Project Report

On

Bayesian Community Detection in Social Network by Using Stochastic Blockmodel

By

Md. Mehrab Hossain
ID.NO. 300506
Student in University of Eastern Finland

Course Teacher
Ville Hautamäki, PhD
Senior researcher in University of Eastern Finland

Introduction

Modern network science has brought considerable progress to our understanding of complex systems. One of the most relevant characteristics of graphs representing real systems is community structure, or clustering, i.e. organizing vertices in clusters, with many edges joining vertices of the same cluster and comparatively few edges joining vertices of various clusters. Such clusters, or communities, may be regarded as a graph's fairly independent compartments, playing a similar role as, e. g., the human body's tissues or organs. In sociology, biology and computer science, disciplines where systems are often represented as graphs, the detection of communities is of great importance. A social network can be represented by a set of people where one member is connected from the same set to one or more members. By analyzing a social network, we can obtain visual and mathematical models of human relationships. Social networks have several inherent properties such as distribution of power law, centrality, small world network, modularity, etc. Another important property of the social network is the Community structure, which has gained tremendous popularity with regard to current research trends. As the community structure becomes increasingly popular, online social network services such as Facebook, Google+, MySpace and Twitter are also becoming equally complex [1].

Stochastic Block Model

The stochastic block model generates all networks that I consider in this project. This model divides a network's vertices into a number of blocks and then edges connect vertices with probabilities depending on the blocks in which the vertices are located. A community structure with the blocks representing the communities and classes is created in this way. I focus primarily on networks generated by the Clustering model, a special case of the stochastic model [2].

For any $n \in N, [n]$ denotes the set $\{1, \ldots\ldots n\}$. A network with n vertices is defined by the pair$([n], A)$, where A is an $n \times n$ matrix with, for$i, j \in [n]$,

$$A_{ij} = \begin{Bmatrix} 1 \text{ if the edge } (i,j) \text{is contained in the network} \\ 0 \text{ otherwise.} \end{Bmatrix}$$

The matrix A is called the adjacency matrix of the network [2].

In the Stochastic block model the set of vertices is divided into K ≥ 1 classes. This means that each vertex $i \in [n]$ gets a label Z_i $[K]$ that indicates what class vertex i is in. Let, $\underset{a}{\rightarrow} = \left(a_{k, \dots\dots\dots\dots,} \quad a_k\right)$ with $\sum_{t=1}^{K} a_i = 1$, be the vector of block proportions, such that

$$Z = (Z_i)_{t \in n} \widetilde{\iota.\iota d.} \mathcal{M}(1 \underset{a}{\rightarrow})$$

Here $\mathcal{M}$ denotes the multinomial distribution, i.e. $Z_i = k$ with probability a_k for all $i \in [n]$ $k \in [K]$.

Let $P = (P_{kl})_{k,l \in [k]}$ be a $K \times K$-matrix of probabilities, such that

$$\mathbb{P} = \left(A_{ij} = 1 \middle| Z_i = k, Z_j = l\right) = P_{kl}.$$

This way the probability of an edge between two vertices only depends on the classes the vertices are in. I assume that the probabilities on the diagonal of P are the largest, such that the classes form communities with relatively many connections within them [2].

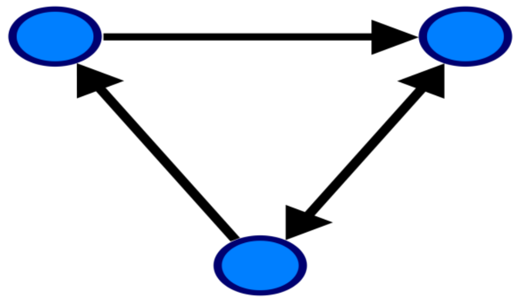
Newman-Girvan Algorithm

Newman Girvan algorithm method uses when the number of classes is known and unknown to identify the communities. This method is more difficult to calculate, but more reliable than the LG algorithm is generally considered. The algorithm itself is then used to find a partition for each K{1,......,n} in K classes. To decide which of these partitions is to use the so-called Newman-Girvan modularity as the estimate for the true partition [2].

The Newman-Girvan algorithm takes advantage of the network's edge betweenness measures. An edge's betweenness can roughly be described as the number of shortest paths that pass through this particular edge between all pairs of vertices. In a community-based network, we expect the edge betweenness to be larger for inter-community edges (i.e. edges that connect two vertices in different communities), as many of the shortest paths from one community to the other pass through these edges. The edge betweenness scores could therefore help us identify network classes [2].

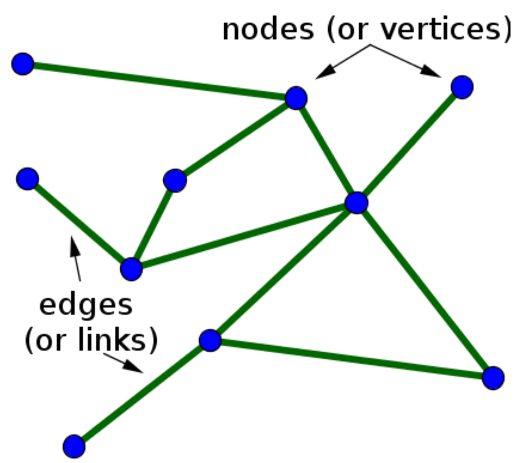
Directed Graph

A directed graph is a set of objects that are connected together (called vertices or nodes) where all edges are directed from one vertex to another. Sometimes a steered graph is called a digraph or a steered network [5].



Undirected Graph

An undirected graph is a set of objects that are connected together (called vertices or nodes) where all edges are bidirectional. Sometimes an undirected graph is called an undirected network. In contrast, a graph is called a directed graph where the edges point in a direction [5].



Data Set Information

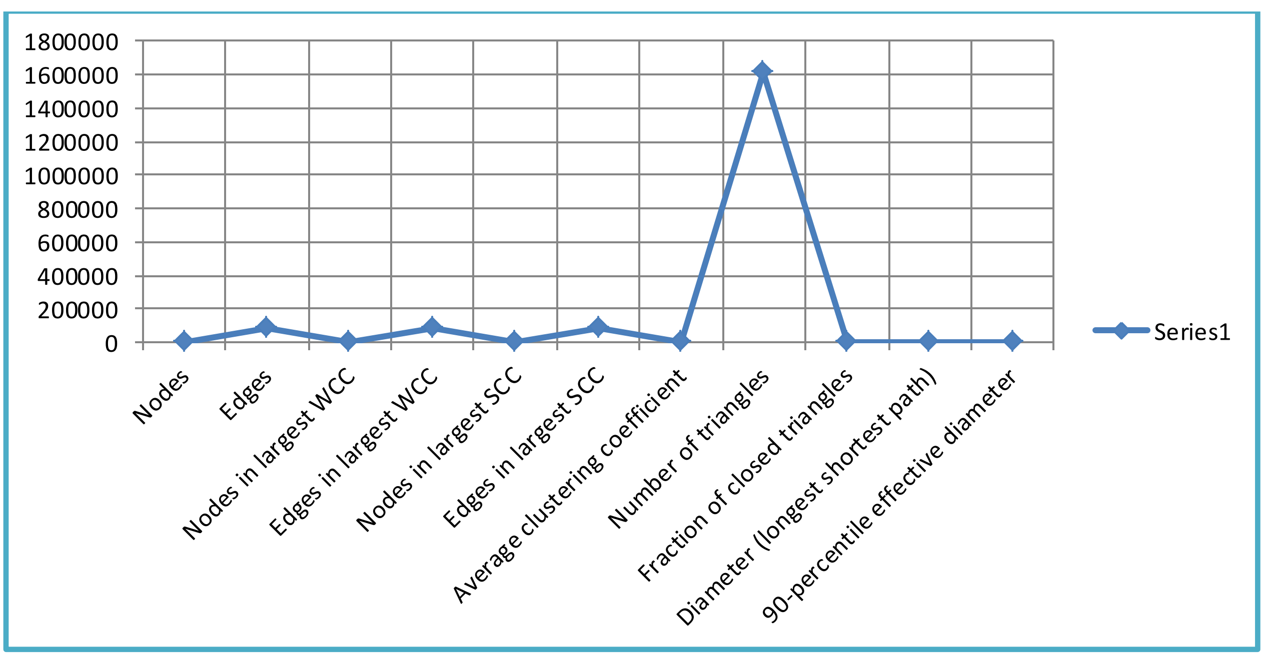
I provided social network datasets but tested Newman-Girvan algorithms on four real world networks: Ego-Facebook, Ego-Twitter, Soc-sign-bitcoin-alpha, and Wiki-Vote. A brief description of all four datasets is given below [4]:

Social networks:

Name	Type	Nodes	Edges	Description
ego-Facebook	Undirected	4,039	88,234	Social circles from Facebook (anonymized)
ego-Gplus	Directed	107,614	13,673,453	Social circles from Google+
ego-Twitter	Directed	81,306	1,768,149	Social circles from Twitter
soc-Epinions1	Directed	75,879	508,837	Who-trusts-whom network of Epinions.com
soc-LiveJournal1	Directed	4,847,571	68,993,773	LiveJournal online social network
soc-Pokec	Directed	1,632,803	30,622,564	Pokec online social network
soc-Slashdot0811	Directed	77,360	905,468	Slashdot social network from November 2008
soc-Slashdot0922	Directed	82,168	948,464	Slashdot social network from February 2009
wiki-Vote	Directed	7,115	103,689	Wikipedia who-votes-on-whom network
wiki-RfA	Directed, Signed	10,835	159,388	Wikipedia Requests for Adminship (with text)
soc-RedditHyperlinks	Directed, Signed, Temporal, Attributed	55,863	858,490	Hyperlinks between subreddits on Reddit
soc-sign-bitcoin-otc	Weighted, Signed, Directed, Temporal	5,881	35,592	Bitcoin OTC web of trust network
soc-sign-bitcoin-alpha	Weighted, Signed, Directed, Temporal	3,783	24,186	Bitcoin Alpha web of trust network
gemsec-Deezer	Undirected	143,884	846,915	Gemsec Deezer dataset
gemsec-Facebook	Undirected	134,833	1,380,293	Gemsec Facebook dataset

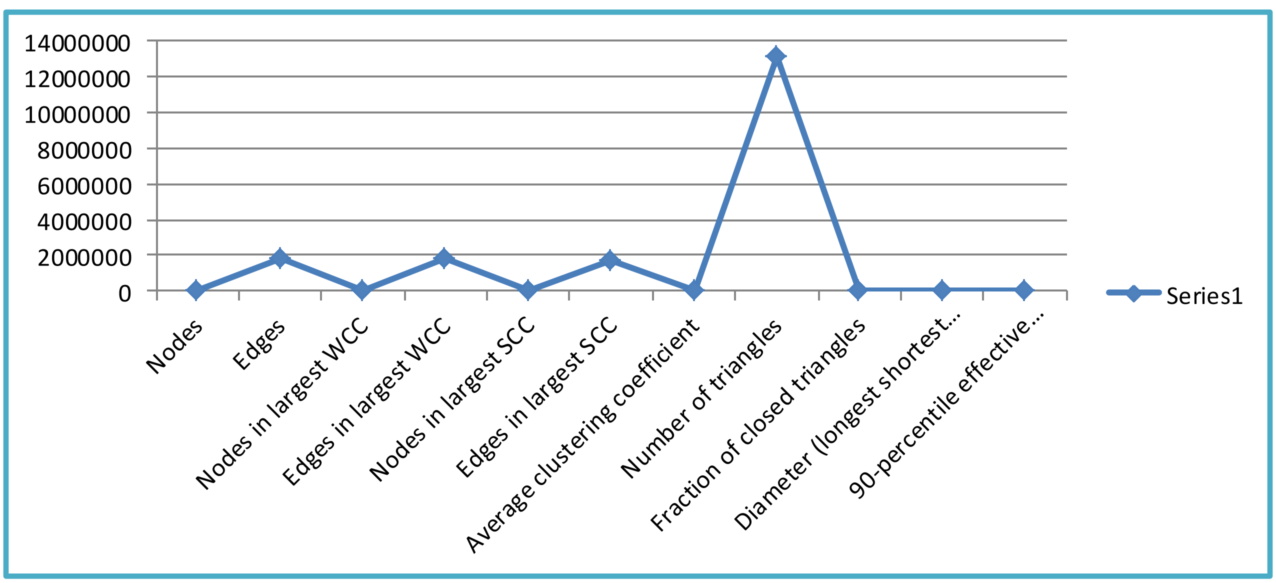
1) **Ego-Facebook:** This dataset is made up of Facebook's' circles' (or' friend lists'). Using this Facebook app, Facebook data was gathered from survey participants. Node features (profiles), circles, and ego networks are included in the dataset. Facebook data was anonymised by replacing the internal Facebook ids with a new value for each user [4].

Dataset statistics	
Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7



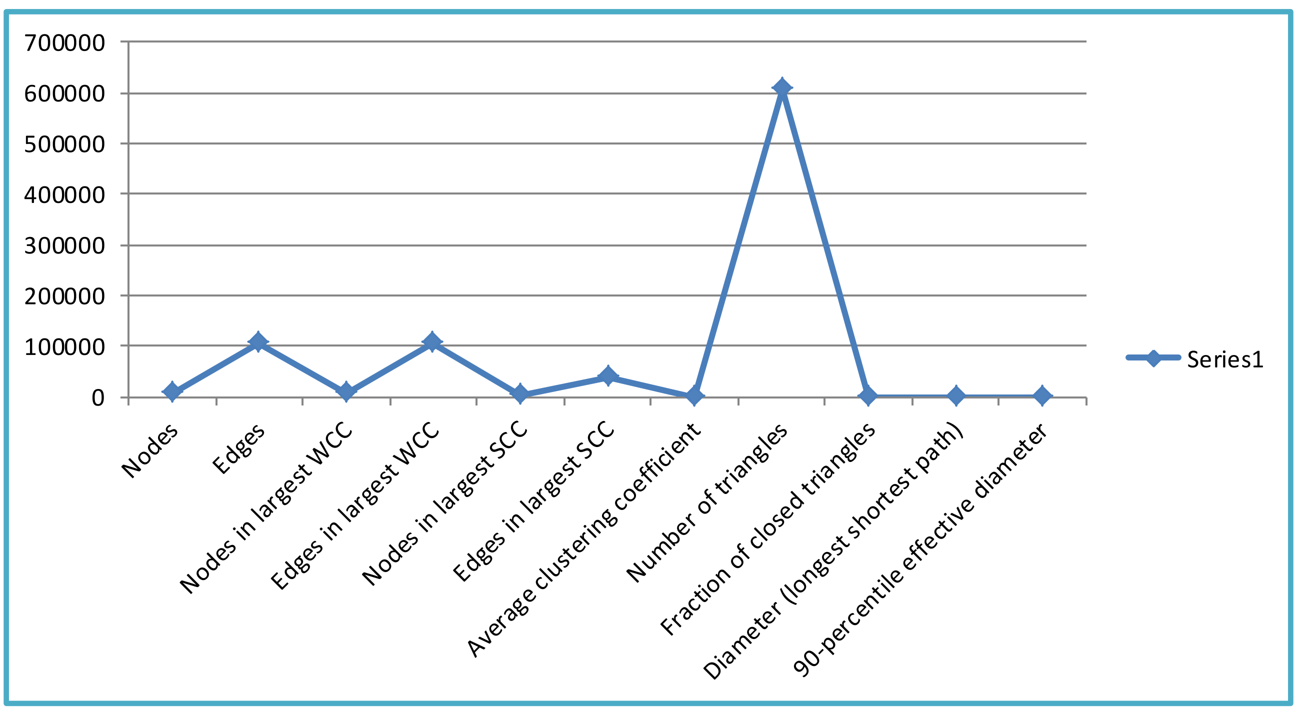
2) **Ego-Twitter**: This dataset is made up of Twitter ' circles ' (or ' lists '). Public sources crawled Twitter data. There are node features (profiles), circles, and ego networks in the dataset [4].

Dataset statistics	
Nodes	81306
Edges	1768149
Nodes in largest WCC	81306 (1.000)
Edges in largest WCC	1768149 (1.000)
Nodes in largest SCC	68413 (0.841)
Edges in largest SCC	1685163 (0.953)
Average clustering coefficient	0.5653
Number of triangles	13082506
Fraction of closed triangles	0.06415
Diameter (longest shortest path)	7
90-percentile effective diameter	4.5



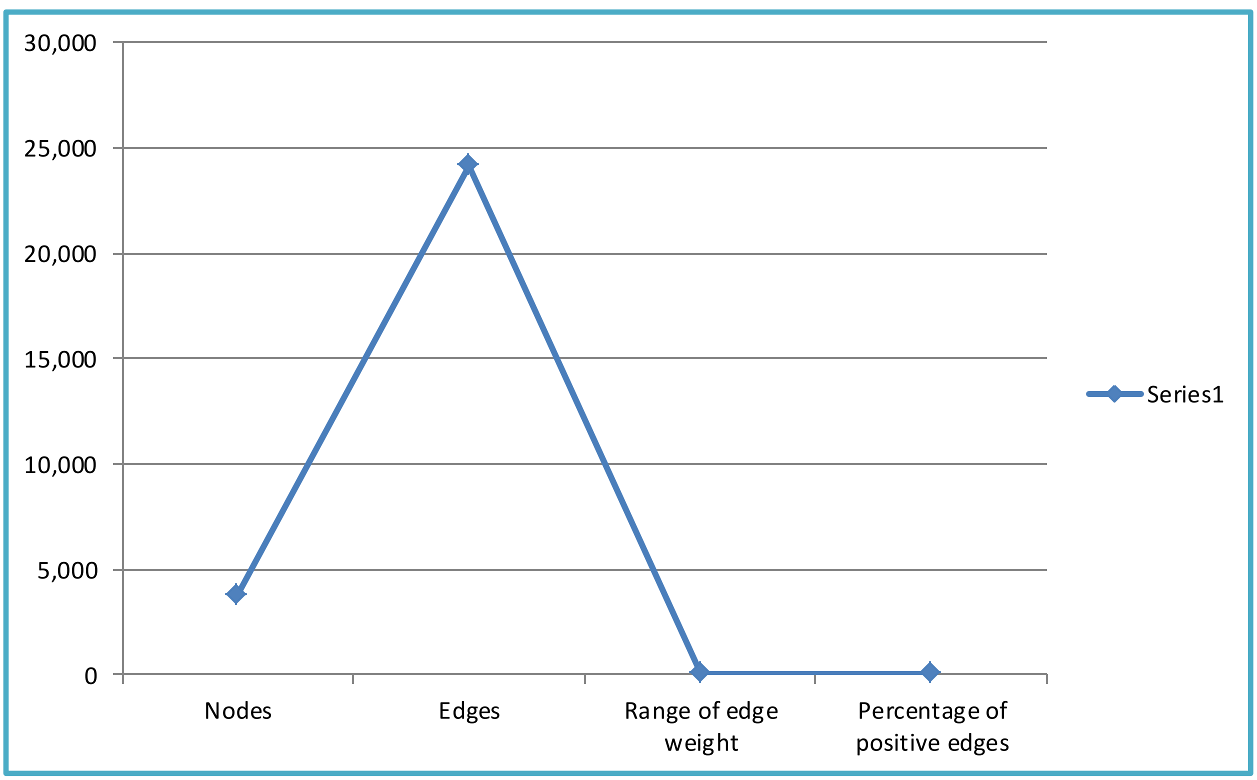
3) Wiki-Vote: The network contains all the voting data from Wikipedia's inception until January 2008. Nodes in the network represent users of Wikipedia and a guided edge from node I to node j shows that user I voted on user j [4].

Dataset statistics	
Nodes	7115
Edges	103689
Nodes in largest WCC	7066 (0.993)
Edges in largest WCC	103663 (1.000)
Nodes in largest SCC	1300 (0.183)
Edges in largest SCC	39456 (0.381)
Average clustering coefficient	0.1409
Number of triangles	608389
Fraction of closed triangles	0.04564
Diameter (longest shortest path)	7
90-percentile effective diameter	4.7



4) **Soc-sign-bitcoin-alpha:** This is who-trusts-whom network of people trading on a platform called Bitcoin Alpha using Bitcoin. Because Bitcoin users are anonymous, to prevent transactions with fraudulent and risky users, there is a need to keep a record of reputation of users. Bitcoin Alpha members rate other members in steps from -10 (total distrust) to+ 10 (total trust). This is the first signed directly signed explicit weighted network available for research [4].

Dataset statistics	
Nodes	3,783
Edges	24,186
Range of edge weight	-10 to +10
Percentage of positive edges	93%



Experimental Set-up

I presented my work in this project using the Community Detection Toolbox (CDTB), a MATLAB toolbox that can be used for community detection. The CDTB includes several functions from the categories below [3].

1. Graph generators;
2. Clustering algorithms;
2. Cluster number selection functions;
4. Clustering evaluation functions.

In addition, CDTB is parametrically designed to allow me to add my own functions and extensions [3].

The CDTB can be used in at least three ways, such as; user can use the MATLAB command line functions; or user can write their own code incorporating the CDTB functions; or user can use the Graphical User Interface (GUI) that automates community detection and includes some options for data visualization [3].

Algorithms:

```
function VV= GCModulMax3(A)
         N=length(A);
         W=PermMat(N);                  % permute the graph node labels
         A=W*A*W';
         [VV,Q] = fast_newman(A);
         VV=W'*VV;                      % unpermute the graph node labels
end
function [com,Q] = fast_newman(adj)
         cur_com = [1:length(adj)]';
         com = cur_com;
         e = get_community_matrix(adj,com);
         ls = sum(e,2);
         cs = sum(e,1);
        cur_Q = trace(e) - sum(sum(e^2));
         Q = cur_Q;
    while length(e) > 1
        loop_best_dQ = -inf;
        can_merge = false;
        for i=1:length(e)
            for j=i+1:length(e)
                if e(i,j) > 0
                   dQ = 2 * (e(i,j) - ls(i)*cs(j));
                    if dQ > loop_best_dQ
                        loop_best_dQ = dQ;
                        best_pair = [i,j];
                        can_merge = true;
                    end
```

```
                end
            end
        end
        if ~can_merge
            disp('!!! Graph with isolated communities, no more merging
possible !!!');
            break;
        end

        % Merge the pair of clusters maximising Q
        best_pair = sort(best_pair);
        for i=1:length(cur_com)
            if cur_com(i) == best_pair(2)
                cur_com(i) = best_pair(1);
            elseif cur_com(i) > best_pair(2)
                cur_com(i) = cur_com(i) - 1;
            end
        end
        e(best_pair(1),:) = e(best_pair(1),:) + e(best_pair(2),:);
        e(:,best_pair(1)) = e(:,best_pair(1)) + e(:,best_pair(2));
        e(best_pair(2),:) = [];
        e(:,best_pair(2)) = [];
        % Update lines/colums sum
        ls(best_pair(1)) = ls(best_pair(1)) + ls(best_pair(2));
        cs(best_pair(1)) = cs(best_pair(1)) + cs(best_pair(2));
        ls(best_pair(2)) = [];
        cs(best_pair(2)) = [];

        % Update Q value
        cur_Q = cur_Q + loop_best_dQ;

        % If new Q is better, save current partition
        if cur_Q > Q
            Q = cur_Q;
            com = cur_com;
        end

        %fprintf(' completed in %f(s)\n',toc);

    end

end
```

Graph Function:

```
function [A,V0]=GGGirvanNewman(N1,K,zi,ze,Diag)
% Generation of a Girvan Newman graph

%
% INPUT:
% N1 number of nodes in each community
% K  number of communities
% zi number of internal half-edges per node
% ze number of external half-edges per node
```

```
% Diag:  if Diag=1, use self-loops; if Diag=0, don't use self-loops
%
% OUTPUT:
% A  adajcency matrix (N-by-N)
% V0 classification vector (N-by-1)
% W  permutation matrix (N-by-N) to put the nodes in order

pi=zi/(N1-1);
pe=ze/(N1*(K-1));
NN=[0];
for k=1:K
     NN(1,k+1)=k*N1;
end
[A,V0]=GGPlantedPartition(NN,pi,pe,Diag);
```

Cluster Number Selection:

```
function Kbst=CNDistBased(VV,A)
% INPUT
% VV:     N-by-K matrix of partitions, k-th column describes a partition
%         of k clusters
% A:      adjacency matrix of graph

% OUTPUT
% Kbst:   the number of best VV column and so best number of clusters

[N Kmax]=size(VV);
for K=1:Kmax
    V=VV(:,K);
    Q(K)=QFDistBased(V,A);
end
[Qbst Kbst]=min(Q);
```

Evaluation:

```
function Q=PSJaccard(V,V0)
% INPUT
% V:        N-by-1 matrix describes 1st partition
% V0:        N-by-1 matrix describes 2nd partition
% OUTPUT
% Q:         The Jaccard similarity between V and V0

if  ~isvector(V)
    error('V must be a vector');
end
if ~isvector(V0)
    error('V must be a vector');
end
if length(V) ~= length(V0)
    error('V and V0 must have the same size');
end
a11 = 0;
a10 = 0;
```

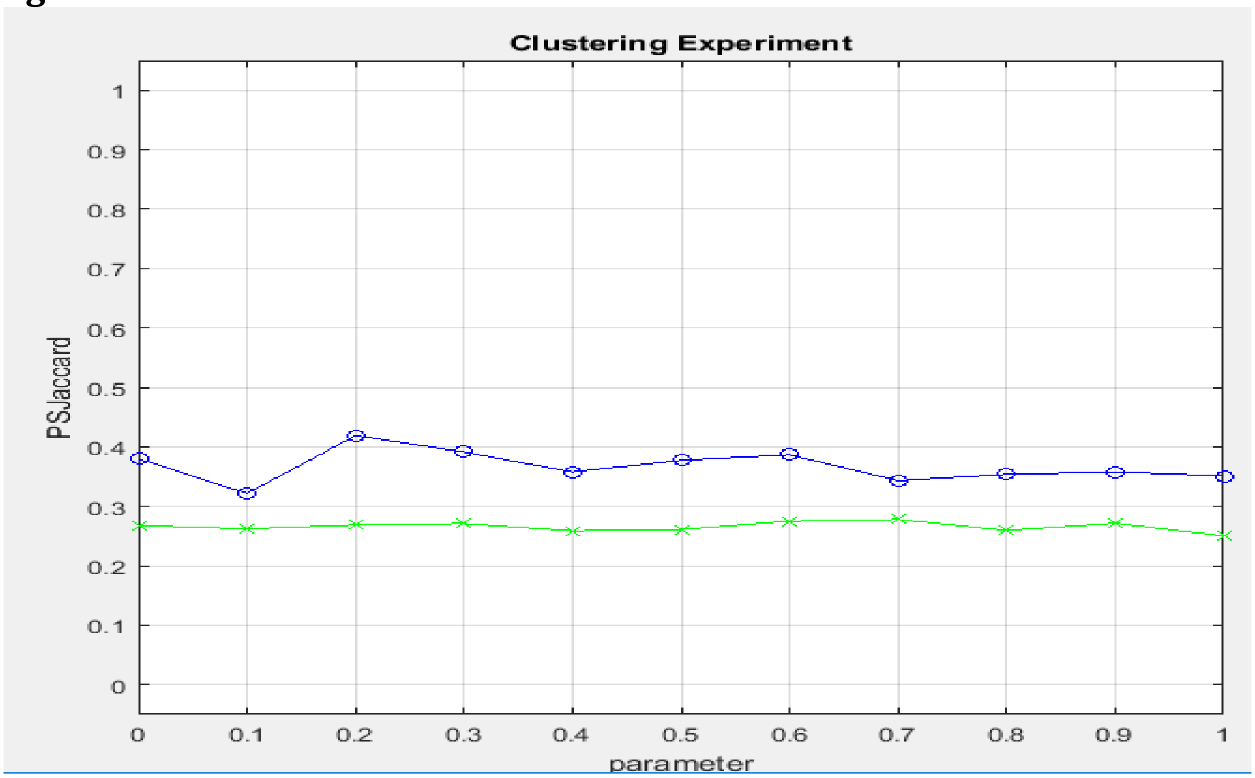
```
a01 = 0;
for i = 1:length(V)
    for j = 1:length(V)
        if i == j
            continue
        end
        sameV = V(i) == V(j);
        sameV0 = V0(i) == V0(j);

        if sameV && sameV0
            a11 = a11 + 1;
        elseif sameV && ~ sameV0
            a10 = a10 + 1;
        elseif ~sameV && sameV0
            a01 = a01 + 1;
        end
    end
end

% Result
Q = a11/(a11+a10+a01);
end
```

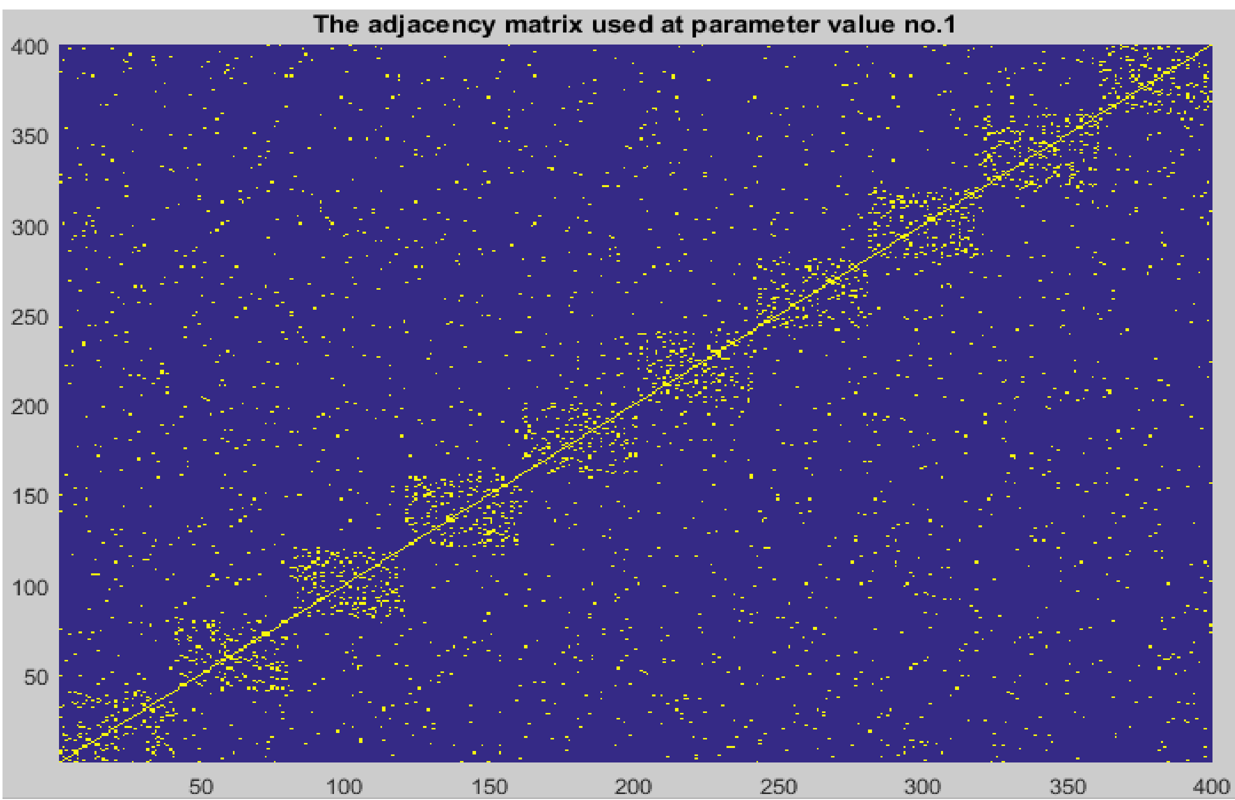
Results:

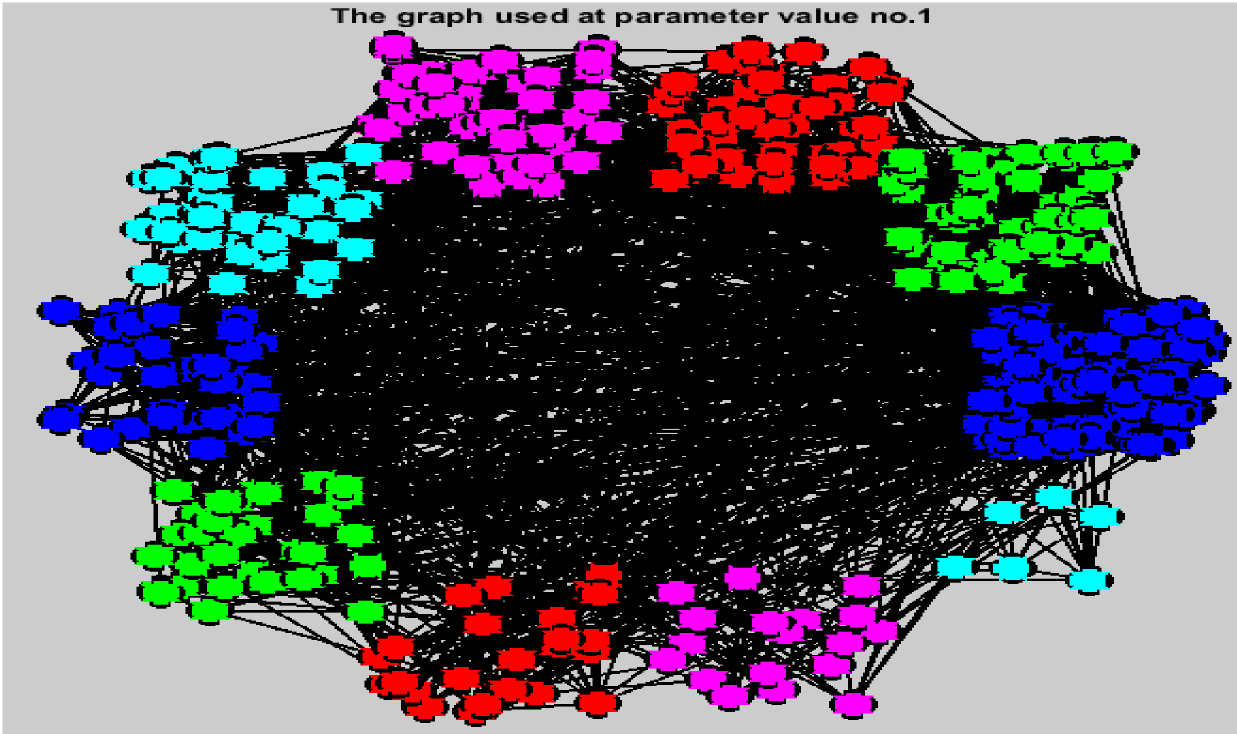
Code:

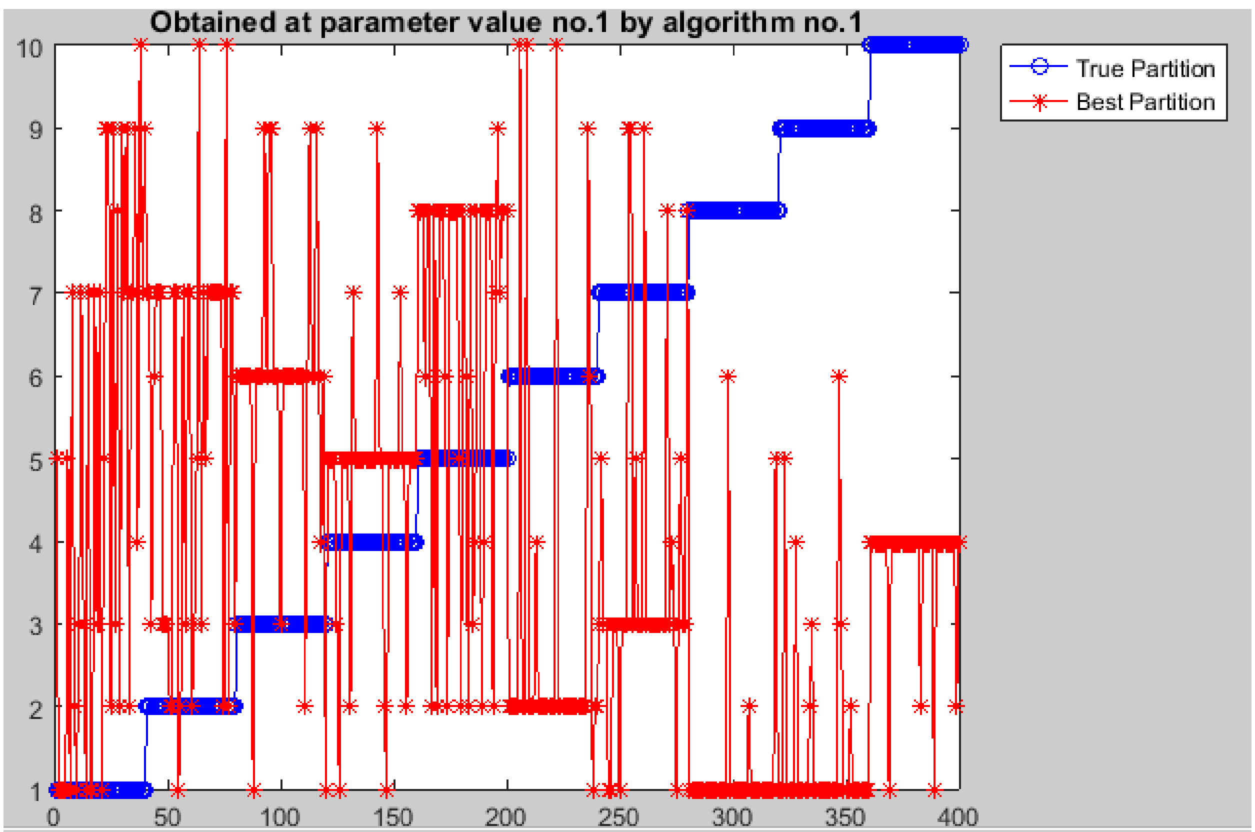
```
N1=30; K=5; Diag=1;
Scale=[2 1.5 0.5 0.4 0.3 0.2];
for i=0:8
    zi=16-i;
    zo=i;
    [A,V0]=GGGirvanNewman(N1,K,zi,zo,Diag);
    N=length(V0);
    VV=GCAFG(A,Scale);
    Mbst=CNLocDens(VV,A);
    V=VV(:,Mbst);
    Q1(i+1,1)=PSNMI(V,V0);
    K1(i+1,1)=max(V);
    figure(1); plot([V V0])
    axis([0 N+1 0 K1(i+1)+1])
    xlabel('Node no.'); ylabel('Cluster membership');   pause(0.5);
end
figure(2); plot(Q1); axis([1 9 -0.05 1.05]);
xlabel('zo'); ylabel('NMI(V,V0)')
figure(3); plot(K1); axis([1 9 0 max(K1)])
xlabel('zo'); ylabel('NMI(V,V0)')
```

1) Ego-Facebook:

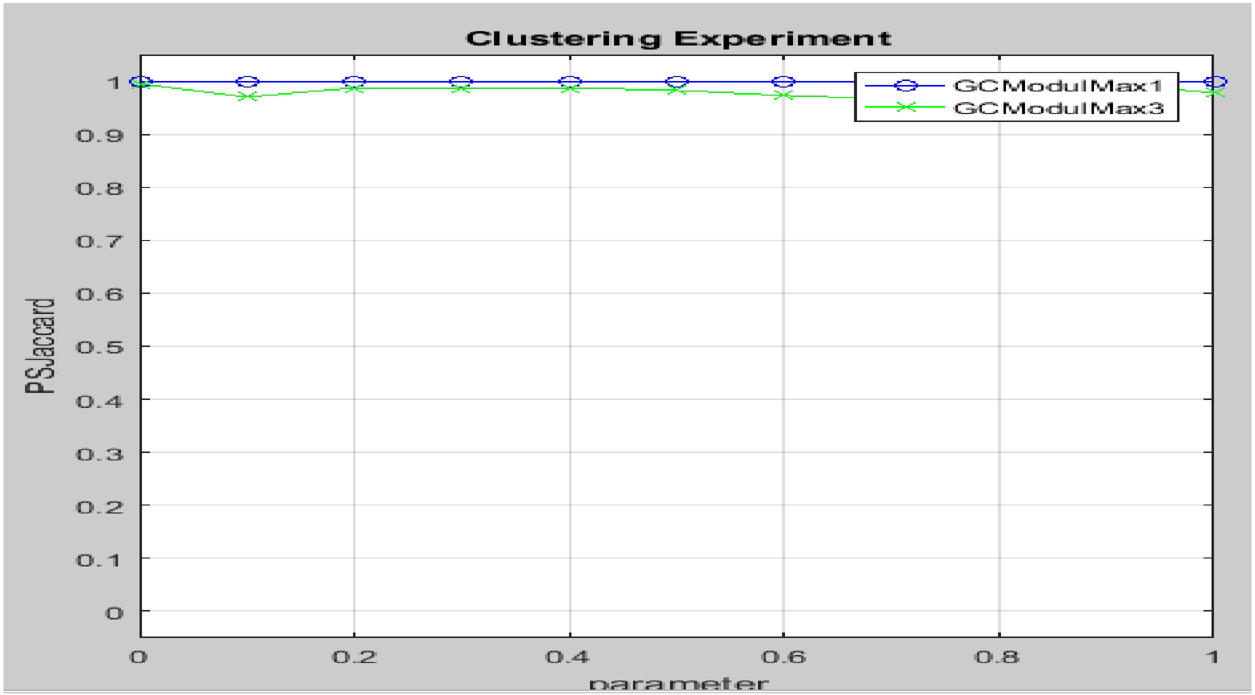


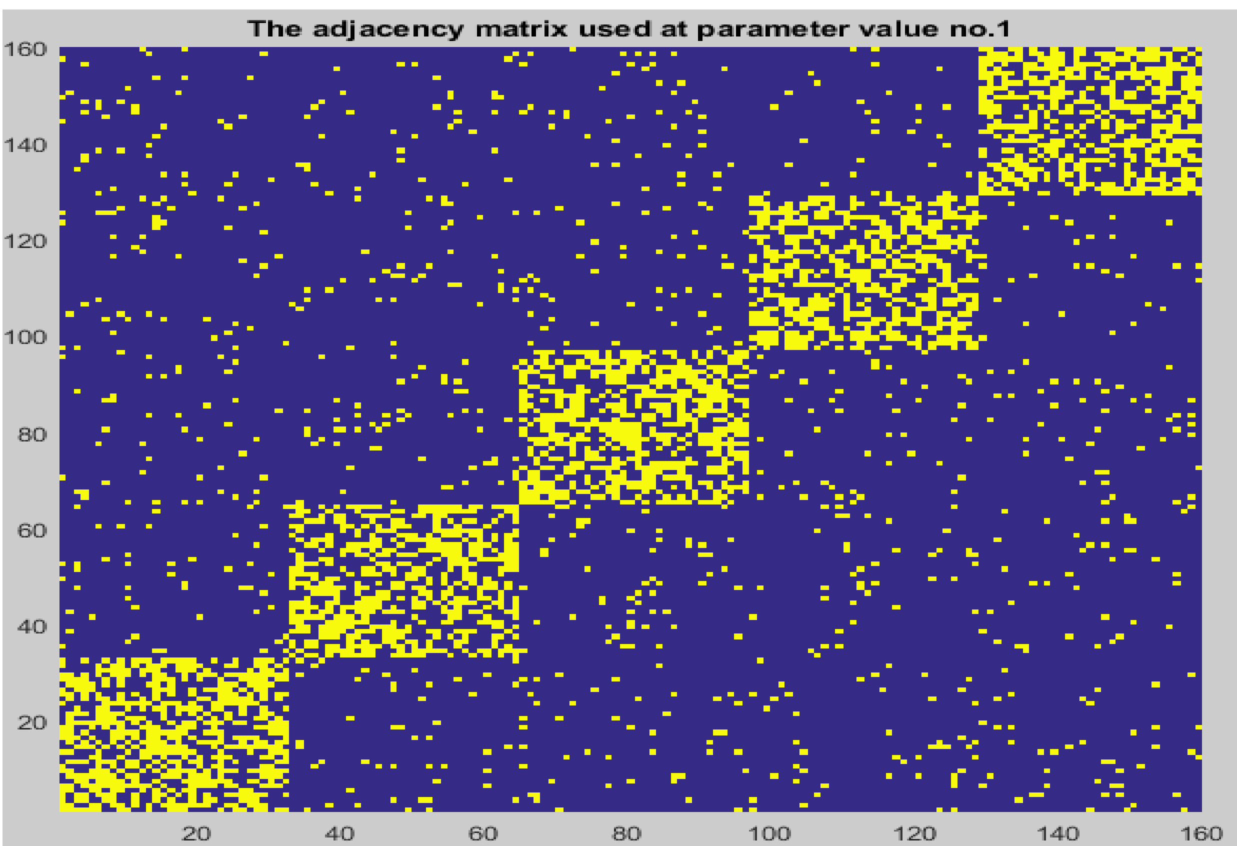


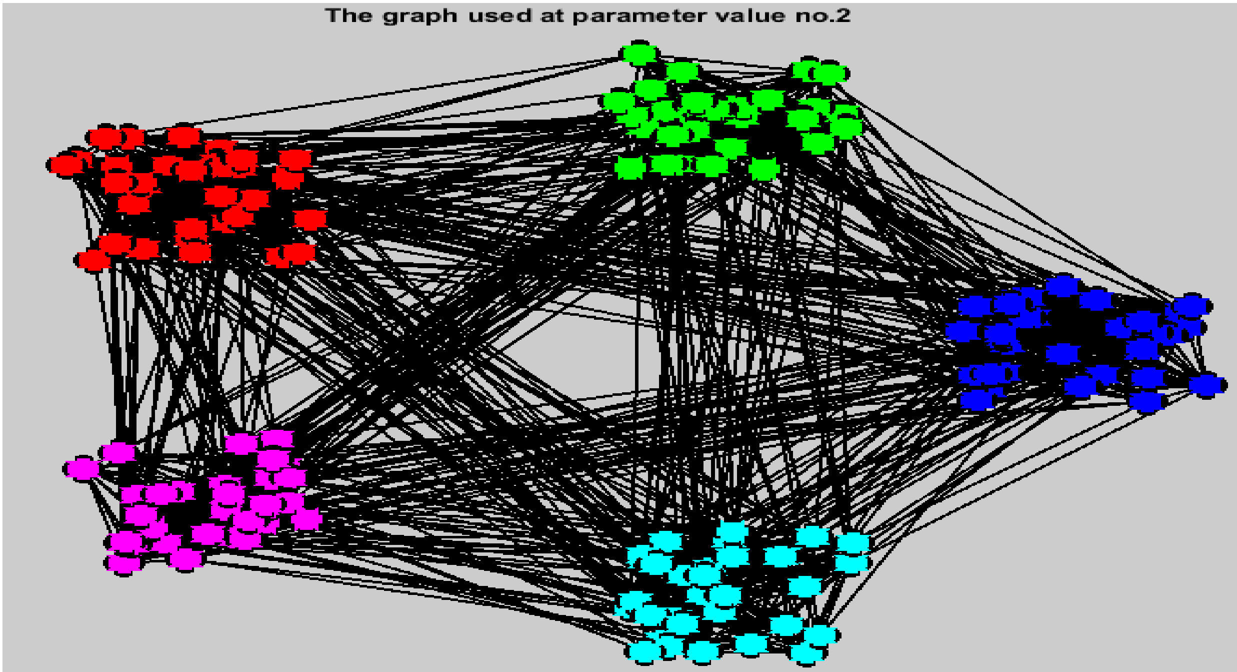

The graph used at parameter value no.1


Obtained at parameter value no.1 by algorithm no.1
True Partition
Best Partition
0
50
100
150
200
250
300
350
400
1
2
3
4
5
6
7
8
9
10

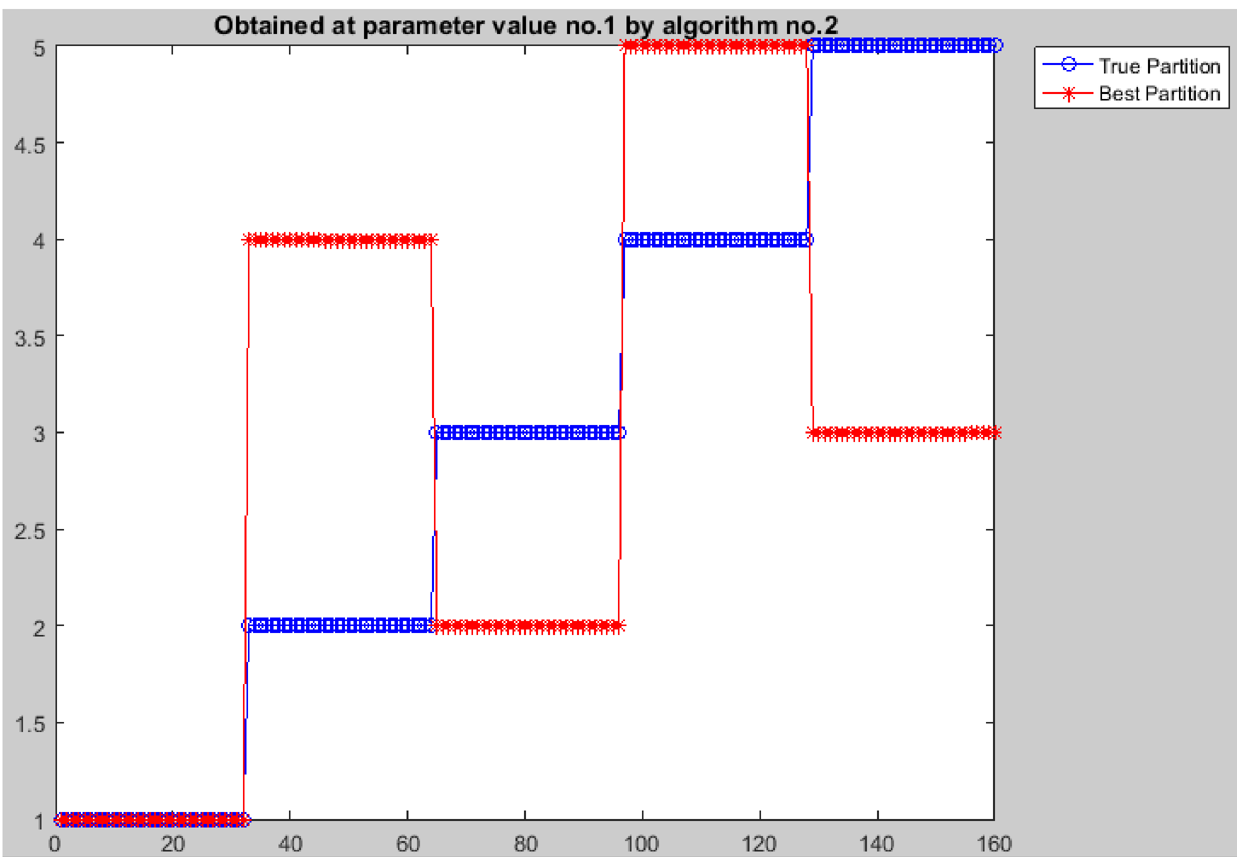
2) Ego-Twitter:





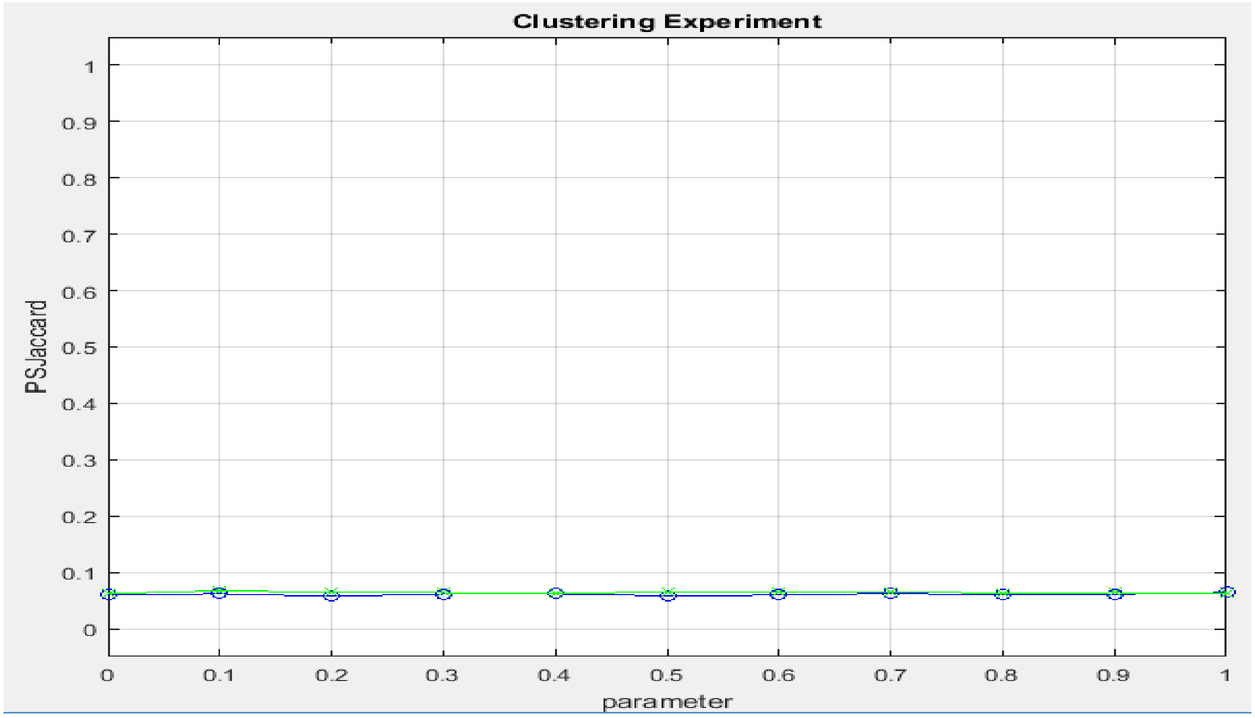


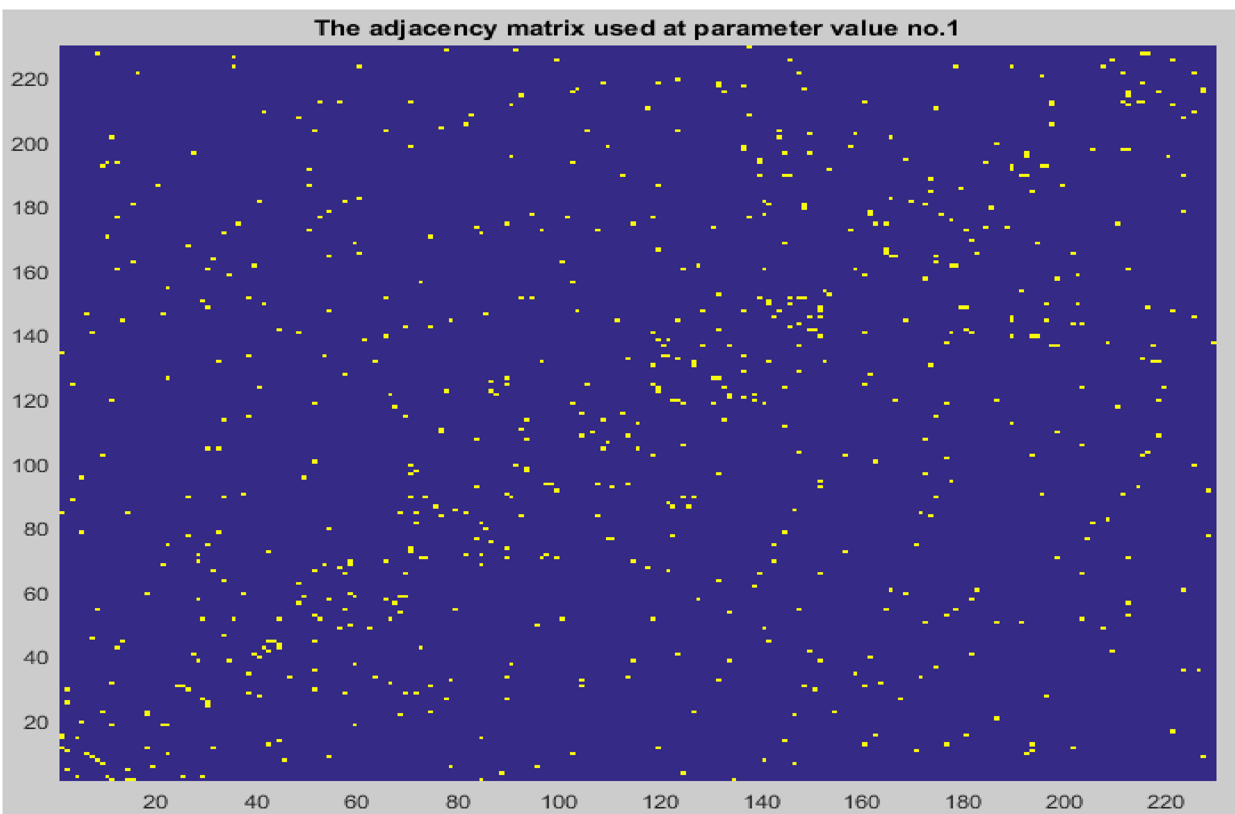
The graph used at parameter value no.2

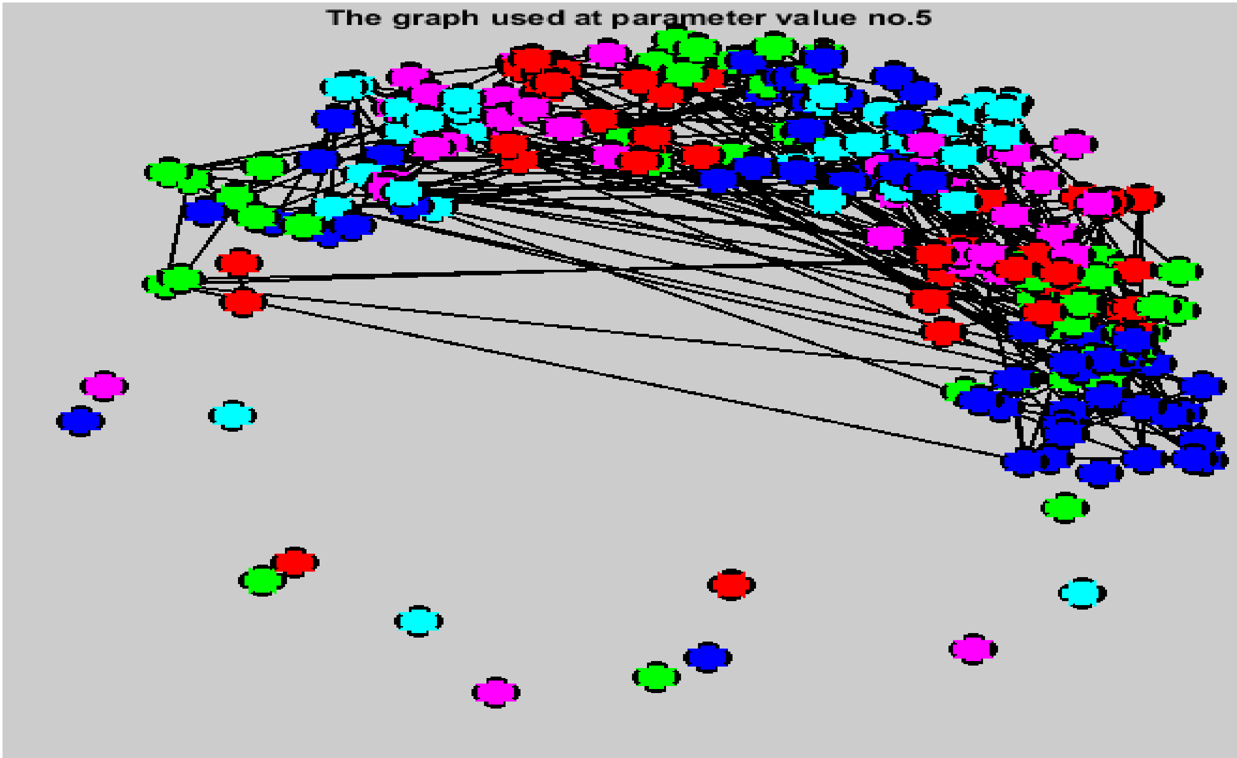


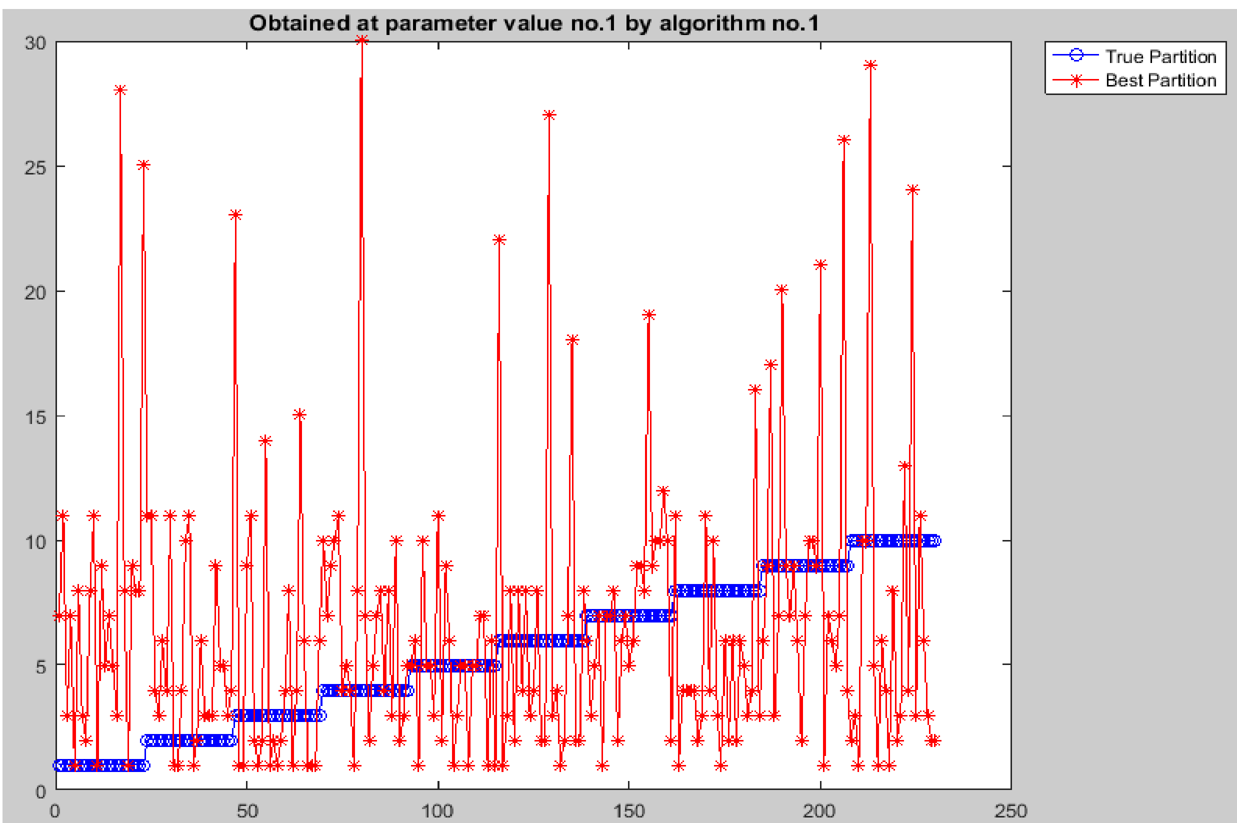
Obtained at parameter value no.1 by algorithm no.2
True Partition
Best Partition
5
4.5
4
3.5
3
2.5
2
1.5
1
0
20
40
60
80
100
120
140
160

3) Wiki-Vote:

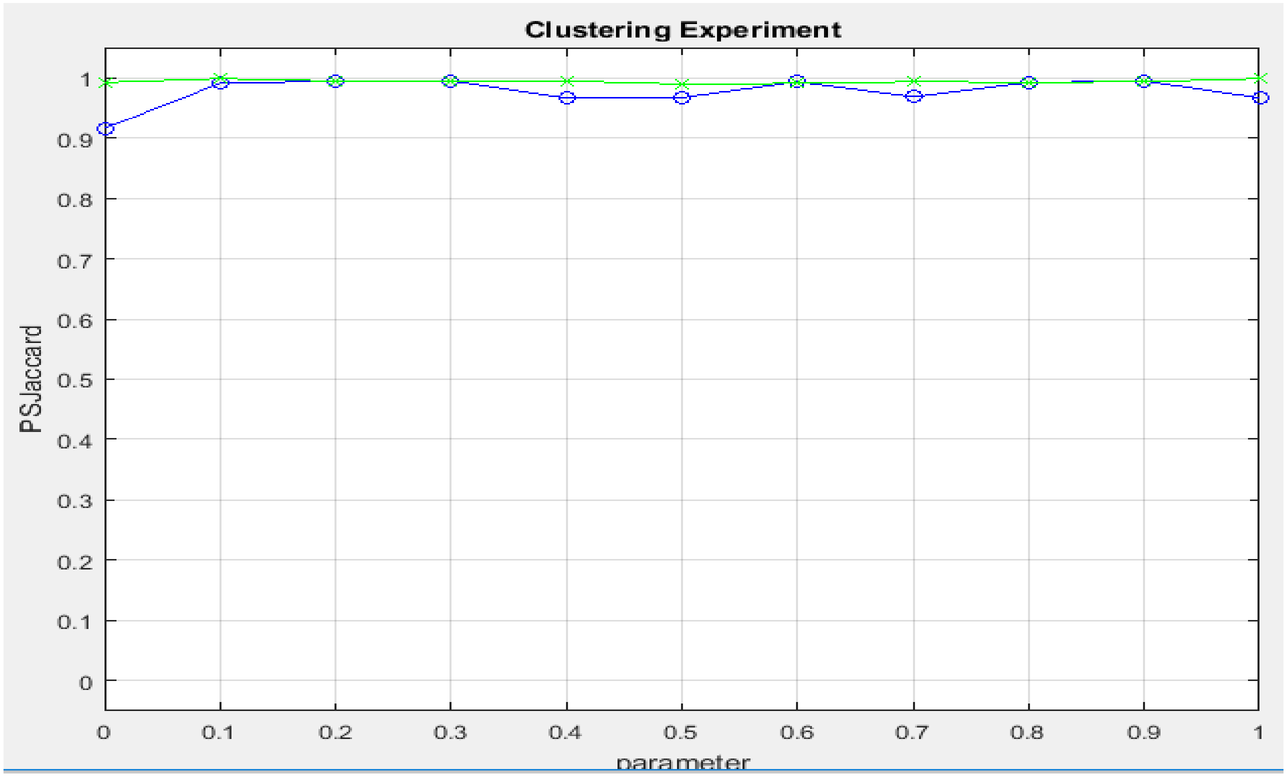

Clustering Experiment
PSJaccard
parameter
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

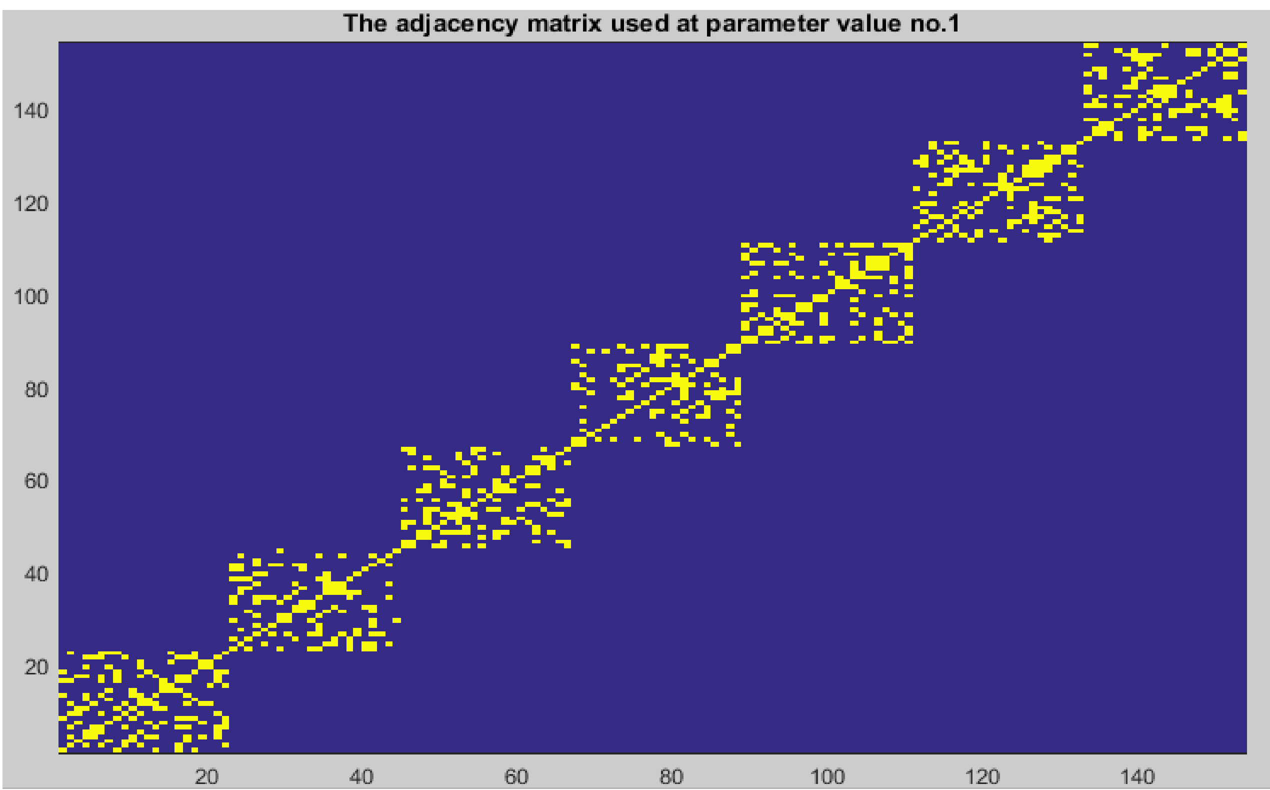

The adjacency matrix used at parameter value no.1
220
200
180
160
140
120
100
80
60
40
20
20
40
60
80
100
120
140
160
180
200
220

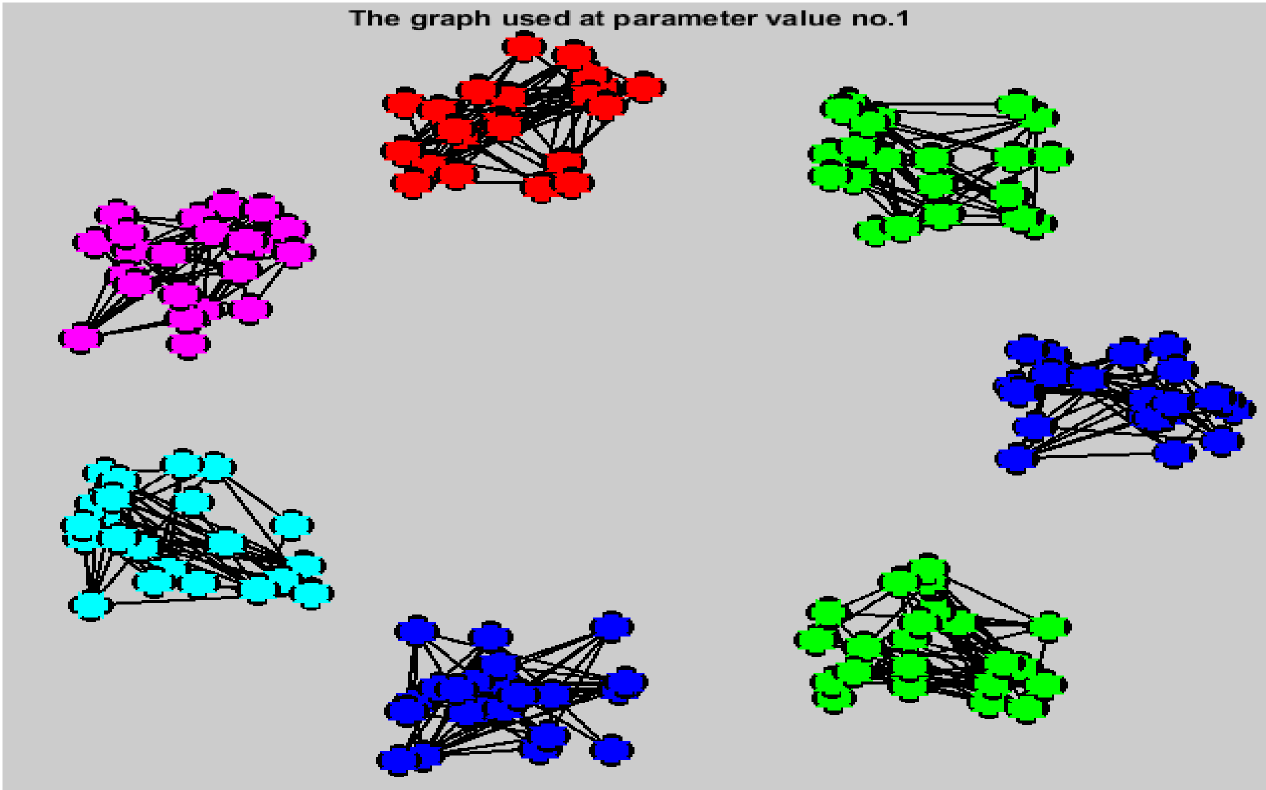

The graph used at parameter value no.5


Obtained at parameter value no.1 by algorithm no.1
True Partition
Best Partition
0
5
10
15
20
25
30
0
50
100
150
200
250

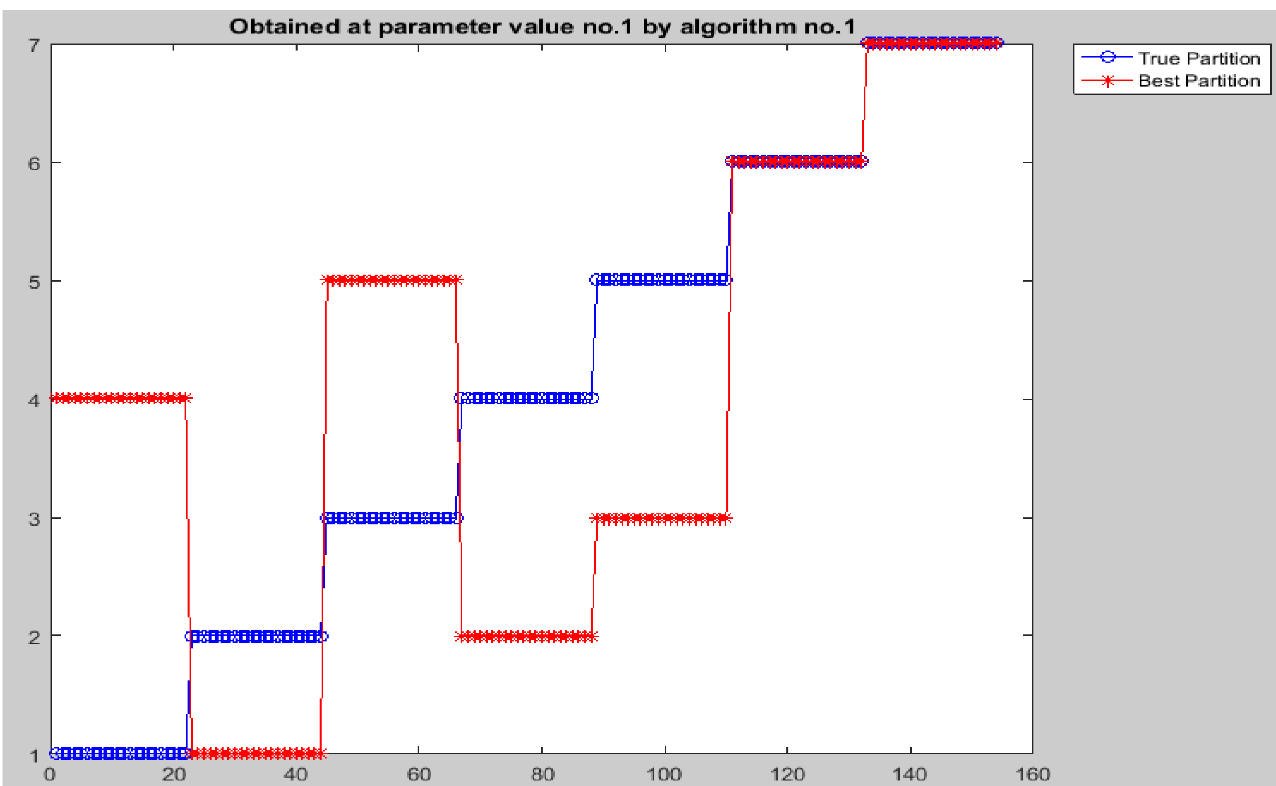
4) Soc-sign-itcoin-alpha:







The graph used at parameter value no.1



Obtained at parameter value no.1 by algorithm no.1
True Partition
Best Partition
1
2
3
4
5
6
7
0
20
40
60
80
100
120
140
160

Conclusion

In this project, I have presented an empirical study of Newman-Girvan algorithm on various data sets. My results differ from those presented earlier in the sense. The main drawback of Newman-Girvan algorithm is the absence of a clear specification on the definition of what constitutes a community.

References

[1] "An Empirical Study of Community and Sub-Community Detection in Social Networks Applying Newman-Girvan Algorithm" *By Deepjyoti Choudhury, Saprativa Bhattacharjee and Anirban Das.*

[2] "Community detection in networks" *By Dr. A.J. Schmidt-Hieber, S.L. van der Pas, MSc MA.*

[3] "Manual for the Community Detection Toolbox v. 0.9" *By M. Mitalidis, Ath. Kehagias, Th. Gevezes and L. Pitsoulis.*

[4] http://snap.stanford.edu/data/index.html

[5]https://www.google.com/search?q=undirected+graph&safe=active&hl=enFI&authuser=0&rlz=1C1GGRV_enFI815FI815&tbm=isch&source=iu&ictx=1&fir=p1MDxLvFM0MtEM%253A%252C3y2hS3DCMw48qM%252C_&vet=1&usg=AI4_kR6YU2KuJglsf3BXRDmyik9cpB61Q&sa=X&sqi=2&ved=2ahUKEwiTzuLc6dDhAhWdAhAIHUCEDLAQ9QEwAHoECA4QBg#imgrc=p1MDxLvFM0MtEM: