```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%
%This program is the implementation of several other programs pasted below.
%This algorithm is based on the nmf solving programs on Dr. Haesun Park website
https://www.cc.gatech.edu/~hpark/nmfsoftware.html
%
%The whole repository for the code implemented for this project is available at
https://github.com/dieguer/CSE6643-project
%
% The matrices used for the calculation are also available in such repository
% This matrices are aggregates that do not allow to identify individually units in
the network in any sense
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

clear
%Code to create the example matrices:
%Two matrices: (mxn) matrix X conteining features and (nxn) matrix S containing the
network
%The minimization probles are given by: min||X-WH|| and min||S-H^tH|| where W is
(mxk) and H is (kxn)

%We need to decide sample size n, features m, and clusters k
n=500;
m=10000;
k=6;

%One approach is to just create random matrices W and H such that X=WH and S=H^tH
rng(106,'twister');
s=rng;
%W = randi([100,1000],m,n);

W = rand(m,k);
H = rand(k,n);
X = W*H;
S = H.'*H;

%Another approach is to create W and H with a particular structure
%If one element in W1 is possitive, it is negative in W2,W3 and W4
D=rand(k,n);
for j=1:n
    mn = min(D(1:end,j));
    for i=1:k
        D(i,j)=D(i,j)==mn;
    end
end

Ht=H.*D;
%w=randi(20,m,k);
%Wt=w./sum(w);
Wt=randi(20,m,k)

Xt = Wt*Ht;
St = Ht.'*Ht;

%Algorithms
%Define parameters
alpha=norm(X,'fro')^2/norm(S,'fro')^2;
beta=alpha*max(max(S));
```

```
alphac=norm(Xt,'fro')^2/norm(St,'fro')^2;
betac=alpha*max(max(St));

%Define values for the loop
inc=10;
iv=10;
fv=100;

%Random matrix
Rw=zeros(fv/inc,1);
Rh=Rw;
Rb=Rw;
count=0;
for i=iv:inc:fv
    count=count+1;
    Rb(count)=i;
    maxitr=i;
    [Wp,Hp]=clustering(X,S,k,maxitr,alpha,beta);
    Rw(count,1)=norm(W,'fro')-norm(Wp,'fro');
    Rh(count,1)=norm(H,'fro')-norm(Hp,'fro');
end

plot(Rb,Rw)
hold on
plot(Rb,Rh)
hold off


hist(clustmem(H))
hist(clustmem(Hp))


%%
%Clustered matrix
Rwc=zeros(fv/inc,1);
Rhc=Rwc;
Rb=Rwc;
count=0;
for i=iv:inc:fv
    count=count+1;
    Rb(count)=i;
    maxitr=i;
    [Wtp,Htp]=clustering(Xt,St,k,maxitr,alphac,betac);
    Rwc(count,1)=norm(Wt,'fro')-norm(Wtp,'fro');
    Rhc(count,1)=norm(Ht,'fro')-norm(Htp,'fro');
end

plot(Rb,Rwc)
hold on
plot(Rb,Rhc)
hold off


hist(clustmem(Htp))
hist(clustmem(Ht))

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Real Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
S_real=dlmread("S_matrix.csv");
X_real=dlmread("X_matrix.csv");
X_real=X_real.'
alpha_r=norm(X_real,'fro')^2/norm(S_real,'fro')^2;
beta_r=alpha_r*max(max(S_real));

k=20;
[W_real,H_real]=clustering(X_real,S_real,k,200,alpha_r,beta_r);

clustmem(H_real)

ceros=find(sum(S_real,2)~=0)

S_R=S_real(ceros,ceros)

X_R=X_real(:,ceros)

alpha_R=norm(X_R,'fro')^2/norm(S_R,'fro')^2;
beta_R=alpha_R*max(max(S_R));

k=40

kas=[ 10 15 20 30 ]

for i=1:4
[W_R,H_R]=clustering(X_R,S_R,kas(i),500,alpha_R,beta_R);
gfc=figure
hist(clustmem(H_R))
hold on
saveas(gcf,sprintf('/home/diegolog/Documents/CSE 6643/project/CSE6643-project/
FIG%d.png',i))
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       CLUSTERING PROGRAM
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% implementation of paper
%   "Hybrid clustering based on content and connection structure"
% minimize ||X-WH||_F^2 + alpha*||S-Hhat'*H||_F^2 + beta*||Hhat - H||_F^2
% where X,S>=0 elementwise.
%
%<Inputs>
%   X: matrix size m * n
%   S: matrix size n * n
%   k: number of features
%   maxitr: max iteration times
%   alpha,beta: parameters for function (8)
%<Outputs>
%   W: matrix size m * k
%   H: matrix size k * n
function [W,H]=clustering(X,S,k,maxitr,alpha,beta)
    [m,n]=size(X);

    % using random initial value
    %H=rand(k,n);
    %Hhat=rand(k,n);
    %W=rand(m,k);
    % instead of using initial value for W and H
    % here using the nmf matrix W,H of matrix X as initial
    [W,H]=nmf(X,k);
    Hhat=H;

    %identical matrix and sqrt(alpha), sqrt(beta)
    Ik=diag(ones(k,1));
    alpha2=sqrt(alpha);
    beta2=sqrt(beta);

    for i=1:maxitr
        % equation (9) from paper
        %[wt,iter]= solveNormalEqComb(H',X');
        [ wt,Y,iter,success ] = nnlsm_blockpivot(H',X');
        %[ wt,Y,iter,success ] = nnlsm_activeset(H',X');
        W=wt';

        % equation (10) from paper
        m1=[alpha2*H';beta2*Ik];
        m2=[alpha2*S;beta2*H];
        %[Hhat,iter]=solveNormalEqComb(m1,m2);
        [ Hhat,Y,iter,success ] = nnlsm_blockpivot(m1,m2);
        %[ Hhat,Y,iter,success ] = nnlsm_activeset(m1,m2);

        % equation (11) from paper
        m1=[W;alpha2*Hhat';beta2*Ik];
        m2=[X;alpha2*S;beta2*Hhat];
        %[H,iter]=solveNormalEqComb(m1,m2);
        [ H,Y,iter,success ] = nnlsm_blockpivot(m1,m2);
        %[ H,Y,iter,success ] = nnlsm_activeset(m1,m2);
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       CLUSTERING Identification Program
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This function takes the H matrix in the algorithm and
% returns the vector X which indicates the cluster to which each
% unit belongs

function S= clustmem(X)

    [m,n]=size(X);
    S=zeros(n,1);
    for i=1:n
        [h,j]=max(X(1:m,i));



        if h==0
         j=99;
        end

        S(i)=j;
     end
    end
```