

成绩 \_\_\_\_\_

江南大学

# 《复杂网络与大数据分析》课程 设计报告

班 级 \_\_\_\_\_ 信计 1901

学 号 \_\_\_\_\_ 1131190111

姓 名 \_\_\_\_\_ 唐川淇

**S1：构建最近邻网络模型、随机网络模型、WS 小世界网络模型、无标度网络模型，并分析度分布、聚类系数、平均路径长度等拓扑性质。（15 分）**

由于在四个网络中都需要计算度分布、聚类系数等内容，首先创建函数。

计算度分布和度的概率分布

```
function [DeD, aver_DeD]=tcq_DD(A)
NN=size(A,2);
DeD=zeros(1, NN);
for i=1:NN
    DeD(i)=sum(A(i,:)) ;
end
aver_DeD=mean(DeD);
if sum(DeD)==0
    disp('该图只含有孤立点');
    return;
else figure;
    bar([1 :NN],DeD,'r');
    xlabel('节点编号');
    ylabel('度');
    title('度分布图');
end
figure;
M=max(DeD);
for i=1:M+1;
    N_DeD(i)=length(find(DeD==i-1));
end
P_DeD=zeros(1,M+1);
P_DeD(:)=N_DeD(:)./ sum(N_DeD);
bar([0:M],P_DeD, 'b') ;
xlabel('度');
ylabel('概率');
title('度的概率分布图');
```

平均路径长度：

计算平均路径长度

```
function [D, aver_D]=tcq_Aver_Path_Length(A)
N=size(A, 2);
D=A;
D(find(D==0))=inf;%将邻接矩阵变为邻接距离矩阵，两点无边相连时赋值为 inf，自身到自身的距离为 0;
for i=1:N
    D(i,i)=0;
end
for k=1:N
```

```

    for i=1:N
        for j=1:N
            if D(i,j)>D(i,k)+D(k,j)
                D(i,j)=D(i,k)+D(k,j);
            end
        end
    end
end
end
aver_D=sum(sum(D))/(N*(N-1))%平均路径长度
if aver_D==inf
    disp('该网络图不是连通图');
end
end

```

计算聚类系数:

计算聚类系数

```

function [c, aver_c]=tcq_clustering_Coefficient(A)
N=size(A, 2);
C=zeros(1,N);
for i=1:N
    aa=find(A(i,:)==1);%寻找子图的邻居节点
    if isempty(aa)
        disp(['节点', int2str(i), '为孤立节点']);
        c(i)=0;
    else
        m=length(aa); if m==1
            disp(['节点', int2str(i), '只有一个邻居节点']);
            c(i)=0;
        else
            B=A(aa, aa); %抽取子图的邻接矩阵
            c(i)=length(find(B==1))/(m*(m-1)); end
        end
    end
end
aver_c=mean(c);

```

最近邻网络模型

代码:

设置节点数为 25，与邻近 8 个节点相连。

最近邻网络

```

clear;
clc;
N=25;

```

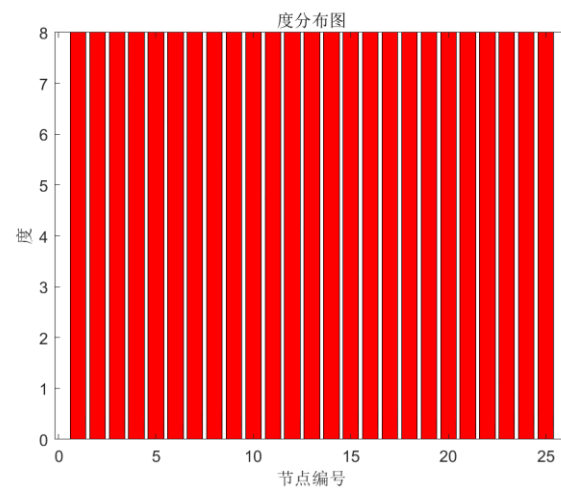
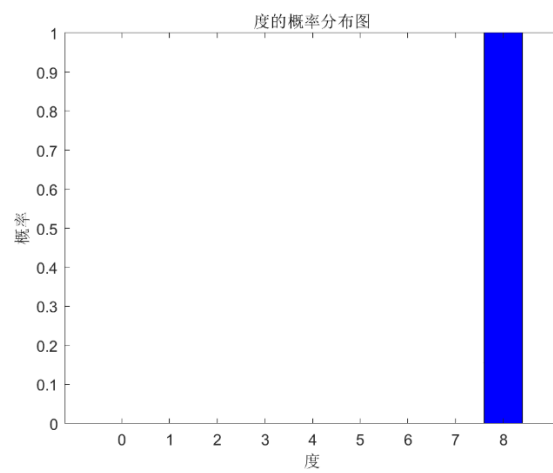
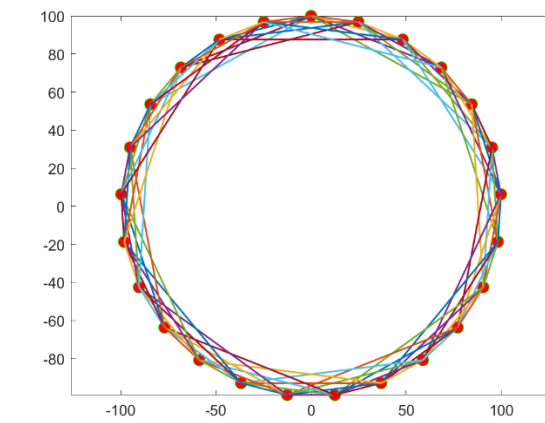
```

K=8;
if K>floor(N-1)|mod(K,2)~=0;
    disp('输入错误');
    return;
end
ang1e=0:2*pi/N :2*pi-2*pi/N ;
x=100*sin(ang1e);
y=100*cos(ang1e);
plot(x, y, 'ro', 'MarkerEdgeColor', 'g','MarkerFaceColor', 'r', 'markersize',8);
hold on;
A=zeros(N);
for i=1: N
    for j=i+1:i+K/2
        jj=j ;
        if j>N
            jj=mod(j,N);
        end
        A(i,jj)=1;A(jj,i)=1;
    end
end
for i=1: N
    for j=i+1: N
        if A(i,j)~=0
            plot([x(i), x(j)], [y(i),y(j)], 'linewidth',1.2);
            hold on
        end
    end
end
axis equal;
hold off



tcq_DD(A);
[c, aver_c]=tcq_clustering_Coefficient(A);
[D, aver_D]=tcq_Aver_Path_Length(A);

```

结果:



聚类系数和平局路径长度如下：

	aver_c	0.6429
	aver_D	2.5617

网络密度为：

0.3333

## 随机网络模型

设置节点数为 50，连接概率为 0.13

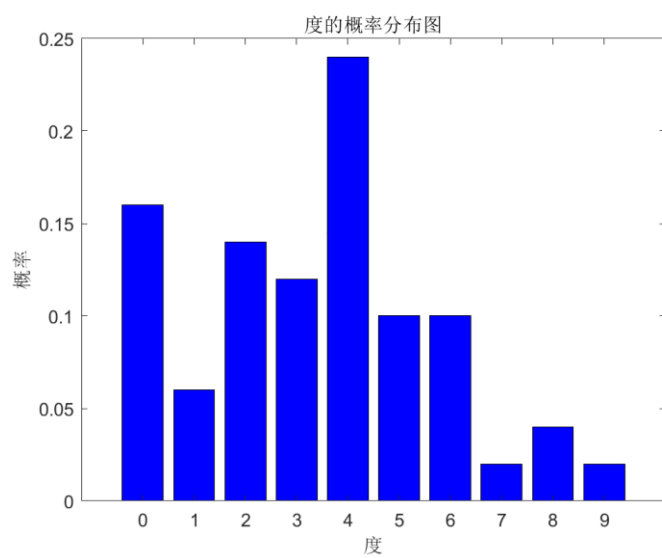
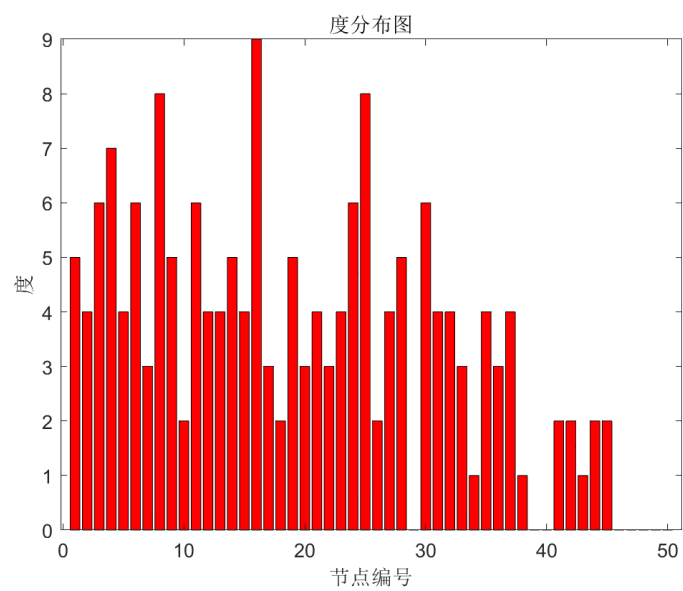
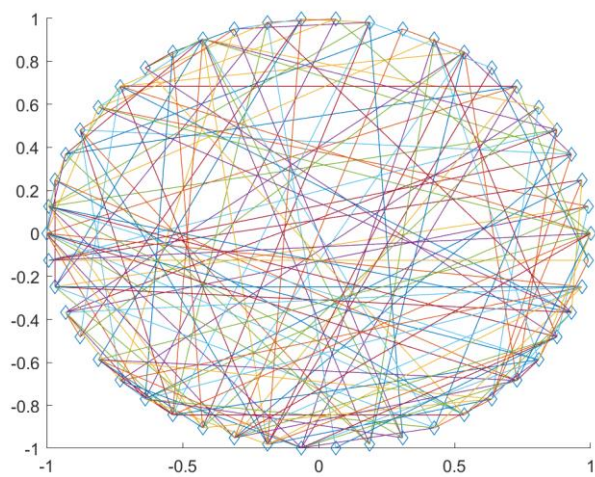
代码：

### 随机网络

```
clear
clc
N=50
p=0.13
position=zeros(N, 2);
A = zeros(N, N);%创建邻接矩阵
for m=1 :N
    %给每个点安排位置，围成一个圆
    position(m,1)=cos(m/N*2*pi);
    position(m,2)=sin(m/N*2*pi);end
figure('name', 'ER 随机图');
hold on;
plot (position(: , 1) , position(: , 2), 'd' )
for m=1:N
    for n=m+1:N
        if (rand(1,1)<p)%以 0.1 的概率生成边
            A(m, n)=1;%这里两句给邻接表赋值 A(n, m)=1;
        end
    end
end
for m = 1:N
    for n = m+1:N
        if (A(m, n)==1)%如果有边就画出来
            plot(position([m,n],1) , position([m, n],2)) ;
        end
    end
end
hold off;

tcq_DD(A);
[c, aver_C]=tcq_clustering_Coefficient(A);
[D, aver_D]=tcq_Aver_Path_Length(A);
```

结果：



其中聚类系数如下：

$$\text{aver\_C} = 0.0654$$

由于该图是不连通的，所以平均路径长度为无穷大。

网络密度：

$$0.0665$$

### WS 小世界网络模型

设置节点数为 100，K 为 8，重连概率为 0.1

代码：

#### WS 小世界网络

```
clear;
clc;
% WS 小世界网络
N=100;
K=8;
if K>floor(N-1)|mod(K,2)~=0;
    disp('输入错误');
    return;
end
ang1e=0:2*pi/N :2*pi-2*pi/N ;
x=100*sin(ang1e);
y=100*cos(ang1e);
plot(x, y, 'ro', 'MarkerEdgeColor', 'g','MarkerFaceColor', 'r', 'markersize',8);%根据坐标画节点
hold on;
A=zeros(N);%N 行 N 列矩阵，作为邻接矩阵用
for i=1: N
    for j=i+1:i+K/2
        jj=j ;
        if j>N
            jj=mod(j,N);
        end
        A(i,jj)=1;A(jj,i)=1;
```

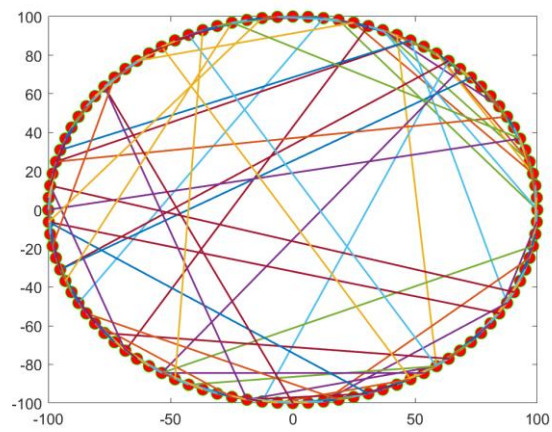
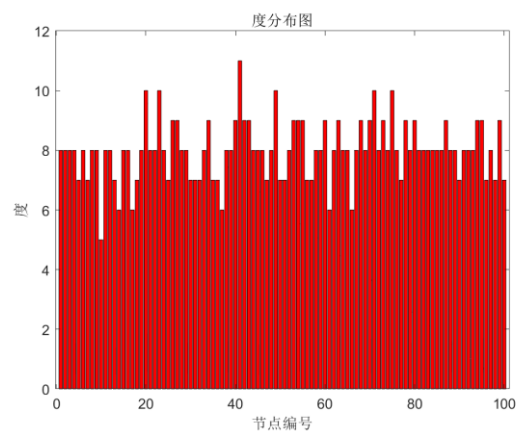
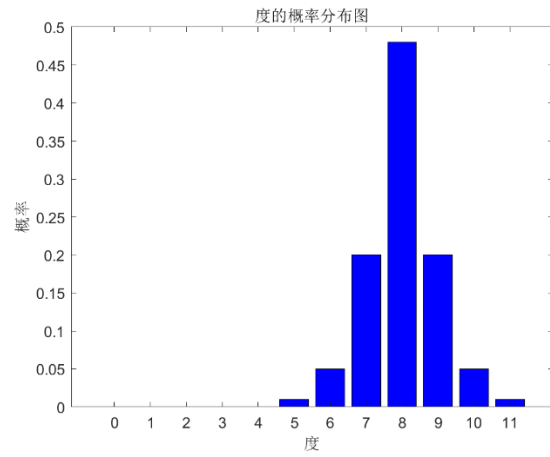


```

        end
    end
    p=0.1;
    for i=1:N
        for j=i+1:i+K/2
            jj=j ;
            if j>N
                jj=mod(j, N);end
            p1=rand(1,1);
            if p1<p
                A(i,jj)=0;
                A(jj,i)=0;
                A(i,i)=inf ;
                a=find(A(i,:)==0);
                rand_data=randi ([1,length(a)],1,1);
                jjj=a (rand_data);
                A(i,jjj)=1;A(jjj,i)=1;A(i,i)=0;
            end
        end
    end
    end
    for i=1:N
        for j=i+1:N
            if A(i,j)~=0
                plot([x(i),x(j)],[y(i),y(j)], 'linewidth',1.2);
                hold on;
            end
        end
    end
    end
    tcq_DD(A);
    [c, aver_C]=tcq_clustering_Coefficient(A);
    [D, aver_D]=tcq_Aver_Path_Length(A);

```

结果:



聚类系数和平均路径长度:

■ aver\_C 0.4554  
■ aver D 3.7872

网络密度:

0.0808

无标度网络模型

代码:

#### BA 无标度网络

```
m_original=2;%未增长前网络节点个数
m_add=2;%每次添加一个点，增加的边数为 m_add
m_after_growth=20;%增长后的网络规模
pp=1;%pp 用来表示初始网络的类型
x=100*rand(1,m_original);%初始网络的横坐标
y=100*rand(1,m_original);%初始网络的纵坐标
A=zeros(m_original);%初始化邻接矩阵 A，A 为全 0 矩阵
switch pp %选择 pp，确定 A 的初始状态
    case 1 %节点孤立
        A=zeros(m_original);
    case 2 %全连接
        A=ones(m_original);
    case 3 %随机图
        for i=1:m_original
            for j=i+1:m_original%操作上半角矩阵
                p=rand(1,1);
                if p>0.5 %以 p 的概率生成边
                    A(i,j)=1;
                    A(j,i)=1;
                end
            end
        end
    end
end
for k=m_original+1:m_after_growth%共生成 m_after_growth 减掉(m_original+1)条边
    M=k-1;%当前要生成第 k 个节点，此时的网络规模 M 为 k-1
    p=zeros(1,M);%初始化每个节点的连接概率为全 0
    x_now=100*rand(1,1);%随机生成第 k 个节点的坐标
    y_now=100*rand(1,1);
    x(k)=x_now;
    y(k)=y_now;
    for i=1:M
        p(i)=(length(find(A(i,:)==1))+1)/(length(find(A==1))+M);
    end %修正孤立节点连接概率为 0，计算一个新节点与一个已经存在的节点 i 相连接
    的概率
    p
    pp=cumsum(p);%叠加概率，计算每行的累加值
```

```

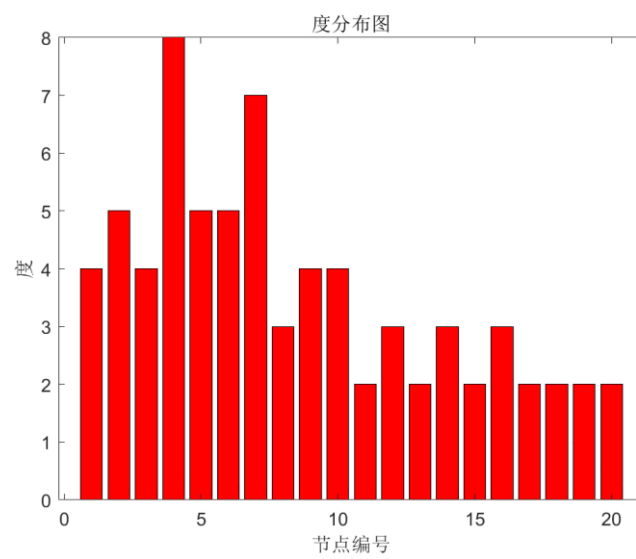
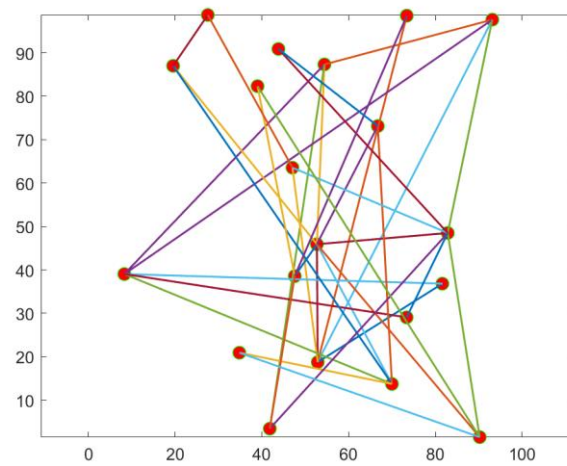
pp
visit=zeros(1,M);%初始化访问数组为全 0 数组
for i=1:m_add %生成 m_add 条边
    random_data=rand(1,1);
    random_data
    aa=find(pp>=random_data);jj=aa(1);%通过叠加概率和随机数的方法得到随机
连边
    A(k,jj)=1;
    A(jj,k)=1;
    visit(jj)=1;%标记访问
    visit          %显示访问情况
    degree=zeros(1,M);%度数组
    total_degree=0;
    for ii=1:M
        if visit(ii)==1
            p(ii)=0;
            degree(ii)=0;
        else
            degree(ii)=length(A(i,:)==1)+1;
        end
        total_degree=total_degree+degree(ii);
    end
    for iii=1:M
        p(iii)=degree(iii)/total_degree;%新一轮的 p 数组计算
    end
    p
    pp=cumsum(p);
    pp
end
end
A
plot(x,y,'ro','MarkerEdgeColor','g','MarkerFaceColor','r','MarkerSize',8);
hold on;
for i=1:m_after_growth
    for j=i+1:m_after_growth
        if A(i,j)~=0
            plot([x(i),x(j)],[y(i),y(j)],'linewidth',1.2);
        end
    end
end
end
axis equal;
hold off;

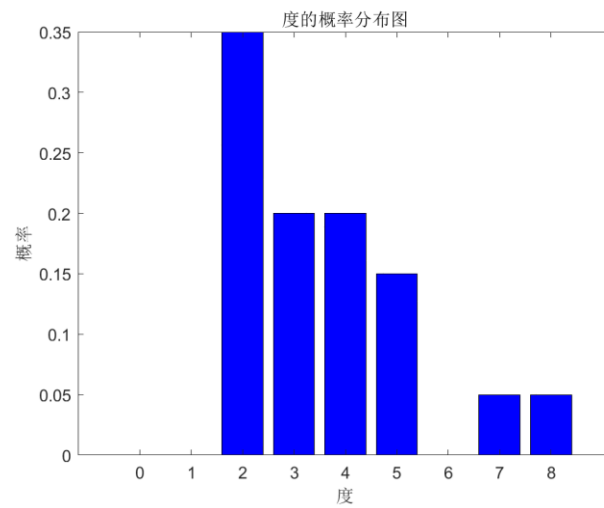
tcq_DD(A);

```

```
[c, aver_C]=tcq_clustering_Coefficient(A);  
[D, aver_D]=tcq_Aver_Path_Length(A);
```

结果:





聚类系数及平均路径长度:

aver_C	0.3131
aver_D	2.2421

网络密度:

```
ans =  
  
0.0808
```

S2. 选择其中一种复杂网络模型，对网络进行社团检测，输出模块度并显示划分社团后的网络图（实现两种算法，所用算法不限）。（15 分）

选择 BA 无标度网络

代码：

```

社团检测
function [M,Q]=community_louvain(W,gamma,M0,B)
    % convert to double format
    n=single(length(W)); % get number of nodes
    s=single(sum(sum(W))); % get sum of edges

    if ~exist('B','var') || isempty(B)
        type_B = 'modularity';
    elseif ischar(B)
        type_B = B;
    else
        type_B = 0;
        if exist('gamma','var') && ~isempty(gamma)
            warning('Value of gamma is ignored in generalized mode.')
        end
    end
    if ~exist('gamma','var') || isempty(gamma)
        gamma = 1;
    end

    if strcmp(type_B,'negative_sym') || strcmp(type_B,'negative_asym')
        W0 = W.*(W>0); %positive weights matrix
        s0 = sum(sum(W0)); %weight of positive links
        B0 = W0-gamma*(sum(W0,2)*sum(W0,1))/s0; %positive modularity

        W1 =-W.*(W<0); %negative weights matrix
        s1 = sum(sum(W1)); %weight of negative links
        if s1 %negative modularity
            B1 = W1-gamma*(sum(W1,2)*sum(W1,1))/s1;
        else
            B1 = 0;
        end
    elseif min(min(W))<-1e-10
        err_string = [
            'The input connection matrix contains negative weights.\nSpecify ' ...
            '"negative_sym" or "negative_asym" objective-function types.'];
        error(sprintf(err_string)) % #ok<SPERR>
    end
end

```

```

end
if strcmp(type_B,'potts') && any(any(W ~= logical(W)))
    error('Potts-model Hamiltonian requires a binary W.')
end

if type_B
    switch type_B
        case 'modularity';      B = single((W-gamma*(sum(W,2)*sum(W,1))/s)/s);
        case 'potts';          B = W-gamma*(~W);
        case 'negative_sym';    B = B0/(s0+s1) - B1/(s0+s1);
        case 'negative_asym';   B = B0/s0      - B1/(s0+s1);
        otherwise;              error('Unknown objective function.');
```

end

```

else
    % custom objective function matrix as input
    B = double(B);
    if ~isequal(size(W),size(B))
        error('W and B must have the same size.')
    end
end

end
if ~exist('M0','var') || isempty(M0)
    M0=1:n;
elseif numel(M0)~=n
    error('M0 must contain n elements.')
end

[~,~,Mb] = unique(M0);
Mb = single(Mb);
M = Mb;

B = (B+B.)/2; % symmetrize modularity
matrix
Hnm=zeros(n,n,'single'); % node-to-
module degree
for m=1:max(Mb) % loop over modules
    Hnm(:,m)=sum(B(:,Mb==m),2);
end

Q0 = -inf;
Q = sum(B(bsxfun(@eq,M0,M0.))); % compute modularity
first_iteration = true;
while Q-Q0>1e-10
    flag = true; % flag for within-hierarchy
search
    while flag

```



```

        flag = false;
        for u=randperm(n)                                % loop over all nodes in
random order
            ma = Mb(u);                                  % current module of u
            dQ = Hnm(u,:) - Hnm(u,ma) + B(u,u);
            dQ(ma) = 0;                                  % (line above) algorithm
condition
            [max_dQ,mb] = max(dQ);                        % maximal increase in
modularity and corresponding module
            if max_dQ>1e-10                               % if maximal increase is
positive
                flag = true;
                Mb(u) = mb;                               % reassign module

                Hnm(:,mb) = Hnm(:,mb)+B(:,u);             % change node-to-
module strengths
                Hnm(:,ma) = Hnm(:,ma)-B(:,u);
            end
        end
    end
    [~,~,Mb] = unique(Mb);                                % new module
assignments
    Mb = single(Mb);
    M0 = M;
    if first_iteration
        M=Mb;
        first_iteration=false;
    else
        for u=1:n                                         % loop through initial
module assignments
            M(M0==u)=Mb(u);                               % assign new modules
        end
    end

    n=max(Mb);                                             % new number of
modules
    B1=zeros(n,'single');                                  % new weighted
matrix
    for u=1:n
        for v=u:n
            bm=sum(sum(B(Mb==u,Mb==v)));                % pool weights of
nodes in same module
            B1(u,v)=bm;

```

```

        B1(v,u)=bm;
    end
end
B=B1;

Mb=1:n; % initial module
assignments
    Hnm=B; % node-to-module strength

    Q0=Q;
    Q=trace(B); % compute modularity
End

mat=A;
n=length(mat);
m=sum(sum(mat))/2;

vs=[1:n]; %vs are the vertices which need to be recalculate
for i1=1:m
    smat=zeros(n);
    %calculate the shortest-path betweenness
    for i2=1:length(vs)
        %BFS
        ln=0; %number of leaves
        D=zeros(1,length(vs))-1;
        D(vs(i2))=0;
        W(vs(i2))=1;
        pah=1;pat=1; %two pointers indicate the index of array
        array(pat)=vs(i2);
        pat=pat+1;
        %calculate the distances and weight of vertices
        while(pah<pat)
            tp=array(pah);
            leaf=1; %assume tp is a leaf
            for i3=1:length(vs)
                if(mat(vs(tp),vs(i3)))
                    if(D(vs(i3))==-1)
                        leaf=0; %tp isn't a leaf
                        array(pat)=vs(i3);
                        pat=pat+1;
                        D(vs(i3))=D(vs(tp))+1;
                        W(vs(i3))=W(vs(tp));
                    elseif(D(vs(i3))==D(vs(tp))+1)

```

```

        W(vs(i3))=W(vs(i3))+W(vs(tp));
    end
end
end
if(leaf)
    ln=ln+1;
    leaves(ln)=tp;
end
pah=pah+1;
end
%calculate the betweenness of the edges
tvs=vs;
for i3=1:length(tvs) %sort vs by des of D
    for i4=i3+1:length(tvs)
        if(D(tvs(i3))<D(tvs(i4)))
            t=tvs(i3);
            tvs(i3)=tvs(i4);
            tvs(i4)=t;
        end
    end
end
tmat=zeros(n); %temp matrix to save the betweenness
T=zeros(1,n); %temp array to save the sum of betweenness of the lower edges
of a vertex
for i3=1:length(tvs)
    tp=tvs(i3);
    for i4=1:length(tvs)
        if(mat(tvs(i3),tvs(i4)) && D(tvs(i3))==D(tvs(i4))-1 && D(tvs(i3))~= -1)
            T(tvs(i3))=T(tvs(i3))+tmat(tvs(i3),tvs(i4));
        end
    end
    for i4=1:length(tvs)
        if(mat(tvs(i3),tvs(i4)) && D(tvs(i3))==D(tvs(i4))+1 && D(tvs(i4))~= -1)
            tmat(tvs(i3),tvs(i4))=(T(tvs(i3))+1)*W(tvs(i4))/W(tvs(i3));
            tmat(tvs(i4),tvs(i3))=tmat(tvs(i3),tvs(i4));
        end
    end
end
smat=smat+tmat;
end
%find the max-betweenness edge
t=0;
for i2=1:n
    for i3=1:n

```

```

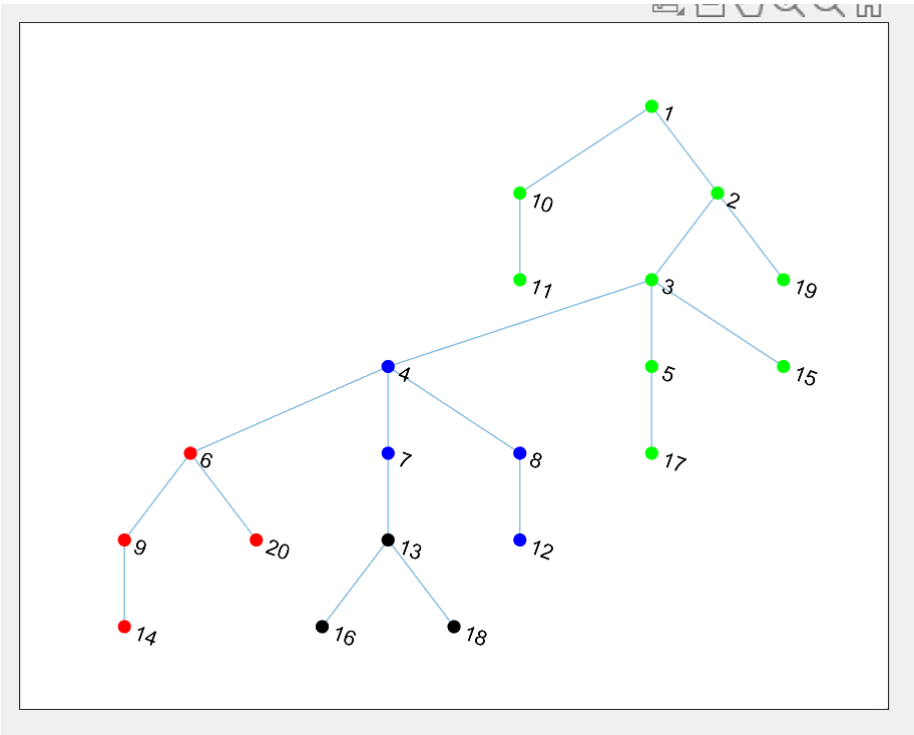
        if(smat(i2,i3)>t)
            t=smat(i2,i3);
            cut(i1,:)= [i2 i3 smat(i2,i3)];
        end
    end
end
    %save('tmat.txt','mat','-ascii');
    mat(cut(i1,1),cut(i1,2))=0;
    mat(cut(i1,2),cut(i1,1))=0;
end

figure;
h = plot(retGraph, 'NodeColor', 'g');
for i=1:19
    if cut(i,1)>=cut(i,2) && cut(i,1)>=cut(i,3)
        highlight(h, i, 'NodeColor', 'r');
    elseif cut(i,2)>=cut(i,3) && cut(i,2)>=cut(i,1)
        highlight(h, i, 'NodeColor', 'b');
    elseif cut(i,3)>=cut(i,1) && cut(i,3)>=cut(i,2)
        highlight(h, i, 'NodeColor', 'black');
    end
end
end

```

结果:

Louvain 社区检测算法:



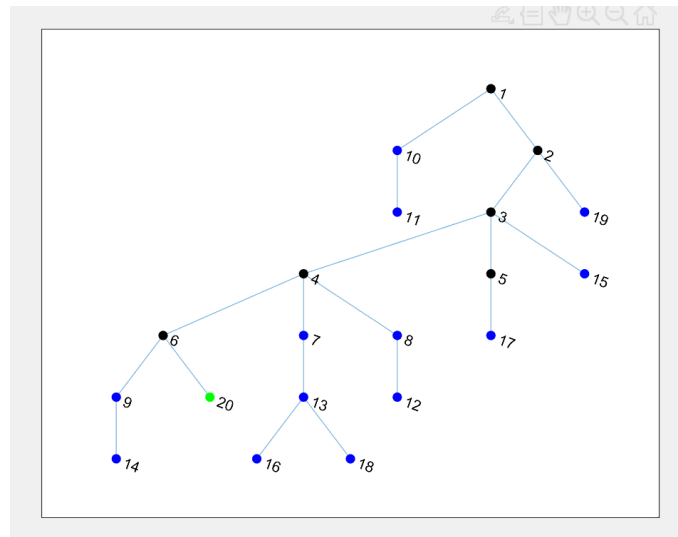
划分结果

M	
20x1 single	
	1
1	5
2	5
3	4
4	2
5	4
6	1
7	2
8	2
9	1
10	5
11	5
12	2
13	3
14	1
15	4
16	3
17	4
18	3
19	5
20	1
21	

模块度

Q	
1x1 single	
	1
1	0.5817
2	

GN 社区检测算法:



划分结果:

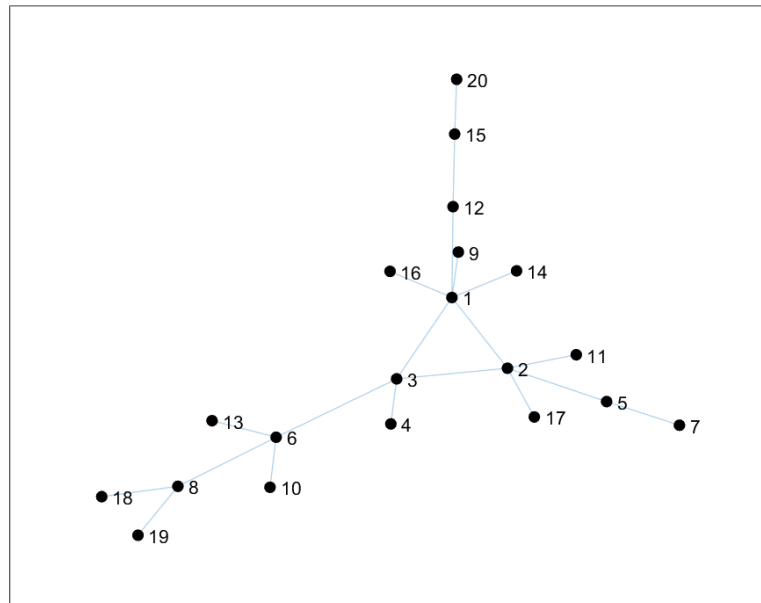
3	4	198
4	6	56
2	3	40
4	7	24
1	2	12
3	5	8
6	9	8
7	13	6
1	10	4
4	8	4
13	16	4
2	19	2
3	15	2
5	17	2
6	20	2
8	12	2
9	14	2
10	11	2
13	18	2

模块度:

Q	
1x1 double	
	1
1	0.6217
2	
3	

S3. 基于 PageRank 算法以及其他重要性指标对一种复杂网络模型的节点重要性进行排序。（15 分）

选择 BA 无标度网络



选择 PageRank、eigenvector、Betweenness 作为评估节点重要性的指标。

代码：

节点重要性

```
clear;
clc;

[sparsematrix, fullmatrix] = tcq_BAgraph_undir(20,3,1);
for i=1:20
    fullmatrix(i,i)=0;
end
retGraph = graph(fullmatrix);
plot(retGraph,'NodeColor','k','EdgeAlpha',0.3);
pg_ranks = centrality(retGraph,'pagerank');
eig_ranks = centrality(retGraph,'eigenvector');
betw_ranks = centrality(retGraph,'betweenness');
retGraph.Nodes.PageRank = pg_ranks;
retGraph.Nodes.Eigenvector = eig_ranks;
retGraph.Nodes.Betweenness= betw_ranks;
retGraph.Nodes
```

结果:

得到的三种重要性值

20×3 [table](#)

PageRank	Eigenvector	Betweenness
0.1342	0.15365	90
0.11248	0.13708	65
0.088314	0.12899	90
0.026272	0.041939	0
0.051666	0.049837	18
0.095863	0.064075	77
0.029452	0.016203	0
0.078349	0.026417	35
0.026499	0.049957	0
0.027863	0.020832	0
0.026626	0.044569	0
0.049807	0.056654	34
0.027863	0.020832	0
0.026499	0.049957	0
0.054902	0.020597	18
0.026499	0.049957	0
0.026626	0.044569	0
0.029707	0.008589	0
0.029707	0.008589	0
0.030808	0.0066966	0

根据三重指标算出的排名:



1	1	1
2	2	2
6	3	3
3	6	6
8	12	12
15	14	14
5	9	9
12	16	16
20	5	5
19	17	17
18	11	11
7	4	4
13	8	8
10	13	13
17	10	10
11	15	15
16	7	7
14	19	19
9	18	18
4	20	20

---

S4. 给定两类复杂网络，从拓扑性质、节点重要性、社团等角度分析两类网络的异同。（55 分）

该部分全部使用 MATLAB 完成，另外增加了 Gephi 丰富结果。

代码：

#### MATLAB

```
%%  
clear;  
clc;  
  
[sparsematrix, fullmatrix] = tcq_BAgraph_undir(20,3,1);  
for i=1:20  
    fullmatrix(i,i)=0;  
end  
retGraph = graph(fullmatrix);  
plot(retGraph,'NodeColor','k','EdgeAlpha',0.3);  
pg_ranks = centrality(retGraph,'pagerank');  
eig_ranks = centrality(retGraph,'eigenvector');  
betw_ranks = centrality(retGraph,'betweenness');  
retGraph.Nodes.PageRank = pg_ranks;  
retGraph.Nodes.Eigenvector = eig_ranks;  
retGraph.Nodes.Betweenness = betw_ranks;  
retGraph.Nodes  
[x1,index]=sort(betw_ranks);  
x1=flipud(x1);  
index=flipud(index);  
%%  
clear;  
clc;  
load T2.mat;  
A=zeros(301);  
for i=1:969  
    A(T2(i,1)+1,T2(i,2)+1)=T2(i,3);  
    A(T2(i,2)+1,T2(i,1)+1)=T2(i,3);  
end  
retGraph = digraph(A);  
plot(retGraph,'NodeColor','k','EdgeAlpha',0.3);  
retGraph = graph(A);  
pg_ranks = centrality(retGraph,'pagerank');  
retGraph.Nodes.PageRank = pg_ranks;  
retGraph.Nodes
```

```

[x1,index]=sort(pg_ranks);
x1=flipud(x1);
index=flipud(index);
%%%
load T4.mat;
A=zeros(301);
for i=1:1053
    A(T4(i,1)+1,T4(i,2)+1)=T4(i,3);
    A(T4(i,2)+1,T4(i,1)+1)=T4(i,3);
end
retGraph = graph(A);
plot(retGraph,'NodeColor','k','EdgeAlpha',0.3);
tcq_DD(A);
[c, aver_c]=tcq_clustering_Coefficient(A);
[D, aver_D]=tcq_Aver_Path_Length(A);
retGraph = graph(A);
pg_ranks = centrality(retGraph,'pagerank');
retGraph.Nodes.PageRank = pg_ranks;
retGraph.Nodes
[x1,index]=sort(pg_ranks);
x1=flipud(x1);
index=flipud(index);
%%%
clc
G=graph(A~=0);
plot(G)
figure;

NN = size(A,2);
DeD = zeros(1,NN);
for i = 1:NN
    DeD(i) = sum(A(i,:));
end

aver_DeD = mean(DeD);
if sum(DeD)==0
    disp('该网络图只是由一些孤立点组成');
    return;
else subplot(3,2,1);
    bar([1:NN],DeD);
    xlabel('节点编号 n');
    ylabel('各节点的度数 K');
    title('网络图中各节点的度的大小分布图');
end

```

```

M = max(DeD);
for i = 1:M+1
    N_DeD(i) = length(find(DeD==i-1));
end
P_DeD = zeros(1,M+1);
P_DeD(:) = N_DeD(:)./sum(N_DeD);
subplot(3,2,2)
bar([0:M],P_DeD,'r');
xlabel('节点的度 K');
ylabel('节点的度为 K 的概率 P(K)');
title('网络图中节点的度的概率分布');

d = size(A,2);
C = zeros(1,d);
for i = 1:d
    aa = find(A(i,:)==1);
    if isempty(aa)
        C(i) = 0;
    else
        m = length(aa);
        if m==1
            C(i) = 0;
        else
            B = A(aa,aa);
            C(i) = length(find(B==1))/(m*(m-1));
        end
    end
end
aver_C = C;
i = 1:d;
subplot(3,2,3)
plot(i,aver_C)
title('网络图中各节点的聚类系数');

d = size(A,2);
D = A;
D(find(D==0))=inf;
for i = 1:d
    D(i,i) = 0;
end
for k = 1:d
    for i = 1:d
        for j = 1:d

```

```

        if D(i,j)>D(i,k)+D(k,j)
            D(i,j)=D(i,k)+D(k,j);
        end
    end
end
end
aver_D = sum(D)/(d-1);
i = 1:d;
subplot(3,2,4)
plot(i,aver_D)
title('网络图中各节点的平均路径长度');

if aver_D==inf
    disp('该网络不是连通图');
end

G = graph(A~=0);
DC = degree(G)./(d-1);
i = 1:d;
subplot(3,2,5)
plot(i,DC)
title('网络图中节点的度中心性值');

d=size(A,1);
B=zeros(1,d);
[D,C]=tcq_Distance_F(A);
for k=1:d
    for i=1:d
        if i~=k
            for j=i+1:d
                if j~=k
                    if D(i,j)==D(i,k)+D(k,j)&&C(i,j)~=0
                        B(k)=B(k)+C(i,k)*C(k,j)/C(i,j);
                    end
                end
            end
        end
    end
end
end
i = 1:d;
subplot(3,2,6)
plot(i,B)
title('网络图中节点的归一化介数');

```

C++

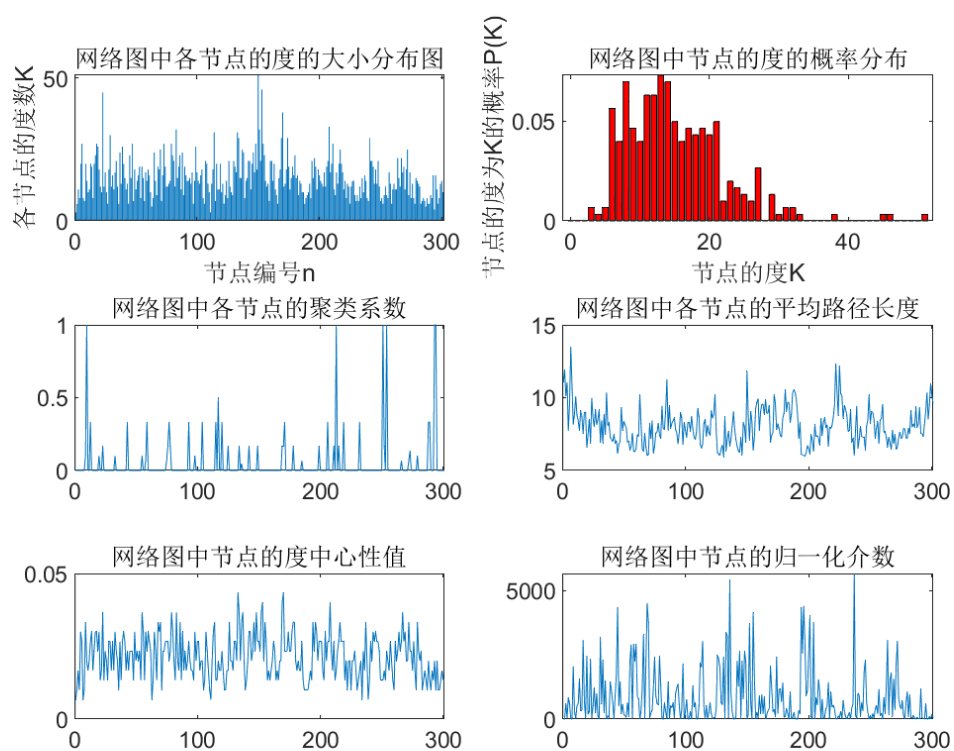
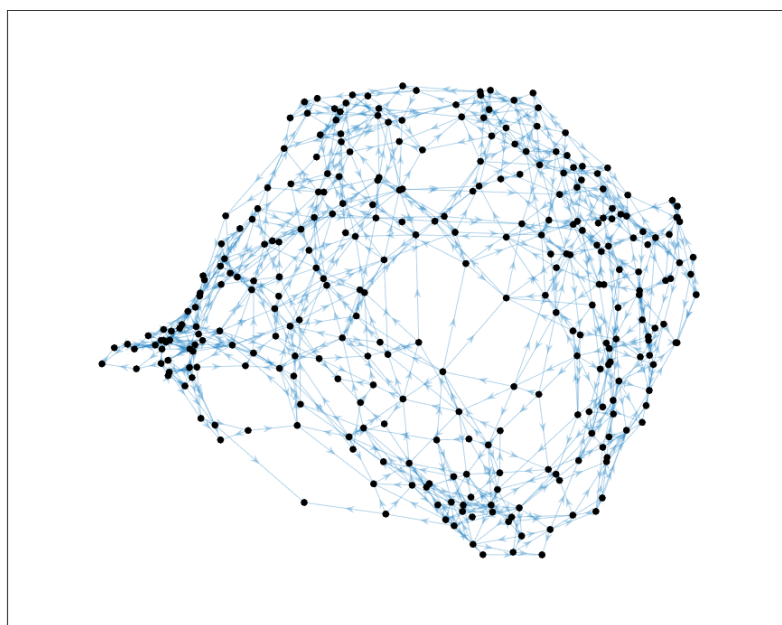
```
#include<iostream>
#include<map>
#define N 969
using namespace std;

string A[N],B[N];
int a[N],b[N];
int w[N];
int cnt=0;
int main(){
    map<string,int> M;
    for(int i=0;i<N;i++){
        cin>>A[i]>>B[i]>>w[i];
        if(M.count(A[i])==0)M[A[i]]=cnt++;
        if(M.count(B[i])==0)M[B[i]]=cnt++;
        a[i]=M[A[i]];
        b[i]=M[B[i]];
    }
    for(int i=0;i<N;i++){
        cout<<a[i]<<"\t"<<b[i]<<"\t"<<w[i]<<endl;
    }
}
```

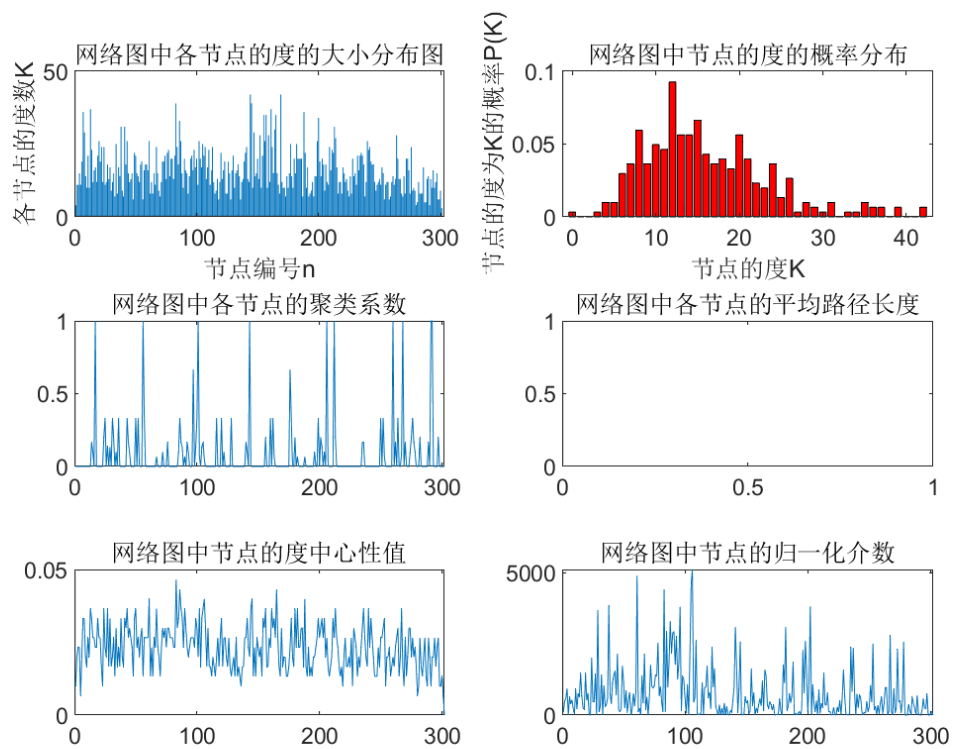
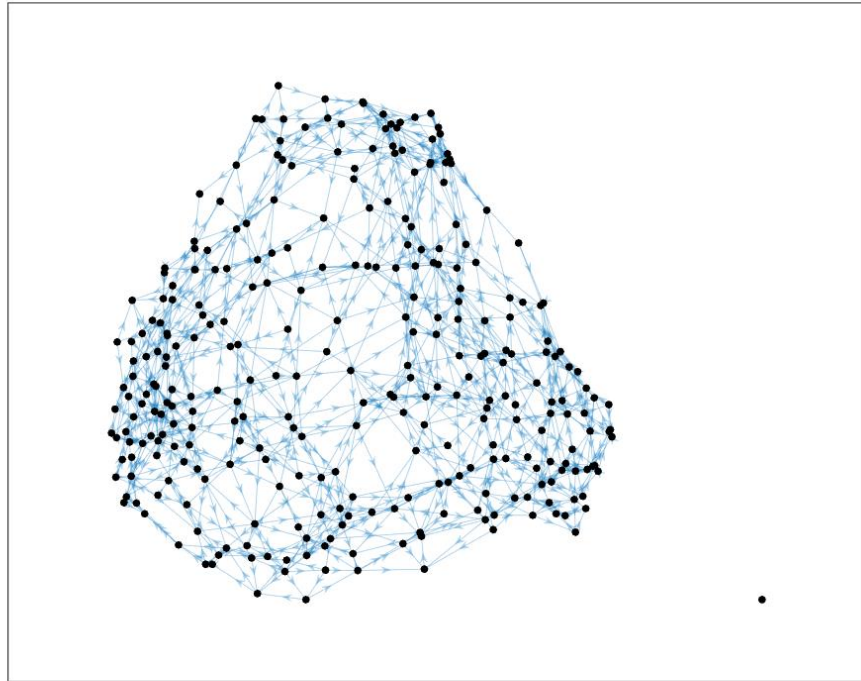
结果：

拓扑性质：

网络 1tux400\_299



网络 1tux400\_000



1tux400\_000 和 1tux400\_299 分析：从网络图的直观来看，两个网络看起来



差别不大，两个网络都具有 300 个左右的节点，网络网络 1tux400\_000 的连边数稍多，多大约 40 左右，由计算得到的数据也能验证这一点。1tux400\_299 中的节点集中于 5-20 之间且在此区间分布较为均匀，1tux400\_299 中的节点也集中于 5-20 之间，但是分布不太均匀。

进一步观察比较这两个网络拓扑性质的差别，两个网络的各项性质都基本相似。由于网络 1tux400\_000 存在一个孤立节点，所以路径长度无法计算，网络 1tux400\_299 的平均路径长度呈现平稳状态。同时这两个网络各节点的聚类系数则显得波动很大，在网络中有着一些重要节点，它们在网络中起着枢纽的作用，应该是进一步研究的重点；同时，度中心性也基本相同，在 0.02-0.04 之间。除此之外，两个网络的归一化介数波动也非常明显，也印证了有一些重要的节点。

由上述分析可知：两个网络在拓扑性质上差异不大，均存在着一些重要节点。

以下是 Gephi 做出的结果

过滤	统计 ×	—
设置		
网络概述		
平均度	5.716	运行 ?
平均加权重度	6.95	运行 ?
网络直径	13	运行 ?
图密度	0.019	运行 ?
点击次数		运行 ●
模块化	0.291	运行 ?
PageRank		运行 ●
连接部件		运行 ●
节点概述		
平均聚类系数	0.246	运行 ?
特征向量中心度		运行 ?
边概述		
平均路径长度	5.865	运行 ?

网络概述		
平均度	5.331	运行 ?
平均加权重度	6.417	运行 ?
网络直径	16	运行 ?
图密度	0.018	运行 ?
点击次数		运行 ?
模块化	0.303	运行 ?
PageRank		运行 ?
连接部件		运行 ●
节点概述		
平均聚类系数	0.23	运行 ?
特征向量中心度		运行 ?
边概述		
平均路径长度	7.314	运行 ?

节点重要性：

对两个网络使用 PageRank 算法，算出的排名靠前的 PageRank 值如下，左边为 1tux400\_000，右边为 1tux400\_299

0.0056	0.0058
0.0055	0.0058
0.0053	0.0056
0.0053	0.0056
0.0053	0.0055
0.0053	0.0054
0.0052	0.0054
0.0052	0.0053
0.0051	0.0053
0.0050	0.0050
0.0049	0.0050
0.0049	0.0049
0.0049	0.0049

算出的排名靠后的 PageRank 值如下，左边为 1tux400\_000，右边为 1tux400\_299

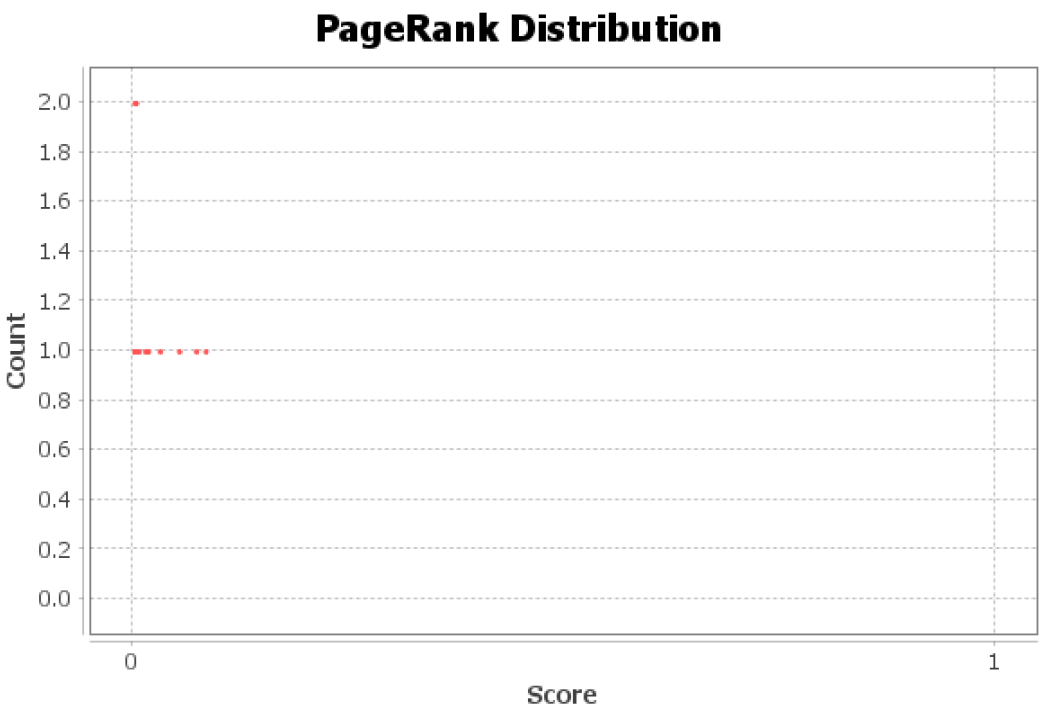
0.0018	0.0019
0.0018	0.0019
0.0018	0.0019
0.0018	0.0019
0.0018	0.0018
0.0018	0.0018
0.0018	0.0018
0.0017	0.0018
0.0017	0.0017
0.0015	0.0016
0.0015	0.0015
0.0014	0.0014
0.0005	0.0014

可以看出两个网络的节点重要性非常相似，最高值在 0.005 左右，最低值在 0.001 左右，重要性分布也非常类似。这说明两个网络背后的行为模式均是幂律，枢纽节点很少，但是却连接着大量的节点，而大量的非枢纽节点却只有不多的几个连接。

以下是 Gephi 做出的结果：

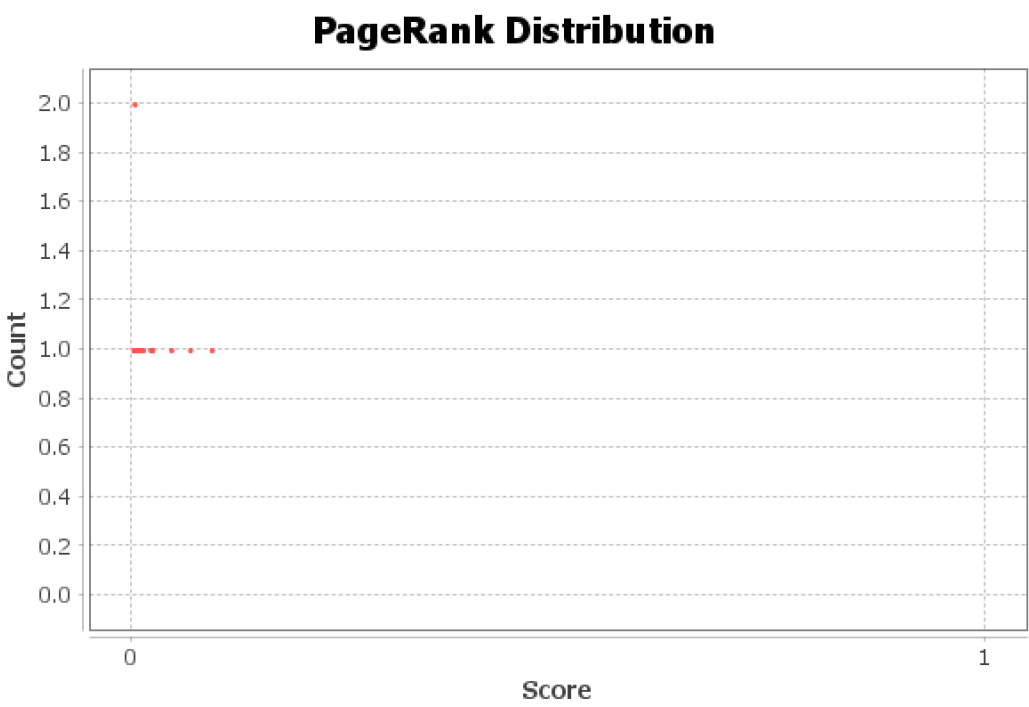
1tux400\_299

Results:



1tux400\_000

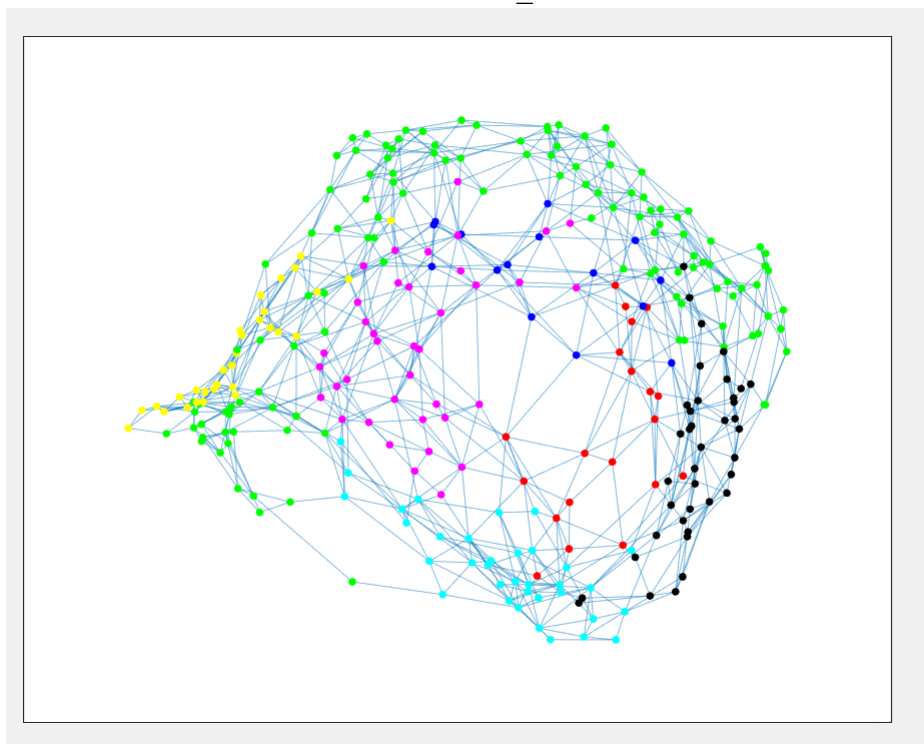
Results:



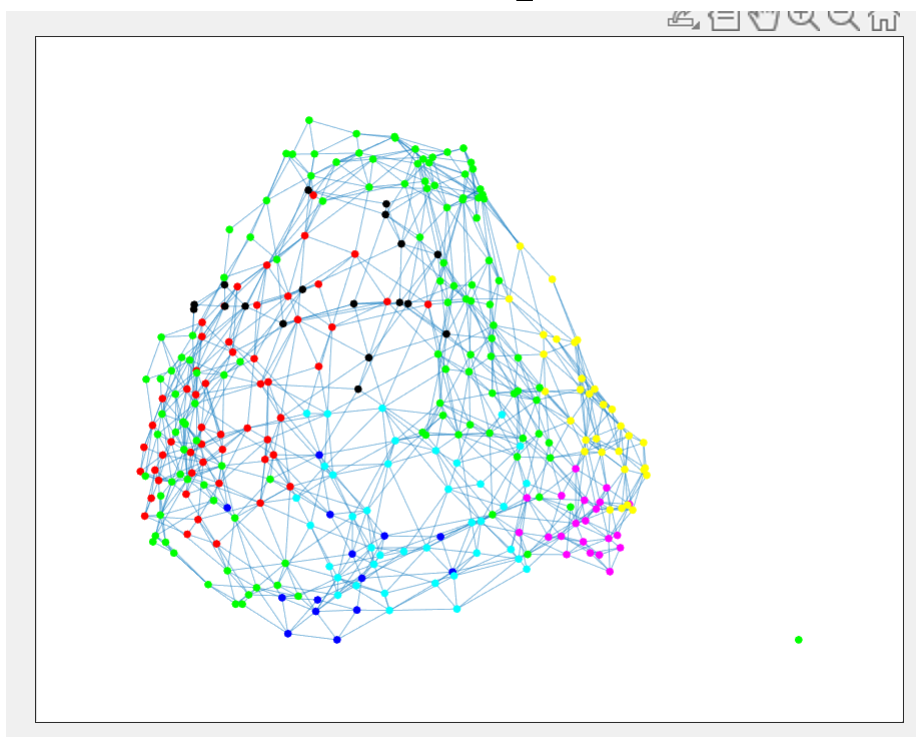
社团检测:

使用 louvain 算法对两个图进行社团检测

1tux400\_000



1tux400\_299



1tux400\_000 分为了 6 个社团，1tux400\_299 分为了 5 个社团。在图一中最

大的社团为 35%，排名第二的社团有 25%。在图二中，最大的社团有 30%，第二大的社团有 28%。这说明图一中包含了一个较大的社团和若干较小规模的社团，而图二中包含了几个规模接近的社团。

以下是 Gephi 做出的结果

