Article

netGen 3.0: The executable Java software for network visualization

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 2 April 2023; *Accepted 21 April 2023*; *Published online 1 May 2023*; *Published 1 June 2024*

Abstract
The executable Java software, netGen 3.0, for network visualization was developed in present study. A network can be generated and interactively visualized from a text file by using netGen. Compared to the previous versions, there are substantial improvements in netGen 3.0. In addition to the visualization of both unweighted and weighted (and both undirected and directed) networks, node size / color, link width / length / color, node name color, canvas size, etc., can be specified by the user. Moreover, network layout can be conducted in artificial or automatic manners. Both netGen 3.0 and demonstration data files were given.

Keywords network visualization; Java; software; netGen.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology /rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Network construction is the fundamental of network analysis. Networks, in particular biological networks, are usually constructed from random sampling (Zhang, 2011, 2012c; Zhang et al., 2014), artificial experiments or reported big data (Huang and Zhang, 2012; Li and Zhang, 2013; Jiang and Zhang, 2015; Jiang et al., 2015; Qi et al., 2018; Zhang and Zhang, 2019; Xin and Zhang, 2020, 2021; Zhang, 2021b, 2016b, 2018; Yang and Zhang, 2022), or theoretical simulation (Zhang, 2016a; Zhang and Li, 2016). Network visualization is the visual construction of networks (Narad et al., 2017; Zhang, 2024 a-c). So far, numerious software and tools for network visualization have been developed and used (Zhang, 2007, 2012 a-b, 2024 a-c; Li and Zhang, 2013; Zhang and Zhang, 2019; Xin and Zhang, 2020, 2021; Yang and Zhang, 2022). For examples, I have presented the online web tools for visualizing user-interface interactive networks in the earlier studies (Zhang, 2021a; Zhang, 2024 a-b). In the generated network, the user can mouse-press any of the nodes to drag the network, examine network topology, evaluate node centrality, etc. They are interactive tools but, the layout of the generated network cannot be easily controlled by the user and, they are not able to be used for weighted networks. Also, in my earlier studies I have developed Java tools (netGen series) for interactive network visualization (Zhang, 2007, 2012a). In the version 1.0 of the netGen (Zhang, 2012a), the data must be loaded with ODBC database and the tool is just a Java application. In the version 2.0 of the netGen (Zhang, 2024c), I have re-developed the executable Java software, which used text file only and the layout of generated network was greatly improved. In present study, I developed netGen 3.0. In netGen 3.0, in addition to the visualization of both unweighted and weighted (and both undirected and directed) networks, node size / color, link width / length / color, node name color, canvas size, etc., can be specified by the user. Moreover, network layout can be conducted in artificial or automatic manners. Both netGen 3.0 and demonstration data files are given.

2 Software and Data

2.1 Software and runtime environment

The Java software, netGen version 3.0, was developed based on JDK (Java Development Kit) and J2SDK1.4.2. Five Java classes, NetGraph, NetVertex, NetEdge, NetPanel, and GraphicsFrame (Zhang, 2007, 2012a, 2024b) are included and loaded by the class, netGenerator. The following are Java codes of the class netGenerator:

```
import java.awt.*;
import java.io.*;

public class netGenerator extends Frame
{

    public netGenerator()
    {
        byte byte0 = 15;
        int i = 1;
        panel = new Panel();
        gbl = new GridBagLayout();
        gbc = new GridBagConstraints();
        panel.setLayout(gbl);
        gbc.fill = 1;
        cbg = new CheckboxGroup();
        cb1 = new Checkbox(" ", cbg, true);
        cb2 = new Checkbox(" ", cbg, false);
        label = new Label("Network Types: ");
        cb1.setLabel("Directed Network");
        cb2.setLabel("Undirected Network");
        layout(0, 0, byte0, i, label);
        layout(0, 1, byte0, i, cb1);
        layout(byte0, 1, byte0 + byte0, i, cb2);
        cbgg = new CheckboxGroup();
        cb11 = new Checkbox(" ", cbgg, true);
        cb22 = new Checkbox(" ", cbgg, false);
        label9 = new Label("Unweighted or Weighted Network: ");
        cb11.setLabel("Unweighted Network");
        cb22.setLabel("Weighted Network");
        layout(0, 2, byte0, i, label9);
        layout(0, 3, byte0, i, cb11);
        layout(byte0, 3, byte0 + byte0, i, cb22);
        cbggg = new CheckboxGroup();
        cb111 = new Checkbox(" ", cbggg, true);
```

```
cb222 = new Checkbox(" ", cbggg, false);
label10 = new Label("Network Layout: ");
cb111.setLabel("Artificially");
cb222.setLabel("Automatically");
layout(0, 4, byte0, i, label10);
layout(0, 5, byte0, i, cb111);
layout(byte0, 5, byte0 + byte0, i, cb222);
label1 = new Label("Node Size: ");
label2 = new Label("Link Length: ");
label3 = new Label("Link Width: ");
edit1 = new TextField("15");
edit2 = new TextField("250");
edit3 = new TextField("1");
layout(0, 6, byte0, i, label1);
layout(byte0, 6, byte0 + byte0, i, edit1);
layout(0, 7, byte0, i, label2);
layout(byte0, 7, byte0 + byte0, i, edit2);
layout(0, 8, byte0, i, label3);
layout(byte0, 8, byte0 + byte0, i, edit3);
label7 = new Label("Canvas Width: ");
label8 = new Label("Canvas Height: ");
edit4 = new TextField("900");
edit5 = new TextField("680");
layout(0, 9, byte0, i, label7);
layout(byte0, 9, byte0 + byte0, i, edit4);
layout(0, 10, byte0, i, label8);
layout(byte0, 10, byte0 + byte0, i, edit5);
label4 = new Label("Choose node color: ");
label5 = new Label("Choose node name color: ");
label6 = new Label("Choose link color: ");
list1 = new List();
list1.addItem("Blue");
list1.addItem("Black");
list1.addItem("Gray");
list1.addItem("Orange");
list1.addItem("Red");
list1.addItem("Magenta");
list1.addItem("Green");
list1.addItem("Cyan");
list1.addItem("White");
list2 = new List();
list2.addItem("Blue");
list2.addItem("Black");
list2.addItem("Gray");
list2.addItem("Orange");
```

```
        list2.addItem("Red");
        list2.addItem("Magenta");
        list2.addItem("Green");
        list2.addItem("Cyan");
        list2.addItem("White");
        list3 = new List();
        list3.addItem("Blue");
        list3.addItem("Black");
        list3.addItem("Gray");
        list3.addItem("Orange");
        list3.addItem("Red");
        list3.addItem("Magenta");
        list3.addItem("Green");
        list3.addItem("Cyan");
        list3.addItem("White");
        list1.select(0);
        list2.select(5);
        list3.select(0);
        layout(0, 11, byte0 + byte0, i, label4);
        layout(0, 12, byte0 + byte0, i, list1);
        layout(0, 13, byte0 + byte0, i, label5);
        layout(0, 14, byte0 + byte0, i, list2);
        layout(0, 15, byte0 + byte0, i, label6);
        layout(0, 16, byte0 + byte0, i, list3);
        buttonok = new Button("OK");
        layout(0, 17, byte0 + byte0, 2 * i, buttonok);
        add(panel);
        resize(200, 600);
        setLocation(200, 80);
        pack();
        show();
    }

    public void layout(int i, int j, int k, int l, Component component)
    {
        gbc.gridx = i;
        gbc.gridy = j;
        gbc.gridwidth = k;
        gbc.gridheight = l;
        gbl.setConstraints(component, gbc);
        panel.add(component);
    }

    public boolean action(Event event, Object obj)
    {
```

```
if(event.target == buttonok)
{
    if(cbg.getCurrent() == cb1)
        sele = 1;
    else
    if(cbg.getCurrent() == cb2)
        sele = 0;
    if(cbgg.getCurrent() == cb11)
        selew = 0;
    else
    if(cbgg.getCurrent() == cb22)
        selew = 1;
    if(cbggg.getCurrent() == cb111)
        selel = 0;
    else
    if(cbggg.getCurrent() == cb222)
        selel = 1;
    nodeDiam = Integer.valueOf(edit1.getText().trim()).intValue();
    linkLen = Integer.valueOf(edit2.getText().trim()).intValue();
    linkWid = Integer.valueOf(edit3.getText().trim()).intValue();
    canvaswid = Integer.valueOf(edit4.getText().trim()).intValue();
    canvashei = Integer.valueOf(edit5.getText().trim()).intValue();
    for(int i = 0; i <= 8; i++)
    {
        if(list1.isIndexSelected(i))
            colorn = i;
        if(list2.isIndexSelected(i))
            colornn = i;
        if(list3.isIndexSelected(i))
            colorl = i;
    }

    hide();
    Frame frame = new Frame();
    FileDialog filedialog = new FileDialog(frame, "Open network data file", 0);
    frame.setLocation(250, 150);
    filedialog.show();
    String s = filedialog.getDirectory() + filedialog.getFile();
    try
    {
        DataInputStream datainputstream = new DataInputStream(new FileInputStream(s));
        int j;
        for(j = 0; datainputstream.readLine() != null; j++);
        datainputstream.close();
        args1 = new String[j + 1];
```

```
                        args2 = new String[j + 1];
                        ai = new int[j + 1];
                        w = new double[j + 1];
                        DataInputStream datainputstream1 = new DataInputStream(new FileInputStream(s));
                        String s1;
                        for(int k = 0; (s1 = datainputstream1.readLine()) != null; k++)
                        {
                            String as[] = s1.split(",");
                            args1[k + 1] = as[0];
                            args2[k + 1] = as[1];
                            ai[k + 1] = Integer.parseInt(as[2]);
                            if(selew == 1)
                            {
                                w[k + 1] = Double.parseDouble(as[3]);
                                continue;
                            }
                            if(selew == 0)
                                w[k + 1] = 0.0D;
                        }

                        datainputstream.close();
                    }
                    catch(Exception exception) { }
                    (new GraphicsFrame(new NetGraph(args1, args2, ai, w, sele, selew, selel, nodeDiam, linkWid, linkLen, colorn,
colornn, colorl, canvaswid, canvashei), "Zhang WJ. 2024. netGen 3.0: The executable Java software for network visualization.
Selforganizology, 11(1-2): 1")).resize(canvaswid, canvashei);
                    return true;
                } else
                {
                    return false;
                }
            }

            public static void main(String args[])
                throws IOException
            {
                new netGenerator();
            }

            public String args1[];
            public String args2[];
            public int ai[];
            public double w[];
            public int sele;
            public int selew;
```

```
        public int selel;
        public int nodeDiam;
        public int linkWid;
        public int linkLen;
        public int colorn;
        public int colornn;
        public int colorl;
        public int canvaswid;
        public int canvashei;
        public CheckboxGroup cbg;
        public CheckboxGroup cbgg;
        public CheckboxGroup cbggg;
        public Checkbox cb1;
        public Checkbox cb11;
        public Checkbox cb111;
        public Checkbox cb2;
        public Checkbox cb22;
        public Checkbox cb222;
        public Button buttonok;
        public TextField edit1;
        public TextField edit2;
        public TextField edit3;
        public TextField edit4;
        public TextField edit5;
        public Label label;
        public Label label1;
        public Label label2;
        public Label label3;
        public Label label4;
        public Label label5;
        public Label label6;
        public Label label7;
        public Label label8;
        public Label label9;
        public Label label10;
        public List list1;
        public List list2;
        public List list3;
        public Panel panel;
        public GridBagLayout gbl;
        public GridBagConstraints gbc;
   }
```

The following are Java codes of the class NetPanel:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.GeneralPath;

class NetPanel extends Applet
    implements Runnable, MouseListener, MouseMotionListener
{

    public NetPanel(double ad[], int i, int j, int k, int l, int i1, int j1,
            int k1, int l1, int i2, int j2, int k2)
    {
        sele = i;
        selew = j;
        selel = k;
        nodeDiam = l;
        linkWid = i1;
        linkLen = j1;
        colorn = k1;
        colornn = l1;
        colorl = i2;
        canvaswid = j2;
        canvashei = k2;
        sign = 0;
        bs1 = new BasicStroke(linkWid);
        switch(k1)
        {
        case 0: // '\0'
            ncolor = Color.blue;
            break;

        case 1: // '\001'
            ncolor = Color.black;
            break;

        case 2: // '\002'
            ncolor = Color.gray;
            break;

        case 3: // '\003'
            ncolor = Color.orange;
            break;
```

```
case 4: // '\004'
        ncolor = Color.red;
        break;

case 5: // '\005'
        ncolor = Color.magenta;
        break;

case 6: // '\006'
        ncolor = Color.green;
        break;

case 7: // '\007'
        ncolor = Color.cyan;
        break;

case 8: // '\b'
        ncolor = Color.white;
        break;
}
switch(l1)
{
case 0: // '\0'
        nncolor = Color.blue;
        break;

case 1: // '\001'
        nncolor = Color.black;
        break;

case 2: // '\002'
        nncolor = Color.gray;
        break;

case 3: // '\003'
        nncolor = Color.orange;
        break;

case 4: // '\004'
        nncolor = Color.red;
        break;

case 5: // '\005'
        nncolor = Color.magenta;
        break;
```

```
case 6: // '\006'
    nncolor = Color.green;
    break;

case 7: // '\007'
    nncolor = Color.cyan;
    break;

case 8: // '\b'
    nncolor = Color.white;
    break;
}
switch(i2)
{
case 0: // '\0'
    lcolor = Color.blue;
    break;

case 1: // '\001'
    lcolor = Color.black;
    break;

case 2: // '\002'
    lcolor = Color.gray;
    break;

case 3: // '\003'
    lcolor = Color.orange;
    break;

case 4: // '\004'
    lcolor = Color.red;
    break;

case 5: // '\005'
    lcolor = Color.magenta;
    break;

case 6: // '\006'
    lcolor = Color.green;
    break;

case 7: // '\007'
    lcolor = Color.cyan;
```

```
                break;

            case 8: // '\b'
                lcolor = Color.white;
                break;
            }
            int l2 = ad.length - 1;
            ww = new double[l2 + 1];
            for(int i3 = 1; i3 <= l2; i3++)
                ww[i3] = ad[i3];

            vertice = new NetVertex[10000];
            edges = new NetEdge[0x186a0];
            addMouseListener(this);
            addMouseMotionListener(this);
        }

        public int findVertex(String s)
        {
            for(int i = 0; i < nvertice; i++)
                if(vertice[i].lab.equals(s))
                    return i;

            return addVertex(s);
        }

        public int addVertex(String s)
        {
            NetVertex netvertex = new NetVertex();
            netvertex.x = (double)(nodeDiam + 10) + (double)(canvaswid - nodeDiam - 50) * Math.random();
            netvertex.y = (double)(nodeDiam + 10) + (double)(canvashei - nodeDiam - 120) * Math.random();
            netvertex.lab = s;
            netvertex.fixed = false;
            vertice[nvertice] = netvertex;
            return nvertice++;
        }

        public void addEdge(String s, String s1, int i)
        {
            NetEdge netedge = new NetEdge();
            netedge.from = findVertex(s);
            netedge.to = findVertex(s1);
            netedge.type = i;
            edges[nedges++] = netedge;
        }
```

```
public void paintVertex(Graphics g, NetVertex netvertex, FontMetrics fontmetrics)
{
    int i = (int)netvertex.x;
    int j = (int)netvertex.y;
    int k = fontmetrics.stringWidth(netvertex.lab) + 10;
    int l = fontmetrics.getHeight() + 4;
    int i1 = fontmetrics.getAscent();
    netvertex.w = k;
    netvertex.h = l;
    g.setColor(ncolor);
    g.fillOval(i, j, nodeDiam, nodeDiam);
    g.drawOval(i, j, nodeDiam, nodeDiam);
    g.setColor(nncolor);
    g.drawString(netvertex.lab, i - (k - 10) / 2, (j - (l - 4) / 2) + i1 + nodeDiam + 10);
}

public synchronized void update(Graphics g)
{
    Dimension dimension = getSize();
    if(offscreen == null || dimension.width != offscreensize.width || dimension.height != offscreensize.height)
    {
        offscreen = createImage(dimension.width, dimension.height);
        offscreensize = dimension;
        offgraphics = offscreen.getGraphics();
        offgraphics.setFont(getFont());
    }
    offgraphics.setColor(getBackground());
    offgraphics.fillRect(0, 0, dimension.width, dimension.height);
    for(int i = 0; i < nedges; i++)
    {
        NetEdge netedge = edges[i];
        int k = (int)vertice[netedge.from].x;
        int l = (int)vertice[netedge.from].y;
        int i1 = (int)vertice[netedge.to].x;
        int j1 = (int)vertice[netedge.to].y;
        if(sele == 0)
        {
            Graphics2D graphics2d = (Graphics2D)offgraphics;
            graphics2d.setStroke(bs1);
            graphics2d.setColor(lcolor);
            graphics2d.drawLine(k + nodeDiam / 2, l + nodeDiam / 2, i1 + nodeDiam / 2, j1 + nodeDiam / 2);
            if(selew == 1)
            {
                graphics2d.setColor(nncolor);
```

```
                    graphics2d.drawString(String.valueOf(ww[i + 1]), k + (i1 - k) / 2, l + (j1 - l) / 2);
                }
                continue;
            }
            if(sele != 1)
                continue;
            if(netedge.type == 1)
            {
                if(i1 >= k)
                    paintArrow(offgraphics, k + nodeDiam / 2, l + nodeDiam / 2, i1, j1 + nodeDiam / 2, i);
                if(i1 < k)
                    paintArrow(offgraphics, k + nodeDiam / 2, l + nodeDiam / 2, i1 + nodeDiam, j1 + nodeDiam / 2, i);
            }
            if(netedge.type == -1)
                paintArrow(offgraphics, i1, j1 + nodeDiam / 2, k + nodeDiam / 2, l + nodeDiam / 2, i);
        }

        FontMetrics fontmetrics = offgraphics.getFontMetrics();
        for(int j = 0; j < nvertice; j++)
            paintVertex(offgraphics, vertice[j], fontmetrics);

        g.drawImage(offscreen, 0, 0, null);
        if((selel == 1) & (sign == 0))
        {
            sign = 1;
            selforganizor = new Thread(this, "");
            selforganizor.start();
        }
    }

    public void paintArrow(Graphics g, int i, int j, int k, int l, int i1)
    {
        double d = nodeDiam;
        double d1 = linkWid + nodeDiam / 5;
        int j1 = 0;
        int k1 = 0;
        int l1 = 0;
        int i2 = 0;
        double d2 = Math.atan(d1 / d);
        double d3 = Math.sqrt(d1 * d1 + d * d);
        double ad[] = rotateVec(k - i, l - j, d2, true, d3);
        double ad1[] = rotateVec(k - i, l - j, -d2, true, d3);
        double d4 = (double)k - ad[0];
        double d5 = (double)l - ad[1];
        double d6 = (double)k - ad1[0];
```

```
            double d7 = (double)l - ad1[1];
            Double double1 = new Double(d4);
            j1 = double1.intValue();
            Double double2 = new Double(d5);
            k1 = double2.intValue();
            Double double3 = new Double(d6);
            l1 = double3.intValue();
            Double double4 = new Double(d7);
            i2 = double4.intValue();
            Graphics2D graphics2d = (Graphics2D)g;
            graphics2d.setStroke(bs1);
            graphics2d.setColor(lcolor);
            graphics2d.drawLine(i, j, k, l);
            if(selew == 1)
            {
                graphics2d.setColor(nncolor);
                graphics2d.drawString(String.valueOf(ww[i1 + 1]), (i - nodeDiam / 2) + (k - i) / 2, (j - nodeDiam / 2) + (l - j) /
2);
            }
            graphics2d.setColor(lcolor);
            GeneralPath generalpath = new GeneralPath();
            generalpath.moveTo(k, l);
            generalpath.lineTo(j1, k1);
            generalpath.lineTo(l1, i2);
            generalpath.closePath();
            graphics2d.fill(generalpath);
        }

        public double[] rotateVec(int i, int j, double d, boolean flag, double d1)
        {
            double ad[] = new double[2];
            double d2 = (double)i * Math.cos(d) - (double)j * Math.sin(d);
            double d3 = (double)i * Math.sin(d) + (double)j * Math.cos(d);
            if(flag)
            {
                double d4 = Math.sqrt(d2 * d2 + d3 * d3);
                d2 = (d2 / d4) * d1;
                d3 = (d3 / d4) * d1;
                ad[0] = d2;
                ad[1] = d3;
            }
            return ad;
        }

        public void run()
```

```
{
    Thread thread = Thread.currentThread();
    do
    {
        if(selforganizor != thread)
            break;
        selfOrganize();
        try
        {
            Thread.sleep(500L);
            continue;
        }
        catch(InterruptedException interruptedexception) { }
        break;
    } while(true);
}

synchronized void selfOrganize()
{
    for(int i = 0; i < nedges; i++)
    {
        NetEdge netedge = edges[i];
        double d = vertice[netedge.to].x - vertice[netedge.from].x;
        double d2 = vertice[netedge.to].y - vertice[netedge.from].y;
        double d4 = Math.sqrt(d * d + d2 * d2);
        double d6 = (double)linkLen - d4;
        double d8 = d6 * d;
        double d10 = d6 * d2;
        vertice[netedge.to].w += d8;
        vertice[netedge.to].h += d10;
        vertice[netedge.from].w += -d8;
        vertice[netedge.from].h += -d10;
    }

    for(int j = 0; j < nvertice; j++)
    {
        NetVertex netvertex = vertice[j];
        double d1 = 0.0D;
        double d3 = 0.0D;
        for(int l = 0; l < nvertice; l++)
        {
            if(j == l)
                continue;
            NetVertex netvertex2 = vertice[l];
            double d7 = netvertex.x - netvertex2.x;
```

```
                double d9 = netvertex.y - netvertex2.y;
                double d11 = d7 * d7 + d9 * d9;
                if(d11 == 0.0D)
                {
                    d1 += Math.random();
                    d3 += Math.random();
                } else
                {
                    d1 += d7 / d11;
                    d3 += d9 / d11;
                }
            }

            double d5 = d1 * d1 + d3 * d3;
            if(d5 > 0.0D)
            {
                d5 = Math.sqrt(d5) / 2D;
                netvertex.w += d1 / d5;
                netvertex.h += d3 / d5;
            }
        }

        Dimension dimension = getSize();
        for(int k = 0; k < nvertice; k++)
        {
            NetVertex netvertex1 = vertice[k];
            if(!netvertex1.fixed)
            {
                netvertex1.x += Math.max(-5D, Math.min(5D, netvertex1.w));
                netvertex1.y += Math.max(-5D, Math.min(5D, netvertex1.h));
                if(netvertex1.x < 0.0D)
                    netvertex1.x = nodeDiam;
                else
                if(netvertex1.x > (double)dimension.width)
                    netvertex1.x = dimension.width - nodeDiam;
                if(netvertex1.y < 0.0D)
                    netvertex1.y = nodeDiam;
                else
                if(netvertex1.y > (double)dimension.height)
                    netvertex1.y = dimension.height - 2 * nodeDiam;
            }
            netvertex1.w /= 2D;
            netvertex1.h /= 2D;
        }
        repaint();
    }

    public void start()
    {
    }

    public void stop()
    {
        if(selel == 1)
            selforganizor = null;
    }

    public void destroy()
    {
        System.gc();
    }

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

    public void mousePressed(MouseEvent mouseevent)
    {
        double d = 1.7976931348623157E+308D;
        int i = mouseevent.getX();
        int j = mouseevent.getY();
        for(int k = 0; k < nvertice; k++)
        {
            NetVertex netvertex = vertice[k];
            double d1 = (netvertex.x - (double)i) * (netvertex.x - (double)i) + (netvertex.y - (double)j) * (netvertex.y -
(double)j);
            if(d1 < d)
            {
                pick = netvertex;
                d = d1;
            }
        }

        pickfixed = pick.fixed;
        pick.fixed = true;
        pick.x = i;
        pick.y = j;
        repaint();
        mouseevent.consume();
    }

    public void mouseReleased(MouseEvent mouseevent)
    {
        pick.x = mouseevent.getX();
        pick.y = mouseevent.getY();
        pick.fixed = pickfixed;
        pick = null;
        repaint();
        mouseevent.consume();
    }

    public void mouseDragged(MouseEvent mouseevent)
    {
        pick.x = mouseevent.getX();
        pick.y = mouseevent.getY();
        repaint();
        mouseevent.consume();
    }

    public void mouseClicked(MouseEvent mouseevent)
    {
    }

    public void mouseEntered(MouseEvent mouseevent)
    {
    }

    public void mouseExited(MouseEvent mouseevent)
    {
    }

    public void mouseMoved(MouseEvent mouseevent)
    {
    }

    public int sele;
    public int selew;
    public int selel;
    public int nodeDiam;
    public int linkWid;
    public int linkLen;
    public int colorn;
    public int colornn;
```

```
        public int colorl;
        public int sign;
        public int canvaswid;
        public int canvashei;
        public double ww[];
        int nvertice;
        int nedges;
        NetVertex vertice[];
        NetEdge edges[];
        NetVertex pick;
        boolean pickfixed;
        Image offscreen;
        Dimension offscreensize;
        Graphics offgraphics;
        Color ncolor;
        Color nncolor;
        Color lcolor;
        BasicStroke bs1;
        Thread selforganizor;
}
```

The following are Java codes of the class NetGraph:

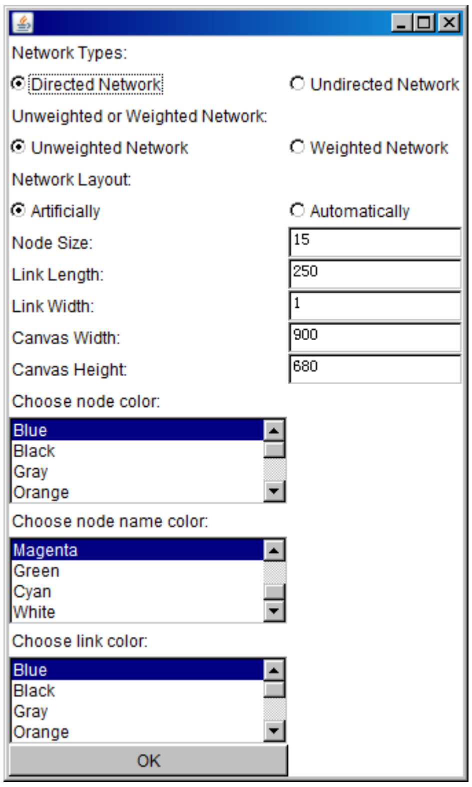
```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;

public class NetGraph extends Applet
        implements ActionListener
{

        public NetGraph(String as[], String as1[], int ai[], double ad[], int i, int j, int k,
                        int l, int i1, int j1, int k1, int l1, int i2, int j2,
                        int k2)
        {
                seles = i;
                selews = j;
                selels = k;
                nodeDiams = l;
                linkWids = i1;
                linkLens = j1;
        colorns = k1;
        colornns = l1;
        colorls = i2;
        canvaswids = j2;
        canvasheis = k2;
        e = as.length - 1;
        s = new String[5][e + 1];
        c = new String[0x186a0];
        t = new int[e + 1];
        ww = new double[e + 1];
        for(int l2 = 1; l2 <= e; l2++)
        {
            s[1][l2] = as[l2];
            s[2][l2] = as1[l2];
            t[l2] = ai[l2];
            ww[l2] = ad[l2];
        }

        v = 1;
        c[1] = s[1][1];
        for(int i3 = 1; i3 <= 2; i3++)
        {
            for(int j3 = 1; j3 <= e; j3++)
            {
                m = 0;
                for(int k3 = 1; k3 <= v; k3++)
                    if(!s[i3][j3].equals(c[k3]))
                        m++;

                if(m == v)
                {
                    v++;
                    c[v] = s[i3][j3];
                }
            }

        }

        begin();
    }

    public void begin()
    {
        save = new Button("Save");
        close = new Button("Close");
```

```
            setLayout(new BorderLayout());
            panel = new NetPanel(ww, seles, selews, selels, nodeDiams, linkWids, linkLens, colorns, colornns, colorls,
canvaswids, canvasheis);
            add("Center", panel);
            controlPanel = new Panel();
            add("South", controlPanel);
            controlPanel.add(save);
            controlPanel.add(close);
            save.addActionListener(this);
            close.addActionListener(this);
            Dimension dimension = getSize();
            resize(dimension.width - 15, dimension.height - 15);
            for(int i = 1; i <= e; i++)
                panel.addEdge(s[1][i], s[2][i], t[i]);

            setLocation(20, 20);
            validate();
            show();
        }

        public void init()
        {
        }

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

        public void destroy()
        {
            remove(panel);
        }

        public void actionPerformed(ActionEvent actionevent)
        {
            Object obj = actionevent.getSource();
            if(obj == save)
            {
                try
                {
                    Frame frame = new Frame();
                    FileDialog filedialog = new FileDialog(frame, "Save network as (*.png, *.jpg, etc)", 1);
                    validate();
                    filedialog.show();
                String s1 = filedialog.getDirectory() + filedialog.getFile();
                BufferedImage bufferedimage = toBufferedImage(panel.offscreen);
                ImageIO.write(bufferedimage, "png", new File(s1));
            }
            catch(Exception exception) { }
            hide();
            getParent().hide();
            System.exit(0);
        }
        if(obj == close)
        {
            hide();
            getParent().hide();
            System.exit(0);
        }
    }

    public BufferedImage toBufferedImage(Image image)
    {
        if(image instanceof BufferedImage)
        {
            return (BufferedImage)image;
        } else
        {
            BufferedImage bufferedimage = new BufferedImage(image.getWidth(null), image.getHeight(null), 1);
            Graphics2D graphics2d = bufferedimage.createGraphics();
            graphics2d.drawImage(image, 0, 0, null);
            graphics2d.dispose();
            return bufferedimage;
        }
    }

    public String getAppletInfo()
    {
        return "Developer: WenJun Zhang";
    }

    public int seles;
    public int selews;
    public int selels;
    public int nodeDiams;
    public int linkWids;
    public int linkLens;
    public int colorns;
    public int colornns;
```

```
        public int colorls;
        public int canvaswids;
        public int canvasheis;
        Button close;
        Button save;
        NetPanel panel;
        Panel controlPanel;
        String s[][];
        String c[];
        int e;
        int v;
        int m;
        public int t[];
        public double ww[];
    }
```

Pack all classes into a JAR package and compile the package into the executable Java software, netGen 3.0.exe.

Before using the software, the Java Runtime Environment (JRE) should be installed. First, download the Java Runtime Environment software from the following website: https://www.java.com/en/. After downloading, click Install Java Runtime Environment to install it. Alternatively, search for "Java Runtime Environment" on the Internet to find the appropriate platform version for downloading. Then, double-click netGen 3.0.exe to run the software (Fig. 1 and 2).



Network Types:
[✓ Directed Network] Undirected Network
Unweighted or Weighted Network:
[✓ Unweighted Network] Weighted Network
Network Layout:
[✓ Artificially] Automatically

Node Size:	15
Link Length:	250
Link Width:	1
Canvas Width:	900
Canvas Height:	680

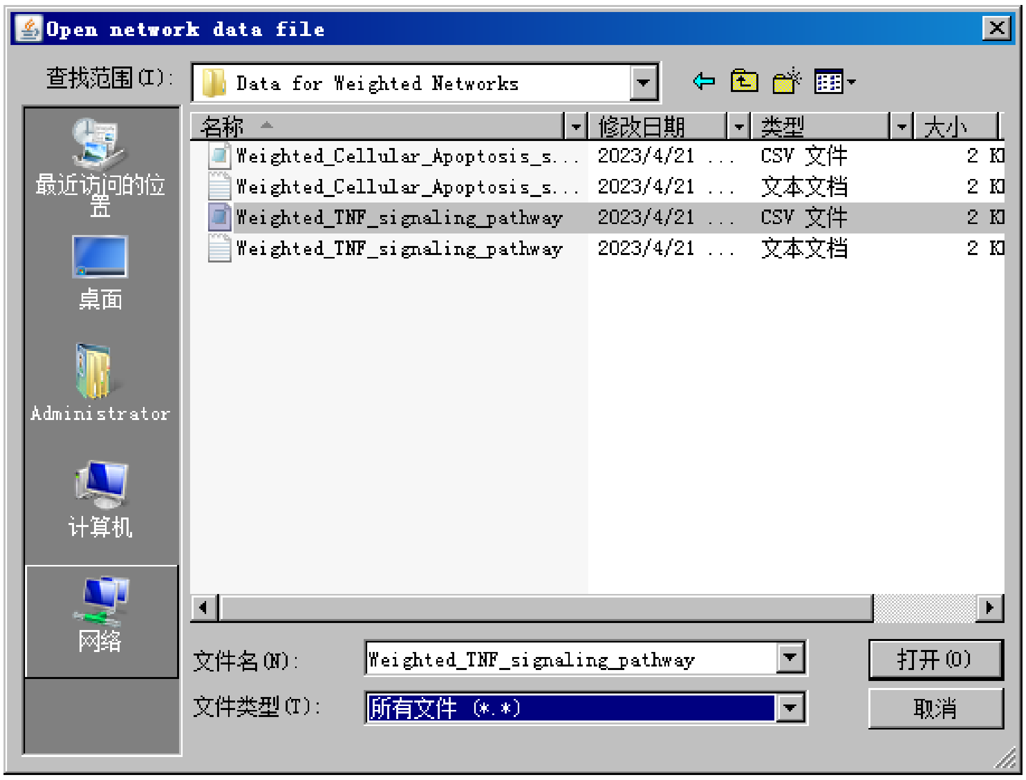
Choose node color:
[✓ Blue] Black Gray Orange
Choose node name color:
[✓ Magenta] Green Cyan White
Choose link color:
[✓ Blue] Black Gray Orange

Fig. 1 The windows interface of netGen 3.0.

2.2 Data

The data for network visualization is stored in a comma delimited text file (in txt or csv formats). Known totally *m* links in the network, thus there should be *m* rows in the data file. There are three items (for unweighted network) or four items (for weighted network) in each row: the first item (node) and the second item (node) represent a link between two nodes and the third item is an integer value *x*. For an undirected network, *x*=1 for each row. For a directed network, *x*=1 if it is a forward link (from the first item (source node) to the second item (target node)); *x*=-1 is for backward link (from the second item (source node) to the first item (target node)). Additionally, there is the fourth item, a weight of the link, for weighted network. If users want to generate an undirected network, *x*=1 and *x*=-1 are equivalent. The three (for unweighted network) or four (for weighted network) items are separated by the comma "," (so by default, it means that any comma is not allowed in each item) (Fig. 3).

The software and demo data files are included in the package: http://www.iaees.org/publications/journals/selforganizology/articles/2024-11(1-2)/e-suppl/Zhang-Supplementary-Material.rar


Open network data file

查找范围(I): Data for Weighted Networks

名称	修改日期	类型	大小
Weighted_Cellular_Apoptosis_s...	2023/4/21 ...	CSV 文件	2 K
Weighted_Cellular_Apoptosis_s...	2023/4/21 ...	文本文档	2 K
Weighted_TNF_signaling_pathway	2023/4/21 ...	CSV 文件	2 K
Weighted_TNF_signaling_pathway	2023/4/21 ...	文本文档	2 K

文件名(N): Weighted_TNF_signaling_pathway

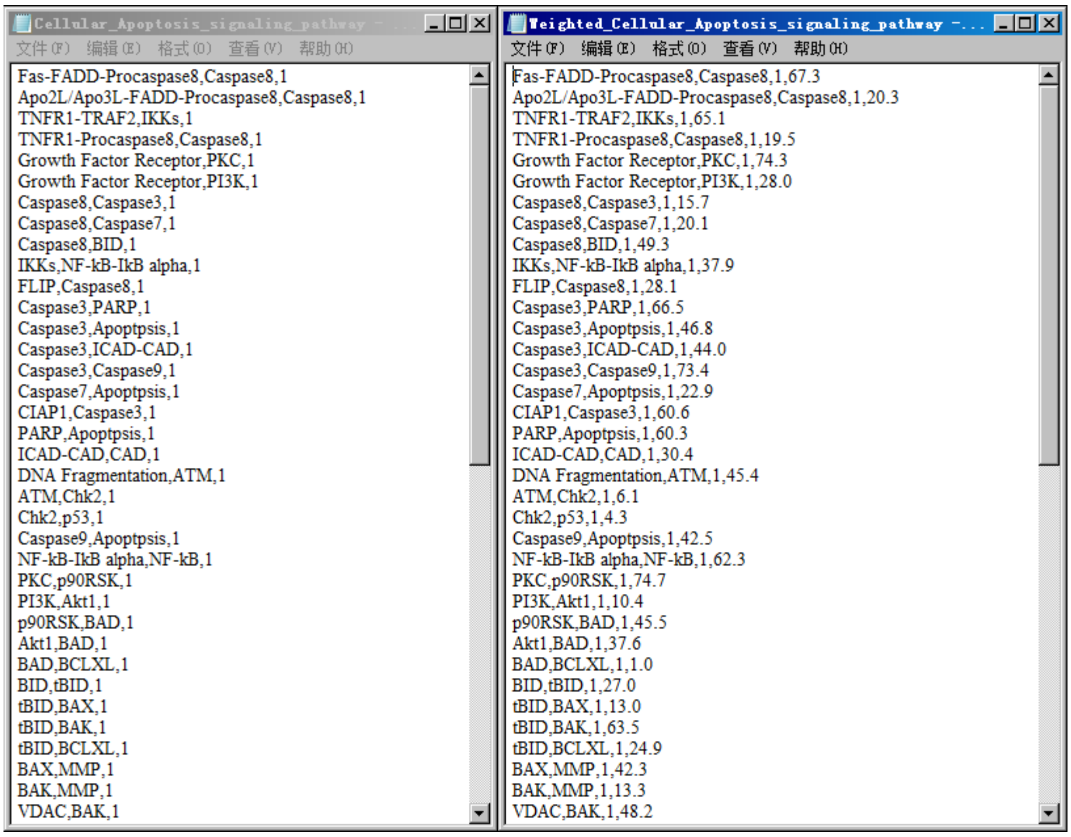
文件类型(T): 所有文件 (*.*)

Fig. 2 The file open dialog of netGen 3.0.

3 Example Demonstration

The network data for TNF and cellular apoptosis signaling pathways were from Huang and Zhang (2012), Li and Zhang (2013), and Zhang and Huang (2023).

Run the software netGen 3.0 and open the data file, an interactive network can be generated and visualized. We may drag the nodes in the network to exhibit a better layout of the network (Fig. 4 and 5). If we choose network layout automatically, the software will automatically adjust layout but we can still drag any node to assist the automatic process.


Cellular_Apoptosis_signaling_pathway - ...
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Weighted_Cellular_Apoptosis_signaling_pathway - ...
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
Fas-FADD-Procaspase8,Caspase8,1
Apo2L/Apo3L-FADD-Procaspase8,Caspase8,1
TNFR1-TRAF2,IKKs,1
TNFR1-Procaspase8,Caspase8,1
Growth Factor Receptor,PKC,1
Growth Factor Receptor,PI3K,1
Caspase8,Caspase3,1
Caspase8,Caspase7,1
Caspase8,BID,1
IKKs,NF-kB-IkB alpha,1
FLIP,Caspase8,1
Caspase3,PARP,1
Caspase3,Apoptpsis,1
Caspase3,ICAD-CAD,1
Caspase3,Caspase9,1
Caspase7,Apoptpsis,1
CIAP1,Caspase3,1
PARP,Apoptpsis,1
ICAD-CAD,CAD,1
DNA Fragmentation,ATM,1
ATM,Chk2,1
Chk2,p53,1
Caspase9,Apoptpsis,1
NF-kB-IkB alpha,NF-kB,1
PKC,p90RSK,1
PI3K,Akt1,1
p90RSK,BAD,1
Akt1,BAD,1
BAD,BCLXL,1
BID,tBID,1
tBID,BAX,1
tBID,BAK,1
tBID,BCLXL,1
BAX,MMP,1
BAK,MMP,1
VDAC,BAK,1
```

```
Fas-FADD-Procaspase8,Caspase8,1,67.3
Apo2L/Apo3L-FADD-Procaspase8,Caspase8,1,20.3
TNFR1-TRAF2,IKKs,1,65.1
TNFR1-Procaspase8,Caspase8,1,19.5
Growth Factor Receptor,PKC,1,74.3
Growth Factor Receptor,PI3K,1,28.0
Caspase8,Caspase3,1,15.7
Caspase8,Caspase7,1,20.1
Caspase8,BID,1,49.3
IKKs,NF-kB-IkB alpha,1,37.9
FLIP,Caspase8,1,28.1
Caspase3,PARP,1,66.5
Caspase3,Apoptpsis,1,46.8
Caspase3,ICAD-CAD,1,44.0
Caspase3,Caspase9,1,73.4
Caspase7,Apoptpsis,1,22.9
CIAP1,Caspase3,1,60.6
PARP,Apoptpsis,1,60.3
ICAD-CAD,CAD,1,30.4
DNA Fragmentation,ATM,1,45.4
ATM,Chk2,1,6.1
Chk2,p53,1,4.3
Caspase9,Apoptpsis,1,42.5
NF-kB-IkB alpha,NF-kB,1,62.3
PKC,p90RSK,1,74.7
PI3K,Akt1,1,10.4
p90RSK,BAD,1,45.5
Akt1,BAD,1,37.6
BAD,BCLXL,1,1.0
BID,tBID,1,27.0
tBID,BAX,1,13.0
tBID,BAK,1,63.5
tBID,BCLXL,1,24.9
BAX,MMP,1,42.3
BAK,MMP,1,13.3
VDAC,BAK,1,48.2
```

Fig. 3 The data files for unweighted network (left) and weighted network (right) respectively, cellular apoptosis signaling pathway.

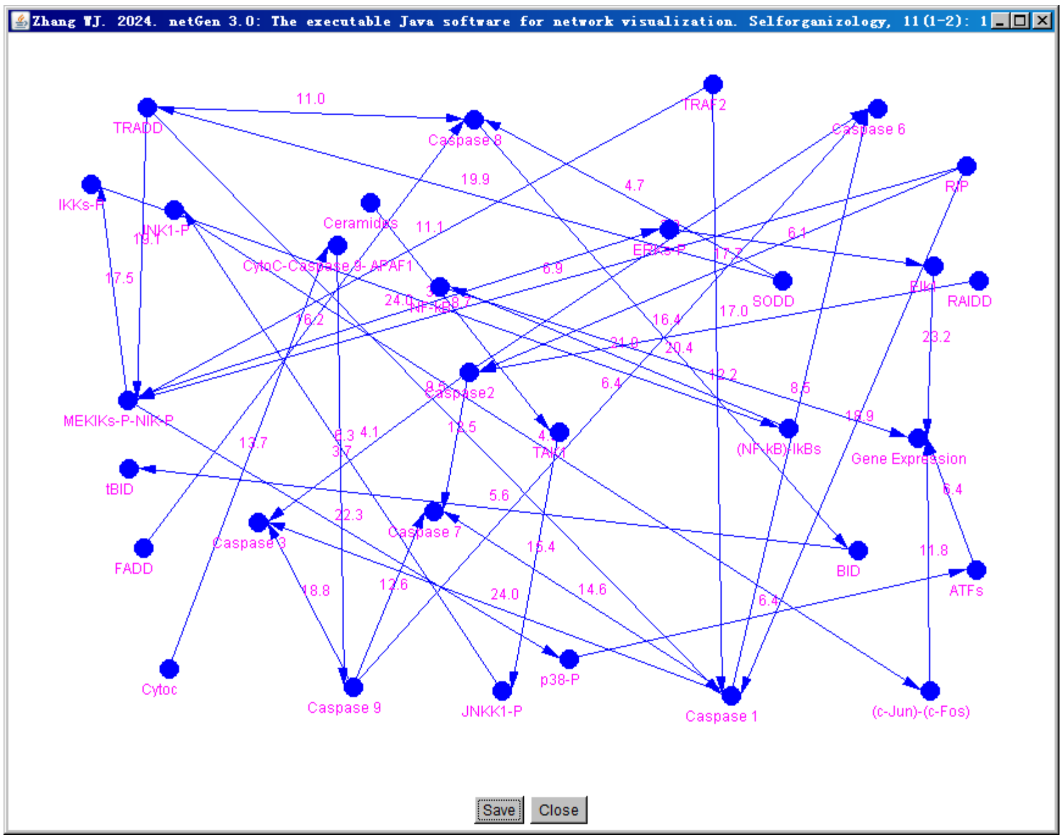


Fig. 4 The weighted network of TNF signaling pathway, generated by netGen 3.0. Weights of links are labeled on links.

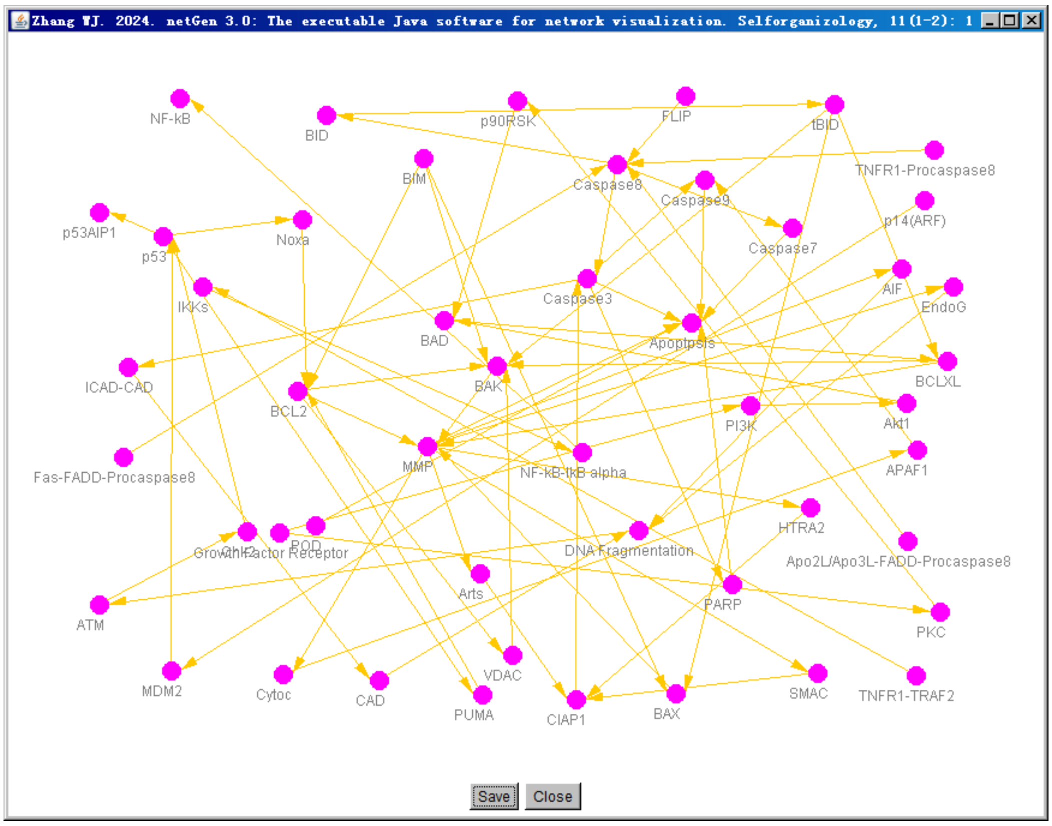


Fig. 5 The unweighted network of cellular apoptosis signaling pathway, generated by netGen 3.0.

4 Discussion

Compared to the previous versions (Zhang, 2012a: netGen version 1.0; Zhang, 2024c: netGen version 2.0), There are substantial improvements in netGen 3.0. In addition to the visualization of both unweighted and weighted (and both undirected and directed) networks, node size / color, link width / length / color, node name color, canvas size, etc., can be specified by the user. Moreover, network layout can be conducted in artificial or automatic manners.

References

Huang JQ, Zhang WJ. 2012. Analysis on degree distribution of tumor signaling networks. Network Biology, 2(3): 95-109

Jiang LQ, Zhang WJ. 2015. Determination of keystone species in CSM food web: A topological analysis of network structure. Network Biology, 5(1): 13-33

Jiang LQ, Zhang WJ, Li X. 2015. Some topological properties of arthropod food webs in paddy fields of South China. Network Biology, 5(3): 95-112

Li JR, Zhang WJ. 2013. Identification of crucial metabolites/reactions in tumor signaling networks. Network Biology, 3(4): 121-132

Narad P, Upadhyaya KC, Som A. 2017. Reconstruction, visualization and explorative analysis of human pluripotency network. Network Biology, 7(3): 57-75

Qi YH, Liu GH, Zhang WJ. 2018. Analysis of word occurrence frequency and word association in English text file: A big data analytics method. Network Biology, 8(3): 126-136

Xin SH, Zhang WJ. 2020. Construction and analysis of the protein-protein interaction network for the olfactory system of the silkworm *Bombyx mori*. Archives of Insect Biochemistry and Physiology, 105(3): e21737

Xin SH, Zhang WJ. 2021. Construction and analysis of the protein-protein interaction network for the detoxification enzymes of the silkworm, *Bombyx mori*. Archives of Insect Biochemistry and Physiology, 108(4): e21850

Yang S, Zhang WJ. 2022. Systematic analysis of olfactory protein-protein interactions network of fruitfly, *Drosophila melanogaster*. Archives of Insect Biochemistry and Physiology, 110(2): e21882

Zhang GL, Zhang WJ. 2019. Protein-protein interaction network analysis of insecticide resistance molecular mechanism in *Drosophila melanogaster*. Archives of Insect Biochemistry and Physiology, 100(1): e21523

Zhang WJ. 2007. Computer inference of network of ecological interactions from sampling data. Environmental Monitoring and Assessment, 124: 253–261

Zhang WJ. 2011. Constructing ecological interaction networks by correlation analysis: hints from community sampling. Network Biology, 1(2): 81-98

Zhang WJ. 2012a. A Java software for drawing graphs. Network Biology, 2(1): 38-44

Zhang WJ. 2012b. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore

Zhang WJ. 2012c. How to construct the statistic network? An association network of herbaceous plants constructed from field sampling. Network Biology, 2(2): 57-68

Zhang WJ. 2016a. A random network based, node attraction facilitated network evolution method. Selforganizology, 3(1): 1-9

Zhang WJ. 2016b. Selforganizology: The Science of Self-Organization. World Scientific, Singapore

Zhang WJ. 2018. Fundamentals of Network Biology. World Scientific Europe, London, UK

Zhang WJ. 2021a. A web tool for generating user-interface interactive networks. Network Biology, 11(4): 247-262

Zhang WJ. 2021b. Construction and analysis of the word network based on the Random Reading Frame (RRF) method. Network Biology, 11(3): 154-193

Zhang WJ. 2024a. A Matlab software for visualizing user-interface interactive networks. Network Biology, 14(1): 13-19

Zhang WJ. 2024b. A standalone executable software for network visualization. Network Pharmacology, 9(1-2): 1-10

Zhang WJ. 2024c. An executable Java software for visualizing networks. Network Biology, 14(1): 1-12

Zhang WJ, Huang XB. 2023. A further study on the topological structure of tumor signaling pathways. Network Pharmacology, 8(1-2): 1-26

Zhang WJ, Li X. 2016. Generate networks with power-law and exponential-law distributed degrees: with applications in link prediction of tumor pathways. Network Pharmacology, 1(1): 15-35

Zhang WJ, Wang R, Zhang DL, et al. 2014. Interspecific associations of weed species around rice fields in Pearl River Delta, China: A regional survey. Selforganizology, 1(3-4): 143-205