NICHING MEMETIC ALGORITHM FOR SIMULTANEOUS CLUSTERING AND FEATURE SELECTION

A PROJECT REPORT

Submitted by

T.ARUNAPRIYANKA	71205205008
M.SUBHASHREE	71205205057
G.SARANYA	71205205305

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2009

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"NICHING MEMETIC ALGORITHM FOR SIMULTANEOUS CLUSTERING AND FEATURE SELECTION"** is the bonafide work of **"T.ARUNAPRIYANKA, M.SUBHASHREE and G.SARANYA"** who carried out the project work under my supervision.

SIGNATURE

Mr.E.A.VIMAL

SUPERVISOR

Lecturer

Information Technology

KumaraguruCollege of Technology

Coimbatore – 641 006

SIGNATURE

Dr.S. THANGASAMY

DEAN

Computer Science and Engineering

Kumaraguru College of Technology,

Coimbatore – 641 006.

The candidates with the university register number **71205205008, 71205205057, 71205205305** was examined by us in the project viva voce examination held on ..27/04/09.....

INTERNAL EXAMINER

27/4/09

EXTERNAL EXAMINER

DECLARATION

We,

T. ARUNAPRIYANKA	71205205008
M. SUBHASHREE	71205205057
G. SARANYA	71205205305

hereby declare that the project entitled **"NICHING MEMETIC ALGORITHM FOR SIMULTANEOUS CLUSTERING AND FEATURE SELECTION"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore
Date: 27/04/09

[T.Arunapriyanka]	[M.Subhashree]	[G.Saranya]

Project Guided by

[Mr.E.A.Vimal, M.E]	[Ms. L.S.Jayashree, M.E,Ph.D.,]

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with all the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for providing us excellent facilities to work with. We express our thanks to our beloved Vice-Principal **Prof. R.Annamalai,** who has been the backbone of all our deeds.

With great veneration and sincere gratitude we express our profound thanks to our beloved Dean of Computer Science and Engineering, **Dr.S.Thangasamy,** B.E(Hons), Ph.D., for his immense encouragement and support for being a source of inspiration all through the course of study.

We also extend our heartfelt thanks to our Project Co-ordinator, **Ms. L.S.Jayashree**, M.E, Ph.D., Asso. Prof., Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide **Mr.E.A.Vimal**, M.E., Lecturer, Department of Information Technology for his invaluable guidance, ideas, suggestions and encouragements in all phases of this project work.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

Clustering is inherently a difficult task and is made even more difficult when the selection of relevant feature is also an issue. Here an approach for simultaneous clustering and feature selection has been proposed based on the Niching Memetic Algorithm. This approach makes feature selection an integral part of clustering and attempts to overcome the problem of identifying less promising locally optimal solutions in both clustering and feature selection, without making any a priori assumptions about the number of clusters. Furthermore, local search operations are introduced to refine feature selection. These local search operations include operators like add and remove. The Genetic Algorithms which incorporates the hybrid methods are called as the Memetic Algorithms. Finally, a niching method is integrated to preserve the population diversity and premature convergence.

LIST OF TABLES

INTRODUCTION

1. INTRODUCTION

1.1 GENERAL:

The process of grouping a set of physical or abstract objects into classes of similar objects is called as Clustering. Clustering or cluster analysis is an important but challenging task in unsupervised learning. The essence of the clustering problem is to partition a set of objects into an a priori unknown number of clusters while minimizing the within-cluster variability and maximizing the between cluster variability. Data clustering is a common technique for statistical data analysis and has been used in a variety of engineering and scientific applications such as biology (eg. To study genome data) and computer vision (eg. To segment images).

Partitional clustering is employed here, where each data object is represented as a vector of features. Pure Genetic Algorithms are not well suited for fine tuning the solution close to optima. So we go in for hybridizing the Genetic Algorithm with local search operations and obtain the Memetic Algorithm. Finally a Niching method, which is used for selecting the pairing parents from the population, is incorporated and we get Niching Memetic Algorithm as a result.

This algorithm possesses unified criterion which performs both clustering and feature selection simultaneously. Also the algorithm works well with variable composite chromosomes.

1.2. PROBLEM DEFINITION:

Clustering is inherently a difficult task and is made even more difficult when the selection of relevant features is also an issue. The essence of the clustering problem is to partition a set of objects into an a priori unknown number of clusters while minimizing the within-cluster variability and maximizing the between cluster variability.

Existing Clustering algorithms do not perform well with high dimensional data as they consider all features to be equally important; indeed, some of the features may be redundant, some may be irrelevant, and some can even mislead the clustering process. Here, we concentrate on a kind of clustering called partitional clustering, which is dynamic and considers the global shape and size of clusters.

For a data set of nontrivial size, finding the optimal clustering solution is a challenging problem and becomes even more difficult, if an appropriate feature set also needs to be selected. Here we suggest a unified criterion for simultaneous clustering and feature selection based on a well-known scatter separability index.

1.3. OBJECTIVE OF THE PROJECT:

In order to allow simultaneous clustering and feature selection without the number of clusters being known a priori, a composite representation is devised to encode both feature selection and cluster centers with a variable number of clusters. As a consequence, the crossover and mutation operators are suitably modified.

Additionally, we hybridize the proposed procedure with local search operations, which are introduced to refine the feature selection and cluster centers, respectively. These local searches move solutions toward local optima and allow a significant improvement in the computational efficiency.

Finally, a niching method is integrated to preserve the population diversity and prevent premature convergence. To evaluate the proposed algorithm, we have conducted a series of experiments on both synthetic and real data.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1. FEASIBILITY STUDY:

A feasibility study is concerned to select the best system that meets performance requirements. These entities are an identification description, an evaluation of candidate systems and the selection of the best system for the job.

2.1.1. CURRENT STATUS OF THE PROBLEM:

Many clustering algorithms have been proposed in the literature. Generally, they can be divided into two main categories, namely, hierarchical and partitional. Hierarchical clustering constructs a hierarchy of partitionings, in which each partitioning is nested within the partitioning at the next level in the hierarchy. In hierarchical clustering, problems due to initialization and local optima do not arise. However, this approach considers only local neighbors in each step and ignores the global shape and size of clusters. Moreover, hierarchical clustering is static; that is, data objects committed to a given cluster in the early stages cannot move to a different cluster. In partitional clustering, each data object is represented by a vector of features.

Most partitional algorithms assume all features to be equally important for clustering in the sense that they do not distinguish among different features, but this approach to clustering can create significant limitations in an unsupervised learning context. The problem is that not all features are equally important; indeed, some of the features may be redundant, some may be irrelevant, and some can even mislead the

clustering process. This is one of the reasons that many clustering algorithms do not perform well in the face of high-dimensional data

2.1.2 PROPOSED SYSTEM AND ITS ADVANTAGE:

We propose a niching MA for simultaneous clustering and feature selection (NMA_CFS) by optimizing the unified criterion J_2. The operation of the algorithm consists of using a niching selection method for selecting pairing parents for reproduction, performing different genetic operators on different parts (i.e., feature selection vector and cluster centers) of the paired parents, applying local search operations (i.e., feature add and remove procedures) to each offspring. The algorithm is terminated when the fitness value of the best solution in the population has not changed for g generations.

ADVANTAGES OF THE PROPOSED SYSTEM:

1. In order to allow simultaneous clustering and feature selection without the number of clusters being known a priori, a composite representation is devised to encode both feature selection and cluster centers with a variable number of clusters.
2. We hybridize the proposed procedure with local search operations, Which are introduced to refine the feature selection and cluster centers, respectively.
3. These local searches move solutions toward local optima and allow a Significant improvement in the computational efficiency
4. Finally, a niching method is integrated to preserve the population Diversity and prevent premature convergence.

2.2 HARDWARE REQUIREMENT: (Minimum Requirements)

PROCESSOR	: Intel Pentium-IV (3.00 GHz)
MEMORY	: 128 MB
HARD DISK	: 40 GB
MONITOR	: EGA/VGA
MOUSE	: LOGITECH MOUSE
KEYBOARD	: LOGITECH KEYBOARD

2.3 SOFTWARE REQUIREMENTS: (Minimum Requirements)

OPERATING SYSTEM	: MICROSOFT WINDOWS 2000
LANGUAGE	: JAVA

2.4 SOFTWARE OVERVIEW:

Introduction to Java:

Java is an object-oriented programming language developed by Sun Microsystems, a company best known for its high-end UNIX/LINUX workstations. Modeled after C++, the java language is designed to be small, simple and portable across platforms and operating systems, both at source and the binary level, which means that java programs can run on any machine that has java virtual machine installed. There are two types of java programs. They are java applets and java applications.

Java is a platform independent at both the source level and the binary level; platform independence means that a program can run on any computer system. Java programs can run on any system for which a Java Virtual Machine has been installed. Unlike other programming languages when java programs are compiled byte codes are generated which is a special set of machine instructions that are not specific to any one-processor or computer system.

Unlike most object-oriented languages, Java includes a set of input and output capabilities and other utility functions. Then basic libraries are part of standard environment, which includes simple libraries from networking, common Internet protocols and user interface toolkit functions. Because the libraries are written in Java, they are portable across platforms as all Java applications are. Apart from these features, Java has the following features:

Features:

1. Simple
2. Object-oriented
3. Distributed
4. Robust
5. Secure
6. Architecture neutral
7. Portable
8. Interpreted
9. High performance
10. Multithreaded
11. Dynamic

- **Java is simple.**

What it means by simple is being small and familiar. Sun designed Java as closely to C++ as possible in order to make the system more comprehensible, but removed many rarely used, poorly understood, confusing features of C++. These primarily include operator overloading, multiple inheritance and extensive automatic coercions. The most important simplification is that Java does not use pointers and implements garbage collection so that we don't need to worry about dangling pointers, invalid pointer references and memory leaks and memory management.

- **Java is secure.**

Java is intended to be used in networked environments. Toward that end, Java implements several security mechanisms to protect us against malicious code that might try to invade your file system. Java provides a firewall between a networked application and our computer.

- **Java is object-oriented.**

This means that the programmer can focus on the data in his application and the interface to it. In Java everything must be done via Method invocation on a Java object. We must view our whole application as an object; an object of a particular class.

- **Java is distributed.**

Java is designed to support applications on networks. Java supports various levels of network connectivity through classes in java.net. For instance, the URL class provides a very simple interface to networking. If we want more control over the downloading data than is through simpler URL methods, we would use a URLConnectionObject which is returned by a URL URL.openConnection() method. Also, you can do your own networking with the Socket and ServerSocket classes.

- **Java is robust.**

Java is designed for writing highly reliable or robust software. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking and eliminating situations that are error prone. The removal of pointers eliminates the possibility of overwriting memory and corrupting data.

- **Java is architecture-neutral.**

Java program are compiled to an architecture neutral byte-code format. The primary advantage of this approach is that it allows a Java application to run on any system that implements the JVM. This is useful not only for the networks but also for single system software distribution. With the multiple flavors of Windows 95 and Windows NT on the PC and the new PowerPC Machintosh, it is becoming increasingly difficult to produce software that runs on all platforms.

- **Java is portable.**

The portability actually comes from architecture-neutrality. But Java goes even further by explicitly specifying the size of each of the primitive data types to eliminate implementation dependence. The Java system itself is quite portable. The Java compiler is written i9n Java, while the Java run-time system is written in ANSI C with a clean portability boundary.

- **Java is interpreted.**

The Java compiler generates byte-codes. The Java interpreter executes the translated byte codes directly on system that implements the Java Virtual Machine. Java's linking phase is only a process of loading classes into the environment.

- **Java is high performance.**

Compared to those high-level, fully interpreted scripting languages, Java is high-performance. If the just-in-time compilers are used, Sun claims that the performance of byte-codes converted to machine code are nearly as good as native C or C++. Java however, was designed to perform well on very low-power CPUs.

- **Java is multithreaded.**

Java provides support for multiple threads of execution that can handle different tasks with a Thread class in the java.lang Package. The thread class supports methods to start a thread, run a thread, stop a thread and check on the status of the thread. This makes programming in Java with threads much easier than programming in the conventional single-threaded C and C++ style.

- **Java is dynamic.**

Java language was designed to adapt to an evolving environment. It is a more dynamic language than C or C++. Java loads in classes, as they are needed, even from across the network. This makes the software much easier and effectively. With the compiler, first we translate a program into an intermediate language called byte-codes, the platform-independent codes interpreted on the Java platform.

The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is being executed.

The Java Platform :

A platform is the hardware or software environment in which a program runs. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that its software only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

1. The Java Virtual Machine (JVM).
2. The Java Application Programming Interface (Java API).

The JVM is the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provides many useful capabilities, such as GUI, the Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

The native code is the code that after we compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

DETAILS OF METHODOLOGY

3. DETAILS OF METHODOLOGY USED

3.1 INITIALIZATION OF FEATURE SUBSET AND CLUSTER CENTER:

Initially, feature subsets are randomly selected from the data set (15 features). The entire data set can be split into k clusters. The data corresponding to the selected feature, entering into the respective cluster is decided by the random number generation.

Class:	**Normal find,Metastases, malignant lymph, fibrosis**
lymphatics:	normal, arched, deformed, displaced
block of afferent:	No, yes
Block of lymph cell:	No, yes
Block of lymph structure:	No, yes
By pass	No, yes
Regeneration of lymph nodes:	No, yes
Lymph nodes diminished:	0-3
Lymph nodes enlarged:	1-4
Changes in lymphocyte:	Bean, oval, round
Defect in node:	No, lacunar, lacunar margin, lacunar central
Changes in node:	No, lacunar, lacunar margin, lacunar central
Changes in structure:	No, grainy, drop-like, coarse, diluted, reticular, stripped, faint
Special forms:	No, chalices, vesicles
Dislocation of:	No, yes
Number of nodes:	0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, >=70

Table .3.1

3.2 CALCULATING FITNESS VALUE:

In this module, we apply a unified criterion for simultaneous clustering and feature selection by investigating a well known scatter separability index. Let $X = \{x_1, x_2, ..., x_n\}$ be a set of n data objects in a D-dimensional feature space. A partitioning of the set X is defined as a set of nonempty clusters of X such that data object S_w xi in X is in exactly one of these clusters. The partitioning is typically achieved by optimizing a specified criterion. Some popular criteria are based on the within-cluster and between-cluster scatter matrices. One criterion is the trace $(S_w^{-1}S_b)$in which the within cluster variation S_w is defined as $S_w = 1/n(\sum_{j=1}^{k}\sum_{i=1}^{n} z_{ji}(x_i - m_j)(x_i - m_j)^T$

S_w measures how scattered the objects are from their cluster means where $Z_{ji} = 1$ if $x_i \in$ cluster j. The between cluster variation S_b is defined as

$S_b = \sum_{j=1}^{k} n_j / n(m_j - m)(m_j - m)^T$ and measures how scattered the cluster means are from the sample mean (separability). In most real-world situations, the number of clusters k in a data set is usually unknown. The unified criterion j2 ie. The fitness value can be calculated using within cluster variability S_w and between cluster variability S_b.

j2= trace (Sw . Sb)*(D-d)/(D-1) * (kmax-k)/(k-1)

kmax -> sqrt(n); maximum number of clusters that can be formed, where n is the number of objects.

k -> number of clusters.

D -> total number of available features.

d -> number of features in the subset.

3.3 APPLYING NICHING SELECTION METHOD:

The operation of the algorithm consists of using a niching selection method for selecting pairing parents, performing different genetic operators on different parts (i.e., feature selection vector and cluster centers) of the paired parents, applying local search operations (i.e., feature add and remove procedures and one step of K Means) to each offspring, and carrying out a niching competition replacement. The evolution is terminated when the fitness value of the best solution in the population has not changed for g generations. The output of the algorithm is the best solution encountered during the evolution.

The flow of the algorithm is given as follows:

Step 1:

Randomly initialize p sets of solutions, which encode both feature selection and cluster centers with different numbers of clusters, by using a variable composite representation.

Step 2:

Calculate J2 according to (4) for each solution in the initial population and set its fitness value as f = J2.

Step 3:

Repeat the following steps until the stopping criterion is met:

a) Select pairing parents based on a niching selection method. This procedure is repeated until p/2 parent pairs are selected.

b) Generate intermediate offspring by applying different genetic operators

on the different parts (i.e., feature selection vector and cluster centers) of the paired parents.

c) Apply feature add and remove procedures to the offspring.

d) Run one step of K Means on the offspring.

e) Pair the offspring with the most similar solution found during a restricted competition replacement

f) Calculate J2 for each of the offspring. If the fitness of the offspring is better than its paired solution, then the latter is replaced.

Step 4:

Provide the feature subset and cluster centers of the solution from the terminal population with the best fitness.

PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION

Testing is a critical element of software quality and assurance and represents the ultimate review of specification design and coding.It is a vital activity that has to be enforced in the development of any system.this could be done in parallel during all phases of system development.The feedback received from these texts can be used for further enchancement of the system under consideration.The testing phase conducts test using the software requirement specification as a reference and with the goal to see whether system satisfies the specified requirements.

Standard procedure have been followed in testing our system.Test cases are generated for each screen.These test cases will cover every possibility which could result in both positive and negative results.These test plans are maintained for any further testing done on the system.

4.1 UNIT TESTING:

A series of stand-alone tests are conducted during unit testing. A unit test is also called a module test because it tests the individual units of code that comprise the application. Unit tests focus on functionality and reliability, unit testing is done in a test environment prior to system integration.

4.2 INTEGRATION TESTING:

Integration testing examines all the components and modules that are new, changed, affected by a change, or needed to form a complete system. It is the testing performed to detect errors on interconnection between modules.

4.3 SYSTEM TESTING:

The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements.the system is tested as a whole for its functionality.

CONCLUSION

5. CONCLUSION

We have designed and implemented NMA_CFS by optimizing the suggested unified criterion J2. The significance of the niching method and local search operations within the proposed algorithm has been clearly shown in the experimental results, which also confirm that the simultaneous global clustering and feature subset optimization mechanism is effective in approaching the problem. The resulting algorithm is generally able to select relevant features and locate appropriate partitionings with the correct number of clusters.

FUTURE ENHANCEMENTS

5. FUTURE ENHANCEMENT

For future work, it will be very interesting to apply the Niching Memetic Algorithm for simultaneous clustering and feature selection procedure to real data sets with an abundance of irrelevant or redundant features. An example of such an application is to cluster gene expression data, in which the goal is to identify the informative gene subset for cluster discovery from a large data set that is contaminated with very-high-dimensional irrelevant features. In this case, identifying a relevant subset that adequately captures the underlying structure in the data can be particularly useful. Additionally, as a general optimization framework, the proposed algorithm can be applied for text mining. In such a case, an unbiased clustering criterion in some sense can be produced by computing the mutual information between clusters, thus enabling a better verification of the properties of the proposed optimization scheme.

APPENDICES

7. APPENDICES:

7.1 SOURCE CODE:

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
public class NichingMain extends JFrame implements ActionListener
{
        public JMenuBar mb;
        public JMenu m,m1,m2,m3,m4;
        public JMenuItem mi,mi1,mi2,mi3,mi4,mi5,mi6,mi7,mi8,mi9;
        public JDesktopPane dp;
        public static Object input;
        public JPanel pan;
        public JPanel pan1;
        public JPanel pan2;
        public JButton but;
        public JButton but1;
        public JComboBox cmb;
        public JLabel lbl;
        public static int i=0;
        database_conn db;
        NichingMain()
        {
                super("Niching&Memetic Algorithm");
                mb=new JMenuBar();
```

```
        m=new JMenu("Feature Selection");
        mi1=new JMenuItem("Start");
        m.add(mi1);
        m2=new JMenu("Table");
        mi6=new JMenuItem("InputDatabase");
        m2.add(mi6);
m4=new JMenu("Exit");
mi9=new JMenuItem("Close");
m4.add(mi9);
        mb.add(m);
        mb.add(m2);
        setJMenuBar(mb);
        db=new database_conn();
        mi1.addActionListener(this);
        mi6.addActionListener(this);
mi9.addActionListener(this);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Dimension ss=Toolkit.getDefaultToolkit().getScreenSize();
        dp=new JDesktopPane();
        setContentPane(dp);
        textArea obj=new textArea();
        display(obj);
        setSize(ss.width,ss.height);
        setVisible(true);
}
```

```
public void actionPerformed(ActionEvent ae)
{
    if(ae.getSource()==mi1)
        {
      NichingInitialization nci=new NichingInitialization();
       }
       if(ae.getSource()==mi6)
           {
              ViewTableInput vi=new ViewTableInput();
              display(vi);
           }
           if(ae.getSource()==mi9)
           {
                System.exit(0);
           }
     }

public static void main(String as[])
    {
        /*try{
    javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.Liquid
LookAndFeel");
        }
        catch(Exception er){er.printStackTrace();}*/
        new NichingMain();
    }
```

```
void display(JInternalFrame obj) {
                new CenterFrame(obj);
                obj.setVisible(true);
                dp.add(obj);
                try {
                        obj.setSelected(true);
                }
                catch(java.beans.PropertyVetoException e2){}
        }
}
```

Niching Initialization:

```
import java.io.*;
import java.sql.*;
import java.util.*;
public class NichingInitialization extends Thread
{
  String query;
  database_conn db;
  String column[];
  int i,column_count;
  Vector vecdata;
  Vector vecbit;
   TreeSet ts;
   FitnessCalculationAdd fca;
 FitnessCalculationRemove fcr;
        GeneticTest gt;
```

```
    //Thread t;
NichingInitialization()
{
        start();
}
public void run()
{
        try
        {
                        db=new database_conn();
                        vecdata=new Vector();
                        vecbit=new Vector();
                        ts=new TreeSet();
                        //t=new Thread();
                        DatabaseMetaData dmd=db.con.getMetaData();
                        ResultSet
                rs=dmd.getColumns(null,null,"Lymphography1","%");
                        ResultSetMetaData rm=rs.getMetaData();
                        column_count=rm.getColumnCount();
                        column=new String[column_count+1];
                        while(rs.next())
                        {
                                String str_column=rs.getString(4);
                                  //System.out.println("Field
                                "+str_column);
                                  column[i]=str_column;//+",";
                                  i++;
```

```
                    }
                    String subset[]=new String[15];
                    while(true)
                    {
                        Random r=new Random();
                        int rnd=r.nextInt(column_count);
                        ts.add(rnd);
                        if(ts.size()==15)
                        break;
                    }
                    query="drop table cluster";
                    db.stat.executeUpdate(query);
                    query="create table cluster(";
                    Iterator iter=ts.iterator();
                    int lev=0;
                    while(iter.hasNext())
                    {
                        int
pos=Integer.parseInt(iter.next().toString());
        subset[lev]=column[pos];
query=query+subset[lev]+" int,";
        lev++;
                    }

query=query.substring(0,query.length()-1);
query=query+")";
db.stat.executeUpdate(query);
```

```
textArea.setTxt("[Cluster Intialization----------]");
                query="select * from Lymphography1 ";
                rs=db.stat1.executeQuery(query);
                 while(rs.next())
                        {
                                Random r=new Random();
                                String str="";
                               for(int j=0;j<5;j++)
                                {
          String
                        str1=String.valueOf(r.nextInt(2));
                                        str=str+","+str1;
                               //vecbit.addElement(r.nextInt(2));
                               }
                               str=str.substring(1,str.length());
                               vecbit.addElement(str);
                               textArea.setTxt(str);
                        }

FirstClusterIntialize tt=new FirstClusterIntialize(vecbit,subset);
            double[] fit=tt.getFit();
      textArea.setTxt("[Genetic Algorithm Started----------]");
                                gt=new
GeneticTest(vecbit,subset,"Lymphography1");
          // for(int i=0;i<2;i++)
          // {
```

```
                    textArea.setTxt("[Local Search----------]");
                fca=new FitnessCalculationAdd(vecbit,subset);
        this.sleep(1000);
                    double[] fit1=fca.getFit();
                    Vector databit=fca.getBitVector();
                    int find=0;
                    for(int len=0;len<fit.length;len++)
                    {
                        if(fit[len]>fit1[len])
                        {
                            find++;
                        }
                    }
                    if(find==3)
                    {
gt=new GeneticTest(vecbit,subset,"cluster");
fcr=new FitnessCalculationRemove(vecbit,subset);
                    }
                    else
                    {
                        gt=new
GeneticTest(databit,subset,"cluster");
fcr= new FitnessCalculationRemove(databit,subset);
                    }
                    /*double[] fit2=fcr.getFit();
                    Vector databit1=fcr.getBitVector();
                    find=0;
```

```
                    for(int len=0;len<fit.length;len++)
                    {
                    System.out.println(fit[len]+" "+fit[len]);
                        if(fit1[len]>fit2[len])
                        {
                            find++;
                        }
                    }
                if(find==3)
                    {
  gt=new GeneticTest(databit,subset,"cluster");
  fca= new FitnessCalculationAdd(databit,subset);
                    }
                    else
                    {
gt=new GeneticTest(databit1,subset,"cluster");
fca= new FitnessCalculationAdd(databit1,subset);
                    }
                }*/
            Vector res=fcr.getBitVector();
            for(int i=0;i<res.size();i++)
            {
                System.out.println(res.elementAt(i).toString());
            }
        query="select * from cluster";
        ResultSet rs1=db.stat1.executeQuery(query);
        textArea.setTxt("\n");
```

```
            textArea.setTxt("---------- Final Clusters---------- ");
            while(rs1.next())
          {
          textArea.setTxt("\n");

                              for(int j=0;j<15;j++)
                              {
                                    //System.out.print(rs1.getInt(j+1)+"\t");
                                    textArea.setTxt1(rs1.getInt(j+1));
                              }
                              //System.out.println("\n");
                  }
        }catch(Exception e){e.printStackTrace();}
          }
          public static void main(String args[])
          {
                NichingInitialization nn=new NichingInitialization();
          }
  }
import Jama.*;
import Jama.util.*;
import java.io.*;
import java.sql.*;
import java.util.*;
public class FirstClusterIntialize
{
  String query;
```

```
database_conn db;
String columnname[];
int i,column_count;
Vector vecdata;
Vector vecbit;
int mean=0;
Vector bitvector;
String[] feature;
double[] fit=new double[3];
FirstClusterIntialize(Vector bitvec,String[] feat)
{
        try
        {
                db=new database_conn();
                bitvector=new Vector();
                feature=feat;
                bitvector=bitvec;
                feature=feat;
                DatabaseMetaData dmd=db.con.getMetaData();
       ResultSet rs=dmd.getColumns(null,null,"Lymphography1","%");
                    ResultSetMetaData rm=rs.getMetaData();
                    column_count=rm.getColumnCount();
                    columnname=new String[column_count+1];
                    while(rs.next())
                    {
                                String str_column=rs.getString(4);
                                ////System.out.println("Field "+str_column);
```

```
                    columnname[i]=str_column;//+",";
                    i++;
                }
        int pos=0;
        for(int outer=0;outer<3;outer++)
        {
            String subset[]=new String[5];
                    for(int j=0;j<5;j++)
                    {
                        subset[j]=feature[pos];
                        pos++;
                    }
            float mean[]=new float[5];
            //query="select avg(";
            for(int j=0;j<5;j++)
             {
query="select avg("+subset[j]+") from Lymphography1 where id between 1
and 150";
//System.out.println("query "+query);
        rs=db.stat1.executeQuery(query);
        rs.next();
        mean[j]=rs.getFloat(1);
        //System.out.println("mean "+mean[j]);
                }
query="select ";
for(int j=0;j<5;j++)
 {
```

```
        query=query+subset[j]+",";
 }
 query=query.substring(0,query.length()-1);
 query=query+" from Lymphography1 where id between 1 and 150";
//System.out.println("query "+query);
 rs=db.stat1.executeQuery(query);
 float[][] matrix=new float[150][5];
 float[][] tmatrix=new float[5][150];
int row=0,column=0,next=0;
while(rs.next())
 {
  column=0;
 next=0;
 float first=rs.getFloat(1);
 float second=rs.getFloat(2);
 float third=rs.getFloat(3);
 float fourth=rs.getFloat(4);
 float five=rs.getFloat(5);
 //double mean=(first+second+third+fourth+five)/(double)5;
 ////System.out.println("mean "+mean);
 matrix[row][column]=first-mean[next];
 matrix[row][++column]=second-mean[++next];
 matrix[row][++column]=third-mean[++next];
 matrix[row][++column]=fourth-mean[++next];
 matrix[row][++column]=five-mean[++next];
 row++;
 }
```

```
  for(int i=0;i<matrix.length;i++)
   {
      for(int j=0;j<matrix[i].length;j++)
        {
              tmatrix[j][i]=matrix[i][j];
        ////System.out.print(matrix[i][j]);
        }
////System.out.println("\n");
}
double[][] res=new double[150][150];
int incr=0;
for(int i=0;i<matrix.length;i++)
  {
   //int bit[]=new int[5];
  String str=bitvector.elementAt(i).toString();
  String bit[]=str.split(",");
  for(int j=0;j<tmatrix[0].length;j++)
   {
     double temp=0;
     for(int k=0;k<matrix[0].length;k++)
       {
         temp+=Integer.parseInt(bit[k])*(matrix[i][k]*tmatrix[k][j]);
         //temp+=(double)(matrix[i][k]*tmatrix[k][j]);
     }
res[i][j]=(1/((double)150))*temp;
//System.out.print("\t"+res[i][j]);
```

```
 //System.out.println("\n");
}
float meanval=0.0f;
 for(int j=0;j<5;j++)
  {
       meanval+=mean[j];
 }

float betmatrix[]=new float[150];
float tbetmatrix[]=new float[150];
query="select ";
for(int j=0;j<5;j++)
 {
   query=query+subset[j]+"+";
 }
 query=query.substring(0,query.length()-1);
 query=query+" from Lymphography1 where id between 1 and 150";
 ////System.out.println("query "+query);
 rs=db.stat1.executeQuery(query);
 int l=0;
 while(rs.next())
 {
        betmatrix[l]=((rs.getFloat(1)/(float)5)-meanval);
       //System.out.println("betmatrix[l] "+betmatrix[l]);
        l++;
 }
float[] resbet=new float[150];
```

```
for(int i=0;i<betmatrix.length;i++)
 {
        float temp=0;
        for(int j=0;j<betmatrix.length;j++)
        {
        temp+=betmatrix[i]*betmatrix[j];

        }
 resbet[i]=(((i+1))/((float)150))*temp;
}
double resmul[][]=new double[150][150];
for(int i=0;i<res.length;i++)
 {
    for(int j=0;j<res[0].length;j++)
     {
         double temp=0;
       for(int k=0;k<resbet.length;k++)
         {
           temp+=res[i][j]*resbet[k];
         }
resmul[i][j]=(double)temp;
//System.out.println("\t"+temp);
   }
//System.out.println("\n");
}
Matrix m=new Matrix(resmul);
double trace=m.trace();
```

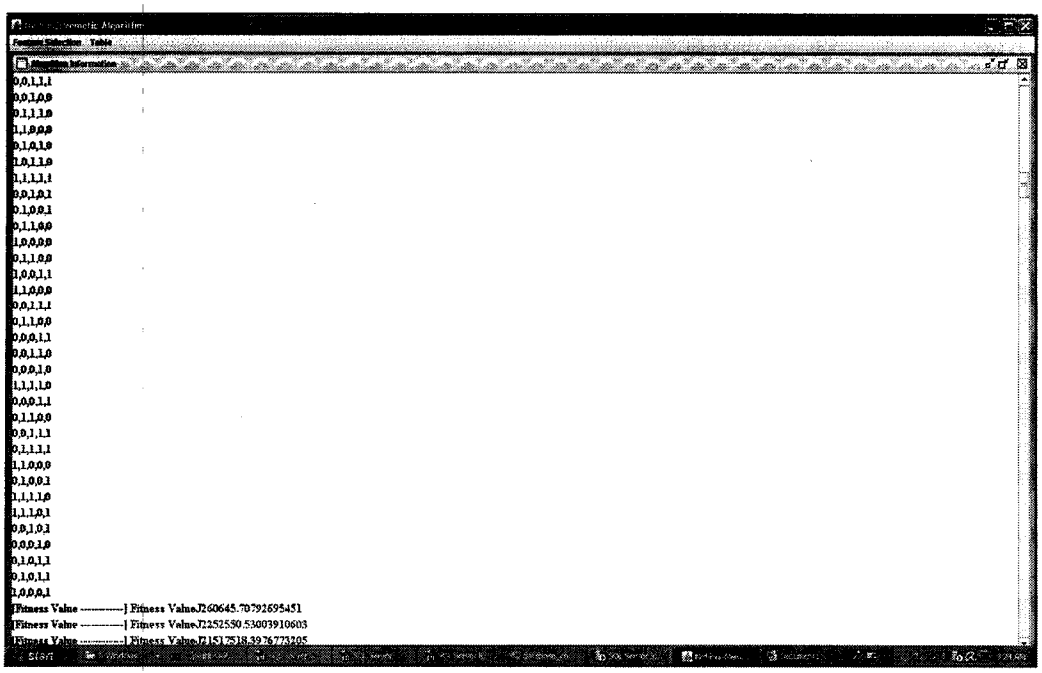
```
//System.out.println(trace);
double J2=(trace*(15-5))/((double)((15-1)*(4-3)))/((double)(3-1));
//System.out.println(J2);
textArea.setTxt("[Fitness Value ----------]"+" Fitness Value"+"J2"+J2);
fit[outer]=J2;
  }
 }catch(Exception e){e.printStackTrace();
}
}
 double[] getFit()
 {
   return fit;
 }
 /*public static void main(String args[])
    {
         ClusterWithin nn=new ClusterWithin();
   }*/
}

import java.sql.*;
import java.io.*;
class database_conn
{
        Connection con;
        Statement stat,stat1,stat2,stat3;
        ///public static void main(String args[])
        database_conn()
```

```
        {
                //ResultSet rs;
                try
                {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:wavlet");
stat=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSe
t.CONCUR_UPDATABLE);
stat1=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultS
et.CONCUR_UPDATABLE);
stat2=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultS
et.CONCUR_UPDATABLE);
stat2=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultS
et.CONCUR_UPDATABLE);
                }
catch(Exception e){e.printStackTrace();}
        }
}
```

7.2 SCREENSHOTS:

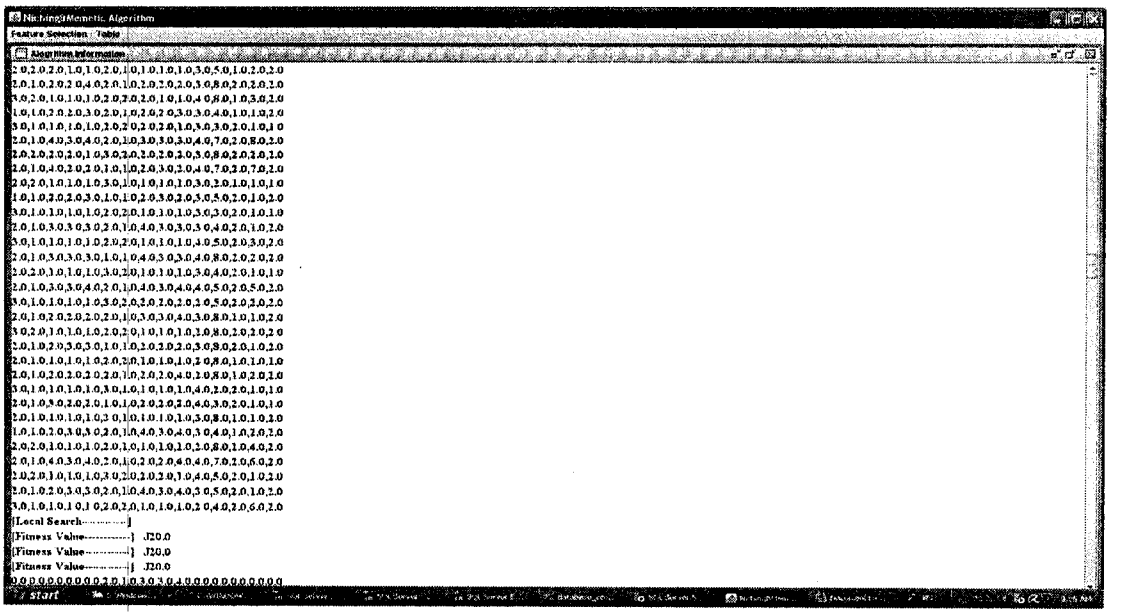
Initialize feature subset and cluster centers:

Calculating Fitness Value:

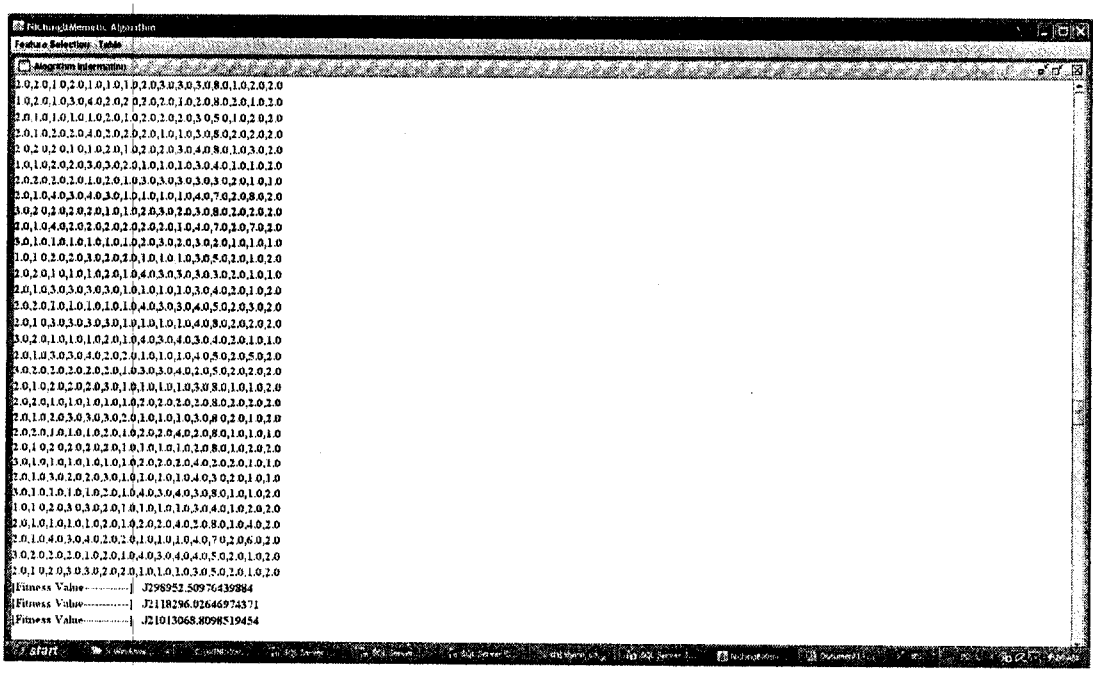


Applying Niching Selection method:

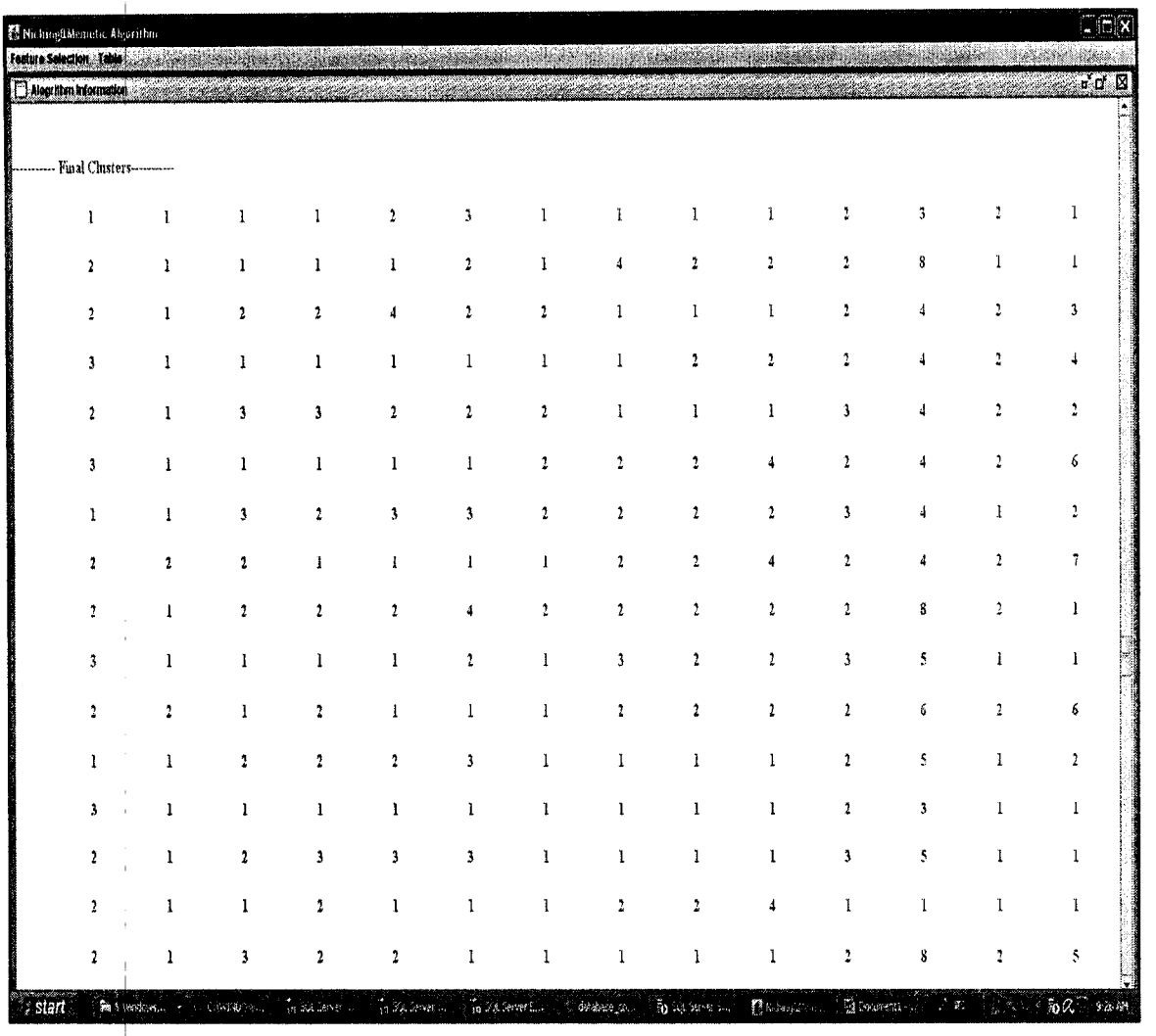
i) local search operator - add



ii) local search operator – remove



Final clusters:



REFERENCES

7. REFERENCES

1. Weiguo Sheng, Xiaohui Liu, Michael Fairhurst, "A Niching Memetic Algorithm for Simultaneous Clustering and Feature Selection", IEEE Trans. Vol 20, NO 7,JULY 2008.

2. P.M. Murphy and D.W. Aha, "UCI Repository for Machine Learning Databases," technical report, Dept. Information and Computer Science, Univ. of California, I
Irvine, 1994.

3. J.H. Yang and V. Honavar, "Feature Subset Selection Using a Genetic Algorithm", IEEE Trans. Information Technology in Biomedicine, vol.8, no. 1,2004.

4. S. Areibi and Z. Yang, "Effective Memetic Algorithms for Local Search + Multi-Level Clustering," Evolutionary Computation, vol. 12, no. 3, pp. 327-353, 2004.

5. W. Sheng , A. Tucker, and X. Liu, "Clustering with Niching Genetic Algorithm", Proc. Genetic and Evolutionary Computation Conf., pp. 162-173, 2004.

6. J.H. Yang and V. Honavar, "Feature Subset Selection Using a Genetic Algorithm", IEEE Intelligent Systems, vol. 13, no,2, pp. 44-49.

WEBSITES:

1. http://www.ics.uci.edu/mlearn/MLRepository.html
2. www.wikipedia.org