

SP07065

Jaypee University of Information Technology
Waknaghat, Distt. Solan (H.P.)

Learning Resource Center

Class Num :

Book Num :

Accession No.: SP07065 / SP0711064

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date

ROUGH SET THEORETIC ANALYSIS OF BIOLOGICAL DATA (PROTEIN)

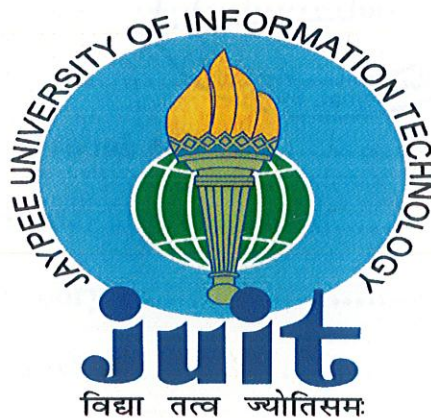
BY:

Group No. 49

Swarit Jasial(071520)

Prateek Mahajan(071529)

Name of supervisor – Mrs. Pooja Jain



Submitted in partial fulfillment of the Degree of

Bachelor of Technology

Department of Bioinformatics and Biotechnology

**JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH**

TABLE OF CONTENTS

Certificate.....	iv
Acknowledgement.....	v
List of figures.....	vi
List of Abbreviations.....	vii
Abstract.....	viii
Introduction.....	1
Objective and Scope.....	2
1.0 Basic concepts of rough sets.....	3-19
1.1 Lower and Upper approximation.....	3
1.2 Rough Set Theory.....	5
1.3 Information system.....	5
1.4 Indiscernibility and Independence.....	6
1.5 Opticians decision table.....	8
1.6 Simplification of decision table.....	10
1.7 Decision algorithm.....	18
2.0 Methods of Protein Classification.....	20
3.0 Rough Sets – A Better Method.....	21-23
4.0 Resources Used	24-26

5.0 Methodology.....	27-51
5.1 Classes in protein architecture.....	27
5.2 Conditional attribute.....	27
5.3 Process.....	28
5.3.1 Code for extraction of features.....	28
5.3.2 SCOP.....	35
5.3.3 MySQL.....	37
5.3.4 Database Connectivity.....	39
5.4 Data Flow diagram.....	51
5.5 Limitations.....	51
Testing.....	52
Conclusion.....	53
References.....	54

**JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY
WAKNAGHAT
SOLAN, HIMACHAL PRADESH**

CERTIFICATE

This is to certify that the work titled “**Rough Set Theoretic Analysis of Biological Data(Protein)**” submitted by “**Swarit Jasial**” and “**Prateek Mahajan**” in partial fulfillment for the award of degree of **B. Tech** of **Jaypee University of Information Technology, Wagnaghat** has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Pooja jain
.....

Name of Supervisor

Mrs. Pooja Jain

Date

25/5/2011
.....

ACKNOWLEDGEMENT

"Success of any endeavor is always due to contribution from different people"

Some time expressions are more reliable than words. Today we are short of words to express our deepest and numerous feelings to Almighty (The most beneficial and merciful) who gave us the mind, ability and courage to perform in a creative way.

We would like to express our sincere thanks to **Mr. Suman Saha** and **Mrs. Pooja Jain** for their invaluable guidance, patience, support and encouragement extended to us throughout the duration of the project.

We want to thank our parents for being with us and having faith in us everytime. We are highly thankful to them for their moral support.

Finally, we wish to express our gratitude to all those who have in one-way or other helped us in the successful completion of interim part of our project.

Swarit Jasial

Prateek Mahajan

List of Figures

Figure No.	Figure Caption	Page No.
1	Lower and Upper approximation	3
2	Rough Set Exploration System	24
3	RSES internal architecture	25
4	SCOP Homepage	36
5	Hierarchy of SCOP	37
6	Protein sequences database in WAMP	38
7	Feature extraction table generated in WAMP	47
8	.tab format file view	48
9	Table opened in RSES	49
10	Reduct set in RSES	50
11	Rules generated in RSES	50
12	Data Flow Diagram	51

List of Abbreviation/Symbols

Abbreviation	Meaning
RSES	Rough Set Exploration System
IS	Information System
SCOP	Structural Classification Of Proteins
SVM	Support Vector Machine
JVM	Java Virtual Machine
WAMP	Windows Apache Mysql Php

ABSTRACT

Every classification of biological data involves deep rooted study of its biological significance. So, in order to make classification easier some computational techniques must be applied to develop some general rules or algorithm for classification. For this purpose rough set theory can prove to be an ideal technique.

In this project first of all thorough study of Rough Set Theory techniques and fundamentals was done and later on it was implemented on biological data (protein classification) . As such the main focus of the project has been to find an easy way of classification of proteins.

Through this project it was tried to develop new decision rules for structural classification of proteins. Also methods to increase accuracy of the algorithm were studied and were taken into consideration.

INTRODUCTION

There are many ways of classification of biological data into different classes according to their biological significance. But this method is quite tedious as one has to do deep study of biological data and then according to its properties one has to classify it. Several details of the data has to be studied which may take a long time for classification.

Proteins are also classified into corresponding classes in the same way. Thus in order to make the process of classification simpler a new system must be developed which takes protein as input and classifies it into its protein class. A system has to be developed in which we do not have to consider the biological significance nor we have to do deep study of all the properties of each protein. Thus , such a system of classification will be much easier than classification by biological functions.

Several algorithms were developed in past out of which most of them were not able to predict the biological data classification accurately. Most of them gave the accuracy rate very low which is a big problem. So, such a method must be chosen whose accuracy is high.

OBJECTIVE AND SCOPE

The objective of our project is to develop some minimum decision rules for classification of biological data with the help of rough set theory and verify its accuracy. As earlier said that it is a tedious process to classify biological data by taking into consideration their biological significance, therefore we want to generate some decision rules by using rough set theory using which we can classify the biological data easily by following these rules.

These rules must be followed by any protein taken as input and it must be classified according to these rules. Thus, classification can become easy for users by following these rules. For this purpose our biological data must be converted to numerical form and then rough set concepts must be used on it.

Here we are concerned with protein data only therefore the main aim of our project is to classify a given protein sequence i.e. a sequence of amino acids into one of the four protein classes by following minimum decision rules developed using rough set theory.

These minimum decision rules can be developed by taking training data i.e. some protein sequences from a database and then extracting features from it and converting the data into numerical form and finally applying rough set theory to it to get some decision rules. Once these are generated then there is no need to study the relevance of proteins for classification as they can be easily classified by any user using these decision rules.

Also, these decision rules can also be developed not only for proteins but for other classifications in biology. We can use rough set theory for each and every process of classification in biological data based on rough set theory which will be much user friendly and save a lot of time.

Chapter 1

BASIC CONCEPTS OF ROUGH SET THEORY

1.1 Lower and Upper Approximation

Human knowledge about a domain is expressed by **classification**. Rough set theory treats knowledge as an ability to **classify** perceived objects into categories. Objects belonging to the same category are considered to be indistinguishable to each other. The primary notions of rough set theory are the **approximation space: lower and upper approximations of an object set**.

The lower approximation of an object set (S) is a set of objects surely belonging to S, while its upper approximation is a set of objects surely or possibly belonging to it. An object set defined through its lower and upper approximations is called a rough set. The boundary is defined as the difference between upper and lower approximations and contains elements which are in upper but not in lower approximation.

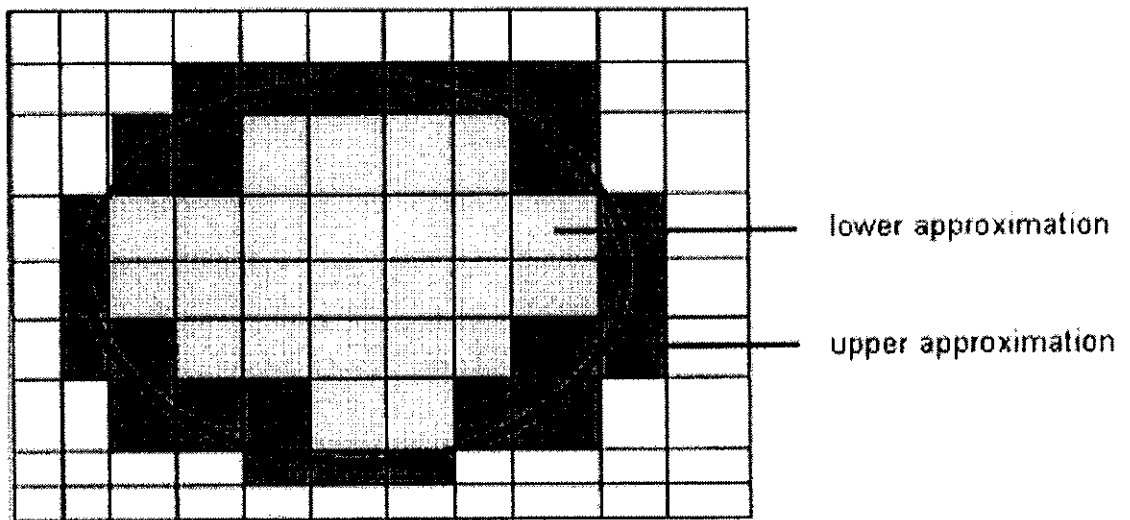


FIGURE 1

Let us assume that we are interested in the subset X of five objects $\{X=x_1,x_3,x_4,x_5,x_9\}$. Can we distinguish this set from whole data set in space of three attributes ($B=\{a_1,a_2,a_3\}$)? One can calculate the lower and upper approximation in the following way :

The elementary sets present in table 2 which are also contained in X are: $\{x_1,x_3,x_9\}$, $\{x_4\}$.

TABLE 2

U/A	a1	a2	a3
$\{x_1,x_3,x_9\}$	2	1	3
$\{x_2,x_7,x_{10}\}$	3	2	1
$\{x_4\}$	2	2	3
$\{x_5,x_8\}$	1	1	4
$\{x_6\}$	1	1	2

It means the lower approximation is given by following set of objects :

$$\{x_1,x_3,x_4,x_9\}$$

To calculate the upper approximation of subset X, one has to find in Table 2 all elementary sets which have atleast 1 element in common with the subset X.

These are :

$$\{x_1,x_3,x_9\} , \{x_4\}, \{x_5,x_8\}$$

So the upper approximation is :

$$\{x_1,x_3,x_4,x_5,x_8,x_9\}$$

The boundary is given by $BNX = \{x_1,x_3,x_4,x_5,x_8,x_9\} - \{x_1,x_3,x_4,x_9\} = \{x_5,x_8\}$

An accuracy measure of the set X in $B \subseteq A$ is defined as :

$$\mu_B(X) = \text{card}(\underline{BX})/\text{card}(\overline{BX})$$

The number of objects contained in the lower approximation of given example equals 4. The cardinality of the upper approximation equals 6. The accuracy of set X therefore is :

$$\mu_B(X) = 4/6.$$

1.2 Rough Set Theory :

Often, information on the surrounding world is

- **Imprecise**
- **Incomplete**
- **Uncertain**

We should be able to **process uncertain and/or incomplete information**. When **dealing with inexact, uncertain, or vague knowledge**, the rough set theory is used. Rough sets represent a different mathematical approach to **vagueness and uncertainty**. Rough set theory was introduced by Pawlak in 1985.

The rough set methodology is based on the premise that **lowering the degree of precision in the data** makes the data pattern more visible. Consider a simple example. **Two acids with pKs of respectively pK 4.12 and 4.53** will, in many contexts, be **perceived as so equally weak**, that they are **indiscernible** with respect to this attribute. **They are part of a rough set 'weak acids'** as compared to 'strong' or 'medium' or whatever other category, relevant to the context of this classification.

1.3 Information system

$IS=(U,A)$

U is the universe (a finite set of objects, $U=\{x_1,x_2,\dots, x_m\}$)

A is the set of attributes (features, variables)

V_a is the set of values a , called the domain of attribute a .

Consider a data set containing the results of three measurements performed for 10 objects. The results can be organized in a matrix 10×3 .

2	1	3
3	2	1
2	1	3
2	2	3
1	1	4
1	1	2
3	2	1
1	1	4
2	1	3
3	2	1

$IS=(U,A)$

$U=\{x_1, x_2, x_3, x_4, x_5, x_6, \dots, x_{10}\}$

$A=\{a_1, a_2, a_3\}$

The domains of attributes are :

$V_1=\{1,2,3\}$

$V_2=\{1,2\}$

$V_3=\{1,2,3,4\}$.

1.4 Indiscernibility and Independence

For every set of attributes $B \subseteq A$, indiscernible relation $Ind(B)$ is defined in the following way: two objects, x_i and x_j , are indiscernible by the set of attributes B

in A , if $b(x_i) = b(x_j)$ for every $b \in B$. The notation U/A means that we are considering elementary sets of the universe U in the space A .

TABLE 2

U/A	a1	a2	a3
{x1,x3,x9}	2	1	3
{x2,x7,x10}	3	2	1
{x4}	2	2	3
{x5,x8}	1	1	4
{x6}	1	1	2

Independence of attributes : In order to check , whether the set of attributes are independent or not , one checks for every attribute whether its removal increases the number of elementary sets in the IS or not.

TABLE 4

REMOVED ATTRIBUTE

	None	a1	a2	a3
No. of elementary sets	5	5	4	4

If the set of attributes is dependent, one can be interested in finding all possible minimal subsets of attributes.

The concept of core and reduct are two fundamental concepts of rough set theory.

Opticians Decision table is a good example of explaining core and reducts.

1.5 Opticians Decision Table

The example , which follow concerns an optician decisions as to whether or not a patient is suited to contact lens use. The set of all possible decisions is listed in Table 1.

U	a	b	c	d	e
1	1	1	2	2	1
2	1	2	2	2	1
3	2	1	2	2	1
4	3	1	2	2	1
5	1	1	1	2	2
6	1	2	1	2	2
7	2	1	1	2	2
8	2	2	1	2	2
9	3	2	1	2	2
10	1	1	1	1	3
11	1	1	2	1	3
12	1	2	1	1	3
13	1	2	2	1	3
14	2	1	1	1	3
15	2	1	2	1	3
16	2	2	1	1	3
17	2	2	2	1	3
18	2	2	2	2	3
19	3	1	1	1	3
20	3	1	1	2	3
21	3	1	2	1	3
22	3	2	1	1	3
23	3	2	2	1	3
24	3	2	2	2	3

The **table 1** is in face a decision table in which a,b,c,d are condition attributes where as e is a decision attributes. The attribute e represents the optician's decisions which are the following :

1. Hard Contact lenses
2. Soft contact lenses
3. No contact lenses

These decision are based on some facts concerning the patients which are expressed by the condition attributes given below together with corresponding attribute values.

1. Age

- (a) young
- (b) pre-presbiopic
- (c) presbiopic

2. Spectacle

- (a) Myope
- (b) Hypermetrope

3. Astigmatic

- (a) no
- (b) yes

4. Tear production rate

- (a) reduced
- (b) normal

The problem we are going to discuss here is the elimination of conditions from a decision table, which are unnecessary to make decision specified in the table.

The first method is based on some operation in discernibility (equivalence) relations and has an algebraic flavor where as the second is embedded in logic. The algebraic approach to decision table simplification is straight forward; however, algorithms based on this method are sometimes not very efficient. The logical approach is easier to implement and gives faster algorithms than the algebraic method, but its intuitive meaning seems not to be so obvious as in the previous case therefore, for the sake of clarity we recommend combinations of both

methods, employing decision tables notation, and logical algorithms.

This means that we treat the decision tables as a set of formulas, which can be treated on a logical way. For example, row one in table one can be treated as the formula as the formula $a_1b_1c_2d_2 \rightarrow e_1$. In order to see whether this formula is true or not we could check the inclusion $|a_1b_1c_2d_2| = |e_1|$. However this position is rather inefficient. It is better in this case to employ proposition to check whether there is in the table another implication with the same predecessor and different successor. Because this is not the case, the implication is true.

1.6 Simplification of Decision Table

In view of what has been said so far our problem consists of:

- Checking whether elementary categories of decision attributes (decision categories) can be defined (expressed) in terms of basic categories defined by the set of all condition attributes (condition categories)
- Reduction of the set of condition categories necessary to define decision categories.

Obviously problem a. reduces in our case to the question of whether decision attribute e depends on condition attributes a, b, c and d, i.e., whether the dependency $\{a, b, c, d\} \rightarrow \{e\}$ holds as there is a total dependency between condition and decision attributes, i.e. the dependency $\{a, b, c, d\} \rightarrow \{e\}$ is valid. This means that the elementary categories of the attribute e (decision categories) can be uniquely defined by the categories of condition attributes (condition categories), or in other words, the values of decision attribute e are uniquely determined by means of values of condition attributes a,b,c and d.

As to the problem b. one can compute that the set of condition attributes is e-independent, that means none of the condition attributes can be eliminated, as a whole, without destroying the classificatory properties of the decision table; that is, removing any of the condition attributes from table 1 makes this decision table inconsistent. Let us try analyze this in detail.

Removing attribute a in table 1 we get a decision table (table 2), which is inconsistent because we have in table 2 the following pairs of inconsistent decision rules

$b2c2d2 = e1(\text{rule}2)$ and

$b2c2d2 = e3(\text{rule } 18 \text{ \& } 24)$

$b1c1d2 = e2(\text{rule } 5)$ and

$b1c1d2 = e3(\text{rule } 20)$

Thus the attribute a cannot be dropped.

U	b	c	d	e
1	1	2	2	1
2	2	2	2	1
3	1	2	2	1
4	1	2	2	1
5	1	1	2	2
6	2	1	2	2
7	1	1	2	2
8	2	1	2	2
9	2	1	2	2
10	1	1	1	3
11	1	2	1	3
12	2	1	1	3
13	2	2	1	3
14	1	1	1	3
15	1	2	1	3
16	2	1	1	3
17	2	2	1	3
18	2	2	2	3
19	1	1	1	3
20	1	1	2	3
21	1	2	1	3
22	2	1	1	3
23	2	2	1	3
24	2	2	2	3

TABLE 2

Similarly by removing the attribute b we get the following decision table

U	a	c	d	e
1	1	2	2	1
2	1	2	2	1
3	2	2	2	1
4	3	2	2	1
5	1	1	2	2
6	1	1	2	2
7	2	1	2	2
8	2	1	2	2
9	3	1	2	2
10	1	1	1	3
11	1	2	1	3
12	1	1	1	3
13	1	2	1	3
14	2	1	1	3
15	2	2	1	3
16	2	1	1	3
17	2	2	1	3
18	2	2	2	3
19	3	1	1	3
20	3	1	2	3
21	3	2	1	3
22	3	1	1	3
23	3	2	1	3
24	3	2	2	3

TABLE 3

In this table the following pairs of decision rules are incorrect.
 $a2c2d2 = e1$ (rule 3) and
 $a2c2d2 = e3$ (rule 18)

a3c2d2 = e1(rule 4) and
a3c2d2 = e3(rule 24)

a3c1d2 = e2(rule 9) and
a3c1d2 = e3(rule 20)

U	a	b	d	e
1	1	1	2	1
2	1	2	2	1
3	2	1	2	1
4	3	1	2	1
5	1	1	2	2
6	1	2	2	2
7	2	1	2	2
8	2	2	2	2
9	3	2	2	2
10	1	1	1	3
11	1	1	1	3
12	1	2	1	3
13	1	2	1	3
14	2	1	1	3
15	2	1	1	3
16	2	2	1	3
17	2	2	1	3
18	2	2	2	3
19	3	1	1	3
20	3	1	2	3
21	3	1	1	3
22	3	2	1	3
23	3	2	1	3
24	3	2	2	3

TABLE 4
In which there are the following pairs of inconsistent rules

a1b1d2 = e1(rule 1) and
a1b1d2 = e2(rule 5)

$a1b2d2 = e1(\text{rule } 2)$ and
 $a1b2d2 = e2(\text{rule } 6)$

$a2b1c2 \rightarrow e1(\text{rule } 3)$
 $a2b1c2 \rightarrow e3(\text{rule } 15)$

$a3b1c2 \rightarrow e1(\text{rule } 4)$
 $a3b1c2 \rightarrow e3(\text{rule } 21)$

$a1b1c1 \rightarrow e2(\text{rule } 5)$
 $a1b1c1 \rightarrow e3(\text{rule } 10)$

$a1b2c1 \rightarrow e2(\text{rule } 6)$
 $a1b2c1 \rightarrow e3(\text{rule } 12)$

$a2b1c1 \rightarrow e2(\text{rule } 7)$
 $a2b1c1 \rightarrow e3(\text{rule } 14)$

$a2b2c1 \rightarrow e2(\text{rule } 8)$
 $a2b2c1 \rightarrow e3(\text{rule } 16)$

$a3b2c1 \rightarrow e2(\text{rule } 9)$
 $a3b2c1 \rightarrow e3(\text{rule } 22)$

As we can see from the analysis, none of the condition attributes can be removed from Table 0. Hence the set of condition attributes is e-independent. It is interesting to note that by dropping of various condition attributes we introduce inconsistency of different "depth

Coming back to our initial problem of simplification of the decision table, we have now to check whether we can eliminate some elementary condition categories, i.e. some superfluous values of condition attributes in Table 1. To this end, first we have to compute e-cores of each elementary condition category. In other words, we have to check which values of condition attributes are indispensable in order to discern the values of the decision attributes. Because the value core is the set of all indispensable values with respect to e, hence in order to check whether a specific value is dispensable or not (i.e. whether it belongs to the core) we have to remove the value from the table and see if the remaining values in the same row uniquely determine the decision attribute values in this row. If not, this value belongs to the core. Speaking in logical terms, we have to compute core values of each decision rule in the decision table i.e. find all those condition attribute values in the decision rule which make the decision rule false, if they are removed.

For example in the first decision rule $a1b1c2d2 \rightarrow e1$ values c2 and d2 are core values because the rules.

$b1c2d2 \rightarrow e1$
 $a1c2d2 \rightarrow e1$ are true, whereas the rules
 $a1b1d2 \rightarrow e1$
 $a1b1c2 \rightarrow e1$ are false.

All core values of each decision rule in Table 1 are given in Table 6.

U	a	b	c	d	e
1	-	-	2	2	1
2	1	-	2	2	1
3	-	1	2	2	1
4	-	1	2	2	1
5	-	-	1	2	2
6	-	-	1	2	2
7	2	-	1	2	2
8	-	-	1	2	2
9	-	2	1	2	2
10	-	-	-	1	3
11	-	-	-	1	3
12	-	-	-	1	3
13	-	-	-	1	3
14	-	-	-	1	3
15	-	-	-	1	3
16	-	-	-	1	3
17	-	-	-	1	-
18	2	2	2	-	3
19	-	-	-	-	3
20	3	1	1	-	3
21	-	-	-	1	3
22	-	-	-	1	3
23	-	-	-	-	3
24	3	2	2	-	3

TABLE 6

Reducts of each decision rule are given in table 7 below :-

U	a	b	c	d	e
1	X	1	2	2	1
1'	1	X	2	2	1
2	1	X	2	2	1
3	X	1	2	2	1
4	X	1	2	2	1
5	1	X	1	2	2
6	1	X	1	2	2
6'	X	2	1	2	2
7	2	X	1	2	2
8	2	X	1	2	2
8'	X	2	1	2	2
9	X	2	1	2	2
10	X	X	X	1	3
11	X	X	X	1	3
12	X	X	X	1	3
13	X	X	X	1	3
14	X	X	X	1	3
15	X	X	X	1	3
16	X	X	X	1	3
17	X	X	X	1	3
17'	2	2	2	X	3
18	2	2	2	X	3
19	3	1	1	X	3
19'	X	X	X	1	3
20	3	1	1	X	3
21	X	X	X	1	3
22	X	X	X	1	3
23	X	X	X	1	3
23'	3	2	2	X	3
24	3	2	2	X	3

TABLE 7

It can be seen from the table that decision rules 1, 6, 8, 17, 19, and 23 have two reducts, and the remaining decision rules have one reduct each.

In order to find the minimal decision algorithms, we have to remove from the table all superfluous decision rules.

From Table 7 we see that, in the first decision class, decision rules 1 and 1' are superfluous, because these rule are identical to rules 2, 3 and 4 respectively.

In the second decision class rules 6 and 5 are identical, rules 6', 8' and 9 are also identical, and so are rules 8 and 7; hence it is enough to have one representative rule from each group of identical rules.

Similarly, we can remove superfluous decision rules from decision class three and we obtain the following set of minimal decision rules (minimal decision algorithms) as shown in Table 7.

U	a	b	c	d	e
1',2	1	X	2	2	1
1,3,4	X	1	2	2	1
---	---	---	---	---	---
5,6	1	X	1	2	2
7,8	2	X	1	2	2
6',8',9	X	2	1	2	2
---	---	---	---	---	---
10- 17,19'					
21,22,23	X	X	X	1	3
17',18	2	2	2	X	3
19,20	3	2	2	X	3
23',24	3	2	2	X	3

TABLE 8

U	a	b	c	d	e
1	1	X	2	2	1
2	X	1	2	2	1
3	1	X	1	2	2
4	2	X	1	2	2
5	X	2	1	2	2
6	X	X	X	1	3
7	2	2	2	X	3
8	3	2	2	X	3
9	3	2	2	X	3

TABLE 9

Crosses in the table denote “don’t care” values of attributes. What we have obtained finally is the minimal set of decision rules (minimal decision algorithms) which is equivalent to the original table as far as the decisions are concerned. That means that in the simplified table only the minimal set of conditions, necessary to make decisions specified in the table, are included.

It is worthwhile to stress once more that there is only one solution to the discussed problem. In general many solutions are possible, as we will see in the next chapter.

1.7 Decision Algorithm

As we already stated, we are not interested in decision tables, but rather in decision algorithms, and tables are used only as a convenient way of notation for decision algorithms. Tabular form of presentation of decision algorithms allows us for easy computation of truth of decision rules and, consequently, provide an easy way for decision algorithms simplification.

Thus our final result, presented in Table 9, can be rewritten as a minimal decision algorithm in normal form :

$$a1c2d2 = e1$$

$$b1c2d2 = e1$$

$$a1c1d2 = e2$$

$$a2c1d2 = e2$$

$$b2c1d2 = e2$$

$$d1 = e3$$

$$a2b2c2 = e3$$

$$a3b1c1 = e3$$

$$a3b2c2 = e3$$

Combining all decision rules for one decision class we get the following decision algorithm:

$$(a1 \vee b1)c2d2 = e1$$

$$(a1 \vee a2 \vee b2)c1d2 = e2$$

$$d1 \vee (a3b2c1) \vee ((a2 \vee a3)b2c2) = e3$$

Chapter 2

METHODS OF PROTEIN CLASSIFICATION

Because there is a gap between sequence and structure, the prediction of protein structural classes is still a hot research field today. One protein usually can be classified into one of the four structural classes: all- α , all- β , α/β and $\alpha + \beta$. Many different algorithms and efforts have been made to address this problem so far.

A review about prediction of protein structural class and subcellular locations by Chou presented this problem systematically, and introduced and compared some existing methods.

In 1986, Klein and Delisi first put forward the prediction of protein structural classes, and shortly afterward, Klein brought discriminate analysis method to this problem. A new weighting method was proposed to predict protein structural classes from amino acid composition in 1992. After that, another new method, called maximum component coefficient method, was proposed by Zhang and Chou, which had a higher correct rate than other methods. Later, a new neural networks based algorithm was developed that considers six hydrophobic amino acid patterns together with amino acid compositions, and a cross-validation test was used to verify the accuracy of this method. Chou brought a novel approach to predict protein structural class in a (20-1)-D amino acid composition space, which takes into account the coupling effect among different amino acid components of a protein by a covariance matrix. A method based on the scale of Mahalanobis distance is proposed by Chou and Zhang in 1994, and it also incorporates the correlative effect among different amino acids automatically. Chou proposed the component-coupling algorithm that took into account the coupling effect among different amino acid components. This method was ever thought to be one of the most accurate algorithms to predict protein structural classes. Later, Zhou and Assa-Munt revealed the subtle relation among the Mahalanobis algorithm, the component-coupled algorithm, and the Bayes decision rule, and that the component-coupled algorithm is much more efficient than the simple geometry algorithm in protein structural class prediction.

In 2001, introduced Support Vector Machine, a machine learning method based on statistical learning theory, to deal with this problem. Functional domain composition was introduced by Chou and Cai to predict protein structural class.

But later, rough set theory was introduced as a method which can classify the protein into its protein class which has been used in our project as its accuracy of prediction was quite high.

Chapter 3

ROUGH SETS – A BETTER METHOD

It has been proven from the research by Youfang Cao, Shi Liu , Lida Zhang , Jie Qin , Jiang Wang , and Kexuan Tang that prediction of protein structural class using rough sets is an accurate method of prediction.

In their study two datasets of protein sequences were extracted from SCOP database one consisted of 277 sequences, and another 498 sequences. The two datasets were used to test this method through self consistency test, jackknife test, and induced decision rules from datasets, and comparison of the total accuracy and accuracies of each structural class with other algorithms was done.

As the 3D structure of proteins is uniquely determined by their amino acid sequences, i.e. the primary sequences so in order to predict the structural classes of proteins, primary sequences were converted into numerical vectors, in other words, features information was extracted from primary sequences as the representation of proteins.

They used some of the properties as the features to describe amino acid sequences, i.e. conditional attributes set. The decision attribute set was taken as the four existing protein classes : all- α , all- β , α/β and $\alpha + \beta$.

Then Rough set models were built which were based on the concept of indiscernibility. Given a decision system $A = (U, A \cup \{d\})$, they defined $IND_A(A,x,d)$ to be the set of objects that are indiscernible from x with respect to the decision attribute d . From the definition of indiscernibility they derived for each object $x \in U$ the set of reducts $RED_A(x,d)$ to be the set of minimal sets of attributes $B \subseteq A$ such that $IND_A(B,x,d) = IND_A(A,x,d)$. Hence, a reduct of x is a minimal set of attributes B with the same discriminatory power as A .

Finally, the model was evaluated. In a self-consistency test, training sets are predicted with those decision rules trained by them. The accuracy of self-consistency test tell us how well the rules captured the characteristics of training sets. Jackknife test is deemed to be the most objective and rigorous way to estimate the performance of a classifier. Each object in the dataset is the test set, and the left as training set. In other words, each protein of the dataset is predicted once by the classifier trained with the left proteins. After this process, average of all iterations is performed to obtain an unbiased number of performance estimates.

The results were finally compared with other algorithms including component coupled algorithm , neural network and SVM.

Use of Perl language scripts was done to deal with protein sequences and extract features from them, all of computation concerning on Rough Sets was implemented on Rosetta system, which is a toolkit for data mining and knowledge discovery using rough sets.

In Self Consistency test , all the percentages of correct prediction on both datasets reach 100%, which is the same as the results of SVM method . The results indicated that Rough Sets captured the characteristics between sequences and their classes.

Dataset	Algorithm	Rate of correct prediction for each class				Overall rate of accuracy
		All- α	All- β	α/β	$\alpha+\beta$	
277 domains	Component coupled	95.7%	93.4%	95.1%	92.3%	94.2%
	Neural network	98.6%	93.4%	96.3%	84.6%	93.5%
	SVM	100%	100%	100%	100%	100%
	Rough Sets	100%	100%	100%	100%	100%
498 domains	Component coupled	95.8%	95.2%	94.9%	95.4%	95.8%
	Neural network	100%	98.4%	96.3%	84.5%	94.6%
	SVM	100%	100%	100%	100%	100%
	Rough Sets	100%	100%	100%	100%	100%

Through the reduction of decision tables, two sets of decision rules are generated as classifiers. The classifier trained by 277 domains contains 46651 decision rules in total, and the one of 498 domains contains 52474 decision rules. These rules can be used to classify new protein sequences to the 4 structural classes.

From the results of jackknife tests, we can see that α/β class has the highest accuracy, no matter compared with whichever class or algorithms. This may be related to the proportion of α/β class in the training sets in which α/β class occupied the biggest part. As a supervised learning method, it makes it easier to capture characteristics that feed more training objects to Rough Sets.

Although the average accuracy of 498 domains of Rough Sets is slightly lower than the SVM, they are still much better than others. So, from the results of jackknife, we can conclude that the performance of Rough Sets should have exceeded the component-coupled algorithm and neural networks in this study, and parallel with SVM algorithm.

Dataset	Algorithm	Rate of correct prediction for each class				Overall rate of accuracy
		All- α	All- β	α/β	$\alpha+\beta$	
277 domains	Component coupled	84.3%	82.0%	81.5%	67.7%	79.1%
	Neural network	68.6%	85.2%	86.4%	56.9%	74.7%
	SVM	74.3%	82.0%	87.7%	72.3%	79.4%
	Rough Sets	77.1%	77.0%	93.8%	66.2%	79.4%
498 domains	Component coupled	93.5%	88.9%	90.4%	84.5%	89.2%
	Neural network	86.0%	96.0%	88.2%	86.0%	89.2%
	SVM	88.8%	95.2%	96.3%	91.5%	93.2%
	Rough Sets	87.9%	91.3%	97.1%	86.0%	90.8%

This proves that Rough set theory is an adequate method and with proper selection of training dataset as well as the conditional attributes one can increase the accuracy further. This suggests that the rough sets approach holds a high potential to become a useful tool in bioinformatics.

The current rough sets approach might also stimulate the development in predicting other protein attributes, such as subcellular location, membrane protein type, and enzyme family classification, among many others.

Chapter 4

RESOURCES USED

RSES (ROUGH SET EXPLORATION SYSTEM)

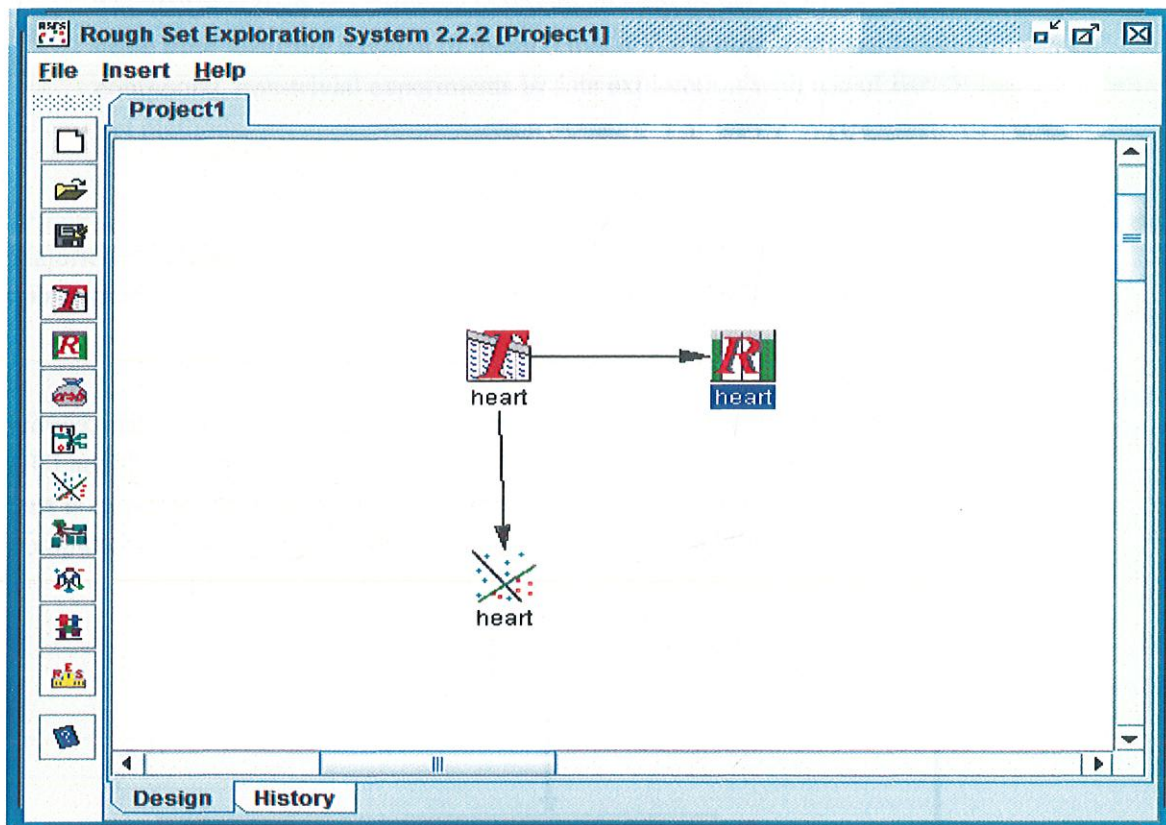


FIGURE 2

The software used in our project is Rough Set Exploration System .

AIMS OF RSES Software

The main aim of RSES is to provide a tool for performing experiments on tabular data sets.

In general, the RSES system offers the following capabilities:

- import of data from text files,
- visualization and pre-processing of data including, among others, methods for discretization and missing value completion, construction and application of classifiers for both smaller and vast data sets, together with methods for classifier evaluation.
- The RSES system is a software tool with an easy-to-use interface, at the same time featuring a bunch of method that make it possible to perform compound, non-trivial experiments in data exploration with use of Rough Set methods.

Majority of RSES is written in Java. Therefore, in order to make it running, an appropriate version of JVM (Java Virtual Machine) is required.

Notice, that Java part of RSES comprises of two logical sub-parts, the RSES 2.2 GUI and the computational kernel powered by the RSES-lib 3.0 library. The GUI part is responsible for interaction with user while the RSES-lib 3.0 serves as computational engine and a wrapper for RSES-lib 2.0 computational routines.

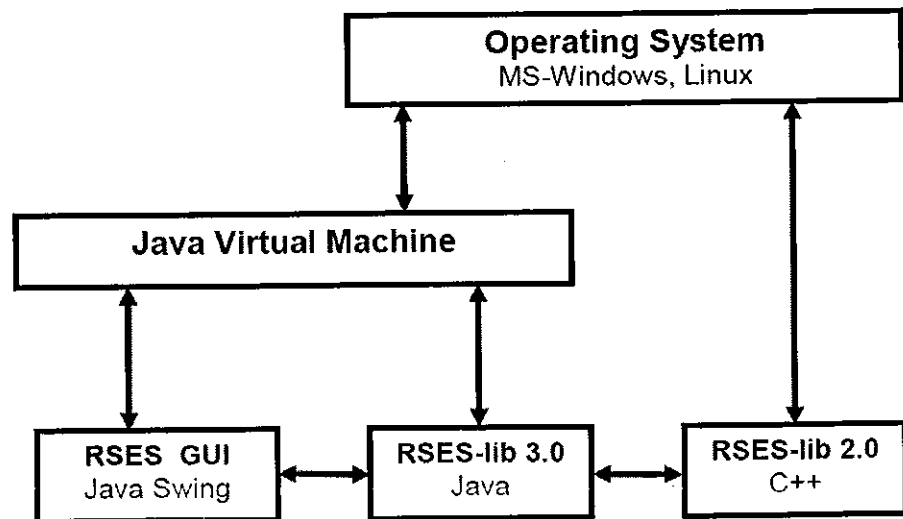


Figure 1.1: RSES internal architecture.

FIGURE 3

We are using RSES software for calculating reducts for an information system which is a subset of attributes which preserves all discernibility information from the information system and none of its proper subsets has that ability.

These reducts can either be calculated based on exhaustive algorithm or genetic algorithm . We can select any algorithm based on our choice. Also we can calculate reducts based on full discernibility matrix or object related discernibility matrix. Dynamic reducts can also be calculated.

This software has several mathematical functions but we are using it to find reducts and decision rules only.

Apart from this software coding also is done in C language for extraction of features where **Turbo C** is used as compiler.

Wamp server is used for creation of database of protein sequences and Java programming is done for establishing connectivity between Mysql(Wamp) and our feature extraction program.

Chapter 5

Methodology

We have to generate minimum decision rules based on which our biological data has to be classified.

First of all, data analysis has to be done as our input will be of protein sequence therefore, it consists of sequence of amino acids. As we want to classify this protein sequence into 4 classes of proteins therefore our decision attributes are as follows :-

5.1 Classes in Protein Architecture :

α -proteins – proteins composed mainly of α - helices (often symmetrical around a central hydrophobic core)

β -proteins – proteins composed mainly of β -sheets

α/β proteins – proteins that are folded with alternating α - helices and β -strands

$\alpha + \beta$ proteins – proteins that combine α - helices and β -strands that are largely separated

5.2 Conditional attributes :

We have to take amino acid compositions of all 20 amino acids present in the sequence and some physico – chemical properties of the protein as conditional attributes.

We selected hydrophobicity, hydrophilicity, positive charge, negative charge, total charge as the conditional attributes..

Thus, Conditional attributes set $S = \{KR, ED, KR-ED, LVOFM, LVIFM\}$, the physicochemical properties: positive charge (KR), negative charge (ED), net charge (KR-ED), major hydrophobic (LVIFM), major hydrophilic (LVOFM).

5.3 Process

As in order to apply rough set concepts to a biological data it has to be converted into numerical form, so data conversion has to be done.

So, codes have to be written for conversion of data.

All the conditional attributes or the features of a protein sequence have to be extracted to get these attributes in numerical form. So, codes in C language are written for the extraction of features from every protein sequence.

5.3.1 CODE for extraction of features

The following code in C extracts amino acid composition and 5 physico-chemical properties i.e. no. of +ve amino acids, no. of -ve amino acids, no. of hydrophilic amino acids, no. of hydrophobic amino acids and net charge on the protein sequence from any given protein-sequence in a file.

```
#include<stdio.h>
#include<conio.h>
```

```
#include<stdlib.h>

void main()
{
I int i,j,length=0,k,hpho=0,hphi=0,neg=0,pos=0,net=0;
  int a[20];
  char ch;
  char *p;
  FILE *fp;

  fp=fopen("seq.txt","r");
  while((ch=fgetc(fp))!=EOF)
  {
    length=length+1;
  }
  fclose(fp);

  p=(char *)malloc((sizeof(char)) * length);

  fp=fopen("seq.txt","r");
  for(k=0;k<length;k++)
  {
    fscanf(fp,"%c",&p[k]);
  }
  fclose(fp);

  for(i=0;i<20;i++)
```

```
{
  a[i]=0;
}
for(j=0;j<length;j++)
{
  if(p[j]=='C')
  {
    a[0]=a[0]+1;
    hpho=hpho+1;
  }
  else if(p[j]=='S')
  {
    a[1]=a[1]+1;
    hphi=hphi+1;
  }
  else if(p[j]=='T')
  {
    a[2]=a[2]+1;
    hphi=hphi+1;
  }
  else if(p[j]=='P')
  {
    a[3]=a[3]+1;
    hpho=hpho+1;
  }
  else if(p[j]=='A')
```

```
{
    a[4]=a[4]+1;
    hpho=hpho+1;
}
else if(p[j]=='G')
{
    a[5]=a[5]+1;
}
else if(p[j]=='N')
{
    a[6]=a[6]+1;
    hphi=hphi+1;
}
else if(p[j]=='D')
{
    a[7]=a[7]+1;
    neg=neg+1;
    hphi=hphi+1;
}
else if(p[j]=='E')
{
    a[8]=a[8]+1;
    neg=neg+1;
    hphi=hphi+1;
}
else if(p[j]=='Q')
```



```
{  
    a[9]=a[9]+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='H')  
{  
    a[10]=a[10]+1;  
    pos=pos+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='R')  
{  
    a[11]=a[11]+1;  
    pos=pos+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='K')  
{  
    a[12]=a[12]+1;  
    pos=pos+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='M')  
{  
    a[13]=a[13]+1;  
    hpho=hpho+1;
```

```
    }  
    else if(p[j]=='I')  
    {  
        a[14]=a[14]+1;  
        hpho=hpho+1;  
    }  
    else if(p[j]=='L')  
    {  
        a[15]=a[15]+1;  
        hpho=hpho+1;  
    }  
    else if(p[j]=='V')  
    {  
        a[16]=a[16]+1;  
        hpho=hpho+1;  
    }  
    else if(p[j]=='F')  
    {  
        a[17]=a[17]+1;  
        hpho=hpho+1;  
    }  
    else if(p[j]=='Y')  
    {  
        a[18]=a[18]+1;  
        hpho=hpho+1;  
    }  
}
```

```

        else if(p[j]=='W')
        {
            a[19]=a[19]+1;
            hpho=hpho+1;
        }
    }
    net=pos+neg;

    printf("Compositions of amino acids in the order C S T P A G N D E Q H
R K M I L V F Y W:\n");

    for(i=0;i<20;i++)
    {
        printf("%d ",a[i]);
    }
    printf("\nNo. of positive amino acids : %d",pos);
    printf("\nNo. of negative amino acids : %d",neg);
    printf("\nNet charge on protein sequence : %d",net);
    printf("\nNo. of hydrophobic amino acids : %d",hpho);
    printf("\nNo. of hydrophilic amino acids : %d",hphi);
    getch();
}

```

Output of the above program is as follows :-

Suppose we take an input of lysosomal protein consisting of following amino acids from a file :

MKVILLFVLA VFTVFVSSRGIPLEEQSFLEFQDKFNKKYSHEEYLERFEIF
 KSNLKGIEELNLI AINH KADTKFGV NKFADLSSDEFKNYYLNNKEAIFTDD
 LPVADYLDDEFINSIPTAFDWRTRGAVTPVKNQGCQSCWSFSTTGNVEG
 QHFISQNKLVSLSEQNLVDCDHECMEYEGEQACDEGCNGGLQPNA YNYII
 KNGGIQTESSYPYTAETGTQCNFN SANIGAKISNFTMIPKNETVMAGYIVS
 TGPLAIAADAVEWQFYIGGVFDIPCNPNSLDHGILIVGYS AKNTIFRKNMP
 YWIVKNSWGADWGEQGYIYLRRGKNTCGVSNFVSTSI

```

C:\C:\WIN45\BIN\NRSTFINAL.EXE
Compositions of amino acids in the order C S T P A G N D E Q H R K M I L U F Y
:
9 25 18 11 20 27 29 17 24 14 5 7 20 5 27 21 20 22 16 6
No. of positive amino acids : 32
No. of negative amino acids : 41
Net charge on protein sequence : 73
No. of hydrophobic amino acids : 157
No. of hydrophillic amino acids : 159_
  
```

We want to find all these features for a dataset taken from SCOP database and finally develop a table in the form of condition and decision attributes .

5.3.2 SCOP

Next step is to take a training dataset from a database and find features of all the sequences in that dataset. SCOP database is used as an appropriate database for this purpose.

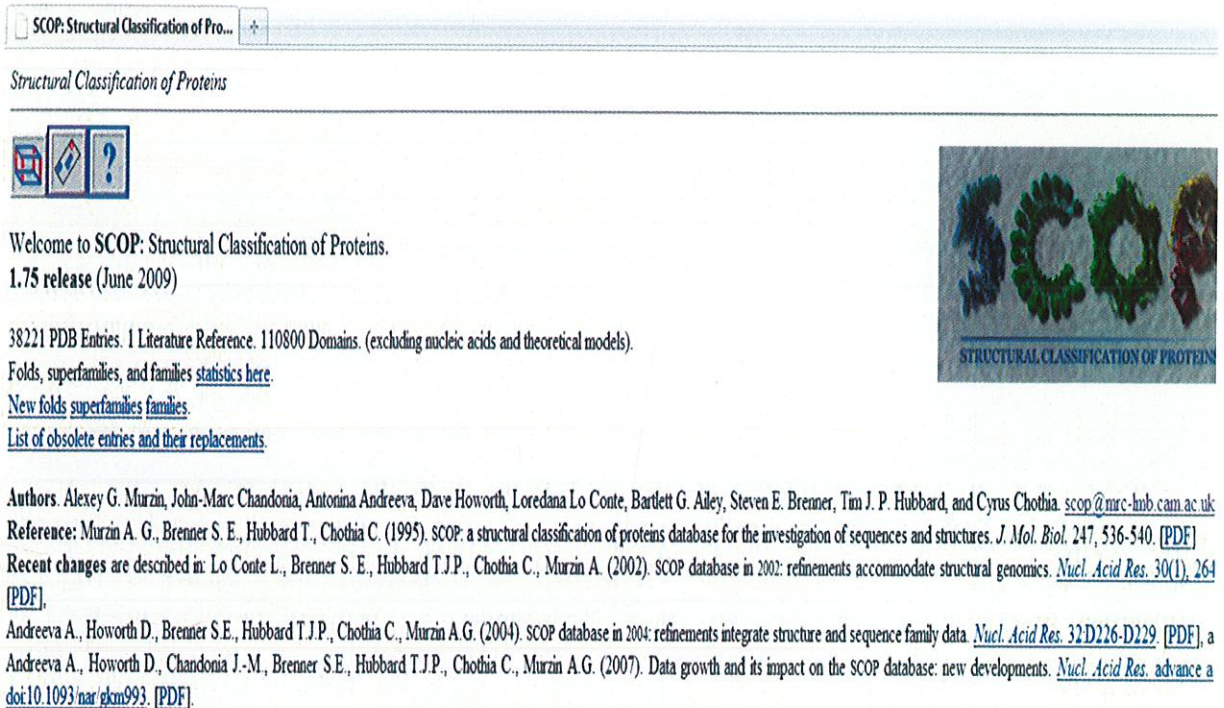
SCOP means Structural Classification Of Proteins.

SCOP organizes proteins in a hierarchy from class down to fold , superfamily , and family.

Although the SCOP protein classification is essentially a manual process using visual inspection and comparison of structures, some automation is used for the

most routine tasks such as clustering protein chains on the basis of sequence similarity.

Following screenshot(FIGURE 4) illustrate homepage and hierarchy of SCOP database :-



SCOP: Structural Classification of Prote...

Structural Classification of Proteins

Welcome to SCOP: Structural Classification of Proteins.
1.75 release (June 2009)

38221 PDB Entries. 1 Literature Reference. 110800 Domains. (excluding nucleic acids and theoretical models).
Folds, superfamilies, and families [statistics here](#).
[New folds superfamilies families](#).
[List of obsolete entries and their replacements](#).

Authors: Alexey G. Murzin, John-Marc Chandonia, Antonina Andreeva, Dave Howorth, Loredana Lo Conte, Bartlett G. Ailey, Steven E. Brenner, Tim J. P. Hubbard, and Cyrus Chothia. scop@mrc-lmb.cam.ac.uk
Reference: Murzin A. G., Brenner S. E., Hubbard T., Chothia C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536-540. [PDF]
Recent changes are described in: Lo Conte L., Brenner S. E., Hubbard T.J.P., Chothia C., Murzin A. (2002). SCOP database in 2002: refinements accommodate structural genomics. *Nucl. Acid Res.* 30(1), 264 [PDF].
Andreeva A., Howorth D., Brenner S.E., Hubbard T.J.P., Chothia C., Murzin A.G. (2004). SCOP database in 2004: refinements integrate structure and sequence family data. *Nucl. Acid Res.* 32:D226-D229 [PDF], and
Andreeva A., Howorth D., Chandonia J.-M., Brenner S.E., Hubbard T.J.P., Chothia C., Murzin A.G. (2007). Data growth and its impact on the SCOP database: new developments. *Nucl. Acid Res.* advance a doi:10.1093/nar/gkm993. [PDF].

Access methods

- Enter SCOP at the [top of the hierarchy](#)
- [Keyword search of SCOP entries](#)
- [SCOP parseable files](#) (MRC site)
- [All SCOP releases and reclassified entry history](#) (MRC site)

SCOP: Root: scop

Structural Classification of Proteins

Root: scop

Classes:

1. [All alpha proteins](#) [46456] (284)
2. [All beta proteins](#) [48724] (174)
3. [Alpha and beta proteins \(a/b\)](#) [51349] (147)
Mainly parallel beta sheets (beta-alpha-beta units)
4. [Alpha and beta proteins \(a+b\)](#) [53931] (376)
Mainly antiparallel beta sheets (segregated alpha and beta regions)
5. [Multi-domain proteins \(alpha and beta\)](#) [56572] (66)
Folds consisting of two or more domains belonging to different classes
6. [Membrane and cell surface proteins and peptides](#) [56835] (58)
Does not include proteins in the immune system
7. [Small proteins](#) [56992] (90)
Usually dominated by metal ligand, heme, and/or disulfide bridges
8. [Coiled coil proteins](#) [57942] (7)
Not a true class
9. [Low resolution protein structures](#) [58117] (26)
Not a true class

FIGURE 5 Hierarchy of SCOP

5.3.3 MYSQL

We have taken about 100 sequences as training dataset from SCOP(25 from alpha class , 25 from beta class , 25 from alpha/beta class and 25 from alpha + beta class).

After taking the dataset from SCOP we have drawn a table of the dataset in MySQL.

As we have to convert our data into a form that can be used by RSES so we have done it by adding training dataset to a table and later on when features are found we have updated the parameters in a new table.

For implementing the code of feature extraction on training dataset we have used Wamp server for creation of our database as well as connectivity between our code and MySQL.

Following is a screenshot(FIGURE 6) of part of the database made in Wamp server through MySQL :-

			sequence	class
<input type="checkbox"/>			MGEVYEKNNIDFDSIAKMLLIKYKDFILSKFKKAAPVENIRFQNLVHTNQ...	A+B
<input type="checkbox"/>			MASATLGSSSSASPVAELCQNTPETFLEASKLLTYADNILRNPSEK...	A+B
<input type="checkbox"/>			MQLKPMENPEMLNKVLSRLGVAGQWRFVDVLGLEEE'SLGSVPAPACALL...	A+B
<input type="checkbox"/>			MSKGTTSDAPFGTLLGYAPGGVAIYSSDYSSLDPEYEDDAVFRSYIDD...	A+B
<input type="checkbox"/>			MKHTYNPFYLIYFAFFITSLSSYSSQTTNIKEQTDGTLAPPAPAITNS...	A+B
<input type="checkbox"/>			MSLRRGIYHIENAGVPSAIDLKDGSSSDGTPVIGWQFTPTINWHQLWLA...	A+B
<input type="checkbox"/>			MLTYQVKQGDTLNSIAADFRISTAALLQANPSLQAGLTAGQSIVPGLPD...	A+B
<input type="checkbox"/>			MILSVCSYELGLYQARTVKENNRVSYIDGKQATQKTENLTPDEVSKREGI...	A+B
<input type="checkbox"/>			MSLRKLTGDLDEISSFLHNTISDFILKRVSAKEIVDIDITVLEVTDEL...	A+B
<input type="checkbox"/>			MHCAENCIFCKIAGDIPSAKVYEDEHVLAFLDISQVTKGHTLVIPKTHI...	A+B
<input type="checkbox"/>			MTRTMSFHKNCELCTTAGGAILWQDALCRVHVHVENQDYPGFCRVILNRHV...	A+B
<input type="checkbox"/>			MREMLQVERPKLILDDGKRTDGRKPDELRSIKIELGVLKNADGSAIFEMG...	A+B
<input type="checkbox"/>			MPEDILVDIKRDYVLSKLRDNERIDGRGFDEFKVEIIPNVIEKAEGSAL...	A+B
<input type="checkbox"/>			MGLEKTVKEKLSFEGVGIHTGEYSKLIHPEKEGTGIRFFKNGVYIPARH...	A+B
<input type="checkbox"/>			MEMAGCGEIDHSINMLPTNKKANESCSNTAPSLTVPECAICLQTCVHPVS...	A+B
<input type="checkbox"/>			MAPRKFVGGNWKMGDKKSLGELIHTLNGAKLSADTEVCGAPSIYLDL...	A/B
<input type="checkbox"/>			MARKYFVAANWKCNGTLESIKSLTNSFNLDLDFPSKLDVVVFPVSVHYDH...	A/B
<input type="checkbox"/>			MRLPILINFKAYGAAAGKRAVELAKAAERAARELGVNIVVAPNHLELGL...	A/B
<input type="checkbox"/>			MAQTPAFNPKVELHVHLDGAIKPETILYGRKRGIALPADTPEELQNI...	A/B
<input type="checkbox"/>			MSLLNPVLLPPVKAYLSQGERFIKWDETTVASPVILRVDPKGYLYWT...	A/B
<input type="checkbox"/>			MVEATAQETDRPRFSFSAAREGKARTGTIEMKRGVIRTPAFMPVGTAA...	A/B
<input type="checkbox"/>			MEILDLTQTLINFPYPGPELRIIEKKIDGFIVSEIIMGSHLCTHIDYPK...	A/B

FIGURE 6

Here in this diagram sequence is listed along with its class. A+B means alpha + beta class and A/B means alpha/beta class. (A=alpha and B = beta)

5.3.4 DATABASE CONNECTIVITY

In Wamp , we have first of all created our table through Mysql tool and have later on updated that table through code by establishing connectivity.

For establishing connectivity between Java and Mysql a code is written in Java.

We have used methods of Java now to change any attribute in the table.

Also our feature extraction code is written in Java which has been used to extract features of each and every training sequence present in the table.

Following code in Java helped in connectivity between database named 'rough' and feature extraction code and also helped in feature extraction of each sequence containing row into a new table 'counters':-

```
import java.sql.*;
import java.io.*;

public class MysqlConnect1
{
    public static void main(String[] args)
    {
        System.out.println("FEATURE EXTRACTION...");
        Connection conn = null;
        String url = "jdbc:mysql://localhost:3306/";
        String dbName = "rough_sets";
        String driver = "com.mysql.jdbc.Driver";
        String userName = "root";
        String password = "";
        char p[];
```

```

int i,j,length=0,k,hpho=0,hphi=0,neg=0,pos=0,net=0;

int a[] = new int[20];

try {
    Class.forName(driver).newInstance();
    conn = DriverManager.getConnection(url+dbName,userName,password);
    System.out.println("Connected to the database");
    BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter sequence:");
    String sequence = bf.readLine();
    try {
        Statement st2 = conn.createStatement();
int val = st2.executeUpdate("INSERT into rough VALUES('"+sequence+"')");
        System.out.println("1 row affected");
    }
    catch (SQLException s)
    {
        System.out.println("SQL statement is not executed!");
    }
}try{
    Statement st = conn.createStatement();
    ResultSet res = st.executeQuery("SELECT * FROM rough");
    System.out.println("SEQUENCES: ");
    while (res.next())
    {
        length=0;
        hpho=hphi=neg=pos=net=0;
    }
}

```

```
String str = res.getString("sequence");
    p = str.toCharArray();
length=str.length();
System.out.println(str);
    for(i=0;i<20;i++)
    {
        a[i]=0;
    }
    for(j=0;j<length;j++)
    {
        if(p[j]=='C')
        {
            a[0]=a[0]+1;
            hpho=hpho+1;
        }
        else if(p[j]=='S')
        {
            a[1]=a[1]+1;
            hphi=hphi+1;
        }
        else if(p[j]=='T')
        {
            a[2]=a[2]+1;
            hphi=hphi+1;
        }
        else if(p[j]=='P')
```

```
{  
    a[3]=a[3]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='A')  
{  
    a[4]=a[4]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='G')  
{  
    a[5]=a[5]+1;  
}  
else if(p[j]=='N')  
{  
    a[6]=a[6]+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='D')  
{  
    a[7]=a[7]+1;  
    neg=neg+1;  
    hphi=hphi+1;  
}  
else if(p[j]=='E')  
{
```

```
        a[8]=a[8]+1;
        neg=neg+1;
        hphi=hphi+1;
    }
    else if(p[j]=='Q')
    {
        a[9]=a[9]+1;
        hphi=hphi+1;
    }
    else if(p[j]=='H')
    {
        a[10]=a[10]+1;
        pos=pos+1;
        hphi=hphi+1;
    }
    else if(p[j]=='R')
    {
        a[11]=a[11]+1;
        pos=pos+1;
        hphi=hphi+1;
    }
    else if(p[j]=='K')
    {
        a[12]=a[12]+1;
        pos=pos+1;
        hphi=hphi+1;
    }
```

```
}  
else if(p[j]=='M')  
{  
    a[13]=a[13]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='I')  
{  
    a[14]=a[14]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='L')  
{  
    a[15]=a[15]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='V')  
{  
    a[16]=a[16]+1;  
    hpho=hpho+1;  
}  
else if(p[j]=='F')  
{  
    a[17]=a[17]+1;  
    hpho=hpho+1;  
}
```



```
        else if(p[j]=='Y')
        {
            a[18]=a[18]+1;
            hpho=hpho+1;
        }
        else if(p[j]=='W')
        {
            a[19]=a[19]+1;
            hpho=hpho+1;
        }
    }
    net=pos-neg;
```

```
for(i=0;i<20;i++)
{
    System.out.println(a[i]);
}
//System.out.println("");
System.out.println("\n"+pos);
System.out.println(neg);
System.out.println(net);
System.out.println(hpho);
System.out.println(hphi);
//System.out.println("");
```

```

        try
        {
            Statement st1 = conn.createStatement();
            int val = st1.executeUpdate("INSERT into counters
VALUES("+pos+", "+neg+", "+net+", "+hpho+", "+hphi+"");

            // System.out.println("1 row affected");
        }
        catch (SQLException s)
        {
            System.out.println("SQL statement is not executed!");
        }

    }

}

catch (SQLException s){
    System.out.println("SQL code does not execute.");
}

conn.close();

System.out.println("Disconnected from database");
}

catch (Exception e) {
    e.printStackTrace();
}

```

```
}
}
```

Following table(counters) is generated when we run the above program which contains values of all the features. This screenshot just shows a part of a table generated :-



FIGURE 7

As we know the classification of each and every sequence taken in training database therefore, once the table consisting of features extracted is found we can easily convert it into a form where it can be imported directly to RSES software for applying rough set concepts on it.

RSES generally takes input in .tab format of the text file. For this purpose we had to convert our text file of our feature extraction table into .tab format so that it can be imported into RSES software.

The .tab file generally required for this software is depicted in the following figure(Figure 8) :-

```

TABLE "rone"
ATTRIBUTES 6
"pos" numeric 0
"neg" numeric 0
"net" numeric 0
"hpho" numeric 0
"hphi" numeric 0
"clas" symbolic
OBJECTS 100
14 13 1 50 46 A
43 40 3 85 138 A
85 64 21 299 328 A
42 42 0 189 186 A
23 31 -8 66 78 A
200 142 58 277 462 A
83 89 -6 228 300 A
66 58 8 141 184 A
84 68 16 142 218 A
74 60 14 184 241 A
160 168 -8 621 561 A
43 41 2 110 109 A
97 91 6 294 331 A
157 162 -5 254 483 A
41 72 -31 144 193 A
15 20 -5 30 54 A
22 12 10 64 52 A
20 16 4 70 53 A
22 22 0 86 87 A
106 102 4 444 364 A
20 21 -1 80 73 A
23 18 5 73 73 A
9 14 -5 50 33 A
47 28 19 138 141 A
111 91 20 224 345 A
43 40 3 228 248 B
127 125 2 395 378 B
12 9 3 63 52 B
22 12 10 42 49 B
17 14 3 73 49 B

```

FIGURE 8

Here Table name is rone and 6 attributes are there in which first five are the conditional attributes i.e. pos, neg, net, hpho and hphi and last one i.e. class is our decision attribute. First five are numerical therefore are mentioned under numeric category in .tab file whereas classes are A(alpha), B(beta), A+B (alpha+beta), A/B(alpha/beta) so are mentioned under symbolic category in .tab text file.

The above .tab file was then imported to RSES.

In RSES software once the table containing conditional and decision attributes is imported we can easily find the reducts using exhaustive algorithm(or genetic algorithm) and can generate the minimum decision rules by processing the rules option on reduct set found.

These rules found are the rules through which we can classify any protein sequence based on its feature information.

The table imported in RSES is shown as below. The screenshot(FIGURE 9) below just shows the part of table imported :-

100 / 6	pos	neg	net	hpho	hphi	clas
O:22	23	18	5	73	73	A
O:23	9	14	-5	50	33	A
O:24	47	28	19	138	141	A
O:25	111	91	20	224	345	A
O:26	43	40	3	228	248	B
O:27	127	125	2	395	378	B
O:28	12	9	3	63	52	B
O:29	22	12	10	42	49	B
O:30	17	14	3	73	49	B
O:31	19	15	4	82	55	B
O:32	18	15	3	68	69	B
O:33	13	8	5	47	52	B
O:34	6	11	-5	61	53	B
O:35	86	78	8	316	303	B
O:36	19	23	-4	103	82	B
O:37	72	84	-12	300	267	B
O:38	18	20	-2	105	75	B
O:39	97	90	7	467	444	B
O:40	121	110	11	426	410	B
O:41	85	84	1	320	325	B
O:42	165	126	39	511	532	B
O:43	51	52	-1	185	174	B
O:44	153	151	2	418	585	B
O:45	23	18	5	94	89	B
O:46	32	28	4	112	99	B
O:47	34	41	-7	98	115	B
O:48	10	4	6	40	30	B
O:49	55	54	1	162	187	B
O:50	64	50	14	165	180	B
O:51	7	12	-5	42	46	A+B
O:52	19	9	10	71	63	A+B
O:53	27	18	9	74	79	A+B
O:54	24	16	8	76	76	A+B
O:55	32	32	0	86	90	A+B
O:56	18	20	-2	108	87	A+B
O:57	47	33	14	146	154	A+B
O:58	71	61	10	208	247	A+B
O:59	98	102	-4	324	339	A+B
O:60	38	35	3	109	119	A+B

FIGURE 9

The reduct set generated is shown as below. These are generated with the help of exhaustive algorithm. There is also an option to generate reducts using genetic algorithm :-

Reduct set: rone				
(1-5)	Size	Pos.Reg.	SC	Reducts
1	2	1	1	{ pos, hpho }
2	3	1	1	{ neg, net, hpho }
3	2	1	1	{ pos, hphi }
4	2	1	1	{ neg, hphi }
5	2	1	1	{ net, hphi }

FIGURE 10

Finally the minimum decision rules generated are shown as below. Following screenshot just show a part of decision rules generated:-

9	1	(pos=84)&(hpho=142)=>(clas={A[1]})
10	1	(pos=74)&(hpho=184)=>(clas={A[1]})
11	1	(pos=160)&(hpho=621)=>(clas={A[1]})
12	1	(pos=43)&(hpho=110)=>(clas={A[1]})
13	1	(pos=97)&(hpho=294)=>(clas={A[1]})
14	1	(pos=157)&(hpho=254)=>(clas={A[1]})
15	1	(pos=41)&(hpho=144)=>(clas={A[1]})
16	1	(pos=15)&(hpho=30)=>(clas={A[1]})
17	1	(pos=22)&(hpho=64)=>(clas={A[1]})
18	1	(pos=20)&(hpho=70)=>(clas={A[1]})
19	1	(pos=22)&(hpho=86)=>(clas={A[1]})
20	1	(pos=106)&(hpho=444)=>(clas={A[1]})
21	1	(pos=20)&(hpho=80)=>(clas={A[1]})
22	1	(pos=23)&(hpho=73)=>(clas={A[1]})
23	1	(pos=9)&(hpho=50)=>(clas={A[1]})
24	1	(pos=47)&(hpho=138)=>(clas={A[1]})
25	1	(pos=111)&(hpho=224)=>(clas={A[1]})
26	1	(pos=43)&(hpho=228)=>(clas={B[1]})
27	1	(pos=127)&(hpho=395)=>(clas={B[1]})
28	1	(pos=12)&(hpho=63)=>(clas={B[1]})
29	1	(pos=22)&(hpho=42)=>(clas={B[1]})
30	1	(pos=17)&(hpho=73)=>(clas={B[1]})
31	1	(pos=19)&(hpho=82)=>(clas={B[1]})
32	1	(pos=18)&(hpho=68)=>(clas={B[1]})

FIGURE 11

Based on the reducts generated and the rules developed we will try to classify proteins into their respective classes and test how much the above decision algorithm is accurate and what improvements need to be done.

5.4 DATA FLOW DIAGRAM

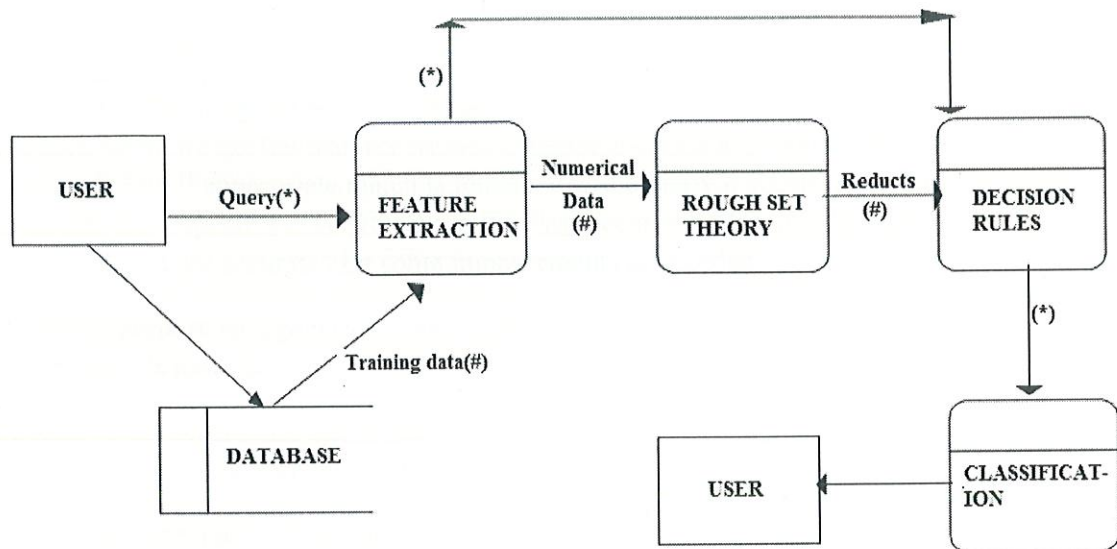
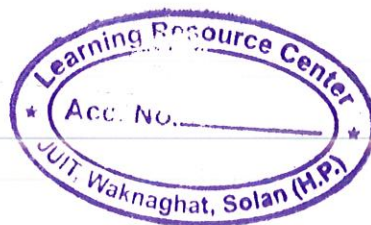


FIGURE 12

5.5 LIMITATIONS

The major limitation of rough set theory is that if the data is variable then rough set theory is not quite efficient in solving the problem. In such a case we have to look for an alternative for generating minimum decision rules.



TESTING

In order to test how well the decision rules are developed we can take any sequence from any database whose classification is known and can check whether according to the decision rules developed it classifies it correctly or not.

For testing, the sequences selected for the purpose have to be taken as input in the feature extraction program and then features have to be calculated for each sequence. Once we get the features we will compare it with our decision rules developed. And if appropriate match is found we will classify it according to that match into the respective class. If it gives the class accurately then our decision rules developed are accurate else some improvements are needed.

For testing purpose we again took some sequences from SCOP database as their classification is known.

We took 10 sequences and found that 6 out of 10 sequences were predicted correctly by matching with our decision rules.

Although the accuracy was ok but still our algorithm can be improved by taking more and different number of conditional attributes.

CONCLUSION

We have tried to develop a generalized algorithm for classification of proteins into their structural class.

This algorithm developed is easy and much more user friendly than traditional classification methods. This algorithm developed can help in classification of new proteins found based on certain characteristics only. By classification through this algorithm we are not required to go deep into more biological details of our system. Rough set theory can be used for other biological classifications also and can become an efficient tool in Bioinformatics.

Although the decision rules developed were able to predict some sequences correctly but still more precise decision rules can be developed by slight changes in methodology specially in the conditional attributes. According to the study conducted by Youfang Cao , Shi Liu, Lida Zhang , Jie Qin , Jiang Wang and Kexuan Tang in their paper "**Prediction of protein structural class with Rough Sets**", it was told that we can improve the accuracy by taking more number of condition attributes which includes more physico-chemical properties of amino acid sequences. This process of taking different combinations of conditional attributes is included in our future work.

REFERENCES

1. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer , 1992
2. *Biological Data Classification Using Rough Sets and Support Vector Machines* by Yanjun Zhao , Yanqing Zhang, Naixue Xiong Department of Computer Science Georgia State University Atlanta , USA (NAFIPS 2009).
3. *Prediction of protein structural class with Rough Sets* by Youfang Cao , Shi Liu, Lida Zhang , Jie Qin , Jiang Wang and Kexuan Tang (BMC Bioinformatics , 2006)
4. RSES software(Developed by Logic Group , Warsaw University , Poland)
5. *Rough set theory and its applications*(by Zdzisław Pawlak)
6. Ning Qian and Terrence J. Sejnowski (1988), "*Predicting the Secondary Structure of Globular Proteins Using Neural Network Models*" in *Journal of Molecular Biology* 202, 865-884. Academic Press.