A Java software for drawing graphs

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China; International Academy of Ecology and Environmental Sciences, Hong Kong
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 7 May 2011; Accepted 13 August 2011; Published online 1 March 2012
IAEES

Abstract
In this study the software for drawing graphs, which is run as a Java application, was described. It can be freely downloaded and run on Windows platforms. The software can be used to draw directed, undirected, cyclic and acyclic graphs.

Keywords software; graph; drawing; Java.

1 Introduction
Java is an object oriented and platform independent programming language. It has been widely used in the development of various biological software (Liu and Zhang, 2011; Zhang, 2011a, b). A lot of network and graph related software have been developed using Java, e.g., NetLogo, Repast, et al.

In the following sections, the software for drawing various graphs, running as a Java application, is described (Zhang, 2012). It can be freely downloaded and run on various Windows platforms.

2 Array Storage of Graph
The graph in present study is stored in computer using a method, Two Linear Array (Zhang, 2012). For instance, the directed graph in Fig. 1 can be expressed in Two Linear Array as:

\[ S_1 = (v_1, v_1, v_1, v_2, v_2, v_2, v_3, v_3, v_5, v_6), \]
\[ S_2 = (v_2, v_4, v_3, v_2, v_4, v_4, v_4, v_5, v_6), \]
\[ S = (1, 1, -1, 3, 1, 1, 2, -2, 3, 0). \]

Fig. 1 A directed graph (Zhang, 2012)
Changing the graph in Fig. 1 to undirected graph, its Two Linear Array representation will be:

\[ S_1 = (v_1, v_1, v_2, v_2, v_2, v_3, v_4, v_5, v_6), \]
\[ S_2 = (v_2, v_3, v_4, v_2, v_3, v_4, v_3, v_5, v_6), \]
\[ S = (1, -1, 1, 3, 1, 1, 2, 2, 5, 4). \]

where 1 denotes there is only one edge between two vertices, and the edge is a positive-directed edge; -1 denotes there is only one edge between two vertices, but the edge is a negative-directed edge; 2 denotes two parallel edges; 3 denotes self-loop; 4 denotes isolated vertex, and 5 denotes the self-loop of an isolated vertex.

### 3 Java Implementation

The software was implemented based on JDK 1.1.8, in which several classes were included (http://www.iaees.org/publications/software/index.asp; In BioNetAnaly). The following are Java codes, netGenerator, re-invented from JDK and Zhang (2007), for generating a graph (Zhang, 2012):

```java
public class netGenerator {
    public static void main(String[] args) {
        String tablename = args[0];
        readDatabase readdata = new readDatabase("dataBase", tablename, 3);
        int mm = readdata.m;
        String s1[] = new String[mm + 1];
        String s2[] = new String[mm + 1];
        int s[] = new int[mm + 1];
        for (int i = 1; i <= mm; i++) {
            s1[i] = readdata.data[i][1];
            s2[i] = readdata.data[i][2];
            s[i] = (Integer.valueOf(readdata.data[i][3])).intValue();
        }
        new GraphicsFrame(new NetGraph(s1, s2, s), "Black and blue edges are forward and backward edges respectively. Self-loop is labeled by semicircle on the vertex. Parallel edges are labeled by between-vertex parallel lines.").resize(720, 550);
    }
}
```

The following are Java classes, NetGraph, NetVertex, NetEdge, and NetPanel, loaded by above class, netGenerator:

```java
import java.util.*;
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
public class NetGraph extends Applet implements ActionListener {
    Button close;
    NetPanel panel;
    Panel controlPanel;
    String s[][], c[];
    int e, v, m;
    public int t[];

```
public NetGraph(String s1[], String s2[], int tt[]) {
    e=s1.length;
    s=new String[5][e+1];
    c=new String[2000];
    t=new int[e+1];
    for(int i=1;i<=e;i++) {
        s[1][i]=s1[i];
        s[2][i]=s2[i];
        t[i]=tt[i];}
    v=1;
    c[1]=s[1][1];
    for(int i=1;i<=2;i++)
        for(int j=1;j<=e;j++) {
            m=0;
            for(int k=1;k<=v;k++)
                if (!(s[i][j].equals(c[k]))) m++;
            if (m==v) {
                v++;
                c[v]=s[i][j];}
        }
    begin();
}

draw begin) {
    close=new Button("Close");
    setLayout(new BorderLayout());
    panel=new NetPanel();
    add("Center", panel);
    controlPanel=new Panel();
    add("South",controlPanel);
    controlPanel.add(close); close.add ActionListener(this);
    for(int k=1;k<=e;k++)
        panel.addEdge(s[1][k],s[2][k],t[k]);
    Dimension d=getSize();
    resize(d.width-15, d.height-15);
    setLocation(20,20);
    validate();
    show();
}

draw init() {
}

draw repaint(Graphics g) {
    repaint();
}

draw destroy() {
    remove(panel);
    remove(controlPanel); 
}

draw actionPerformed(ActionEvent e) {
    Object src=e.getSource();
    if (src==close) {
        this.hide();
        this.getParent().hide();
        System.exit(0); }
    return; 
}

draw getAppletInfo() {
    return "Ecological Network Generator"; 
}
class NetVertex {
    double x, y, w, h;
    boolean fixed;
    String lab;
}

class NetEdge {
    int from, to, type;
}

class NetPanel extends Panel implements MouseListener, MouseMotionListener {
    int nVertex, nEdges;
    NetVertex vertex[] = new NetVertex[2000];
    NetEdge edges[] = new NetEdge[50000];
    NetVertex pick;
    boolean pickFixed;
    Image offscreen;
    Dimension offscreensize;
    Graphics offgraphics;
    Color posColor = Color.black;
    Color negColor = Color.blue;

    public NetPanel() {
        addMouseListener(this);
        addMouseMotionListener(this);
    }

    public int findVertex(String lab) {
        for (int i = 0; i < nVertex; i++)
            if (vertex[i].lab.equals(lab)) return i;
        return addVertex(lab);
    }

    public int addVertex(String lab) {
        NetVertex v = new NetVertex();
        v.x = 100 + 350 * Math.random();
        v.y = 100 + 350 * Math.random();
        v.lab = lab;
        vertex[nVertex] = v;
        return nVertex++;
    }

    public void addEdge(String from, String to, int type) {
        NetEdge e = new NetEdge();
        e.from = findVertex(from);
        e.to = findVertex(to);
        e.type = type;
        edges[nEdges++] = e;
    }

    public void paintVertex(Graphics g, FontMetrics mtr) {
        int x = (int) v.x;
        int y = (int) v.y;
        g.setColor(Color.white);
        int w = mtr.stringWidth(v.lab) + 10;
        int h = mtr.getHeight() + 4;
        v.w = w;
        v.h = h;
        g.fillRect(x - w / 2, y - h / 2, w, h);
        g.setColor(Color.black);
        g.drawRect(x - w / 2, y - h / 2, w - 1, h - 1);
        g.drawString(v.lab, x - (w - 10) / 2, (y - (h - 4) / 2) + mtr.getAscent());
    }
public void update(Graphics g) {
    Dimension d=getSize();
    if ((offscreen==null) || (d.width!=offscreensize.width) || (d.height!=offscreensize.height)) {
        offscreen=createImage(d.width, d.height);
        offscreensize=d;
        offgraphics=offscreen.getGraphics();
        offgraphics.setFont(getFont());
        offgraphics.setColor(getBackground());
        offgraphics.fillRect(0,0,d.width,d.height);
        for(int i=0;i<nedges;i++) {
            NetEdge e=edges[i];
            int x1=(int)vertice[e.from].x;
            int y1=(int)vertice[e.from].y;
            int x2=(int)vertice[e.to].x;
            int y2=(int)vertice[e.to].y;
            if ((e.type==1) | (e.type==-1)) {
                if (e.type==1) offgraphics.setColor(posColor);
                else offgraphics.setColor(negColor);
                offgraphics.drawLine(x1,y1,x2,y2);
            }
            if (e.type==2) {
                offgraphics.setColor(posColor);
                offgraphics.drawLine(x1,y1-5,x2,y2-5);
                offgraphics.setColor(negColor);
                offgraphics.drawLine(x1,y1+5,x2,y2+5);
            }
            if ((e.type==3) | (e.type==5)) {
                int w=(int)vertice[e.from].w;
                int h=(int)vertice[e.from].h;
                int rad=w-1;
                offgraphics.setColor(posColor);
                offgraphics.drawArc(x1-(int)(0.5*w),y1-h,rad,rad,0,360);
            }
        }
        FontMetrics mtr=offgraphics.getFontMetrics();
        for(int i=0;i<nvertice;i++) paintVertex(offgraphics,vertice[i],mtr);
        g.drawImage(offscreen,0,0,null);
    }
}

public void paint(Graphics g) {
    repaint();
}

public void mousePressed(MouseEvent e) {
    double bestdist=Double.MAX_VALUE;
    int x=e.getX();
    int y=e.getY();
    for(int i=0;i<nvertice;i++) {
        NetVertex v=vertice[i];
        double dist=(v.x-x)*(v.x-x)+(v.y-y)*(v.y-y);
        if (dist<bestdist) {
            pick=v;
            bestdist=dist;
        }
        pickfixed=pick.fixed;
        pick.fixed=true;
        pick.x=x;
        pick.y=y;
        repaint();
e.consume();
    }
}

public void mouseReleased(MouseEvent e) {
    pick.x=e.getX();
    pick.y=e.getY();
    pick.fixed=pickfixed;
}
public void mouseDragged(MouseEvent e) {
pick.x = e.getX();
pick.y = e.getY();
repaint();
e.consume(); } public void mouseClicked(MouseEvent e) {} public void mouseEntered(MouseEvent e) {} public void mouseExited(MouseEvent e) {} public void mouseMoved(MouseEvent e) {} }

The following is the class GraphicsFrame, used by class netGenerator:

import java.awt.*;
import java.applet.*;
public class GraphicsFrame extends Frame {
public GraphicsFrame(Applet applet) {
this.resize(600,400);
add(applet);
setVisible(true); }

public GraphicsFrame(Applet applet, String str) {
this.resize(600,400);
this.setTitle(str);
add(applet);
setVisible(true); }
}

4 Application

Some ecological graphs have been drawn using the software, for example, the graphs in Fig. 2 and Fig. 6 of Zhang (2011c), and the graph in Fig. 2.

![Fig. 2 An arthropod interactions graph generated by software](image-url)
References


Zhang WJ. 2011a. A Java program to test homogeneity of samples and examine sampling completeness. Network Biology, 1(2): 127-129

Zhang WJ. 2011b. An algorithm for calculation of degree distribution and detection of network type: with application in food webs. Network Biology, 1(3-4): 159-170
