Automatic Protein Structure Classification through Structural Fingerprinting

Zeyar Aung Kian-Lee Tan
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
Email: {zeyaraun, tankl}@comp.nus.edu.sg

Abstract

In this paper, we present a new scheme named "CP-Mine" for automatic three-dimensional (3D) protein structure classification using structural fingerprints. We represent a 3D protein structure as a CPset*, which is a set of inter-SSE* contact patterns (CPs) *existing in the protein. Suppose we have a database of protein structures whose class labels are already known, and suppose there are n distinct protein structure classes in the database. For each class, we generate its* fingerprint *by mining the* frequent CPsets *from all the member protein structures belonging to this class. When we want to predict the class label of an unknown protein, we also generate the CPset of this protein, and find the intersection between this CPset and the fingerprint of each protein structure class one by one. Then, the labels of the classes with the highest degree of intersection are returned as the answer. The proposed method is a pure classification scheme in that any kind of structural comparison, alignment or searching is not needed to be performed. The preliminary experimental results shows that our method can classify the protein structures accurately and efficiently.*

1. Introduction

Three dimensional (3D) protein structure analysis is an important area in bioinformatics. Protein structure analysis involves such tasks as protein structure comparison, classification, modelling, and prediction. Among them, researchers have been developing numerous automated methods for structural comparison, modelling and prediction since the last decade. But, only a few automated structural classification methods have been proposed up until now. Instead, people rely on the manual and semi-automatic classification methods such as *SCOP* [12] and *CATH* [15]. Or, they use the traditional structural comparison or alignment methods such as *DALI* [10] and *VAST* [7] for classification purpose.

Because of the advancements in the laboratory methods to determine the 3D structures of the proteins, the sizes of the protein structure databases such as *PDB* [2] are growing at very rapid rates. Nowadays, 10 to 20 proteins are added to PDB everyday. Thus, determining the respective structural classes[1] of these new proteins by the manual or the semi-automatic methods becomes inadequate. Classifying a new protein through traditional structural comparison also becomes inefficient because it involves the comparison of the new protein against all the proteins (or the representative set) in the large database. Although faster database searching methods such as *ProtDex* [1] are available, they are not designed specifically for classification, and thus still require quite some time to search through the entire database.

Our objective is to develop an efficient, dedicated and automated protein structure classifier without having to perform any structural comparison or searching. We use data mining techniques to generate the structural *fingerprints* for the various structural classes. A fingerprint is a set of concise and representative features that are common to all the protein structures in a given class. When we want to classify an unknown protein structure, we can use these fingerprints as the indicators for efficient and effective classification.

Our proposed *CPMine* scheme can be briefly described as follows. Suppose we have a database of 3D protein structures (in PDB format) whose class labels are already known. We use SCOP as the golden standard in assigning the class labels to the proteins. A class label of a protein is its SCOP designation – Class, Fold, Superfamily, or Family – depending on the level of detail required. We represent each protein structure in the database as a *CPset*. A CPset is a set of

1 The term 'class' in the machine learning literatures refers to a group in general. It should not be confused with the term 'Class' in SCOP hierarchy.

Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04)
0-7695-2173-8/04 $ 20.00 © 2004 IEEE

discretized feature vectors representing the inter-SSE *contact patterns (CPs)* that exist in the protein. For each distinct protein structure class in the database, we find the *maximal frequent CPsets* from all the member protein structures (or some representative members) in the class. A *frequent itemset mining* algorithm named *Eclat* [18] is used to mine these frequent CPsets whose numbers of occurrences exceed the predefined *minimum support (minsup)*. The resultant multiple frequent CPsets are collectively designated as the fingerprint of this class. When we want to predict the class label of an unknown protein, we again generate the CPset of this protein. For each class, we take each frequent CPset in its fingerprint at a time, and probe it into the CPset of the unknown protein. Then we calculate the score of each class based on the count of the probes that are matched. The labels of the classes that receive the best scores (after some adjustments, weightings and normalization) are returned as the answer. The experiential results shows that we can always achieve the correct classification in the best 20% classes out of the total number of classes (e.g., the best 3 classes out of total 15 classes) with 81% reliability.

The rest of the paper is organized as follows. Section 2 gives some background information relevant to the proposed scheme. Section 3 describes the proposed scheme in detail. Section 4 presents the experimental results of the scheme, and Section 5 concludes the paper.

2. Background

This section discusses the basic concepts regarding the protein structures, classification of the protein structures, and some of the previous works that are related to the proposed scheme.

2.1. 3D Protein Structures

A protein is composed of a sequence of *amino acid (AA)* residues, which folds into a 3D structure by the various forces of nature. The 3D structure of a protein is represented as a set of 3D positions (x, y, z coordinates) of all the atoms in a protein in a 3D reference frame. The shape of a protein can be roughly determined by the positions of the central carbon atoms (C_α atoms) of the AA residues. A protein structure can be logically represented as a sequence of 3D positions of all the C_α atoms in the protein.

In a protein, certain portions of its AA sequence folds into specific shapes such as *helices* (denoted as 'H') and *sheets* (denoted as 'E') due to the forces of nature such as hydrogen bonds and disulphide bonds. Such a shape is called a *secondary structure element (SSE)*. The arrangement of SSEs within a protein is called its *topology*. Identifying SSEs within a protein structure is subjective depending on the assumption of the various parameters. In our system, we use *STRIDE* [6] algorithm to identify and annotate SSEs in the protein structures.

Protein Data Bank (PDB) [2] is the most popular and widely-used resource in protein structure analysis. It stores the 3D structural information and annotations on several proteins. For each protein, PDB stores the 3D coordinates of all the atoms that compose the molecule, alongside with other information (such as chain information, SSE information and annotations). Nowadays (as of March 2004), PDB stores over 24,000 protein structures.

2.2. Protein Structure Classification

According to the machine learning literatures, classification is a kind of supervised learning. First, we have a collection of objects with their class labels already known. Then, for each unique class, we train the classifier with the positive examples of the objects that belong to the class, and optionally, the negative examples of the objects that do not belong to the class. The classifier learns the classification rules from the training data and reapplies these rules to classify the new unknown objects. *Support vector machines (SVMs)*, *neural networks* and *Bayesian networks* are some commonly used classification methods.

In our case, we have a database of 3D protein structures with their structural class labels (Class, Fold, Superfamily or Family) already known according to the existing classification databases. (*SCOP* [12] and *CATH* [15] are two well-known classification databases. They are manually and semi-automatically constructed respectively.) For each unique structural class, we train our classifier with the positive and, optionally, the negative example structures. We predict the structural classes of the new coming structures using the rules learned by the classifier from the training examples. Structural classification is different from structural clustering schemes such as Fischer *et al.* [5] in which unsupervised learning is used to build the classes of the structures whose class labels are not known in advance.

2.3. Related Work

There are quite a number of protein structure classification methods such as [4, 8] that use the AA sequences as their input data. But, on the other hand, to our knowledge there only exists a few pure protein structure classifiers that take the 3D structures as their input data. Many people use the traditional structural comparison methods such as SSAP and DALI for classification purpose. This comparison approach incurs high computational cost unnecessarily because structural comparison is a computationally expensive task. A pure or dedicated classifier does not need to carry out any comparison and can accomplish the required classification task with much lower computational cost.

Huan *et al.* [11] proposed a dedicated protein structure classifier based on *coherent subgraph analysis*. It presents a 3D protein structure as a feature vector of the number occurrences of the coherent subgraphs representing the peptide and proximity relations among the AA residues. It then uses SVMs to train and classify these feature vectors.

Røgen and Fain [16] described another dedicated protein structure classifier that uses *Gauss integrals* to represent a 3D protein structure as a 30-dimensional feature vector. For each structural class, it constructs a cluster containing the member proteins in 30-dimensional space. It then uses nearest cluster search to predict the most appropriate class for a new coming protein.

3. CPMine Method

In this section, we will discuss our proposed CPMine method in detail. Section 3.1 describes how a 3D protein structure can be represented as a CPSet. Section 3.2 explains how structural fingerprints of the protein structure classes are built by frequent CPset mining. Section 3.3 describes how the class of a new unknown protein can be predicted by using these structural fingerprints.

3.1. Representing a Protein Structure

In this sub-section, we will discuss how a 3D protein structure can be represented by a *CPset*.

3.1.1. Distance Matrix. A 3D protein structure can be represented as a 2D *distance matrix*. This representation is popular and has been used in DALI [10] and other protein structure analysis methods. A distance matrix is rotation and translation invariant. The formal definition of a distance matrix is given below.

Definition 1 *Distance Matrix*
A distance matrix D representing a particular protein with n AA residues is an $n \times n$ matrix, in which each cell $D(i,j)$, where $1 \leq i,j \leq n$, stores the Euclidean distance[2] *$Dist(i,j)$ between the C_α atom of the i^{th} residue and C_α atom of the j^{th} residue. The distance matrix is symmetrical along the main diagonal, i.e., $D(i,j) \equiv D(j,i)$.*

For example, let us consider a small fragment of protein structure P which is excerpted from protein *1tph1* (residue id 90–100). The fragment has 11 AA residues and 2 SSEs, with C_α atom coordinates and SSE assignments as shown in Figure 1. We can construct a distance matrix for protein fragment P as shown in Figure 2. All the measurements are in Angstrom (Å).

2 Distances that are greater than 50Å are considered not very significant and are just taken as merely 50Å.

Residue No.	x	y	z	SSE Type
1	13.6	27.1	7.7	E
2	13.3	25.8	11.2	E
3	16.0	25.4	13.9	E
4	15.7	22.0	15.5	E
5	17.7	20.4	18.4	-
6	19.0	23.7	19.8	-
7	20.9	23.2	23.1	H
8	18.3	25.2	25.0	H
9	15.5	22.8	23.7	H
10	17.7	19.8	24.7	H
11	18.9	20.9	28.1	H

Figure 1. 3D coordinates of C_α atoms in protein fragment P.

		E 1	E 2	E 3	E 4	5	6	H 7	H 8	H 9	H 10	H 11
E	1	0.0	3.7	6.9	9.6	13.3	13.7	17.5	18.0	16.7	18.9	22.0
E	2	3.7	0.0	3.8	6.2	10.0	10.5	14.4	14.7	13.0	15.4	18.5
E	3	6.9	3.8	0.0	3.8	6.9	6.8	10.7	11.3	10.2	12.3	15.2
E	4	9.6	6.2	3.8	0.0	3.9	5.7	9.3	10.4	8.2	9.7	13.0
	5	13.3	10.0	6.9	3.9	0.0	3.8	6.3	8.2	6.2	6.3	9.8
	6	13.7	10.5	6.8	5.7	3.8	0.0	3.8	5.5	5.3	6.4	8.8
H	7	17.5	14.4	10.7	9.3	6.3	3.8	0.0	3.8	5.4	4.9	5.9
H	8	18.0	14.7	11.3	10.4	8.2	5.5	3.8	0.0	3.9	5.4	5.3
H	9	16.7	13.0	10.2	8.2	6.2	5.3	5.4	3.9	0.0	3.9	5.9
H	10	18.9	15.4	12.3	9.7	6.3	6.4	4.9	5.4	3.9	0.0	3.8
H	11	22.0	18.5	15.2	13.0	9.8	8.8	5.9	5.3	5.9	3.8	0.0

C^{11} C^{12} C^{21} C^{22}

Figure 2. Distance matrix of protein fragment P.

3.1.2. Inter-SSE Contact Pattern (CP). We can approximate the overall 3D shape of a protein structure by the shapes and topology of its constituent SSEs. Consequently, when a 3D protein structure is presented as a distance matrix, we can approximate the overall structure of the protein by the combination of the properties of the individual inter-SSE *contact patterns (CPs)* that are formed by the interaction of the SSEs. In other words, we can capture most of the important structural properties of a protein structure in a set of CPs that present in its distance matrix. In our example protein fragment P with 2 SSEs, we have four CPs: C^{11}, C^{12}, C^{21} and C^{22}, which are shown as the gray areas in Figure 2. The formal definition of a CP is given below.

Definition 2 *Contact Pattern (CP)*
Within a distance matrix D (see Definition 1) of size $n \times n$, a contact pattern C^{ab} of two SSEs a and b is a sub-matrix containing the cells $D(s_a, s_b), \ldots, D(s_a + l_a - 1,\ s_b + l_b - 1)$, where s_a and s_b are starting AA residue numbers of SSE a and SSE b respectively $(s_a, s_b \leq n)$, and l_a and l_b are the lengths or the numbers of residues in SSE a and SSE b respectively $(s_a + l_a - 1 \leq n,\ s_b + l_b - 1 \leq n)$.

Since the distance matrix is symmetrical, only the CPs that are on or above the main diagonal are needed to be taken into account. In our example only the CPs C^{11}, C^{12}, and C^{22} are needed to be taken into account.

3.1.3. CP Feature Vector. Since a protein structure can be approximately represented by a set of its constituent CPs, we can encode and manipulate them in analyzing the protein structures. It is vital that we can encode these CPs in a concise manner while able to capture their important properties. For this purpose, we present a CP as a 7-dimensional *feature vector*. We can effectively and efficiently determine the similarity or compatibility between two given CPs by comparing their respective feature vectors. The feature vector of a CP C^{ab}, which is formed by SSE a and SSE b, consists of 7 attributes as shown in Table 1.

The first two attributes are derived from *3D vector representation* of the SSEs. Since a CP essentially represents the inter-relationship between the two SSEs, we can logically associate it with the angle and the vertex distance between the two SSE vectors. Angle and distance are the natural and most commonly used properties for the relationship between two SSE vectors, and are also used in many other methods such as *VAST* [7] and *LOCK* [17].

An SSE can be roughly approximated by its representative vector or line segment in 3D space. We adopt the equations for calculating the start point $\mathbf{V}_{start}$ and the end point $\mathbf{V}_{end}$ of an SSE vector $\mathbf{V}$ from Singh and Brutlag [17].

After having $\mathbf{V}_{start}$ and $\mathbf{V}_{end}$ of a vector, we can represent it as a *point vector* $\mathbf{V}$ with respect to the origin $\mathbf{0}$.

$$\mathbf{V} = \mathbf{V}_{end} - \mathbf{V}_{start} \tag{1}$$

Given two point vectors $\mathbf{V}$ and $\mathbf{U}$, representing two SSEs, we can calculate the *angle* θ between them as:

$$\theta(\mathbf{V}, \mathbf{U}) = \begin{cases} 0 & \text{if } \mathbf{V} = \mathbf{U} \\ \cos^{-1}\left(\frac{\mathbf{V} \cdot \mathbf{U}}{Dist(\mathbf{V},\mathbf{0}) \cdot Dist(\mathbf{U},\mathbf{0})}\right) & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{V} \cdot \mathbf{U}$ is the *dot product* of vectors $\mathbf{V}$ and $\mathbf{U}$, and $Dist$ is the Euclidean distance between two point vectors. Also, the *vertex distance* (VD) between two SSE vectors $\mathbf{V}$ and $\mathbf{U}$ can be calculated as:

$$VD(\mathbf{V}, \mathbf{U}) = \min \begin{cases} Dist(\mathbf{V}_{end}, \mathbf{U}_{start}) \\ Dist(\mathbf{V}_{start}, \mathbf{U}_{end}) \\ Dist(\mathbf{V}_{end}, \mathbf{U}_{end}) \\ Dist(\mathbf{V}_{start}, \mathbf{U}_{start}) \end{cases} \tag{3}$$

The rest of the attributes are derived directly from the distance matrix. Square-root of area and aspect ratio of CP attributes are related to the lengths of the SSEs. These two attributes are used in order to avoid the matching of two CPs that are formed by the SSE pairs with very different lengths. Mean and standard deviation of C_α-C_α distance attributes are related to the internal configuration of a CP. They are used in order to avoid the matching of two CPs which store very different C_α-C_α distance values. CP type is a natural property that is used for avoiding the comparison of two CPs formed by different types of SSEs respectively. The functions used to calculate these attribute values of the feature vector are defined as follows.

$$SA(a, b) = \sqrt{l_a \cdot l_b} \tag{4}$$

$$AR(a, b) = \min\left(\frac{l_a}{l_b}, \frac{l_b}{l_a}\right) \tag{5}$$

$$MD(a, b) = \frac{\sum_{i=0}^{l_a - 1} \sum_{j=0}^{l_b - 1} D(s_a + i, s_b + j)}{l_a \cdot l_b} \tag{6}$$

$$SD(a, b) = \sqrt{\frac{\sum_{i=0}^{l_a - 1} \sum_{j=0}^{l_b - 1} (D(s_a + i, s_b + j) - MD(a, b))^2}{l_a \cdot l_b}} \tag{7}$$

$$CT(a, b) = \begin{cases} 0 & \text{if } (a \text{ is H}) \wedge (b \text{ is H}) \wedge (a = b) \\ 1 & \text{if } (a \text{ is H}) \wedge (b \text{ is H}) \wedge (a \neq b) \\ 2 & \text{if } (a \text{ is E}) \wedge (b \text{ is E}) \wedge (a = b) \\ 3 & \text{if } (a \text{ is E}) \wedge (b \text{ is E}) \wedge (a \neq b) \\ 4 & \text{if } ((a \text{ is H}) \wedge (b \text{ is E})) \vee ((a \text{ is E}) \wedge (b \text{ is H})) \end{cases} \tag{8}$$

where the definitions of all the symbols are the same as their previous definitions respectively. Now, let

$$\mathbf{K} = (k_\theta,\ k_{VD},\ k_{SA},\ k_{AR},\ k_{MD},\ k_{SD},\ k_{CT})$$

be a feature vector. We can generate feature vector $\mathbf{K}^{ab}$ for CP C^{ab} as follows.

$$\mathbf{K}^{ab} = (\ \ \theta(a, b),\ VD(a, b),\ SA(a, b),\ AR(a, b),\ MD(a, b),\ SD(a, b),\ CT(a, b)\ \ ) \tag{9}$$

The feature vectors $\mathbf{K}^{11}$, $\mathbf{K}^{12}$, and $\mathbf{K}^{22}$ for their respective CPs C^{11}, C^{12}, and C^{22} in our example protein fragment P are shown in Table 1.

The feature vector we use is a good approximation of the original CP in an abstract form. Each of the 7 attributes in the feature vector is important in its own. We have conducted an experiment by excluding each attribute from the feature vector at a time, and observed that the expressive power of the feature vector degrades with each exclusion. (The results are not shown here.)

3.1.4. Discrete CP Feature Vector. In the original CP feature vector, except CP type attribute, all the other attributes are continuous-valued. However, the frequent itemset mining algorithm, which will be used in the next step, requires discrete values as input. Thus, we discretize each attribute value by mapping the value from the original space to the discretized space. The possible ranges for the original space and the discretized space for each attribute are given in Table 1. Some attributes are allocated larger discretized spaces (i.e., more number of bits) than the others

Dim	Attribute	Function	Equation	Original Space		Discretized Space		#Bits	Example continuous and discrete feature vectors in protein P					
				Min	Max	Min	Max		$\mathbf{K}^{11}$	$\mathbf{T}^{11}$	$\mathbf{K}^{12}$	$\mathbf{T}^{12}$	$\mathbf{K}^{22}$	$\mathbf{T}^{22}$
1	Angle between a and b	θ	2	0.0	180.0	0	15	4	0.0	0000	42.9	0011	0.0	0000
2	Vertex distance between a and b	VD	3	0.0	50.0	0	7	3	0.0	000	9.7	001	0.0	000
3	Square-root of the area of C^{ab}	SA	4	0.0	60.0	0	15	4	4.0	0001	4.5	0001	5.0	0001
4	Aspect ratio of C^{ab}	AR	5	0.0	1.0	0	7	3	1.0	111	0.8	110	1.0	111
5	Mean C_α-C_α distance in C^{ab}	MD	6	0.0	50.0	0	15	4	4.2	0001	14.0	0100	3.9	0001
6	Standard deviation of C_α-C_α distance in C^{ab}	SD	7	0.0	25.0	0	7	3	3.2	001	3.8	001	2.1	000
7	Type of C^{ab}	CT	8	0	4	0	4	3	2	010	4	100	0	000
	Total							24						

Table 1. Attributes of a CP feature vector.

because they are found to be relatively more important according to our experiments.

As an example, for angle attribute, we map an original real number angle value between 0.0 to 180.0 into one of the discrete *bins* numbered between 0 to 15 (i.e. 4 bit space). We use simple *equal partition discretization* as shown in Equation 10.

$$bin(Val) = floor(Val \times \frac{NumDiscreteBins}{OriginalUpperBound \times 1.001}) \quad (10)$$

where 1.001 is a safety factor. For instance, we can calculate the discretized value of 42.9 degree angle as: $floor(42.9 \times 16/(180.0 + 0.18)) = 3$.

Every attribute, except CP Type, is discretized in this way. Finally, we can present a CP as a 24-bit *discrete feature vector*. We can define a discrete CP feature vector $\mathbf{T}$ as a bit vector.

$$\mathbf{T} = (\quad \begin{matrix} bin(k_\theta)|\, bin(k_{VD})|\, bin(k_{SA})|\, bin(k_{AR})| \\ bin(k_{MD})|\, bin(k_{SD})|\, bin(k_{CT}) \end{matrix} \quad) \quad (11)$$

where bin is the discretization function (Eq.10) and $|$ is the concatenation operator of the bit strings. The resultant $\mathbf{T}^{11}$, $\mathbf{T}^{12}$, and $\mathbf{T}^{22}$ respectively for CPs C^{11}, C^{12}, and C^{22} in our example protein fragment P can be seen in Table 1.

We gain two fringe benefits from this discretization process. Firstly, it enables a compact and discrete encoding of a CP. Secondly, we can easily approximate the compatibility or incompatibility of two CPs by simply comparing their respective discretized feature vectors for equality.

3.1.5. CPset. Now, we can represent a 3D protein structure as a set of discrete CP feature vectors called a *CPset*.

Definition 3 *CPset*
A k-CPset S_k is a set of k *discrete CP feature vectors that occurs within a protein.*
$S_k = \{\mathbf{T_1}, \mathbf{T_2}, \ldots, \mathbf{T_k}\}$, *where* $\mathbf{T_i}$ $(1 \leq i \leq k)$ *is a discrete CP feature vector.*

The concept of CPset is borrowed from that of *itemset* which is used in general data mining. We can also draw an analogy between a protein and a *transaction*. In this way, we can view a protein as a set of CPs (CPset) – as we can view a transaction as a set of items (itemset).

It should be noted that we use sequence-order independent approach as in [13]. (The advantage of sequence-order independence is that some non-sequential sub-structures such as surface motifs can be easily detected.) In this approach, we only take care of the spatial arrangement of the SSEs in a 3D structure, and neglect the AA-sequence order of them. Hence, we also neglect the sequence order of the CPs in a distance matrix. Consequently, the order of the discrete CP feature vectors ($\mathbf{T_i}$) in a CPset is not important.

In this way, we can represent all the known protein structures in the database as their respective CPsets.

3.2. Building Structural Fingerprints

Now we have all the protein structures in the database represented as their respective CPsets. For each distinct protein structure class in the database, we apply *frequent itemset mining* on all the CPsets of the proteins belonging to this class in order to find the *maximal frequent CPsets* that occur in this class. We regard these frequent CPsets as the *structural fingerprint* representing this particular class. Usually, a fingerprint includes more than one CPset. Subsequently, we can use these fingerprints as the classification indicators to classify the new coming protein structures.

We do not use the standard classification tools such as SVMs and neural networks for two reasons. Firstly, the number of CPs in a CPset is variable. A large protein contains a large number of CPs in its CPset, whilst a small protein contains only a small number of them. Secondly, the order of CPs in a CPset is not fixed. A particular CP that occurs first in one CPset may occur last in another yet compatible CPset. These two reasons make our CPset representation inappropriate to be used with the standard classification tools.

Although frequent itemset mining is not a standard classification tool, it has been used successfully for classification purpose in some instances such as *CBA* [14].

3.2.1. Frequent CPset Mining. Frequent CPset mining is merely a specialization of the general frequent itemset min-

ing by treating CPsets as the itemsets. We adopt the following formal definition of frequent itemset mining from Han and Kamber, 2000 [9].

Let $\mathcal{J} = \{i_1, i_2, \ldots, i_m\}$ be a set of all possible items. Let $\mathcal{D}$, the task-relevant data, be a set of database *transactions* where each transaction T is a set of items such that $T \subseteq \mathcal{J}$. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. A set of items is referred to as an *itemset*. An itemset that contains k items is a k-itemset. The *occurrence frequency* of an itemset is the number of transactions that contain the itemset. An itemset satisfies *minimum support* ($minsup$) if the occurrence frequency of the itemset is greater than or equal to the product of $minsup$ and the total number of transactions in $\mathcal{D}$. If an itemset satisfies minimum support, then it is a *frequent itemset*. A *maximal frequent pattern* is a frequent pattern, p, such that any proper superpattern of p is not frequent.

In our CPMine method, we use *Eclat* [18] frequent itemset mining algorithm. We choose it because of its efficiency. We will not discuss Eclat algorithm in detail in this paper. The interested readers can refer to the original paper [18].

3.2.2. Fingerprint Generation Algorithm. Given a database of protein structures with known class labels, we can generate a structural fingerprint for each distinct class in terms of a set containing maximal frequent CPsets occurring in this class. The fingerprint generation algorithm is describes in Figure 3.

We first partition the protein structure database $\mathcal{D}$ into n partitions – each for a distinct structural class (line 1). We assign the minimum and maximum number of desired maximal frequent CPsets in a fingerprint as 10 and 30 respectively (line 2), and the initial minimum support ($minsup$) as 50% (line 4). (These values are determined experimentally.) Then for each structural class i, we build a set of CPsets S_i by compiling all the CPsets S_{ij}'s from the protein structures $\mathcal{D}_{ij}$'s belonging to this class (line 3–9). (The detailed procedure for **GenerateCPset** routine (line 7) is described in Section 3.1.) Then, we run Eclat algorithm on S_i with the initial $minsup$ in order to generate the fingerprint F_i for class i (line 10). If the number of frequent CPsets in the fingerprint are too few or too many, we adjust $minsup$ by a factor of 1.1 and run Eclat algorithm again (line 11–16).

1. Let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ be the database of protein structures where $\mathcal{D}_i$ $(1 \le i \le n)$ is a set of protein structure with class label i
2. $MinRow = 10$, $MaxRow = 30$
3. for $i = 1$ to n do
4. $\quad minsup = 0.5$, $S_i = \phi$
5. $\quad$ for $j = 1$ to $|\mathcal{D}_i|$
6. $\quad\quad$ Let $\mathcal{D}_{ij}$ $(1 \le j \le |\mathcal{D}_i|)$ be a particular protein structure in $\mathcal{D}_i$
7. $\quad\quad$ S_{ij} = **GenerateCPset**($\mathcal{D}_{ij}$)
8. $\quad\quad$ $S_i = S_i \bigcup S_{ij}$
9. $\quad$ end for
10. $\quad F_i$ = **Eclat**($S_i, minsup$)
11. $\quad$ if $|F_i| < MinRow$
12. $\quad\quad minsup = minsup/1.1$
13. $\quad\quad$ goto 10
14. $\quad$ else if $|F_i| > MaxRow$
15. $\quad\quad minsup = minsup \times 1.1$
16. $\quad\quad$ goto 10
17. $\quad$ end if
18. end for
19. return $F = \{F_1, F_2, \ldots, F_n\}$

Figure 3. Fingerprint generation algorithm.

3.3. Classification using Fingerprints

Now we have a set of structural fingerprints – one for each known structural class. Each fingerprint consists of 10–30 maximal frequent CPsets occurring in the protein structures of the respective class. If we want to classify a new protein structure whose class is not known yet, we represent it as a CPset, and try to predict its class by using algorithm described in Figure 4. It should be noted that we use multi-classification approach (as opposed to multiple binary classifications). The scores for all the classes are calculated in a single classification process.

We first generate the CPset of the query protein Q (line 1). Then for each distinct class i, we take its fingerprint F_i (line 2–5), and probes its constituent maximal frequent CPsets one-by-one into the CPset of Q (line 7–11), and count the hits (line 12). The score of class i is calculated based on the number of hits it receives (line 15). Some adjustments and weightings are done in calculating the score. Squaring of the hit percentage makes partial matches less significant (line 14). Multiplying with the log of number of discrete CPs in the frequent CPset ($lg(p)$) makes the longer frequent CPsets more significant than the shorter ones (line 15). Also, multiplying with the support of the frequent CPset ($Support(F_{ij})$) makes the frequent CPsets with greater support more important (line 15). Finally, we normalize the score of class i by dividing it with the number of frequent CPsets in its fingerprint (line 17). The class label of the fingerprint that receives most of the best score is returned as the most relevant class (line 21).

The algorithm shown in Figure 4 returns only the label of the single best class. We can optionally return the multiple labels of the best b classes for wider coverage. We use $b = 3$ in our experiments.

The overview of the entire CPMine method is presented in Figure 5.

1. Let $S = \{\mathbf{T_1}, \mathbf{T_2}, \ldots, \mathbf{T_a}\}$ be the CPset
   representing the protein Q with unknown class label,
   where $\mathbf{T_i}$ $(1 \leq i \leq a)$ is a discrete CP feature vector.
2. Let $F = \{F_1, F_2, \ldots, F_n\}$ be a set of fingerprints
   where F_i $(1 \leq i \leq n)$ is
   the fingerprint of structural class i
3. $MaxScore = 0$, $MaxClass = NULL$
4. for $i = 1$ to n do
5. Let us define $F_i = \{F_{i1}, F_{i2}, \ldots, F_{im}\}$
   where F_{ij} $(1 \leq j \leq m)$ is a particular
   frequent CPset within fingerprint F_i
6. $ClassScore = 0$
7. for $j = 1$ to m do
8. Let us define $F_{ij} = \{\mathbf{F_{ij1}}, \mathbf{F_{ij2}}, \ldots, \mathbf{F_{ijp}}\}$
   where $\mathbf{F_{ijk}}$ $(1 \leq k \leq p)$ is a discrete
   CP feature vector within frequent CPset F_{ij}
9. $numHit = 0$
10. for $k = 1$ to p do
11. if $\mathbf{F_{ijk}} \in S$ then
12. $numHit$++ /* if $\mathbf{F_{ijk}}$ exists in CPset of
    protein Q, count it as a hit */
13. end for
14. $CPScore = (numHit\,/\,p)^2$
15. $ClassScore += CPScore \cdot lg(p) \cdot Support(F_{ij})$
16. end for
17. $ClassScore = ClassScore\,/\,m$
18. if $ClassScore > MaxScore$ then
19. $MaxScore = ClassScore$, $MaxClass = i$
20. end for
21. return $MaxClass$

Figure 4. Classification algorithm.

4. Experimental Results

In order to assess the accuracy and efficiency of the proposed CPMine scheme, we conduct an experiment involving 600 protein structures extracted from *ASTRAL* data set [3] with less than 40% sequence homology. (ASTRAL provides the PDB-style 3D coordinate files for the protein domains as defined by SCOP.) From this data set, we choose protein *Folds* (according to SCOP designation) each having 40 or more member protein structures. There are 15 such Folds. We use these 15 Folds as our target structural classes.

From each Fold, we take 40 of its member proteins. (For Folds with more than 40 members, we randomly select 40 from them.) Thus, we have a pool of $15 \times 40 = 600$ protein structures whose class labels (Folds) are known.

We conduct our experiment using 10-*fold cross validation* strategy. We split each Fold into 10 partitions each having 4 protein structures. We conduct 10 sub-experiments. In

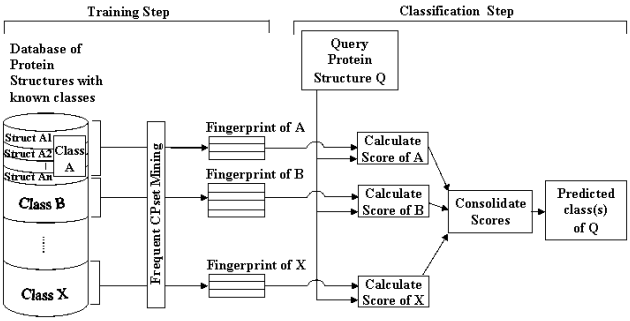

Figure 5. Overview of CPMine method.

each sub-experiment, we build the *testing data set* by choosing a partition from each Fold and combining them together. In this way, we have a testing data set consisting of 60 proteins for each sub-experiment. The remaining 540 proteins are used as *training data set*. The training data set is made up of 36 proteins from each of 15 Folds.

We use the training data set to build 15 structural fingerprints for these 15 Folds. Then, we classify protein structures in the testing data set using the constructed structural fingerprints.

In this way, 10 sub-experiments are conducted by using the different testing sets and training sets in each run. Then, we consolidate the results from 10 sub-experiments and calculate the average accuracy of the scheme. We look at the top 3 scorer classes (which is 20% of the total number of classes), and examine whether they are actually the correct classifications. The results are shown in Table 2. Column 'Top 1' shows the accuracy of the scheme if only the top 1 scorer is examined, and Column 'Top 2' shows the accuracy if both top 1 and 2 scorers are examined, etc.

From the experiments, it is observed that the proposed method performs quite accurately. In overall, it can give the correct classification within the top 3 answers with 81.33% accuracy. It performs very good on certain Folds such as 47472 and 48725 with 100% accuracy. However, it classifies poorly on certain Folds such as 53066. This is because the method produces some false positives in matching of the CPset belonging to a particular Fold to the fingerprints of the other Folds. We will try to reduce such false matchings in the future version of the system.

It is also observed that the proposed scheme works very efficiently. All the experiments are done on Pentium IV 2.6GHz machine with 1GB RAM and 80GB HDD running Windows XP. The average time for a training and testing cycle is only 2:30 minutes. The time statistics are also shown in Table 2.

Fold	Class	Average Accuracy (in Percents)			Average Time (in Seconds)	
		Top 1	Top 2	Top 3	Training	Testing
46688	All-α	80.0	85.0	92.5	2.01	0.21
47472	All-α	47.5	97.5	100.0	3.73	0.39
48370	All-α	50.0	72.5	85.0	10.78	1.18
48725	All-β	72.5	97.5	100.0	4.09	0.43
50198	All-β	30.0	62.5	85.0	4.45	0.47
51350	α/β	47.5	65.0	77.5	26.24	2.89
51734	α/β	52.5	85.0	87.5	12.14	1.33
51904	α/β	32.5	55.0	60.0	16.12	1.77
52171	α/β	42.5	72.5	95.0	8.79	0.95
52539	α/β	32.5	60.0	82.5	15.94	1.75
52832	α/β	25.0	37.5	62.5	4.63	0.49
53066	α/β	30.0	37.5	47.5	11.50	1.26
53473	α/β	47.5	60.0	70.0	24.80	2.73
54235	$\alpha+\beta$	50.0	70.0	85.0	3.46	0.36
54861	$\alpha+\beta$	32.5	67.5	90.0	4.63	0.49
Overall		44.83	68.33	81.33	133.30	16.71

Table 2. Experiential results on 15 protein Folds.

5. Conclusion

In this paper, we have presented a new fingerprinting scheme for protein structure classification. We use frequent itemset mining approach to build the structural fingerprints for the protein structure classes, and use them as the indicators for future structural classifications. Our system is a pure and dedicated classifier without requiring a costly structural comparison process. The preliminary experimental results shows that our method is accurate and very efficient.

As a future work, we can further assess the general behaviour of our scheme by trying it with a greater number of training and testing proteins taken from a greater number of structural classes. We can also improve the accuracy and efficiency of classification by introducing a better CPset feature vector representing scheme, a better discretization scheme, a better CPset probing and scoring scheme, etc.

Finally, we believe our scheme can become a very useful automatic tool for rapid and accurate protein structure classification in the age of very large protein structure databases.

References

[1] Z. Aung, W. Fu, and K. L. Tan. An efficient index-based protein structure database searching method. In *Proceedings of 8th International Conference on Database System for Advanced Applications (DASFAA'03)*, pages 311–318, 2003.

[2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[3] S. E. Brenner, P. Koehl, and M. Levitt. The astral compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

[4] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17:349–358, 2001.

[5] D. Fischer, C. J. Tsai, R. Nussinov, and H. J. Wolfson. A 3d sequence-independent representation of the protein data bank. *Protein Engineering*, 8:981–997, 1995.

[6] D. Frishman and P. Argos. Knowledge-based secondary structure assignment. *Proteins: Structure, Function and Genetics*, 23:566–579, 1995.

[7] J. F. Gibrat, T. Madej, and H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6:377–385, 1996.

[8] J. Grassmann, M. Reczko, S. Suhai, and L. Edler. Protein fold class prediction: new methods of statistical classification. In *Proceedings of 7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 106–112, 1999.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[10] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233:123–138, 1993.

[11] J. Huan, W. Wang, A. Washington, J. Prins, and A. Tropsha. Accurate classification of protein structural families using coherent subgraph analysis. In *Proceedings of 9th Pacific Symposium on Biocomputing (PSB'04)*, 2004.

[12] T. J. P. Hubbard, B. Ailey, S. E. Brenner, A. G. Murzin, and C. Chothia. Scop: a structural classification of proteins database. *Nucleic Acids Research*, 25(1):236–239, 1997.

[13] N. Leibowitz, Z. Fligelman, R. Nussinov, and H. J. Wolfson. Automated multiple structure alignment and detection of a common substructural motif. *Proteins: Structure, Function, and Genetics*, 43:235–245, 2001.

[14] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 80–86, 1998.

[15] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. Swindells, and J. Thornton. Cath: a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[16] P. Røgen and B. Fain. Automatic classification of protein structure by using gauss integrals. *Proceedings of the National Academy of Sciences of the United States of America*, 100(1):119–124, 2003.

[17] A. P. Singh and D. L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proceedings of 5th International Conference on Intelligent Systems for Molecular Biology (ISMB'97)*, pages 284–293, 1997.

[18] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 283–296, 1997.