APPENDIX 1

SOURCE CODES

A1.1 MATLAB CODE FOR THE CALCULATION OF LYAPUNOV EXPONENT

The algorithm employed in this m-file for determining Lyapunov exponents was proposed in A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov Exponents from a Time Series," Physica D, Vol. 16, pp. 285-317, 1985. The main segment of the code is shown here. This code is a part of tstool. Tstool is a software package for nonlinear time series analysis, though it has a lot of features a general signal analysis package would have. Tstool is written almost completely in MATLAB, a powerful language for scientific computing.

```matlab
function [Texp,Lexp]=lyapunov(n,tstart,stept,tend,ystart,ioutp);
global DS;
global P;
global calculation_progress first_call;
global driver_window;
global TRJ_buffer Time_buffer buffer_i;

options = odeset('RelTol',DS(1).rel_error,'AbsTol',DS(1).abs_error,'MaxStep',DS(1).
max_step,...
'OutputFcn',@odeoutp,'Refine',0,'InitialStep',0.001);
```

The code snippet above demonstrates how to set up options for an ODE solver using `odeset` in MATLAB. This function is then called with the appropriate parameters to calculate the Lyapunov exponents for a given time series.
n_exp = DS(1).n_lyapunov;
n1=n; n2=n1*(n_exp+1);
neq=n2;

% Number of steps
nit = round((tend-tstart)/stept)+1;

% Memory allocation
y=zeros(n2,1);
cum=zeros(n2,1);
y0=y;
gsc=cum;
znorm=cum;

% Initial values
y(1:n)=ystart(:);
for i=1:n_exp y((n1+1)*i)=1.0; end;
t=tstart;

Fig_Lyap = figure;
set(Fig_Lyap,'Name','Lyapunov exponents','NumberTitle','off');
set(Fig_Lyap,'CloseRequestFc','");
hold on;
box on;
timeplot = tstart+(tend-tstart)/10;
axis([tstart timeplot -1 1]);
title('Dynamics of Lyapunov exponents');
xlabel('t');
ylabel('Lyapunov exponents');
Fig_Lyap_Axes = findobj(Fig_Lyap,'Type','axes');
for i=1:n_exp
    PlotLyap{i}=plot(tstart,0);
ITERLYAP = 0;
% Main loop
calculation_progress = 1;

while t<tend
  tt = t + stept;
  ITERLYAP = ITERLYAP+1;
  if tt>tend, tt = tend; end;
  % Solution of extended ODE system

  % [T,Y] = feval(fcn_integrator,rhs_ext_fcn,[t t+stept],y);
  while calculation_progress == 1
    [T,Y] = integrator(DS(1).method_int,@ode_lin,[t tt],y,options,P,n,neq,n_exp);
    first_call = 0;
    if calculation_progress == 99, break; end;
    if ( T(size(T,1))<tt ) & (calculation_progress==0)
      y=Y(size(Y,1),:);
      y(1,1:n)=TRJ_buffer(bufer_i,1:n);
      t = Time_buffer(bufer_i);
      calculation_progress = 1;
    else
      calculation_progress = 0;
    end
  end
end;

uu=findobj(Fig_Lyap,'Type','line');
for i=1:size(uu,1)
  set(uu(i),'EraseMode','none');
  set(uu(i),'XData',[],'YData',[]);
  set(uu(i),'Color',[0 0 rand]);
end

ITERLYAP = 0;
% Main loop
calculation_progress = 1;

while t<tend
  tt = t + stept;
  ITERLYAP = ITERLYAP+1;
  if tt>tend, tt = tend; end;
  % Solution of extended ODE system

  % [T,Y] = feval(fcn_integrator,rhs_ext_fcn,[t t+stept],y);
  while calculation_progress == 1
    [T,Y] = integrator(DS(1).method_int,@ode_lin,[t tt],y,options,P,n,neq,n_exp);
    first_call = 0;
    if calculation_progress == 99, break; end;
    if ( T(size(T,1))<tt ) & (calculation_progress==0)
      y=Y(size(Y,1),:);
      y(1,1:n)=TRJ_buffer(bufer_i,1:n);
      t = Time_buffer(bufer_i);
      calculation_progress = 1;
    else
      calculation_progress = 0;
    end
  end
end;
end;
end;
if (calculation_progress == 99)
    break;
else
    calculation_progress = 1;
end;
t=tt;
y=Y(size(Y,1),:);

first_call = 0;

% construct new orthonormal basis by gram-schmidt
znorm(1)=0.0;
for j=1:n1 znorm(1)=znorm(1)+y(n1+j)^2; end;
znorm(1)=sqrt(znorm(1));
for j=1:n1 y(n1+j)=y(n1+j)/znorm(1); end;
for j=2:n_exp
    for k=1:(j-1)
        gsc(k)=0.0;
        for l=1:n1 gsc(k)=gsc(k)+y(n1*j+l)*y(n1*k+l); end;
    end;
    for k=1:n1
        for l=1:(j-1)
            y(n1*j+k)=y(n1*j+k)-gsc(l)*y(n1*l+k);
        end;
    end;
znorm(j)=0.0;
for k=1:n1 znorm(j)=znorm(j)+y(n1*j+k)^2; end;
znorm(j)=sqrt(znorm(j));
for k=1:n1 y(n1*j+k)=y(n1*j+k)/znorm(j); end;
%       update running vector magnitudes
for k=1:n_exp cum(k)=cum(k)+log(znorm(k)); end;
%       normalize exponent
rescale = 0;
u1 =1.e10;
u2 =-1.e10;

for k=1:n_exp
    lp(k)=cum(k)/(t-tstart);
% Plot
    Xd=get(PlotLyap{k},'Xdata');
    Yd=get(PlotLyap{k},'Ydata');
    if timeplot<t
        u1=min(u1,min(Yd));
        u2=max(u2,max(Yd));
    end;
    Xd=[Xd t]; Yd=[Yd lp(k)];
    set(PlotLyap{k},'Xdata',Xd,'Ydata',Yd);
end;
if timeplot<t
    timeplot = timeplot+(tend-tstart)/20;
    figure(Fig_Lyap);
    axis([tstart timeplot u1 u2]); end;
drawnow;
% Output modification
if ITERLYAP==1
    Lexp=lp;
    Texp=t;
else
    Lexp=[Lexp; lp];
    Texp=[Texp; t];
end;

if (mod(ITERLYAP,ioutp)==0)
    for k=1:n_exp
        txtstring{k}=['
\lambda_' int2str(k) '=' num2str(lp(k))];
    end
    legend(Fig_Lyap_Axes,txtstring,3);
end;
end;

ss=warndlg('Attention! Plot of lyapunov exponents will be closed!','Press OK to continue!');
uiwait(ss);
delete(Fig_Lyap);

fprintf('

 Results of Lyapunov exponents calculation: 
 t=%6.4f',t);
for k=1:n_exp fprintf(' L%d=%f; ',k,lp(k)); end;
fprintf('
');
if ~isempty(driver_window)
    if ishandle(driver_window)
        delete(driver_window);
        driver_window = [];
    end;
end;
calculation_progress = 0;
update_ds;

A1.2  JAVA CODE FOR GENERATING COBWEB, TIME SERIES GRAPHS FOR A DISCRETE LOGISTIC EQUATION AND CALCULATION OF LIMITING PROBABILITIES

/**  Cobweb.java
Description: Generates cobweb and time-series graphs for a discrete logistic equation. Also calculates the limiting probabilities of the transitions. */

import java.awt.event.*;
import java.awt.*;
import java.applet.*;
import java.lang.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
import java.text.*;
import javax.swing.table.*;

public class cobweb11 extends Applet{

public void init(){
setLayout(new BorderLayout());
CobwebTimeSeriesCanvas
dc = new CobwebTimeSeries Canvas();
CobwebCanvas c = new CobwebCanvas();
DefaultTableModel model = new DefaultTableModel(1,2);
add("West", c );
}
add("East", dc);
add("South", new CobwebControls(c,dc,model));
}

// Main entry point when running standalone
public static void main(String[] args) {
    cobweb11 applet = new cobweb11();
    Frame frame = new Frame();
    frame.addWindowListener(new java.awt.event.WindowAdapter()
    {
        public void windowClosing(java.awt.event.WindowEvent e)
        {
            Frame f = (Frame) e.getSource();
            f.setVisible(false);
            f.dispose();
            System.exit(0);
        }
    });
    frame.setTitle("Logistic Map");
    frame.add( applet, BorderLayout.CENTER );
    applet.init();
    applet.start();
    frame.setSize( 1000, 800 );
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    frame.setLocation( (d.width - frame.getSize().width) / 2,
        (d.height - frame.getSize().height) / 2);
    frame.setVisible( true );
}
}

class CobwebControls extends Panel {
    Label k1 = new Label("a");
    Label k2 = new Label("Initial Condition");
    Label k3 = new Label("Iteration Number");
TextField a;
TextField x0;
TextField In;
DefaultTableModel model
CobwebCanvas canvas1;
CobwebTimeSeriesCanvas canvas2;

Public CobwebControls ( CobwebCanvas canvas1,
CobwebTimeSeriesCanvas canvas2,DefaultTableModel model)
{
    this.canvas1 = canvas1;
    this.canvas2 = canvas2;
    this.model = model;
    add(k1);  add(a = new TextField("3.9999", 4));
    add(k2); add(x0 = new TextField("0.3", 4));
    add(k3); add(In = new TextField("100", 4));
    add(new Button("Draw"));
    JTable jt = new JTable(model);
    int v =
        ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDE;
    int h =
        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
    JScrollPane jsp = new JScrollPane(jt,v,h);
    add(jsp);
}

public boolean action(Event ev, Object arg)
{
    if (ev.target instanceof Button) {
        String label = (String)arg;
        canvas1.redraw(Double.valueOf(a.getText().trim()).doubleValue(),
            Double.valueOf(x0.getText().trim()).doubleValue(),
            Double.valueOf(In.getText().trim()).doubleValue());
}
canvas2.redrawer(Double.valueOf(a.getText().trim()).doubleValue(),
Double.valueOf(x0.getText().trim()).doubleValue(),
Double.valueOf(In.getText().trim()).doubleValue(),model);
return true;
}
return false;
}
}
class CobwebCanvas extends Canvas{
double
fixed_a=3.999,Initial_Condition=0.3,Iteration_Number=100;
public CobwebCanvas(){
    resize(410,280);
}
public void paint(Graphics g) {
    Logistic_Map ex = new Logistic_Map(g, fixed_a,
    Initial_Condition, Iteration_Number);
}
public void redraw(double a, double x0, double In) {
    fixed_a = a;
    Initial_Condition = x0;
    Iteration_Number = In;
    repaint();
}
}
class Logistic_Map {
    double fixed_a;
    double Initial_Condition;
    double Iteration_Number;
Logistic_Map(Graphics g, double a, double x0,double In) {
    double y2,y1,y0;
    double fixed_a = a;
    double Initial_Condition = x0;
    double Iteration_Number = In;
    double deltax = .0001;
    y0=Initial_Condition;
    g.setColor(Color.black);
    //Make Axes and Label them
    g.drawLine(10,10,10,260);
    g.drawLine(10,260,410,260);
    g.drawString("1",0,15);
    g.drawString("0",0,255);
    g.drawString("0",15,270);
    g.drawString("1",400,270);
    //Make Function and Diagonal
    for (int i=0;i<10000;i++) {
        g.setColor(Color.black);
        g.drawLine((int)(400*i*deltax)+10,
                    (int)(260-250*(fixed_a*i*deltax*(1-i*deltax))),(int)(400*(i)*deltax)+10,
                    (int)(260-250*(fixed_a*i*deltax*(1-i*deltax))));
        g.setColor(Color.red);
        g.drawLine((int)(400*i*deltax)+10,
                    (int)(260-250*i*deltax),
                    (int)(400*(i)*deltax)+10,
                    (int)(260-250*i*deltax));
    }
    //Draw Cobweb Diagram
    g.setColor(Color.blue);
for(int i=0;i<=Iteration_Number;i++){
    y1=(fixed_a*y0*(1-y0));
    y2=(fixed_a*y1*(1-y1));
    g.drawLine((int)(400*y0+10),(int)(260-250*y1),
               (int)(400*y1+10),(int)(260-250*y1));
    g.drawLine((int)(400*y1+10),(int)(260-250*y1),
               (int)(400*y1+10),(int)(260-250*y2));
    y0=y1;
    y1=y2;
}

class CobwebTimeSeriesCanvas extends Canvas {
    double fixed_a=3.999,Initial_Condition=0.3,Iteration_Number=100;
    DefaultTableModel model = new DefaultTableModel(1,2);
    public CobwebTimeSeriesCanvas()
    {
        resize(510,270);
    }

    public void paint(Graphics g) {
        LogisticTimeSeries_Map ex = new LogisticTimeSeries_Map(g,fixed_a,Initial_Condition,Iteration_Number,model);
    }

    public void redrawer(double a1, double x0, double In, DefaultTableModel ml)
    {
        fixed_a = a1;
        Initial_Condition = x0;
        Iteration_Number = In;
        model = ml;
    }
repaint();
}
}

class LogisticTimeSeries_Map {
    double fixed_a;
    double Initial_Condition;
    double Iteration_Number;
    DefaultTableModel model = new DefaultTableModel(1,2);
    int NumOfFun=25;
    double fstart[]=new double[101];
    double fend[]=new double[101];
    double f_start=0;int return_1,return_2;
    int trans_matrix[][]=new int[NumOfFun+1][NumOfFun+1];
    double prob_matrix[][] = new
double[NumOfFun+1][NumOfFun+1];
    double c[][] = new double[NumOfFun+1][NumOfFun+1];
    double Limit_prob_matrix[][] = new
double[NumOfFun+1][NumOfFun+1];
    static int row = 0;
LogisticTimeSeries_Map()
{
}
}
void initial_Map()
{
    int intervel = 1/NumOfFun;
    for(int i=1;i<=NumOfFun;i++)
    {
        fstart[i]=f_start;
        fend[i]=(double)(i*(1.0/NumOfFun));
        f_start = fend[i];
    }
for(int i=1;i<=NumOfFun;i++)
{
for(int j=1;j<=NumOfFun;j++)
{
    trans_matrix[i][j]=0;
}
}

int find_Y0_position(double arg1)
{
for(int i=1;i<=NumOfFun;i++)
{
    if(arg1>=fstart[i] && arg1<fend[i])
    {
        return_1=i;
    }
}
return return_1;
}

int find_Y1_position(double arg1)
{
for(int i=1;i<=NumOfFun;i++)
{
    if(arg1>=fstart[i] && arg1<fend[i])
    {
        return_2=i;
    }
}
return return_2;
}

void ShowMatrix()
{
DecimalFormat Currency = new DecimalFormat("0.0000");
double rsum = 0.0;
int limitPower = 100;
System.out.println("Transition Matrix ");
for(int i=1;i<=NumOfFun;i++)
{
for(int j=1;j<=NumOfFun;j++)
    System.out.print(trans_matrix[i][j]+"t");
System.out.println();
}
System.out.println("probability Matrix");
for(int i=1;i<=NumOfFun;i++)
{
    rsum = 0;
    for(int j=1;j<=NumOfFun;j++)
    {
        rsum = rsum + trans_matrix[i][j];
    }
    for(int j=1;j<=NumOfFun;j++)
    {
        if(rsum != 0)
        {
            if(i==j) c[i][j] = trans_matrix[i][j]/rsum;
        else prob_matrix[i][j]=0;
        String val=Currency.format(prob_matrix[i][j]);
        System.out.print(val + "\t");
        double value=Double.parseDouble(val);
        prob_matrix[i][j] = value;
        System.out.print(prob_matrix[i][j]+"\t");
    }
System.out.println();
}
System.out.println(" Limiting probability Matrix");
//Creating Identity Matrix
for (int i = 0; i < NumOfFun; i++)
    for (int j = 0; j < NumOfFun; j++)
        if (i==j) c[i][j] = 1;
        else c[i][j] = 0;
//Calculating the matrix power of probability Matrix to limit Power;
for (int z = 1; z < limitPower; z++)
{
    for (int i = 0; i < NumOfFun; i++)
        for (int j = 0; j < NumOfFun; j++)
            for (int k = 0; k < NumOfFun; k++)
                Limit_prob_matrix[i][j] += c[i][k] * prob_matrix[k][j];
    for (int i = 0; i < NumOfFun; i++)
        for (int j = 0; j < NumOfFun; j++)
            { c[i][j] = Limit_prob_matrix[i][j];
              Limit_prob_matrix[i][j] = 0;
            }
}
for (int i = 1; i <= NumOfFun; i++)
{
    for (int j = 1; j <= NumOfFun; j++)
        { String val = Currency.format(c[i][j]);
            System.out.print(val + "\t");
            System.out.println();
        }
}

LogisticTimeSeries_Map(Graphics g, double a, double x0, double In, DefaultTableModel m) {
    double y1, y0;
    double fixed_a = a;
    double Initial_Condition = x0;
    model = m;
    model.removeRow(model.getRowCount() - 1);
    y0 = x0;
    double Iteration_Number = In;
    g.setColor(Color.black);
//Draw and Label Axes
g.drawLine(10,10,10,260);
g.drawLine(10,260,510,260);
g.drawString("1",0,15);
g.drawString("0",0,255);
g.drawString("x",0,120);
g.drawString("0",15,270);
g.drawString(""+Iteration_Number,482,272);
g.drawString("Iteration Number",240,272);
g.setColor(Color.green);
DecimalFormat Currency = new DecimalFormat("0.00");
initial_Map();
for(int i=0;i<=Iteration_Number;i++)
{
y1=(fixed_a*y0*(1-y0));
System.out.println(y0+"---"+y1);
String copy_y0=Currency.format(y0);
String copy_y1=Currency.format(y1);
double yo1=Double.parseDouble(copy_y0);
double y02=Double.parseDouble(copy_y1);
int get1=find_Y0_position(yo1);
int get2=find_Y1_position(yo2);
trans_matrix[get1][get2]=trans_matrix[get1][get2]+1;
System.out.println("g1"+get1+"---"+get2);
g.drawLine((int)((500/Iteration_Number)*i)+10,
(int)(260-260*y0),
(int)((500/Iteration_Number)*(i+1))+10,
(int)(260-260*y1));
Object[] ob = {i,y1};
model.addRow(ob);
A1.3 CODE FOR GETTING VERTEX COVER OF THE GIVEN GRAPH

using System;
namespace VertexCover
{
    public class Vertex
    {
        public static int BLACK = 2;
        public static int GRAY = 1;
        public static int WHITE = 0;
        public static int limit = 0;
        public int xcoordinate;
        public int ycoordinate;
        public int x;
        public int color;
        public int discoverTime;
        public int finishTime;
        public Vertex child;
        public Vertex parent;
        public Vertex()
        {
            this.color = Vertex.WHITE;
            this.discoverTime = Vertex.limit;
            this.finishTime = Vertex.limit;
            this.child = null;
            this.parent = null;
        }
        public Vertex(int val)
        {
            this.color = Vertex.WHITE;
            this.discoverTime = Vertex.limit;
        }
    }
}
A1.4 CODE FOR VERTEX COVER FORM

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Data;
namespace VertexCover
{
    public class VCover : System.Windows.Forms.Form
    {
        private static bool coverFound = false;
        private System.Windows.Forms.Button button1;
        private bool[,] edges;
        private int dimension;
        private int coverToFind;
        private OleDbConnection cnn;
        private OleDbDataAdapter adpcover, adpmatrix;
        private DataSet dsMatrix, dsCover;
        private System.ComponentModel.Container components = null;

        public VCover()
        { InitializeComponent(); }

        protected override void Dispose( bool disposing )
        { if( disposing )
            { if(components != null)
                { components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            protected override void Dispose( bool disposing )
            { if( disposing )
                { if(components != null)
                    { components.Dispose();
                    }
                }
            }
            base.Dispose( disposing );
        }
    }
}
this.button1.Location = new System.Drawing.Point(176, 96);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(160, 40);
this.button1.TabIndex = 0;
this.button1.Text = "Find Cover";
this.button1.Click += new System.EventHandler(this.button1_Click);
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(496, 237);
this.Controls.AddRange(new System.Windows.Forms.Control[] { this.button1 });
    this.Name = "VCover";
    this.Text = "VCover";
    this.Load += new System.EventHandler(this.VCover_Load);
}
#endregion

private void VCover_Load(object sender, System.EventArgs e)
{
    dsMatrix = new DataSet();
dsCover = new DataSet();
cnn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=data.MDB");
cnn.Open();
adpcover = new OleDbDataAdapter("Select * From cover",cnn);
adpmatrix = new OleDbDataAdapter("Select * From adjacencyMatrix",cnn);
adpcover.Fill(dsCover);
adpmatrix.Fill(dsMatrix);
adpcover.Dispose();
adpmatrix.Dispose();
cnn.Close();
dimension = Convert.ToInt32(dsCover.Tables[0].Rows[0]["v"]); //get the number of vertices
edges = new bool[dimension, dimension]; //allocate memory to the adjacency matrix
for(int i=0;i<dimension-1;i++)
    for(int j=0;j<dimension-1;j++)
        edges[i, j] = false; //set all the entries in the adjacency matrix to false
int k = Convert.ToInt32(dsCover.Tables[0].Rows[0]["k"]); //get k from the database
coverToFind = k;
int w = 0;
while(w < dsMatrix.Tables[0].Rows.Count)

// this is to set the values in the adjacency matrix to true where there is connection between the nodes
{
edges[Convert.ToInt32(dsMatrix.Tables[0].Rows[w][0]), Convert.ToInt32(dsMatrix.Tables[0].Rows[w][1])] = true;
edges[Convert.ToInt32(dsMatrix.Tables[0].Rows[w][1]), Convert.ToInt32(dsMatrix.Tables[0].Rows[w][0])] = true;
}

private void button1_Click(object sender, System.EventArgs e)
{
    coverFound = false;
    findCover(new int[0], coverToFind);
}

public int[] addCover(int[] cover1, int[] cover2)
// returns an array of integers containing the sum of the 2 covers (the arguments)
{
    int[] finalCover = new int[cover1.Length + cover2.Length];
    int k = 0;
    for (int i = 0; i < cover1.Length; i++)
    {
        finalCover[k] = cover1[i];
        k++;
    }
    for (int j = 0; j < cover2.Length; j++)
    {
        finalCover[k] = cover2[j];
        k++;
    }
    return finalCover;
}

public int[] addCover(int[] cover1, int val)
// returns an array of integers containing the old cover + the vertex with the highest degree
{
    int[] finalCover = new int[cover1.Length + 1];
    int k = 0;
    for (int i = 0; i < cover1.Length; i++)
    {
        finalCover[k] = cover1[i];
        k++;
    }
    finalCover[k] = val;
    return finalCover;
}

public void findCover(int[] cover, int k)
{
    int nodeIndex = -1;
}
if (k == 0 && coverFound == false)
{  bool edgeInCover = false;
    int conditionViolated = 0;
    for (int i = 0; i < dimension; i++)
    {
      for (int j = 0; j < dimension; j++)
      {
        if (edges[i, j] == true)
        // found neighbors
        {  edgeInCover = false;
          for (int v = 0; v < cover.Length; v++)
            // check if one of the vertices is in the cover
            {  if (cover[v] == i || cover[v] == j)
              {  edgeInCover = true;
                // one of the vertices is in the cover
                break;  }  }
          if (edgeInCover == false)
          // this means not all the edges are covered in the cover
          {  conditionViolated = 3; break;  }
        }
      }
    }
    if (conditionViolated != 3) // all the edges are covered
    {  String outputstr = "";
      for (int i = 0; i < cover.Length; i++)
        outputstr = outputstr + "," + cover[i].ToString();
      MessageBox.Show("found a cover");
      coverFound = true;
    }  else if (k < 0)
    {  }
    else if (coverFound == true){}
  else
  // if the cover is found, the below code will not be executed
  {  bool neighborInCover = false;
    // get highest degree
    bool inCover = false;
    int max = 0;
    int count = 0;
    for (int i = 0; i < dimension; i++)
    {  inCover = true;
      while ((inCover == true) && i < dimension)
        {  }
    }
  // we don't want the vertices that are in the cover
    inCover = false;
    for (int v = 0; v < cover.Length; v++)
    {  if (i == cover[v])
      {  inCover = true;
        i++;  }  }  }  }  }  }
break;
}}
if(i>=dimension) break;
if(inCover == true) break;
for(int j=0;j<dimension;j++)

//to get the highest degree
{
    if(edges[i,j] == true)
    {
        neighborInCover = false;
        for(int v=0;v<cover.Length;v++)
            //we want to check if the neighbor of this vertex is in the cover, if it is, we don't increase the degree
            {
            if((i == cover[v]) || (j == cover[v]))
                neighborInCover = true;
                break; }
    }

if(neighborInCover == false)
    //if none of its neighbors is in the cover
    count++;  }
if (count>max)
    //if until now this is the node with the highest degree
    {
        max = count;
        nodeIndex = i;
    }

//nodeindex is highest degree
}
count = 0;}
if(nodeIndex == -1)
{
    coverFound = false;
    findCover(cover,0);
}
else
{
    findCover(addCover(cover,nodeIndex),k-1);
    //find neighbors
    count=0;
    int[] retVal;
    for(int i = 0;i<dimension;i++)
    {
        inCover = true;
        while((inCover == true) && (i<dimension))
        //this "while" loop is to skip the vertices that are in the cover
        {
            inCover = false;
        for(int v=0;v<cover.Length;v++)
            {
            if(i == cover[v])
                {
                inCover = true;
                i++;;
                break;  }
            }
        }
    }
    if(i>=dimension) break;
    if(inCover == true) break;
if(edges[nodeIndex,i] == true)
//a neighbor is found. next we want to check if this neighbor is in the cover
{
    neighborInCover = false;
    for(int v=0;v<cover.Length;v++)
    {
        if((i == cover[v]))
        {
            neighborInCover = true;break;
        }
    }
    if(neighborInCover == false)
    //if the neighbor is not in the cover
    count++; }
retVal = new int[count];
//now we allocated memory to store the neighbors of the vertex
the next step is to store them
    neighborInCover = false;
    int m = 0;
    for(int i = 0;i<dimension;i++)//find the neighbors of i
    {
        if(edges[i,nodeIndex] == true)
        {
            neighborInCover = false;
            //check if i is in cover
            for(int v=0;v<cover.Length;v++)
            {
                if((i == cover[v]))

            }
            if(neighborInCover == false)
            {
                retVal[m] = i; //store this vertex as being a neighbor
                m++;}
        }
    }
int[] cover2 = retVal; //cover2 contains the neighbors of the vertex find neighbours
findCover(addCover(cover,cover2),k-cover2.Length);
}
}
}
}
static void Main()
{
    Application.Run(new VCover());
}
}
### A1.5 JAVA SAMPLE CODE FOR MULTIPLE WEB RESOURCE ALLOCATION IN WORLD WIDE WEB

```java
package threading;
import java.util.Random;

class Thread1 implements Runnable {
    Thread t;
    String name;
    static int cpos, cneg, tothits, nusers, notoken, tokenobtained;
    Thread1(String n) {
        t = new Thread(this, n);
        t.start();
    }

    public void run() {
        System.out.println("Entering to thread " + t);
        for (int i = 0; i < 333; i++) {
            System.out.println(t);
            User u = new User();
            nusers++;
            Random r = new Random();
            synchronized(u) {
                int nhits = r.nextInt(10);
                tothits += nhits;
                u.sethits(nhits);
                for (int j = 0; j < nhits; j++)
                    u.RequestObj();
                if (u.robj.ID == -1) {
                    int c = 0;
                    System.out.println("token not available for " + u.robj.ID + " retrying");
                    while (u.robj.ID == -1) {
                        c++;
                        try {Thread.sleep(u.processing());} catch (Exception e) {}
                    }
                    u.robj.tokenRequest(u.robjID);
                }
            }
        }
    }
}
```
u.updatecount();
System.out.println("token available for " + u.robjID + " after " + c + " trails");
}
u.accessgain();
try{
    Thread.sleep( u.processing());
catch(Exception e){}
}

u.tokenReturn();
} u.display();
} cpos = cpos+u.getpositive();
 cneg = cneg+u.getnegative();
 notoken = notoken+u.getnotoken();
 tokenobtained = tokenobtained+u.gettokenobtained();

} System.out.println("Existing from " + t);
}
}

public class Main
{
    public static void main(String arg[])
    {
        Thread1 t1 = new Thread1("one") ;
        Thread1 t2 = new Thread1("two");
        Thread1 t3 = new Thread1("three");
        try
        {
            t1.t.join();
            t2.t.join();
            t3.t.join();
        } catch(InterruptedException e)
        {
        }
        System.out.println("Number of Users" + Thread1.nusers);
        System.out.println("total hits " +Thread1.tothits);
        System.out.println("cumulative positive " +Thread1.cpos);
        System.out.println("cumulative negative " +Thread1.cneg);
        System.out.println("No token negative" + Thread1.notoken);
        System.out.println("token obtained" + Thread1.tokenobtained);
    }
}
//User.java
package threading;
import java.util.Random;
public class User
{
    int
    access,category,nhits,positive,negative,notoken,robjID,tokenobtained,accessright;
    boolean processover;
    float ID;

    static int ResPositive[][][];
    static int ResNegative[][][];

    Robject robj;
    User()
    {
        Random ruser  = new Random();
        ID = ruser.nextFloat();
        if (ID <=0.25)
            category = 1;
        else if (ID >0.25 && ID <=0.50 )
            category =2;
        else if (ID >0.50 && ID <=0.75)
            category = 3;
        else category = 4;
        System.out.print("u"+category);
        positive = 0;
        negative =0;

    }

    void sethits(int hits)
    {
        nhits = hits;
        System.out.print(" no hits "+nhits);
    }
    void RequestObj()
    {
        Random r = new Random();
        robj = new Robject(r);
        if (robj.ID == -1)
        {
            System.out.print("Requested object have no token");
            notoken++;
        }
    }
robjID = robj.next;
System.out.print(" "+robj.ID);
}

void accesgain()
{
    Random raccess = new Random();
    accessright = raccess.nextInt(3);
    AccessCheck ac = new AccessCheck(category,robjID,accessright);
    access= ac.accessed;
    if(access == 1)
        { positive++;
        }
    else
        { negative++;
            }
}

void updatecount()
{
    positive++;  
    negative--;  
    tokenobtained++;
}

int processing()
{
    Random rtime = new Random();
    return(rtime.nextInt(50));
}

    @SuppressWarnings("static-access")
public void tokenReturn()
{
    int rID = robjID;
    robj.tokenReturnfn(rID);
}

public int getpositive()
{return(positive);}

public int getnegative()
{return(negative);}

public int getnotoken()
{return(notoken);}

public int gettokenobtained()
{return(tokenobtained);}

public void display()
{ System.out.println("\\t"+positive +"\\t"+negative);}
}
package threading;
import java.util.Random;
public class Robject
{
static int token[] = { 4,4,4,4,4,4,4,4};
int ID,next,tokenreturn;
    Robject(Random r) {
        next =r.nextInt(8);
        if (token[next] >= 1) //Checking token availability
        {   ID = next;
            token[next]  = token[next] -1;
        }
        else
            ID = -1;
    }
    public boolean tokenRequest(int rID)
    {
        boolean tokenavailable = false;
        if (token[rID] >= 1) //Checking token availability
        {   ID = rID;
            token[rID] = token[rID] -1;
            tokenavailable = true;
        }
        else
            ID = -1;
        return(tokenavailable);
    }
    public void tokenReturnfn(int rID)
    {
        if(token[rID] <=1) //Maximun no of tokens - 1;
        {   token[rID]= token[rID] +1;
        }
        else
        {
            System.out.println("token cant update since reached the maximum tokens");
        }
    }
    public void Robject()
    {
        Random r = new Random();
        next =r.nextInt(8);
        if (token[next] > 0)
        {   ID = next;
            token[next] = token[next] -1;
        }
    }
else
    ID = -1;
}

//AccessCheck.java
package threading;
public class AccessCheck
{
    int accessMatrix[][][] = {
        {{1,1,1},{1,1,1},{1,1,1},{1,0,0}},
        {{1,1,1},{0,0,0},{1,1,1},{1,0,0}},
        {{1,1,1},{1,1,1},{0,0,0},{1,0,0}},
        {{1,1,1},{0,0,0},{0,0,0},{1,0,0}},
        {{1,1,1},{1,1,1},{1,1,1},{1,0,0}},
        {{1,1,1},{1,1,1},{0,0,0},{1,0,0}},
        {{1,1,1},{0,0,0},{1,1,1},{1,0,0}},
        {{1,1,1},{1,1,1},{0,0,0},{1,0,0}}};

    int accessed;
    int obj,usercategory,accessright;
    AccessCheck(int uc,int o,int ac)
    {
        obj =o;
        usercategory = uc-1;
        accessright = ac;
        accessed = accessMatrix[obj][usercategory][accessright];
    }
}
APPENDIX 2

AGENA RISK TOOL INTRODUCTION

A2.1 AGENARISK TOOL FOR SIMULATION

AgenaRisk enables the user to build models that contain a mixture of discrete and continuous nodes and functions. The past generation of Bayesian Network tools require users to define the states of any numeric node (whether it is continuous or discrete) as a sequence of pre-defined intervals. For instance, if we have a numeric variable like “cost” then instead of just specifying that the node ranges from, say 0 to 100000, you would have to break up 0 to 100000 into a manageable number of discrete intervals. The more no of intervals that is defined, more the accuracy could be achieved, but at a heavy cost of computational complexity. A thumb rule says that it is never advisable to allow any node to have more than 100 states. The discretization process is not only time-consuming, but is made worse by the fact that the user do not necessarily know in advance which ranges require the finer intervals. It follows that where a model contains numerical nodes having a potentially large range, results are necessarily only crude approximations. We can still model numerical quantities in this way with AgenaRisk, however AgenaRisk also solves this critical problem by allowing us to define numeric nodes as simulation nodes AgenaRisk then automatically simulates the correct states for nodes so we do not need to pay much attention to the specification of node intervals.
A2.2  STATISTICAL DISTRIBUTIONS

The continuous and discrete statistical distributions that are supported in AgenaRisk are listed in Table A2.1.

Table A2.1 Statistical Distribution

<table>
<thead>
<tr>
<th>Continuous Uniform</th>
<th>Chi Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
<td>Exponential</td>
</tr>
<tr>
<td>Gamma</td>
<td>Normal</td>
</tr>
<tr>
<td>Weibull</td>
<td>Truncated Normal</td>
</tr>
<tr>
<td>Logistics</td>
<td>Extreme Value</td>
</tr>
<tr>
<td>Triangular</td>
<td>Log Normal</td>
</tr>
<tr>
<td>Integer Uniform</td>
<td>Binomial</td>
</tr>
<tr>
<td>Negative Binomial</td>
<td>Hyper geometric</td>
</tr>
<tr>
<td>Poisson</td>
<td>Geometric</td>
</tr>
</tbody>
</table>
APPENDIX 3

PROLOG RULES AND QUERY PROCESSING TECHNIQUES

A3.1 RULES BASED ON CONJUNCTIONS

Rules based on conjunctions are illustrated with the following example. Consider the statement “A man is happy if he is rich and famous” might be translated to prolog form as given below.

\[
\text{happy(Person):-}
\]

\[
\text{man(Person),}
\]

\[
\text{rich(Person),}
\]

\[
\text{famous(Person).}
\]

The ‘:-’ means implication and read as implies. The ‘,’ indicates the conjunction and is roughly equivalent to the $\land$ of predicate calculus. Therefore, read ‘,’ as ‘and’. The whole of the above is one (non-unit) single clause. It has three sub goals in its body. These sub goals are ‘conjoined’. In this case, conjunctions are represented using an AND tree. Here is an AND tree that represents the above.

![AND Tree Diagram](image)

Figure A3.1 First order predicate logic AND conjunction Tree
A3.2 RULES BASED ON DISJUNCTIONS

Rules based on disjunctions are illustrated with the following example. Consider the statement “Someone is happy if he/she is healthy, wealthy or wise” might be translated to prolog form as given below.

happy(Person):-healthy(Person).
happy(Person):-wealthy(Person).
happy(Person):-wise(Person).

This prolog translation is obtained by rewriting the original informal statement into something like:

Someone is happy if he/she is healthy or
Someone is happy if he/she is wealthy or
Someone is happy if he/she is wise

A3.3 RULES BASED ON BOTH DISJUNCTIONS AND CONJUNCTIONS

Rules could be formed based on both conjunctions and disjunctions. This could be illustrated by the example statement “woman is happy if she is healthy, wealthy or wise.”

This statement is translated in prolog as below.

happy(Person):-healthy(Person),woman(Person).
happy(Person):-wealthy(Person),woman(Person).
happy(Person):-wise(Person),woman(Person).

The combination of the OR tree representation together with an AND tree representation forms an AND/OR tree that shows the structure of the definition of happy person.
A3.4 DISJUNCTIONS BASED QUERIES

Informally, a query is a goal which is submitted to prolog in order to determine whether this goal is true or false. At top level, prolog normally expects queries by prompting the symbol “?- “. When it is prompted type in one or more goals. This is given by an example

?- woman(Jane).

Here the meaning of the query is to check out whether Jane is a woman or not. In the process of obtaining the result to the query, one must search through the rules and facts known by prolog to find out whether this is so. It is important to note the distinction between facts and rules in prolog. For example, prolog does not search through the facts before the rules.

Here are some facts assumed to be known

<table>
<thead>
<tr>
<th>Program Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman( jean)</td>
</tr>
<tr>
<td>man( fred)</td>
</tr>
<tr>
<td>woman( jane)</td>
</tr>
<tr>
<td>woman( joan)</td>
</tr>
<tr>
<td>woman( pat)</td>
</tr>
</tbody>
</table>
The answer is simple. Prolog searches through the set of clauses in the same way that read, that is from top to bottom. The search space might appear using the AND/OR tree representation. The tree might look like Figure A3.3.

![Figure A3.3 First order logic OR tree (search space tree) for query processing](image)

In the given search space representation tree it is understood that the search would zigzag across the page from left to right and stop when the solution is found.

### A3.5 CONJUNCTIONS BASED QUERIES

Now to look at a query that requires prolog to solve two sub goals. Here given is a set of facts and a rule.

<table>
<thead>
<tr>
<th>Program Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman(jean)</td>
</tr>
<tr>
<td>man(fred)</td>
</tr>
<tr>
<td>wealthy(fred)</td>
</tr>
<tr>
<td>happy(Person): - woman(Person),Wealthy(Person).</td>
</tr>
</tbody>
</table>

The prolog interpretation of the query “Is Jean happy?“ is given below

?- happy(Jean)
In this case the match produces a substitution, Person to Jean, and two sub goals replace the current goal. The substitution of Person by Jean is known as a unifier and often written Person/Jean. The process of replacing a single goal by one or more sub goals with whatever substitutions are applicable is part of the resolution process.

Prolog searches the tree for AND choices are to zigzag from left to right across the page. This is similar to how it processes the OR choices except that Prolog must satisfy all the AND choices at a node before going on. Zigzagging from left to right is not the whole story for the goal. Once wealthy (Person) is reached with Person/jean and failed then move back (backtracking) to the goal woman (Person) and break the binding for Person which is made already. Now the tree for AND choice is explored from left to right again.

A3.6 QUERIES BASED ON BOTH CONJUNCTIONS AND DISJUNCTIONS

Consider the following basic facts

<table>
<thead>
<tr>
<th>PROGRAM DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman(jean).</td>
</tr>
<tr>
<td>woman(jane).</td>
</tr>
<tr>
<td>woman(joan).</td>
</tr>
<tr>
<td>woman(pat).</td>
</tr>
<tr>
<td>wise(jean).</td>
</tr>
<tr>
<td>wealthy(jane).</td>
</tr>
<tr>
<td>wealthy(jim).</td>
</tr>
<tr>
<td>healthy(jim).</td>
</tr>
<tr>
<td>healthy(jane).</td>
</tr>
<tr>
<td>healthy(jean).</td>
</tr>
<tr>
<td>happy(P) :- healthy(P), woman(P)</td>
</tr>
<tr>
<td>happy(P) :- wealthy(P), woman(P)</td>
</tr>
<tr>
<td>happy(P) :- wise(P), woman(P)</td>
</tr>
</tbody>
</table>
The standard AND/OR tree representation of the search space is given below.

![AND/OR Tree](image)

**Figure A3.4** First order predicate logic AND/OR tree for query `?-happy(jean)`

Let us look into the solution of the goal “?-happy(jean)”.

It is important to note that

1. The whole goal is succeeded only when both the sub goals `healthy(jean)` and `woman(jean)` are succeeded.

2. Then return to the top level.

Prolog provides the facility to *redo* a goal whenever the top level goal has succeeded and there is still a variable binding is available. This could be achieved by just typing “;” followed by RETURN “;” can be read as *or*. If there is another solution exists, Prolog finds it. This process is repeated until there are no more solutions and provides us the sequence of all possible solutions.

Let us consider the new Goal “?-happy(P);” This query will list out all the happy persons and according to the known program database, the result produced is given as below

```
jane
jean
```
The matching process known as unification has already been met. The basic idea of recursion as a programming technique is reviewed and applied to list processing.