is applied on the classes of previous classification in order to obtain the next one. Finally, a hierarchical organization of all proteins is obtained. The runtime for the whole procedure is not reported, but as it requires all versus all comparisons of the sequences in the database using three above methods, it will be very expensive. The issue associated with multi domain problem is addressed indirectly in the method.

2.3.3 Structure Prediction by Transitive Homology

The study by [18] offers a clustering of protein sequences by using transitive relation of homology. The procedure starts with computing pair-wise sequence similarity using Smith-Waterman alignment method. A weighted directed graph is built using the raw scores in which every edge indicates the percentage of similarity between two nodes. Finally, finding the strongly connected component of this graph does the clustering. The correctness of the clusters is reported by comparing with SCOP database that offers manually curated highly reliable classification of protein sequences. Three different datasets are selected from SCOP database with varying sequence similarity. The results demonstrate that carefully designed methods are able to find sequences in the twilight zone than pairwise sequence comparison methods. Therefore, transitive relation of homology can help identifying the sequences in the twilight zone. However, the total runtime reported in the study for 70,454 sequences are 600 CPU days. The workload was distributed among 40 computers with 55 CPUs. This method provides a procedure for the detection of multi-domain proteins.

2.3.4 CluStr

The **CluSTr** [17] database offers an automatic classification of SWISS-PROT and TrEMBL protein into groups of related proteins. The clustering approach is based on two steps: (1) a similarity matrix of all versus all comparisons of the protein sequences is built using the Smith-Waterman algorithm and statistical significance of similarity between potentially related proteins is estimated using a Monte-Carlo simulation; (2) clusters are built using a single linkage algorithm for different levels of protein similarity. Only clusters with more than one sequence are reported in the database. This method does not provide procedure for the detection of multi domain proteins.

2.4 Hidden Markov Modeling (HMM)

Clustering using HMM was first mentioned in [20] for speech recognition problems. Today, the HMM models have been used extensively to model protein families [21, 22, 23, 24]. The basis of these studies is to build a statistical model of the known protein family and then comparing the model to each sequence in the database. These statistical models have a strong theoretical basis in probability and are supported by efficient algorithms for training, database searching and multiple sequence alignment. One drawback of modeling proteins using HMM is that, it contains lot of free parameters and therefore, require a large amount of training sample.

2.4.1 Pfam

Pfam [19] is a manually curated database of Hidden Markov Models for protein families. The Pfam families are important tool for understanding protein structure and function, form the basis for techniques such as secondary structure prediction, fold recognition, phylogenetic analysis and mutation design. For each family, the process starts from a seed alignment of a non-redundant representative set of known members. The seed alignments are taken either from a published multiple alignment or an alignment from other existing databases such as PROSITE [1,2] or ProDom [7]. The alignments are checked manually by experts for further verification that the conserved features are aligned correctly and the alignment has enough information content to distinguish chance similarities from true relationships. After the manual confirmation, the HMM models are built from the seed alignments. The HMM models are then used to scan SWISS-PROT for all other members of the families. If a true member is missed, it is added to the family manually and the process is repeated. Finally, a full alignment is constructed for the family by aligning all the members to the HMM. This alignment is checked again manually, and for any incorrect alignment, the method is modified or the whole process starts with a new improved seed. To date, Pfam release 8 contains 5193 families that have matches 73% of proteins in SWISS-PROT release 40.44 and TrEMBL release 22.

Chapter 3: Preliminaries of Protein Sequence

3.1 Basics of Protein Sequence and Structure

Proteins are complex molecules that form much of the functional and structural machinery of each cell in every living organism. They are made from small building blocks called **amino acids**. There are 20 different amino acids with diverse physical and chemical properties that allow proteins to exhibit a great variety of structures and functions. **DNA**, the cell's repository of genetic information, contains the code for protein sequence. The DNA sequence is represented by long chain of 4 different small molecules called **nucleic acids**. A segment of DNA that contains the code for a specific protein called **gene**, in which every block of 3 nucleic acids corresponds to an individual amino acid.

A well-defined linear chain of amino acids represents the **primary structure** of a protein sequence. The average length of a protein sequence is 350 amino acids, but it can be short as 50 amino acids and also can be long as 5000 or longer. Each amino acid is represented by one letter and thus, protein can be viewed as a long word over an alphabet of 20 letters. **Secondary structures** are local sequence elements (3-40 amino acids long) that have a well-determined regular shape, such as a α-helix or a β-strand. Other local sequence elements that are neither helices nor strands are usually called loops or coils and they may adopt a large variety of shapes. The **tertiary structure** of a protein sequence is the full 3-dimensional folded structure of the whole sequence. The shape of a 3-dimensional structure is called the **fold**. Each protein has a unique 3-dimensional structure, but several proteins may adopt the same fold i.e. similar 3-

dimensional structure. The **quaternary structure** or **protein complex** can also be formed if two or more proteins join together.

The term **domain** is very common when describing protein structure or function. It is the fundamental unit of structure. It combines several secondary structure elements such as α-helices and β-strands in order to pack together to form a compact globular structure. A domain can fold independently into a stable 3-dimensional structure and can have a specific function. A protein may be comprised of a single domain or several different domains or several copies of the same domain depending on its size.

3.2 Homology and Protein Family

When sequences share a significant similarity they are usually assumed to have a common evolutionary ancestry and are called **homologous proteins**. Several different levels of sequence similarity are defined. A **protein family** is a group of homologous proteins, and proteins that belong to the same family have the same or similar biological function. A group of protein families that are distantly related through evolution make up a **protein superfamily**. Proteins that belong to the same superfamily have close or related biological functions, but sequence similarity is not always detectable. A few protein families may adapt the same fold. Proteins that belong to the same fold may have related biological functions. However, this is not necessarily due to the sequence similarity or evolution relationship, and therefore, this kind of similarity is not always detectable by the standard methods for sequence comparison.

3.3 The Protein Folding Problem

According to the central dogma of protein folding, the protein sequence almost always folds into a characteristic, 3-dimensional structure. It is the specific 3-dimensional structure that enables the protein to function in its particular biological role. This folding prescribes the function of the protein and the way it interacts with other molecules. In other words, a protein can only be active and functional after settling into its final shape. This process is complete almost immediately after proteins are made. Most proteins fold in less than a second, although the largest and most complex protein may take several seconds to fold. There are some exceptions when some proteins remain partially unfolded and complete shape occurs at later time.

As protein sequence always folds into 3-dimensional structure, scientists reason that the instruction for folding a protein sequence must be encoded within the sequence. But, for 50 years scientists have tried and failed to find the special piece of code that governs the folding process of a protein sequence. Scientists call this the "protein folding problem" and it remains one of the great challenges in the structural biology. Although researchers can predict the protein's shape by using some general rules and sometimes, manual investigations, they cannot accurately and reliably predict the final structure from an amino acid sequence. Therefore, determining the 3-dimensional structure of a protein sequence is difficult and not always feasible.

The 3-dimensional structure of a protein sequence gives the most information about its biological function. But, the lack of full understanding of protein folding process cannot advance the structure prediction methods and had only moderate success.

3.4 Protein Sequence Analysis

As the rules of protein folding process have not been fully understood and structure prediction is still not possible, sequence analysis remains the main source of information for most new proteins. Therefore, a significant effort has been made only on the studies of primary structure of protein sequences, their biochemical properties and functional roles in living organisms. Shared evolutionary ancestry has become the basis of such studies. Sequences that share common ancestor are said to be homologous. The homologous sequences are believed to have similar or related functions. In most cases, sequence similarity entails the similar or related functions. Therefore, sequence similarity has become a powerful and common tool for detecting homologous sequences. Today, there are a number of generally accepted methods such as BLAST, FASTA, Smith-Waterman algorithm for comparing sequences and assessing their similarity or distance. These are quite successful in detecting similarity and have become a standard tool for biologists.

However, sequence similarity is not always easily detectable. Homologous proteins may have similar structure without sharing significant or even detectable sequence similarity. Therefore, the inference of homology can be based on sequence similarity, but the converse is not true. This is due to the changes of sequences that have occurred during evolution by insertions, deletions and mutations. But, they share a common ancestor and thus, homologous. Sequences that share similarity greater than 30% can be deduced safely as homologous. This is known as "safe zone" where homology inference can be made without using any further knowledge. When similarity falls between 18-25%, which is called the "twilight zone" where homology cannot be inferred directly. So, the search methods that just use sequence comparison for homology detection fail to identify these homologous proteins.

By definition, homology is a transitive relation, i.e. if A is homologous to B, and B is homologous to C, then A is homologous to C. This simple observation can help identifying homologous protein that does not share significant sequence similarity. However, this relation should be used with great caution, otherwise it will lead to many pitfalls. Similarity is not transitive and it does not necessarily imply homology [26]. Similarity may be quantified and homology is a relation that either holds or does not hold. Significant similarities can be used to infer homology with a level of confidence that depends on the statistical significance [26]. Therefore, similarity should be used carefully in order to infer homology.

Deduction of homology can be even more difficult when proteins are comprised of multiple domains. When protein 1 contains domain A and B, protein 2 contains domain B and C, protein 3 contains domain C and D, should we conclude that protein 1 and 3 are homologous.

Chapter 4: Clustering Method

The basic idea of our clustering method is based on simple observation that sequences under the same cluster should share a set of frequent patterns and sequence properties such that intra-cluster sequences are highly homologous and inter-cluster sequences are non-homologous. The homologous includes all closely related and distantly related proteins in the database that share similar behavior or functionalities. Therefore, our goal is to optimize the homogeneity within the clusters and increase the dissimilarity between the clusters.

Here, we propose an iterative flat clustering method that will solve our protein clustering problem efficiently and accurately. The method has two phases. In first phase, we carefully select a set of sequence to represent the initial clustering. This set of sequence is known as cluster representative. The selection process is based on frequent patterns and sequence property such as length. Then, we compare the remaining set of sequences with each cluster representative and assign them to clusters to which it is the most similar. The sequence comparison is based on a similarity scoring function that uses frequent patterns to detect the level of sequence similarity in the presence of highly conserved regions. We also define a procedure for the detection of multi-domain proteins. The multi-domain procedure is used during the initial clustering process in order to avoid the affect of these proteins in the later part of the clustering.

In second phase, we select a set of new representative for each cluster and search for the similar sequences. The newly discovered similar sequences are basically distantly related proteins to the old representatives. The process iterates until no more sequences can be added to the clusters or a threshold limit is achieved. At the end, we merge the similar clusters if there exists any. The final clustering is flat and overlapping. In order to make our clustering algorithm more robust and accurate we introduce another algorithm that differs only in the second phase. In the second phase, this algorithm creates a consensus sequence for each cluster instead of selecting a set of representative sequence.

We name our first clustering method as CRS i.e. Clustering based on **R**epresentative Sequence and second clustering algorithm as CRCS i.e. Clustering based on **R**epresentative Consensus Sequence. Before we go into the details of clustering method let us identify some definitions and procedures that are needed for the method.

4.1 Problem Definition

A protein sequence is an ordered list of amino acids represented by the alphabet set $A = \{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y\}$. Let $S = \{S_1,S_2 \ldots\ldots S_n\}$ be a database of protein sequences and $P = \{p_1,p_2, \ldots\ldots p_m\}$ be a set of frequent patterns or frequent subsequences occurring in the sequences of S that satisfy the user defined minimum threshold. Each sequence S_i is represented by a set of frequent patterns occurring in S_i such that $S_i \subseteq P$.

Let $f = \{f_1, f_2 \ldots\ldots f_m\}$ be a set of frequency for all frequent patterns in P, and $w = \{w_{1,1} \ldots w_{i,j}, \ldots\ldots w_{n,m}\}$ be a set of frequency weight for all P in database S, where, $w_{i,j}$ is the frequency weight for pattern P_j in sequence S_i. Let $\overline{w} = \{\overline{w}_1 \ldots\ldots \overline{w}_m\}$ be a set of means for all frequency weight of frequent pattern set P. So, $\overline{w}_j$ can be defined as: $\overline{w}_j = \sum_{1 \le i \le n} w_{i,j}/n$. Let $l = \{l_1 \ldots\ldots l_n\}$ be a set of length for all sequence in S and $\bar{l}$ be the mean length such that: $\bar{l} = \sum_{1 \le i \le n} l_i/n$.

Let *min similarity threshold* t be a real number such that, $t \in \Re$. The value of t determines the min similarity that must satisfy in order for two sequences to be considered similar (homologous). Let *min distance threshold* d be an integer number such that, $d \in \Im$. The value of d determines the min number of intermediate sequences required to infer the homology between sequences.

Let $C = \{C_1, C_2, \ldots\ldots C_k\}$ be a *clustering* where each C_i is any subset of the set of all subsets of the database S such that each sequence of S is contained in at least one of the sets (clusters). The clusters of clustering may or may not overlap.

4.2 Finding Frequent Pattern

Find all the patterns that satisfy a user define threshold from sequence database S using well known *apriori* [38,39] algorithm. These patterns are known as *frequent patterns*. Inspiring from document clustering methods, we convert all pattern frequency by the weighted frequency i.e. *term frequency-inverse document frequency* (TF-IDF). The idea is to discriminate all the patterns that appear too many times in the sequence database. The frequency weight $W_{i,j}$ is defined as:

$$w_{i,j} = f_{i,j} \times (\log_2^{\,n} - \log_2^{\,f_j} + 1) \tag{4.1}$$

where, n is the number of sequences in the sequence database S, $f_{i,j}$ is the occurrence of frequent pattern P_j in sequence S_i.

Example 1: The Table 4.1 describes a sequence database S of 5 sequences and the mean length of these sequences are 10. Table 4.1 also presents the frequent patterns of different sizes for each sequence. The *min threshold* for finding the frequent patterns is set to 2. The Table 4.2 displays the occurrence of each frequent pattern in the database *S*.

Sequence database S	***Length***	***fp[1] size =2***	***fp size =3***	***Fp size =4***
ACDZYMNACD	10	AC, CD, YM, MN	ACD	
LCDEFIKCDEM	11	CD, FI, IK, KC	FIK, IKC	FIKC
PQACDILYMIQM	12	AC, CD, DI, YM	ACD	
MNDIKLAC	8	AC, DI, IK, MN		
EZFIKCMCD	9	FI, IK, KC, CD	FIK, IKC	FIKC

Table 4.1: Sequence Database and Frequent Pattern

Frequent pattern	f_i	***Frequent pattern***	f_i	***Frequent pattern***	f_i
AC	3	IK	3	ACD	2
CD	4	KC	2	FIK	2
YM	2	DI	2	IKC	2
FI	2	MN	2	FIKC	2

Table 4.2: Frequency of Frequent Pattern

The frequency weight $w_{i,j}$ of all frequent patterns for each sequence is presented in the following table.

S_i	***Frequency Weight ($w_{i,j}$)***	***Frequency weight***	***Frequency Weight***
S_1	AC=3.47,CD=2.64, YM =2.32, MN=2.32	ACD=4.64	
S_2	CD =2.64, FI =2.32, IK =1.73, KC=2.32	FIK=2.32, IKC=2.32	FIKC=2.32
S_3	AC=1.73, CD=1.32, DI=2.32, YM=2.32	ACD=2.32	
S_4	AC=1.73, DI=2.32, IK=1.73, MN=2.32		
S_5	FI=2.32, IK=1.73, KC=2.32, CD=1.32	FIK=2.32, IKC=2.32	FIKC=2.32

Table 4.3: Frequency Weight $w_{i,j}$

[1] *frequent pattern (fp) and pattern size = 2*

4.3 Selection of Cluster Representative

The goal is to select a set of sequence from database S such that each sequence can represent a group in the database and serve the basis of our clustering method. This set of sequence represents the clusters known as cluster representative sequence. In order to be an ideal sequence for clustering basis, a sequence must show some better quality than other sequences in the database. An ideal sequence would be a sequence that has average length and frequent patterns with high weighted frequencies. So, frequent pattern and sequence property such as length are used as selection criteria for choosing cluster representatives.

Frequent pattern plays an important role for measuring sequence similarity in our clustering method. Therefore, selection of cluster representative with high scoring frequent patterns will result in better initial clustering. Also, sequence length is important for determining the cluster representatives, because protein sequence can be as large as few thousands of amino acid to as short as few hundreds. Large sequences are more likely to contain many patterns and therefore, it is not a good choice to select them as base sequence for initial clustering. For example, a large protein sequence may contain multiple domains and thus, it will cover lot sequences from different domains and as a result, it may distort the quality of initial clustering. So, an average sized sequence is a better candidate for cluster representative. Therefore, the function for selecting a cluster representative (*Rep-Score*) is defined as:

$$\textit{Rep-Score}\ (S_i) = \frac{\sum w_{i,j} \times \log_2{}^{|p_j|}}{|S_i|} + \log^{\frac{1}{|l_i - \bar{l}|+1}} \qquad (4.2)$$

where, $|P_j|$ represents the length of frequent pattern P_j and $|S_i|$ represents the number of frequent patterns in sequence S_i.

The motivation behind this function is to measure the goodness of a sequence by using its sequence property and frequent patterns. The first term of this function is adding up the weighted frequencies of all frequent patterns multiplied by the corresponding pattern length. We multiply frequencies by pattern length in order to reflect the importance of pattern size in the score. The longer pattern gets higher score and vice versa. Then, we normalize the score to avoid the affect of varying number of frequent patterns. When measuring this term we do not consider any subset frequent pattern with lower or equal weighted frequency than its super pattern. For example: AC cannot participate in the calculation because it is a subset and it has lower weighted frequency than ACD. The second term of this function measures the deviation of sequence length from the mean length. When sequence length is equal to mean length score of this term is 0 otherwise, score is negative. Therefore, higher score of this function represents the better candidate for cluster representative.

Example 2: We use the sequence database in Table 4.1 and calculate the *Rep-Score* for sequence S_1. Frequent pattern AC and AD are not used in the calculation because their weighted frequency is less than ACD. The calculation is given below and Table 4.4 contains the *Rep-Score* for all sequences in S.

$$\textit{Rep-Score}\ (S_1) = \frac{2.32 \times 1 + 2.32 \times 1 + 4.64 \times 1.58}{3} + \log^{1/(10-10)+1} = 3.99$$

Sequence database	***Rep-Score***
ACDZYMNACD	3.99
LCDEFIKCDEM	3.34
PQACDILYMIQM	2.29
MNDIKLAC	1.55
EZFIKCMCD	2.68

Table 4.4: *Rep-Score* for Sequence Database

Now, the question is how to select the representative set using the *Rep-Score* function. One easy way to select the representative set is by choosing all the sequences that have score greater than some min threshold. The potential problem in this case is, there may be sequences in the representative set which are highly similar to each other. As a result, it may cause lot of overlaps in the initial clustering. In order to avoid this problem and furthermore, to achieve the goal of our representative set we define the following selection algorithm.

Rep-Selection Algorithm
Input: sequence set S, k, coverage;
Output: Representative set

```
1. Sort the sequence set by Rep-Score
2. Repeat
    a. count = 0
    b. select the highest Rep-Score sequence $S_{high-score}$
    c. for each sequence $s \in S$ do
        i. if $S_{high-score} \cap s$ > min pattern threshold      // # of matching pattern
            1. count++
        ii. if count > min representative threshold
            1. add $S_{high-score}$ to Representative set
            2. remove $S_{high-score}$ and all s covered by $S_{high-score}$ from S
3. Until |Representative set| > k or min database coverage is reached
```

Algorithm 4.1: Representative selection Algorithm

The basic idea behind this algorithm is to select a set of representative sequence that possibly represent a particular group of sequences in the database and minimize the chances of selecting similar sequences. First, we sort all the sequences by their Rep-Score. Then, we select the highest Rep-Score sequence $S_{high\text{-}score}$ and compare each sequence with this to find the number of matching pattern. If the number of matching pattern exceeds the min pattern threshold we increment the counter. The number of matching pattern represents the % of similar patterns between the two comparing sequence. Finally, if counter is greater than min representative threshold we add $S_{high\text{-}score}$ to the representative set. Min representative threshold is identified as the minimum number of sequence that a representative sequence must have in order to represent a group of potential similar sequence. The algorithm terminates when there is k number of representative sequence in the set. The stop condition can also be the database coverage that is minimum number of sequence that a representative set must cover.

The selection scoring function discussed above has captured three important concepts such as sequence length, pattern length and significance of pattern frequencies with respect to their appearance in the sequence. Furthermore, the selection algorithm tries to select the representatives from the database such that each represents a potential group of similar sequences.

4.4 Sequence Comparison Method

The initial clustering is based on the cluster representatives. Every sequence in the database is compared against all the cluster representatives. The comparison between two sequences is based on sequence similarity. The sequence similarity is measured in terms of frequent patterns and its weighted frequencies. The most commonly used similarity measure in document classification is the cosine measure and defined as follows. Let v_1 and v_2 be two sequence vectors represented by the frequency of frequent patterns. Therefore, their cosine similarity is defined as:

$$\textit{Similarity}\ (v_1, v_2) = \frac{v_1 . v_2}{|v_1||v_2|} \tag{4.3}$$

In equation 4.3, inner dot product $v_1 . v_2$ is the standard vector dot product defined as $\sum_{i=1}^{t} v_{1i} . v_{2i}$, and the norm $|v_1|$ in the denominator is defined as $|v_1| = \sqrt{v_1 . v_1}$. However, this measure cannot distinguish the similarity between v_1, v_2 and v_2, v_1 which is important in the case of measuring similarity between two proteins. Therefore, we define a similarity measure as the ratio of matched versus unmatched frequent patterns calculated in terms of their weighted frequencies. The similarity function (*Sim-Score*) of sequence *a, b* is defined as:

$$\text{Sim-Score}\,(a \rightarrow b) = \frac{\sum_{\forall p} w_i \times \log_2^{|p_i|}}{\sum_{\forall \bar{p}} w_j \times \log_2^{|\bar{p}_j|}} \tag{4.4}$$

Here, $p = a \cap b$ is the set of all matched frequent patterns in sequence a and b, w_i is the weighted frequency of common patterns in a and $|p_i|$ is the pattern length. In the denominator, $\bar{p} = a - (a \cap b)$ is the set of all unmatched frequent patterns in sequence a, w_j is the weighted frequency of unmatched pattern in a, and $|\bar{p}_j|$ is the pattern length. This function represents the ratio of similarity over dissimilarity of sequence a against sequence b. The similarity of sequence a against b is not significant when the ratio is less than min threshold i.e. number of patterns matched is not significant in sequence a with respect to the min threshold. When ratio is 1 that means sequence a is approximately 50% similar to sequence b and maximum similarity occurs when denominator is 1 i.e. all the patterns in sequence a is matched with sequence b. The higher score represents higher similarity of sequence a against b. Similarly, we find *Sim-Score* $(b \rightarrow a)$ i.e. similarity of sequence b against a. Finally, we use the geometric mean of above scores and so, the *Sim-Score* $(a \leftrightarrow b)$ is defined as:

$$\text{Sim-Score}\,(a \leftrightarrow b) = [\text{SimScore}\,(a \rightarrow b) * \text{SimScore}\,(b \rightarrow a)]^{½} \tag{4.5}$$

The motivation behind the use of geometric mean is that, two sequences are considered to be highly similar only if both values of *SimScore* $(a \rightarrow b)$ and *SimScore* $(b \rightarrow a)$ are high.

Example 4: Let us calculate the similarity score for sequence S_1 and S_3 given in Table1. S_1 and S_3 have two common frequent patterns i.e. ACD and YM, and one uncommon pattern in both sequences i.e. MN and DI respectively. So, the score is:

$$\textit{Sim-Score}\ (S_1 \rightarrow S_3) = \frac{4.64 \times \log_2{}^3 + 2.32 \times \log_2{}^2}{2.32 \times \log_2{}^2} = 4.16$$

$$\textit{Sim-Score}\ (S_3 \rightarrow S_1) = \frac{2.32 \times \log_2{}^3 + 2.32 \times \log_2{}^2}{2.32 \times \log_2{}^2} = 2.58$$

$$\textit{Sim-Score}\ (S_1 \leftrightarrow S_3) = \sqrt{4.16 \times 2.58} = 3.28$$

***Sim-Score* (a ↔ b)**	***Score***	***Sim-Score* (a ↔ b)**	***Score***
Sim-Score ($S_1 \leftrightarrow S_3$)	*3.28*	*Sim-Score* ($S_2 \leftrightarrow S_3$)	*0.40*
Sim-Score ($S_1 \leftrightarrow S_4$)	*1.58*	*Sim-Score* ($S_2 \leftrightarrow S_4$)	*0.42*
Sim-Score ($S_1 \leftrightarrow S_5$)	*0.40*	*Sim-Score* ($S_2 \leftrightarrow S_5$)	*6.58*

Table 4.5: *Sim-Score* for all sequences against cluster representatives

Table 4.5 contains the *Sim-Score* for all sequences against the cluster representatives. The following scores show that S_3, S_4 are highly similar to cluster representative S_1 and S_5 is highly similar to cluster representative S_2.

4.5 Detection of Multi-Domain Protein

One of our protein clustering challenges is to detect the multi-domain protein efficiently and effectively. We use the similarity scoring function (*Sim-Score*) to detect these proteins. The similarity function is based on the ratio of common over uncommon frequent pattern scores. The multi-domain protein is generally larger than the single domain proteins and contains groups of highly conserved regions in different domains of that protein. If we follow Figure 4.1 we find that

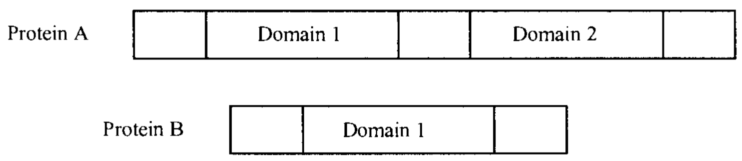

Figure 4.1: Multiple domain protein

protein B is highly similar to protein A but, A is not significantly similar to B because of large non-matching regions. Therefore, we can detect this difference by using the ratio of similarity scores in the following way:

$$\textit{Multi-Domain-Score} = \frac{Sim - Score(b \rightarrow a)}{Sim - Score(a \rightarrow b)} \tag{4.6}$$

If sequence *a* is a multi-domain protein then *Sim-Score (b → a)* will be very high and *Sim-Score (a → b)* will be very low. When the ratio is greater 2, it is highly probable that sequence *a* is a multi-domain protein. But, this may occur by chance. Therefore, in order to conclude that sequence *a* is a multi-domain protein, we select a set of representative such that their comparison scores are higher than the min threshold. Now, for this significant representative set we calculate the *Multi-Domain-Score* and if most of the scores (say at least 75%) are greater than 2, then only we can conclude that sequence *a* is a multi-domain protein. The ratio between 2 to 3 indicates that sequence *a* may contain two domains. Thus, higher ratio follows the higher number of domains.

There are some multi-domain proteins which are not significantly larger than other proteins i.e. similar length. During the representative selection process our selection method discriminates the sequences by their length and frequent patterns. Therefore, there is a little chance that we may choose a multi-domain protein as one of the cluster representatives because of having average length. In order to detect this in later clustering phase, we check if the *Multi-Domain-Score* is greater than 2 and the comparison score is significant for a given sequence then, we mark the representative sequence and wait until we have enough sequences to judge this. Finally, if the representative sequence is a multi-domain protein we split the cluster.

Therefore, we have two levels of screening for detecting the multi-domain proteins. First, we filter the multi-domain proteins during the representative selection stage and finally, we detect them during the building stage of initial clustering.

4.6 Optimize the intra-cluster similarity

We can refine the intra-cluster similarity by improving the appearance of patterns in the cluster. So, the pattern appearance of cluster C_i can be defined as:

$$PatternAppearance\ (C_i) = cluster\text{-}frequen\text{-}\ patterns\ /cluster\text{-}nonfrequent\text{-}patterns \qquad (4.7)$$

where, cluster frequent pattern represents the number of patterns that are frequent within the cluster and vice versa. The higher score of *Pattern-Appearance* represents higher occurrences of patterns in the cluster which means sequences within the cluster are sharing larger number of patterns i.e. higher similarity among the sequences and vice versa. So, maximizing the score of *Pattern-Appearance* means increasing the appearance of patterns and thus, increasing the similarity among sequences in the cluster. The algorithm for measuring intra-cluster similarity *Intra-Sim* for a given cluster C_i is given below:

Intra-Sim Algorithm

Input: Cluster C_i, Output: Refined Cluster

```
1. Calculate Pattern-Appearance
2. Rank the sequence by Sim-Score
```

```
3. Repeat
    a. Select the lowest scoring sequence $S_{lowest}$
    b. Recalculate Pattern-Appearance$_{new}$
    c. Calculate Pattern-Appearance-Ratio (equation 4.8)
    d. If Pattern-Appearance-Ratio > min-ratio-threshold then
        i. Replace the old scores
        ii. Remove the sequence from the cluster
4. Until min optimization threshold is reached
```

Algorithm 4.2: Algorithm for measuring intra-cluster similarity

In Algorithm 4.2, first we calculate the *Pattern-Appearance* for that cluster and then, rank the sequences according to *Sim-Score.* After ranking, we choose the lowest scoring sequence and recalculate the *Pattern-Appearance.* If the score is significantly improved therefore, we can conclude that the selected sequence does not belong to the cluster at this level of cluster quality specified by *Pattern-Appearance.* Hence, we remove this sequence from the cluster. We define the significant scores as a ratio of new and old pattern appearance in the following way:

$$\text{Pattern-Appearance-Ratio} = \frac{PatternAppearance_{new}}{PatternAppearance_{old}} \tag{4.8}$$

Therefore, we consider the new score as significant when the ratio is greater than min ratio threshold which is at least 1. We continue the above procedure until specified optimization threshold is achieved or there is no sequence for which the quality measure can be improved further.

4.7 Clustering Method

Our Clustering method takes the following inputs from user such as sequence set *S*, similarity threshold *t* and distance threshold *d*. Our clustering method completes in two phases. In phase one, a set of sequence is selected in order to represent the initial clustering i.e. each initial cluster is represented by a sequence known as cluster representative. The selection process is solely based on the *Rep-Score* (Representative selection score) function and *Rep-Selection* algorithm defined in earlier section. The initial clustering is constructed using the cluster representatives. Every sequence in database S is compared using the *Sim-Score* sequence comparison function against all cluster representatives. First, we check whether the comparing sequence is a multi-domain protein using the multi-domain detection procedure defined in earlier section. If we find the sequence is a multi-domain protein, we mark and assign it to the corresponding significant clusters for which *Multi-Domain-Score* > 2. If the sequence is not a multi-domain protein we assign it to the best cluster provided the similarity score is above the *min similarity threshold t*. In this case, we are assigning sequence to a single best scoring cluster only. This completes phase one and the construction of initial clustering.

In the second phase, first, we refine the initial clusters in order to improve the closeness among the sequences within a cluster and remove any chance similarities. Then, a set of new representative is selected for each cluster using the *Rep-Scores* except the multi-domain sequences that are marked in the earlier phase. Then, we search for additional sequences that may be distantly related using the new representatives. Here, we are choosing more than one representative per cluster in order to avoid the chance similarities of distant proteins. Therefore, we select only those sequences where search hit is more than one i.e. sequences which are found similar to more than one representative. We apply the same sequence comparison function for comparing sequences against the new representatives. This completes the first iteration for finding the distant relatives and the distance is two. The distance here means the number of intermediate sequences required to find the distant relatives. Then, we continue to the next iteration. This process continues until no more sequences can be found or *min distance threshold d* is achieved.

```
Clustering Algorithm CRS
Input: sequence set S, min similarity threshold t, distance threshold d
Output: Clustering of S
  Phase 1
  1. Compute Rep-Score function for all sequence in S
  2. Select a set of sequence $R \subset S$ using Rep-Selection algorithm. Set R is known as the
     representative set for initial clustering.
  3. For each sequence $S_k$ in set S-R
       a. compare against set R using Sim-Score function
       b. if $S_k$ is a multi-domain protein (using multi-domain detecting procedure)
            i. add $S_k$ to set R for which Multi-Domain-Score > 2
       c. else, assign $S_k$ to best representative $R_j$ provided Sim-Score > t
  4. For each $R_i$ in set R
       a. if $R_i$ is a multi-domain protein (using multi-domain detecting procedure)
            i. split the corresponding cluster represented by $R_i$

  Phase 2
  5. Refine the initial clustering using Pattern-Appearance (section 4.6)
  6. round = 0
  7. Repeat
       a. Select a new representative set $R_i'$ for each cluster $C_i$
       b. for each cluster $C_i$ and each sequence $S_k$ in set S- $C_i$
            i. compare against set $R_i'$ using Sim-Score function
            ii. if $S_k$ is similar to more than one representative in $R_i'$
                 1. assign $S_k$ to $C_j$
       c. round++
  8. Until round > d for all clusters or no more sequence can be added
  9. Merge clusters if necessary (section 4.8)
  10. Return the final clustering
```

Algorithm 4.3: Clustering Algorithm CRS

The setting of distance threshold is a crucial problem and unfortunately, there is no definite answer to solve this problem. However, the study in [33] discussed the issue in great detail and gave some experimental evaluation. It shows that, a substantial proportion of true homologues have distance two or larger, with a significant drop off at distance five. Therefore, one up to four intermediate sequences are needed to cover about 50% of the super-family pairs. However, still a sizable proportion has larger distance up to a maximum of 13. Hence, the distance threshold can be set as low as 2 to maximum of 13. However, we have to be careful in setting the threshold because the larger distance increases the chances of selecting false homologue.

The complexity of our clustering algorithm is $n \times r \times t$, where n is the number of sequences in the database, t is the number of iterations and r is the number of representatives. The efficiency of our algorithm greatly depends on the number of representatives that are needed to compare with the database sequences. The worst case scenario is when r is equal to n then it is equivalent to all against all comparisons.

One of our clustering goals is efficiency that we want to avoid all against all sequence comparisons. Though the above clustering method requires much less than the all against all comparisons, we can improve it by introducing the concept of consensus or imaginary sequence. Instead of selecting a set of representative sequence at each iteration for each cluster we use a consensus sequence and search for the distant relatives. The new clustering method CRCS differs only in the second phase. The algorithm is as follows:

Clustering Algorithm CRCS

```
Phase 2
1. Refine the initial clustering using Pattern-Appearance (section 4.6)
2. Repeat
    a. Build or Update an consensus sequence $I_i$ for each cluster $C_i$ using the cluster
       frequent patterns
    b. for each $I_i$ and for each sequence $S_k$ in S- $C_i$
        i. Use $I_i$ to search for distance relative
```

```
            ii. If $S_k$ and $I_i$ satisfies the min similarity threshold
                1. Assign $S_k$ to $C_i$
3. Until no more sequences OR no changes in consensus sequence for all clusters
4. Merge clusters if necessary (section 4.8)
5. Return the final clustering
```

Algorithm 4.4: Clustering Algorithm CRCS

In CRCS, first we refine or improve the clusters by increasing the average pattern appearance for each cluster. The reason for refining the clusters is to improve the closeness of the sequences within a cluster and to define a better consensus sequence. Now, we define a consensus sequence for each cluster by using all the cluster frequent patterns. Cluster frequent patterns are those that appear more than a specified threshold within a cluster. These cluster frequent patterns can be easily computed during the initial cluster building process.

After defining the consensus sequence, we search for distant relative in the database. We compare the database sequences against the set of consensus sequence and assign them to the corresponding clusters if their similarity holds true. Then, update the consensus sequences and continue with the same procedure until no more sequence can be added to the clusters or no change in the consensus sequences for all clusters.

4.8 Merging Clusters

After completing second phase of the clustering, there may be some clusters which are similar and need to be grouped together for better clustering result. In order to merge the similar clusters we need to find a way to compare the clusters and measure the similarity. One way to measure the similarity of two clusters is measuring the pair-wise similarity of the sequences between the clusters. However, this is all against all comparison of the sequences and computationally expensive. In order to compute the similarity between clusters efficiently we use consensus sequence for the comparison. So, we define a consensus sequence for each cluster if it is not done already.

Now, we use the consensus sequence to compare the similarity between clusters. For each cluster we use the pair-wise comparison of all consensus sequences and select all pairs that have scores more than some min threshold. We use the same comparison function *Sim-Score* (equation 4.5) to compute the similarity. Then, for all chosen pairs we calculate the *Pattern-Appearance* and select one pair that has the best quality measure score i.e. higher pattern appearance. Then, we merge the corresponding clusters. The procedure continues until it finishes comparing all the clusters.

We use the quality measure for determining the merging pair so that the selection process is not solely based on the consensus sequences. Now, there may be a question, why we are not using the quality measure only for selecting the cluster pairs. We cannot use the quality measure directly to deduce the similarity between clusters because the scores are not comparable as a similarity measure.

We can use the above merging procedure to further organize the protein sequences into hierarchy. We can use a range of cluster merging threshold to detect the different classes of hierarchy. For lower class identification we can set very high threshold and low threshold for the higher classes.

4.9 Splitting Clusters

There may be situations where cluster splitting is necessary when multi-domain protein is mistakenly selected as representative sequence. In our clustering method, this threat is eliminated by checking for multi-domain proteins during the initial cluster building process. For splitting a cluster, we apply the representative selection procedure on the corresponding cluster and select a set of representatives. Then, assign the remaining sequences to the representatives based on the scoring function. In this case, the number of representative reflects the possible number of domains present in the corresponding multi-domain representative sequence that causes the cluster to split.

Chapter 5: Experimental Evaluation

This section provides the details of experiments and evaluation using our clustering method. The dataset, evaluation method and running time are also discussed. For the experiments, evaluations and comparisons we consider only CRS and CRCS. We distinguish CRC using multi-domain procedure as $CRC_{multi\text{-}domain}$ and without using multi-domain procedure as $CRC_{original}$. Similarly, we define the $CRCS_{multi\text{-}domain}$ and $CRCS_{original}$.

5.1 Dataset

We choose three experimental dataset from the prosite database. The prosite database is highly hand-crafted and reliable as we mentioned earlier. It provides flat clustering with overlaps. We randomly choose ten clusters for dataset DS1. The details of DS1 are given in Table 5.1.

Prosite AC	# of Sequence
PS00011	57
PS50089	320
PS00221	117
PS00171	121
PS50198	36
PS00012	195
PS00078	223
PS00116	111
PS00447	45
PS00956	16
Total Sequence	1241

Table 5.1: Dataset 1

Each row in the Table 5.1 represents a prosite cluster identified by the accession number (AC) and number of sequences in that cluster. The second dataset DS2 contains first dataset and 10 more randomly chosen clusters identified as: PS00226, PS00795, PS00411, PS00930, PS00414, PS00242, PS00704, PS00162, PS50059 and PS00194. Total number of sequences in DS2 is 2151. We consider third dataset DS3 with 50 clusters and total number of sequences 11400.

Further details of these clusters can be found in the prosite website: http://au.expasy.org/prosite/. The sequences in these clusters are accessed from the Swiss-prot database and their web site is: http://au.expasy.org/sprot/.

5.2 Evaluation method

In order to evaluate the clustering quality we use a commonly used measurement called F-measure first introduced in [36]. It is an oft-used method in the information retrieval and natural language processing communities. It is also standard method for evaluating both flat and hierarchical clustering. As we defined earlier $C = \{C_1, C_2, \ldots\ldots C_m\}$ be some clustering, i.e. a set of clusters and $K = \{K_1, K_2, \ldots\ldots K_k\}$ be the natural classes of a database. Let $n_{i,j}$ be the number of members of natural class K_i in cluster C_j. The recall $R(K_i, C_j)$, precision $P(K_i, C_j)$ of natural class K_i and cluster C_j can be calculated as:

$$R(K_i, C_j) = \frac{n_{i,j}}{|K_i|} \tag{5.1}$$

$$P(K_i, C_j) = \frac{n_{i,j}}{|C_j|} \tag{5.2}$$

Therefore, F-measure $F(K_i, C_j)$ can be defined as:

$$F(K_i, C_j) = \frac{2 \times R(K_i, C_j) \times P(K_i, C_j)}{R(K_i, C_j) + P(K_i, C_j)} \tag{5.3}$$

$F(K_i, C_j)$ represents the quality of cluster C_j for natural class K_i. The overall F-measure *F(C)* is the weighted sum of the maximum *$F(K_i, C_j)$* of all the classes defined as follows:

$$F(C) = \sum \frac{|K_i|}{|S|} \quad \left(\max C_j \in C\{F(K_i, C_j)\}\right) \tag{5.4}$$

where, *S* denote the total number of sequences in the database. The value of *F(C)* lies in [0, 1] and larger value indicates the higher quality of the clustering.

5.3 Experimental results

We compare the experimental results between CRS and CRCS with the presence and absence of multi-domain procedure. The results are presented below:

	Overall F-measure F(C)			
	$CRS_{original}$	$CRCS_{original}$	$CRS_{multi\text{-}domain}$	$CRCS_{multi\text{-}domain}$
DS1	0.42	0.47	0.66	0.69
DS2	0.43	0.46	0.645	0.71
DS3	0.43	0.48	0.65	0.695

Table 5.2: Experimental Results for the use of Multi-Domain Procedure

The results clearly show the importance of using multi-domain detection procedure in clustering protein sequences in both algorithms. Using multi-domain procedure substantially improves the quality of the results. The results also show the consistency over the number of sequences and number of classes. As the number of sequence and classes are increased from DS1 to DS3, the quality of the results does not change significantly.

In order to show that, our careful representative selection procedure results in better clustering result compare to the randomly selection procedure, we perform several experiments using CRS and CRCS with and without using our Rep-Selection-Algorithm. Here, CRS_{random} and

$CRCS_{random}$ represent the algorithms using random selection procedure and consequently, $CRS_{Rep-Selection}$ and $CRCS_{Rep-Selection}$ represent the algorithms using Rep-Selection procedure. The results are presented in the following table 5.3.

	Overall F-measure F(C)			
	CRS_{random}	$CRS_{Rep-Selection}$	$CRCS_{random}$	$CRCS_{Rep-Selection}$
DS1	0.54	0.66	0.58	0.69
DS2	0.55	0.645	0.56	0.71
DS3	0.54	0.65	0.59	0.695

Table 5.3: Experimental Results for the use of Rep-Selection Procedure

For both methods, the results show that, use of Rep-Selection Algorithm gives much better results than the use of random selection procedure. In the case of random selection procedure, the representatives are selected such that there may be sequences which are extremely short or extremely large or do not contain good patterns as we discussed in the earlier chapter. As a result, it distorts the clustering quality.

The results further show that consensus based representative CRCS performs better than the CRS where set of sequence is used as representative. Therefore, consensus based representation of the clusters are better than representing by a single sequence or a set of sequence. In the following table, we show the quality of clusters by using set size as a parameter.

	Overall F-measure F(C)		
	Dataset	Set size	$CRC_{multi-domain}$
$CRC_{multi-domain}$	DS1	1	0.62
	DS1	2-3	0.66
	DS1	6	0.57
	DS1	9	0.50
	DS1	12	0.46

Table 5.4: Experiment results for set size as a parameter

The results indicate that the quality of the clusters degrades as the number of sequence increases in the set. Therefore, the best set size we achieved is 2 or 3. The similar results are also found for the DS2 and DS3.

We also perform experiments in order to show the cut-off points for the distance threshold *d*. The results show that clustering quality improves until *d* is 3 and after that quality degrades dramatically. Therefore, increasing the number of intermediate sequences increases the chances for false positives to be included in the clusters. Hence, we can conclude that, the cut-off point for the distance threshold is 3. The results are shown in the following Table 5.5.

	Overall F-measure F(C)	
	Distance Threshold *d*	$CRC_{multi\text{-}domain}$
$CRC_{multi\text{-}domain}$	1	0.62
	2	0.65
	3	0.66
	4	0.58
	5	0.47
	6	0.39

Table 5.5: Experiment results for distance threshold as a parameter

5.4 Efficiency and Scalability

The algorithm is highly scalable and efficient. We measure the efficiency of the algorithm in terms of sequence numbers. The following chart shows the run time for $CRS_{original}$ and $CRS_{multi\text{-}domain}$. It indicates that run time does not increase exponentially with the increase of sequence numbers. The runtime for CRS is slightly higher than CRCS. This is due to extra computation associated with choosing a group of sequence as the representative set and comparing with other sequences.

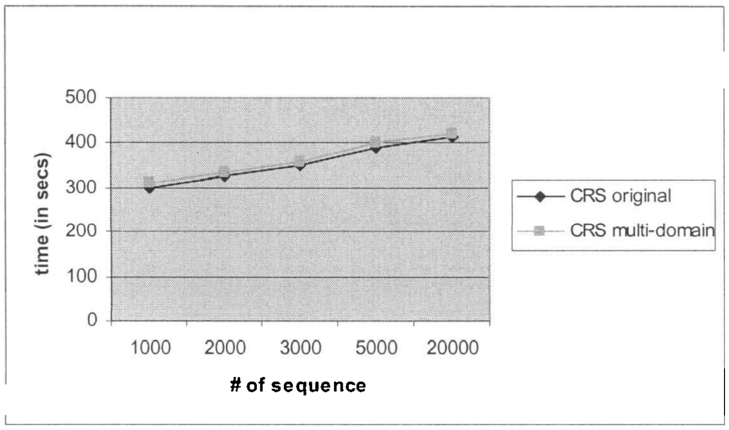

Figure 5.1: Runtime for CRS

Similarly consistent run time holds for the $CRCS_{original}$ and $CRCS_{multi\text{-}domain}$. The graph is as follows:

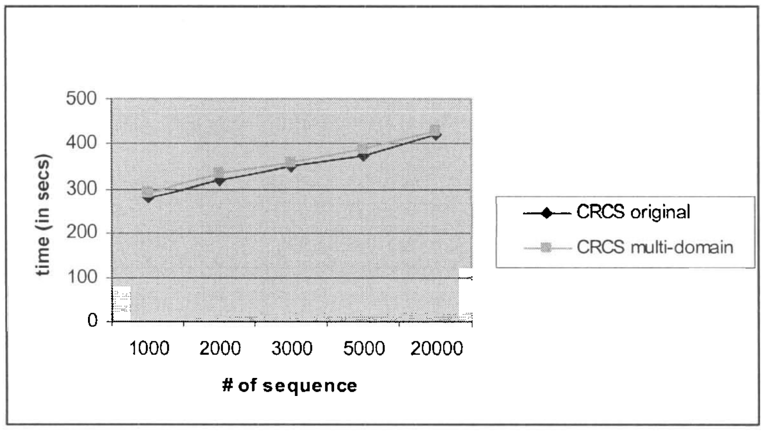

Figure 5.2: Runtime for CRCS

Chapter 6: Discussions and Future Studies

6.1 Solving Protein Clustering Problem

In this project, we have tried to solve the problem of clustering protein sequences by exploiting the sequence similarity in an efficient manner. It is one of the most important and challenging problems in genome studies. We have addressed three major challenges in clustering related proteins. The three main challenges are (1) homology detection problem, (2) multi-domain detection problem and (3) scalability.

We use transitivity to detect the homology and furthermore, we develop two methods one using group of sequence and another using consensus sequence to improve the quality of the homology detection. We successfully exploit the similarity measure to find the multi-domain proteins without introducing extra level of complexity. Multi-domain detection substantially helps to improve the quality of the clustering. Our algorithm is highly scalable compare to other existing methods as it can avoid computing pair-wise sequence comparison. This is evident from the run time of our algorithm.

We also develop a novel sequence comparison technique. Our similarity function captures the significant similarity information embedded in the sequence such as frequent patterns and sequence length. We compute similarity in both ways in order to ensure the quality of the comparison by exploiting geometric mean in the calculation. This also helps later finding the multi-domain proteins. This approach can be used in text mining to find documents containing multiple topics or in other sequence mining to identify different set of events.

In conclusion, we have developed a simple, efficient and effective representative based clustering technique for protein sequences. This method can be easily applied to other areas of the sequence and text mining. Our method is different from other representative based clustering methods such as k-medoids [37] and its variants PAM [37] and CLARA [37]. Unlike these methods, our algorithm does not require pre knowledge of cluster numbers. The main difference is initial selection of the representatives. Instead of randomly choosing a representative and then optimizing it, we carefully choose a set of meaningful representative by analyzing the sequence. Then, we apply one step optimization in order to remove some outliers and further we optimize the representative by selecting a group of sequence or constructing a consensus sequence for each cluster.

6.2 Possible Future Studies

The quality of our clustering results is encouraging and consistent. The results show that we achieve a good quality clustering for all the datasets. It will be really interesting to see how the other methods perform using the same dataset and compare with them.

The clustering quality can further be improved if we employ profile based sequence searching instead of using consensus sequence. The idea is to construct a profile for each cluster by using the local alignment on the sequences within the cluster. Thus, profile captures more information in describing the cluster. Finally, use these profiles in representing the clusters.

A typical protein contains domain and motifs (patterns). In our method, we intend to use domain information in clustering proteins at the higher level. The domain also captures significant information about the proteins. Further studying domains in details can give us crucial information about the functionalities of proteins. However, studying domain in depth requires some expertise in the field.

One important future study would be considering the properties of the amino acids in calculating the sequence similarity and finding the distant relatives. The properties capture significant physico-chemical information of the sequences. Further exploring these properties in an effective manner can dramatically change the resulting quality of the clusters. In the appendix A, we give a detail description of how to use the properties for further improving the clustering quality by capturing more sequence information into the similarity function.

Bibliography

1. Sigrist C.J., Cerutti L., Hulo N., Gattiker A., Falquet L., Pagni M., Bairoch A., Bucher. PROSITE: a documented database using patterns and profiles as motif descriptors.Brief Bioinform. 3:265-274(2002).
2. Falquet L., Pagni M., Bucher P., Hulo N., Sigrist C.J, Hofmann K., Bairoch A.The PROSITE database, its status in 2002.Nucleic Acids Res. 30:235-238(2002).
3. Boeckmann B., Bairoch A., Apweiler R., Blatter M.-C., Estreicher A., Gasteiger E., Martin M.J., Michoud K., O'Donovan C., Phan I., Pilbout S., Schneider M. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003 Nucleic Acids Res. 31:365-370(2003).
4. Attwood, T.K., Blythe, M., Flower, D.R., Gaulton, A., Mabey, J.E., Maudling, N., McGregor, L., Mitchell, A., Moulton, G., Paine, K. and Scordis, P. PRINTS and PRINTS-S shed light on protein ancestry. Nucleic Acids Research, 30(1), 239-241 (2002).
5. Henikoff, S. and Henikoff, J.G. Automated assembly of protein blocks for database searching. Nucleic Acids Research 19, 6565-6572 (1991).
6. Henikoff, J.G., Greene, E.A., Pietrokovski, S. & Henikoff, S. Increased coverage of protein families with the blocks database servers. Nucleic Acids Research 28:228-230 (2000).
7. Sonnhammer, E.L.L. and Kahn, D. Modular arrangement of proteins as inferred from analysis of homology. Protein Science 3:482-492 (1994).
8. Servant F, Bru C, Carrère S, Courcelle E, Gouzy J, Peyruc D, Kahn D. ProDom: Automated clustering of homologous domains. Briefings in Bioinformatics. vol 3, no 3:246-251 (2002).
9. Jérôme Gracy, Patrick Argos. Automated protein database classification: I. Integration of compositional similarity search, local similarity search and multiple sequence alignment. II. Delineation of domain boundaries from sequence similaritites. Bioinformatics, 14(2), 164-187 (1998).
10. Valerie Guralink, George Karypis. A Scalable Algorithm for Clustering Sequential Data. IEEE Data Mining 2001.
11. Valerie Guralink, George Karypis. A Scalable Algorithm for Clustering Protein Sequences. BIOKDD01:Workshop on Data Mining in Bioinformatics, 2001.
12. Krause, A and Vingron, M. A set theoretic approach to database searching and clustering. Bioinformatics, 14:5, 430-438 (1998).
13. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. Basic local alignment search tool. Journal of Molecular Biology, 215, 403-410 (1990).

14. Yona, G., Linial, N., Tishby, N. and Linial, M. A map of the protein space - anautomatic hierarchical classi_cation of all protein sequences. In Proceedings of theSixth International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, CA (1998).

15. Yona, G., Linial, N. and Linial, M. ProtoMap: Automatic classi_cation of protein sequences, a hierarchy of protein families, and local maps of the protein space. PROTEINS: Structure, Function, and Genetics, 37, 360-378 (1999).

16. Yona, G., Linial, N. and Linial, M. ProtoMap: automatic classi_cation of protein sequences and hierarchy of protein families. Nucleic Acids Research, 28, 9-55 (2000).

17. Kriventseva, E. V., Fleischmann, W., Zdobnov, E. M. and Apweiler, R. CluSTr: a database of clusters of SWISS-PROT+TrEMBL proteins. Nucleic Acids Research, 29, 33-36 (2001).

18. E.Bolten, A.Schliep, S.Scheneckener, D.Schomburg, R.Schrader. Clustering Protein Sequences - Structure Prediction by Transitive Homology. (2000).

19. Bateman A, Birney E, Cerruti L, Durbin R, Etwiller L, Eddy SR, Griffiths-Jones S, Howe KL, Marshall M, Sonnhammer EL. The Pfam Protein Families Database. Nucleic Acids Research 30(1):276-280 (2002).

20. Rabiner, L. R., Lee, C. H., Juang, B. H., and Wilpon, J. G. Hmm clustering for connected word recognition. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 1989.

21. Cen Li and Gautam Biswas. Clustering Sequence Data using Hidden Markov Model Representation.

22. krogh, A., Brown, M., Mian, I., Sjolander, K. and D. Haussler. Hidden Markov Models in computational biology: application to protein modeling. Journal of Molecular Biology, 235:1501-1531, 1994.

23. Baldi, P., Chauvin, Y., Hunkapiller, T. and McClure, M.A.. Hidden Markov models of biological primary sequence information. Proceedings of the National Academy of Sciences of the United States of America, 91(3): 1059-1063, 1994.

24. Eddy, S.R.. Multiple alignment using Hidden Markov models. Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, 114-120. AAAI press, 1995.

25. R.Apweiler, T.K.Attwood, A.Bairoch, A.Bateman, E.Birney, M.Biswas,P.Bucher, L.Cerutti, F.Corpet, M.D.R.Croning, R.Durbin, L.Falquet, W.Fleischmann, J.Gouzy, H.Hermjakob, N.Hulo, I.Jonassen, D.Kahn, A.Kanapin, Y.Karavidopoulou, R.Lopez, B.Marx, N.J.Mulder, T.M.Oinn, M.Pagni,F.Servant, C.J.A.Sigrist, E.M.Zdobnov. The InterPro database, an integrated documentation resource for protein families, domains and functional sites. Nucleic Acids Research vol 29(1): 37-40 (2001).

26. Pearson, W.R. Effective protein sequence comparison. Methods Enzymol, 266 227-258 (1996).

27. Pearson, W.J. and Lipman, D.J. Improved tools for biological sequence comparison. Proc. Natl. Acad. Science USA, 85, 2444-2448 (1988).

28. Smith. T.F. and Waterman M.S. Comparison of biosequences. Adv. Appl. Math, 2, 482-489 (1981).

29. Anton J. Enright and Christis A. Ouzounis GeneRAGE: A robust algorithm for sequence clustering and domain detection. Bioinformatics Vol. 16, no. 5, 451-457 (2000).

30. Ori Sasson, Nathan Linial and Michal Linial, The metric space of proteins-comparative study of clustering algorithms. Vol. 18, S14-S21 (2002).

31. Andreas Heger and Lisa Holm, Picasso: generating a covering set of protein family profiles. Bioinformatics Vol. 17, No. 3, 272-279 (2001).

32. Federico Abascal and Alfonso Valencia, Clustering of proximal sequence space for the identification of protein families. Bioinformatics Vol. 18, No. 7, 908-921 (2002).

33. P.Pipenbacher, A.Schliep, S.Schneckener, A. Schonhuth, D. Schomburg and R. Schrader, ProClust: improved clustering of protein sequences with an extended graph-based approach. Bioinformatics Vol. 18, S182-S191, (2002).

34. Weizhong Li, Lukasz Jaroszewski and Adam Godzik. Tolerating some redundancy significantly speeds up clustering of large protein databases. Bioinformatics, Vol. 18, No. 1, 77-82, (2002).

35. Jaroszewski Li W., and Godzik A. Clustering of highly homologous sequences to reduce the size of large databases. Bioinformatics, 16, 458-464, (2000).

36. Van Rijsbergen C. J., Information Retireval. Butterworths, London, 1979.

37. Kaufman, L. and Rousseeuw, P. J., Finding groups in data: An Introduction to Cluster Analysis, New York:John Wiley & Sons, 1990

38. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and A. Verkamo, I., Fast discovery of Association rules. In Advancesin Knowledge Discovery and Data Mining, pages 307–328, 1996.

39. Agrawal, R.and Srikant, R., Fast algorithms for mining association rules. The nternational Conference on Very Large Databases, pages 487–499, 1994.

40. Han, J. and Kamber, M., Data Mining: Concepts and Techniques. Morgan Kaufmann publishers, 2001.

Appendices

Appendix A: Exploring physico-chemical properties of proteins

In addition to sequence similarity, we want to use physico-chemical properties of amino acids to compare sequences. The rationale is amino acids are the basic components of constructing primary structure of a protein sequence. Each amino acid has several different properties and collection of these properties determine the classes and functions of proteins. The properties provide us useful information on sequence characteristics, for example: *how much charge or hydrophobicity is present in a sequence?* Therefore, without considering the properties of amino acids we cannot guarantee a better comparison of sequences. Hence, the properties are incorporated in our sequence comparison function. The detail property table for all amino acids is given below:

Size	***Large***	***Medium-large***	***Medium***	***Small***
	SGA	KEQM	VLDNPTIC	RYHWF
Charge	***Basic***	***Acidic***	***Neutral***	
	HRK	DE	QPGYCVIWLMFATSN	
Polarity	***Non-polar***	***Polar***		
	PAILVF	YHKEQDNRSTGWMC		
Hydrophobicity	***Hydrophilic***	***Neutral***	***Hydrophobic***	
	KEQDNRSTG	PAYHC	IWLMVF	
Aliphaticity	***Non-aliphatic***	***Aliphatic***	***Amphipathic***	
	CHKEQDNRSTG	PAILMV	YWF	
Aromatic	***Yes***	***No***		
	YHWF	ACDEGIKLMNPQRSTV		
Confirmation	***Flexible***	***Inflexible***	***N/A***	
	G	P	ACDEFHIKLMNQRSTVWY	

Figure 7.1: Physico-Chemical Property Table

There are total 7 physico-chemical properties of amino acids that we are discussing here such as *size*, *charge*, *polarity*, *hydrophobicity*, *aliphaticity*, *aromatic* and *confirmation*. Each property is further classified into several groups, for example the *charge* property has three groups i.e. *basic, acidic* and *neutral.* Now, our goal is to compare sequence using the distribution of property groups in the sequence e.g. how much similarity of *basic, acidic* and *neutral* are present in the sequence.

Let $g = \{g_1 \ldots\ldots g_q\}$ be the frequency of amino acids in database S. Amino acid has several different physico-chemical properties and each property is further classified into several groups. Each amino acid supports many different groups. Let $z = \{z_1 \ldots\ldots z_d\}$ be a set of property group for all amino acids and $|z_k|$ is the number of amino acids supporting k-th property group. Let $Z = \{Z_1 \ldots\ldots Z_d\}$ be a set of frequency for property groups i.e. occurrence of each property group in database S and Z_k is defined as:

$$Z_k = \sum_{g_i \in z_k} g_i$$

Let $\overline{Z} = \{\overline{Z}_1 \ldots\ldots \overline{Z}_d\}$ be a set of mean frequency for each property group in the database and so, $\overline{Z}_k$ is defined as:

$$\overline{Z}_k = Z_k / n$$

where, n is the total number of sequences in the database. The property table is discussed more in the later section.

We use z-Score or standard score function to find the score for each property group. The z-score of Z_k property group is defined as:

$$\text{z-Score } (Z_k) = \frac{Z_k - \overline{Z}_k}{s_k}$$

where, Z_k represents the frequency of a sequence of k-th property group, $\bar{z}_k$ represents the mean frequency of k-th property group, and s_k is the standard deviation. We measure the similarity of physico-chemical properties by taking the absolute difference of z-Scores of each property group in the comparing sequences. The higher difference of z-Scores represents less similarity and vice versa. We define the similarity of property groups of sequence a and b as a penalty function in the following way:

$$\text{Prop-Penalty (a, b): } \sum_{i=1}^{d} | zScore_{i,a} - zScore_{i,b} |$$

The penalty is calculated by adding the differences of z-Scores of all property groups for a given comparing sequences. The zero penalties means the distribution of property groups appeared equally in the sequences i.e. similarity is maximal in terms of physico-chemical properties.

Thus, final comparison function can be combined with equation 4.5 i.e. using both *Sim-Score* and *Prop-Penalty* scoring function. Therefore, the *Final-Score* function is defined as:

$$\text{Final-Score (a, b): Sim-Score } (a \leftrightarrow b) - \log(\text{Prop-Penalty} + 1)$$

The second term of this function is 0 when *Prop-Penalty is* 0 i.e. no penalty. When *Prop-Penalty* is greater than 0, the term penalizes the *Sim-Score* accordingly. Therefore, higher penalty will result in lower final score and thus lower similarity. The motivation behind the use of penalty is further discriminating the *Sim-Score* because sometimes the score may not well reflect the sequence similarity. When two sequences are highly similar in terms of *Sim-Score*, it is more likely that penalty for those sequences will also be lower. If penalty is not low that means the *Sim-Score* is overly calculated and thus, minimizing by the penalty function. We can further explain using the same example used in the chapter 4 as follows:

Example 7.1: The following table contains mean $\bar{z}_k$ and standard deviation s of *charge* property for the database given in Table 4.1. We calculate the z-Scores for sequence S_1, S_2 and S_4.in the table. For simplicity we are giving example for one property only.

Property	**Groups**	$\bar{z}_k$	**s**	**z-Score(S_1)**	**z-Score(S_2)**	**z-Score(S_4)**
charge	*Basic*	1	0.7	0	0	0
	Acidic	2	1.22	0	1.6	-0.82
	Neutral	7	2.34	0	-0.38	-0.42

Table 7.1: Mean, Std. Dev. and *z*-Score for *charge* Property

We calculate the *Final-Score* for S_1, S_4 and S_2, S_4 given in table 4.1. The scores are given in the following table. From the table, we can conclude that sequence S_1,S_4 are highly similar compare to S_2, S_4 with respect to their physico-chemical similarity.

Prop-Penalty (a, b)	***Score***	***Final-Score (a, b)***	***Score***
Prop-Penalty (S_1, S_4)	*1.24*	*Final-Score (S_1, S_4)*	*1.23*
Prop-Penalty (S_1, S_4)	*2.46*	*Final-Score (S_2, S_4)*	*-0.14*

Table 7.2: *Prop-Penalty* and *Final-Score* for S_1, S_4 and S_2, S_4.