% importing data from <https://snap.stanford.edu/data/p2p-Gnutella08.html>

gra=importdata('p2p-Gnutella08.txt');

%data contains 1st vertex list data1 contains the 2nd vertex list

G=digraph(data,data1);

D = indegree(G);

D1 = outdegree(G);

%{

First step is anonimizing the degree sequence by computing the matrix of differences.

To compute the matrix ,w e first create sets of 9 elements.

inoutpair is the cell array with indegree-outdegree pairs of every vertex.

%}

a = floor(6300/700);

b = rem(6300, 700);

part = ones(1, 700)\*a;

part(1:b) = part(1:b)+1;

outmatrix=mat2cell(inoutpair, part,2);

%outmatrix the output set containing 700 partitions, each with 9 elements.

sumindeg=0;

avgindeg=cell(700,2);

anoninmat=cell(700,2); % anoninmat is the anonimized indegree matrix

for k=1:700

anoninmat{k,1}=0;

anoninmat{k,2}=0;

end;

for i=1:700

sumindeg=0;

for j=1:9

sumindeg=sumindeg+outmatrix{i,1}(j,1);

end

avgindeg{i,1}=floor(sumindeg/9);

avgindeg{i,2}=ceil(sumindeg/9);

end

for i=1:700

for j=1:9

anoninmat{i,1}=anoninmat{i,1}+outmatrix{i,1}(j,1)-avgindeg{i,1};

anoninmat{i,2}=anoninmat{i,2}+outmatrix{i,1}(j,1)-avgindeg{i,2};

end

end

%second step of degree anonimization is creating the probability distribution matrix .

probdist=cell(700,2);

for i=1:700

for j=1:2

if(anoninmat{i,1}~=0 && anoninmat{i,2}~=0)

probdist{i,j}=1-(abs(anoninmat{i,j})/(abs(anoninmat{i,1})+abs(anoninmat{i,2})));

else

probdist{i,j}=0;

end

end

end

%{

This code segment calculates the p vector based on the probability distribution matrix.It employs the GREEDY METHOD to reduce search complexity.In this, we select the minimum of the 2 columns in every row.

%}

pvector=cell(700,1);

for i=1:700

for j=1:2

if(probdist{i,1}>probdist{i,2})

pvector{i,1}=anoninmat{i,2};

elseif( probdist{i,1}<probdist{i,2})

pvector{i,1}=anoninmat{i,1};

else

pvector{i,1}=0;

end

end

end

%Using the p vector obtained in the previous step we create the new k-degree anonymized in-degree sequence.

%{

candidatevert contains an array of the vertices with minimum and maximum in-degrees of every partition that will alter their in-degree values according to the corresponding value(positive/negative) in the p-vector.

%}

newdegseq=cell(700,1); % anonymized in-degree sequence

minindex=zeros(700,1); %minindex is the index of the minimum in-degree in that partition.

maxindex=zeros(700,1); %maxindex is the index of the maximum in-degree in that partition.

k=1;

min=outmatrix{1,1}(1,1);

max=outmatrix{1,1}(1,1);

candidatevert=zeros(700,1);

for i=1:700

min=outmatrix{i,1}(1,1);

max=outmatrix{i,1}(1,1);

for j=2:9

if(min>=outmatrix{i,1}(j,1))

min=outmatrix{i,1}(j,1);

minindex(i)=j;

end

if(max<=outmatrix{i,1}(j,1))

max=outmatrix{i,1}(j,1);

maxindex(i)=j;

end

end

for j=1:9

if(j==maxindex(i))

if(pvector{i,1}>0)

newdegseq{i,1}(maxindex(i),1)=

outmatrix{i,1}(maxindex(i),1)-pvector{i,1};

candidatevert(k)= maxindex(i)+((i-1)\*9);

k=k+1;

end

elseif(j==minindex(i))

if(pvector{i,1}<0) newdegseq{i,1}(minindex(i),1)=outmatrix{i,1}(minindex(i),1)-pvector{i,1};

candidatevert(k)= minindex(i)+((i-1)\*9);

k=k+1;

end

else

newdegseq{i,1}(j,1)=outmatrix{i,1}(j,1);

end

end

end

%end of degree sequence anonymization.

%Next ,we have to modify the graph to match the anonymized in-degree sequence.

%We calculate the neighbourhood centrality values for the candidate vertices.

NC=cell(6300,20);% NC is the matrix of the neighbourhood centrality values of the candidate vertices.

for i=1:6300

for j=1:20

NC{i,j}=

numel(setdiff(union( pred{candidatevert(j)},pred{i}),intersect(pred{candidatevert(j)},pred{i})))/(182);

end

end

%In this segment we calculate the minimum NC value for every candidate vertex

minncvalues=cell(10,1);

minnc=NC{1,2};

for i=1:10

minnc=NC{1,i};

for j=1:10

if(minnc==0)

if(NC{j,i}~=0)

minnc=NC{j,i};

end

else

if(minnc>NC{j,i})

minnc=NC{j,i};

end

end

end

minncvalues{i,1}=minnc;

end

%Next, we find the list of edges to be removed from the original edge list.

extra=cell(100,2);

k=1;

indexp=[];

for i=1:20

indexp(i)=ceil(candidatevert(i)/9);

end

for i=1:20

if(pvector{indexp(i),1}>0)

count=0;

for j=1:numel(pred{candidatevert(i),1}(j,1))

if(NC{pred{candidatevert(i),1}(j,1),i}==minncvalues{i,1})

extra{k,1}=pred{candidatevert(i),1}(j,1);

extra{k,2}=candidatevert(i);

k=k+1;

count=count+1;

else

extra{k,1}=pred{candidatevert(i),1}(j,1);

extra{k,2}=candidatevert(i);

k=k+1;

count=count+1;

end

end

end

end

%This code segment finds list of edges to be added to the original list

addn=cell(56,2);

k=1;

indexp=[];

for i=1:10

indexp(i)=ceil(x(i)/9);

end

for i=1:10

if(pvector{indexp(i),1}<0)

count=0;

for j=1:10

if(NC{j,i}==minncvalues{i,1}&& count~=abs(pvector{indexp(i),1}))

addn{k,1}=x(j);

addn{k,2}=x(i);

k=k+1;

count=count+1;

end

end

end

end

%Finally , we create the new graph from the original list of edges minus the edges to be removed and appending the list of edges to be added.

newgraph=cell(20825,2);

k=1;

condn=0;

for i=1:20777

condn=0;

for j=1:11

if(oldgraph{i,1}== extra{j,1} && oldgraph{i,2}==extra{j,2})

condn=condn+1;

end

end

if(condn~= 1)

newgraph{k,1}=oldgraph{i,1};

newgraph{k,2}=oldgraph{i,2};

k=k+1;

end

end

for n=1:59

newgraph{k,1}=addn{n,1};

newgraph{k,2}=addn{n,2};

k=k+1;

end

col1=zeros(20825,1);% list of 1st vertex in the edge list

col2=zeros(20825,1);%list of the 2nd vertex in the edge list

for i=1:20825

col1(i)=newgraph{i,1};

col2(i)=newgraph{i,2};

end

G1=digraph(col1,col2);

plot(G1); % plotting the new graph

% {

This section calculates the evaluation metrics-average distance,harmonic mean,largest eigen value of the adjacency matrix of both the graphs.

%}

d=distances(G); % d is the matrix of the length of shortest path between all vertices in the original graph.

sum=0;

d1=distances(G1);

% d1 is the matrix of the length of shortest path between all vertices in the new graph.

sum1=0;

for i=1:6300

for j=1:6300

if(d(i,j)~=inf)

sum=sum+d(i,j);

end

if(d1(i,j)~=inf)

sum1=sum1+d1(i,j);

end

end

end

avgdist=sum/19841850 ;

avgdist2=sum1/19841850 ;

h=0.0;h1=0.0;

for i=1:6300

for j=1:6300

if(i~=j)

if(d(i,j)~=inf)

h=h+(1/d(i,j));

end

if(d1(i,j)~=inf)

h1=h1+(1/d1(i,j));

end

end

end

end

h=h/39683700;

h1=h1/39683700;

h=1/h; %harmonic mean of the original graph

h1=1/h1; %harmonic mean of the anonymized graph

A1=adjacency(G1);

e=eigs(A);

e1=eigs(A1);

%end of program