CS 532: Assignment 5

Dinesh Kumar Paladhi

Spring 2016

Contents

1 Problem 1

We know the result of the Karate Club (Zachary, 1977) split. Prove or disprove that the result of split could have been predicted by the weighted graph of social interactions. How well does the mathematical model represent reality?

Generously document your answer with all supporting equations, code, graphs, arguments, etc.

Useful sources include:

* Original paper

http://aris.ss.uci.edu/~lin/76.pdf

* Slides

http://www-personal.umich.edu/~ladamic/courses/networks/si614w06/ppt/lecture18.ppt

http://clair.si.umich.edu/si767/papers/Week03/Community/CommunityDetection.pptx

* Code and data

http://networkx.github.io/documentation/latest/examples/graph/karate_club.html

http://nbviewer.ipython.org/url/courses.cit.cornell.edu/info6010/resources/11notes.ipynb

http://stackoverflow.com/questions/9471906/what-are-the-differences-between-community-detection-algorithms-in-igraph/9478989#9478989

http://stackoverflow.com/questions/5822265/are-there-implementations-of-algorithms-for-community-detection-in-graphs

http://konect.uni-koblenz.de/networks/ucidata-zachary

http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/ucidata.htm#zachary

https://snap.stanford.edu/snappy/doc/reference/CommunityGirvanNewman.html

http://igraph.org/python/doc/igraph-pysrc.html#Graph.community_edge_betweenness

1.1 Solution

1. Karate club got split into two groups after the dispute.
2. I now implemented Girvan-Newman Algorithm which uses edge betweenness to split a group into sub groups.
3. Edge betweenness is defined as the number of times that a particular edge is used by every node to connect all other nodes.
4. Now using the concept of Girvan-Newman Algorithm, I need to find out the maximum edge betweenness among all the edges.
5. The edge that has the maximum edge betweenness is deleted.
6. This process is continued till a new group or cluster is formed.

7. Now In order to do this I got the karate club data in GraphML format. This can be seen in figure1.
8. In order to plot the graph and calculate the edge betweenness, I found out that a library called "igraph" can be used.
9. I tried to install igraph library in my laptop which has windows operating system, but it did not work out.
10. So, I went on to install igraph on mac but I got an error with "cairo" library.
11. Later on I figured out that it was working fine in ubuntu.
12. I wrote a python code to calculate edge with maximum edge betweenness and deleted it each time in a loop. Code can be seen in listing1
13. This was continued until I got two separate clusters.
14. Graph plotted before splitting can be seen in figure2.
15. The graph after the split can be seen in figure4.
16. I also printed all the nodes in each cluster and number of deletions required to form 2 clusters which can be seen in figure3.

1.2 Code Listing

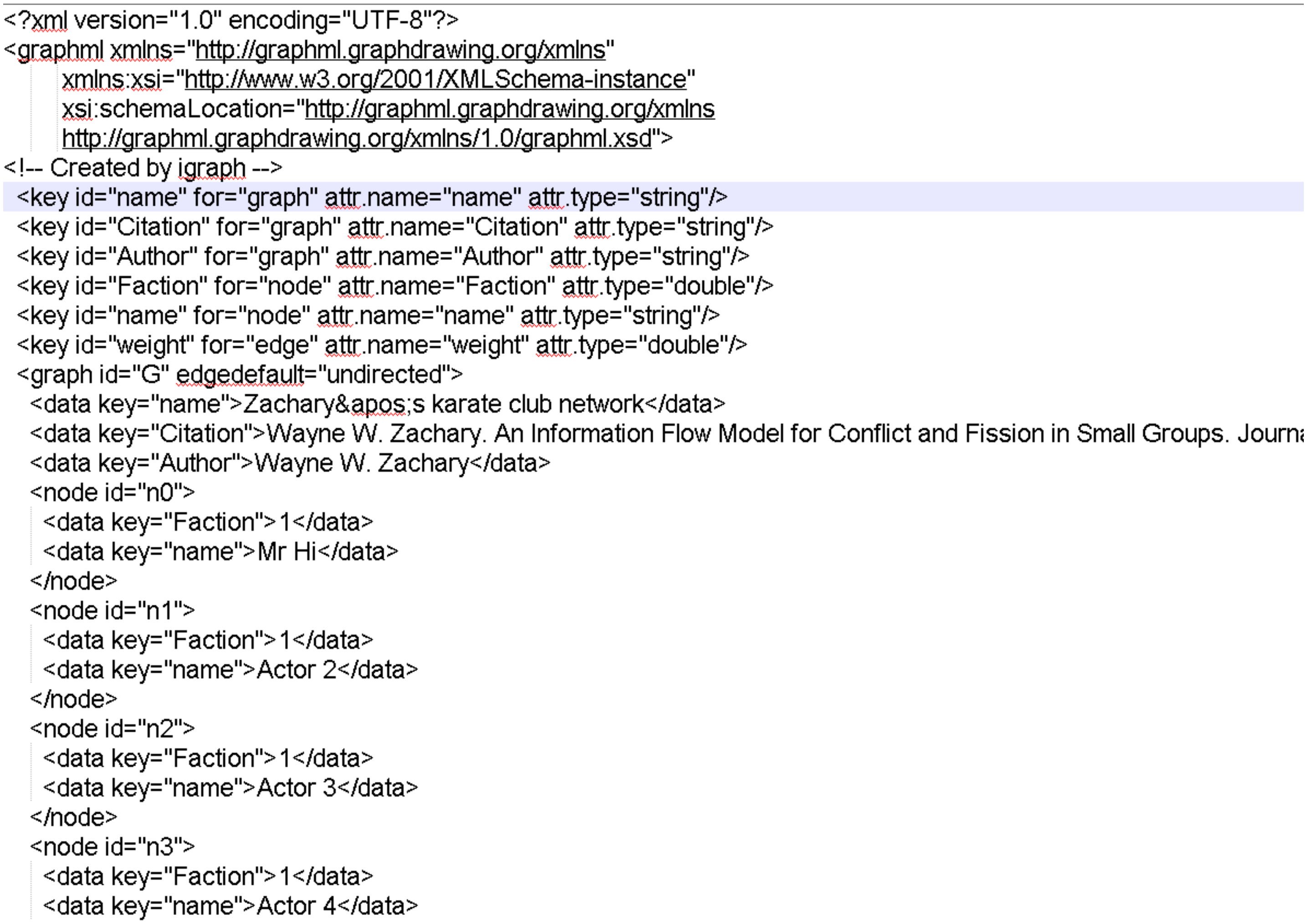
Here is the Python program for dividing 1 cluster into 2

```
from igraph import *
#visual_style = {}
#with open('karate.GraphML','r') as my_data :
#        data=my_data.read()
data=Graph.Read_GraphML('karate.GraphML') #reading graphml data into variable data
#print data
data.vs["label"]=data.vs["name"] #giving names to all the nodes
layout = data.layout("grid_fr") #providing "grid_fr" layout for the graph
color = {1: "green", 2: "orange"}
plot(data, 'group1.png', layout = layout, vertex_color = [color[fact] for fact  in data.vs["Faction"]]) #plotting initial graph
length =len(data.clusters())
count=0
#print length
while length <2 :
        find_eb= data.edge_betweenness()
        #print find_eb
        #max_eb=max(find_eb)
        #print max_eb
        #a=xrange(len(find_eb))
        #print a
        #key = find_eb.__getitem__
        #print key
        maximum_node = max(xrange(len(find_eb)), key = find_eb.__getitem__)
        #print maximum_node
        count=count+1
        data.delete_edges(maximum_node)

        length =len(data.clusters())
layout = data.layout("grid_fr") #providing "grid_fr" layout for the graph
print "Number of edges deleted for getting 2 clusters are : ",+count
print data.clusters()
plot(data, 'output1.png', layout = layout)
```

Listing 1: Python code

1.3 Results

Sample Karate GraphML file given as input to the program



```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<!-- Created by igraph -->
  <key id="name" for="graph" attr.name="name" attr.type="string"/>
  <key id="Citation" for="graph" attr.name="Citation" attr.type="string"/>
  <key id="Author" for="graph" attr.name="Author" attr.type="string"/>
  <key id="Faction" for="node" attr.name="Faction" attr.type="double"/>
  <key id="name" for="node" attr.name="name" attr.type="string"/>
  <key id="weight" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <data key="name">Zachary&apos;s karate club network</data>
    <data key="Citation">Wayne W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. Journa
    <data key="Author">Wayne W. Zachary</data>
    <node id="n0">
      <data key="Faction">1</data>
      <data key="name">Mr Hi</data>
    </node>
    <node id="n1">
      <data key="Faction">1</data>
      <data key="name">Actor 2</data>
    </node>
    <node id="n2">
      <data key="Faction">1</data>
      <data key="name">Actor 3</data>
    </node>
    <node id="n3">
      <data key="Faction">1</data>
      <data key="name">Actor 4</data>
```

Figure 1: Sample Karate GraphML file

Graph before splitting into 2 clusters

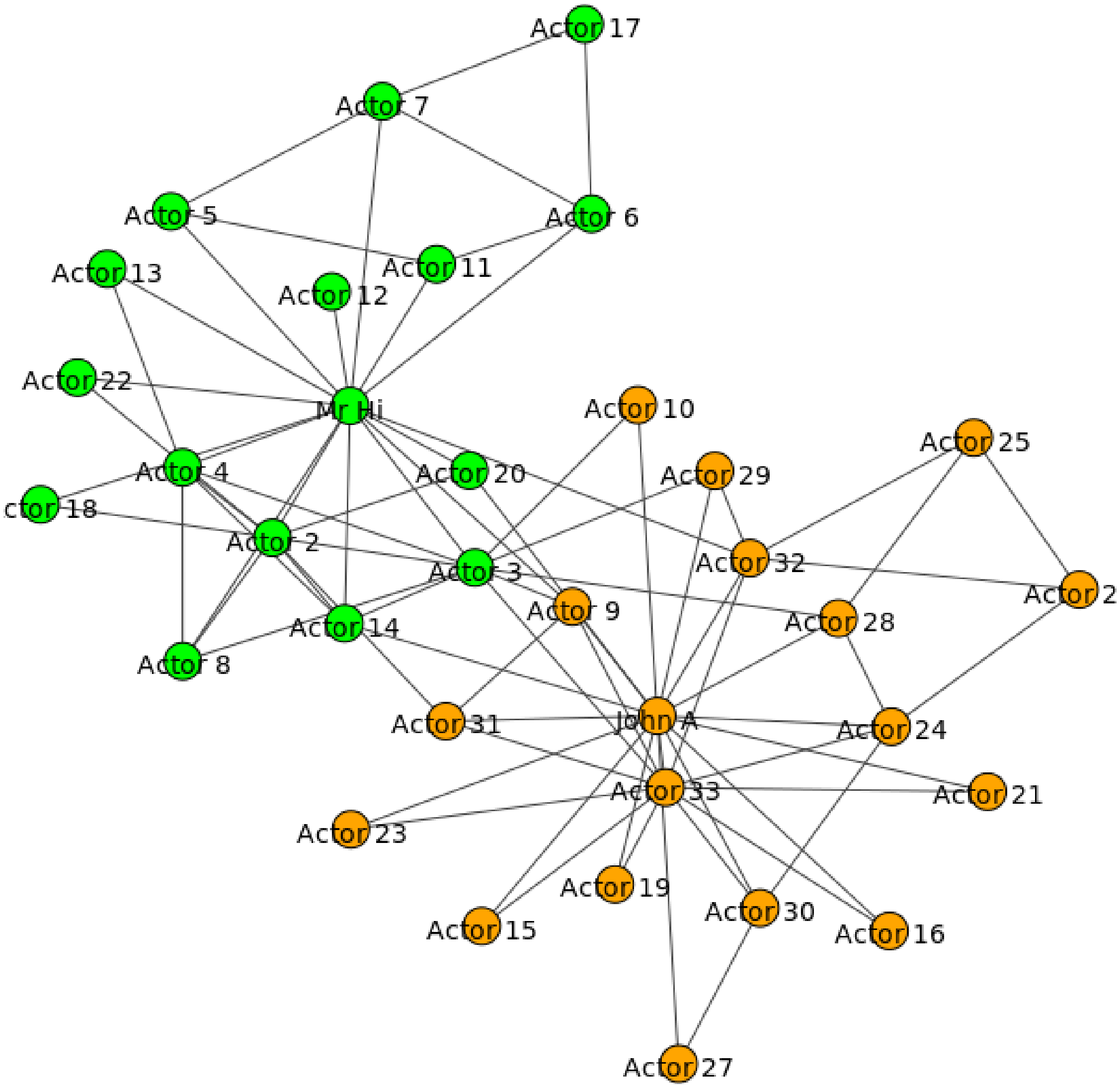
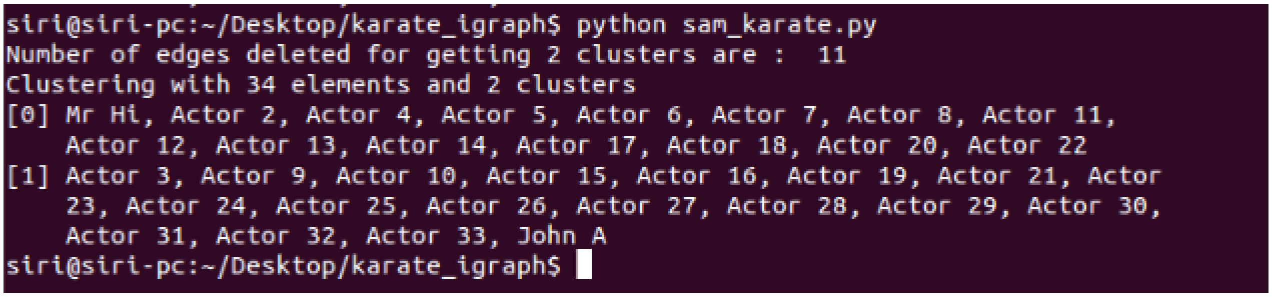


Figure 2: Original Graph

Command line output of nodes in each cluster and number of edges deleted to get 2 clusters



```
siri@siri-pc:~/Desktop/karate_igraph$ python sam_karate.py
Number of edges deleted for getting 2 clusters are :  11
Clustering with 34 elements and 2 clusters
[0] Mr Hi, Actor 2, Actor 4, Actor 5, Actor 6, Actor 7, Actor 8, Actor 11,
    Actor 12, Actor 13, Actor 14, Actor 17, Actor 18, Actor 20, Actor 22
[1] Actor 3, Actor 9, Actor 10, Actor 15, Actor 16, Actor 19, Actor 21, Actor
    23, Actor 24, Actor 25, Actor 26, Actor 27, Actor 28, Actor 29, Actor 30,
    Actor 31, Actor 32, Actor 33, John A
siri@siri-pc:~/Desktop/karate_igraph$ 
```

Figure 3: Command line output

Graph after splitting into 2 clusters

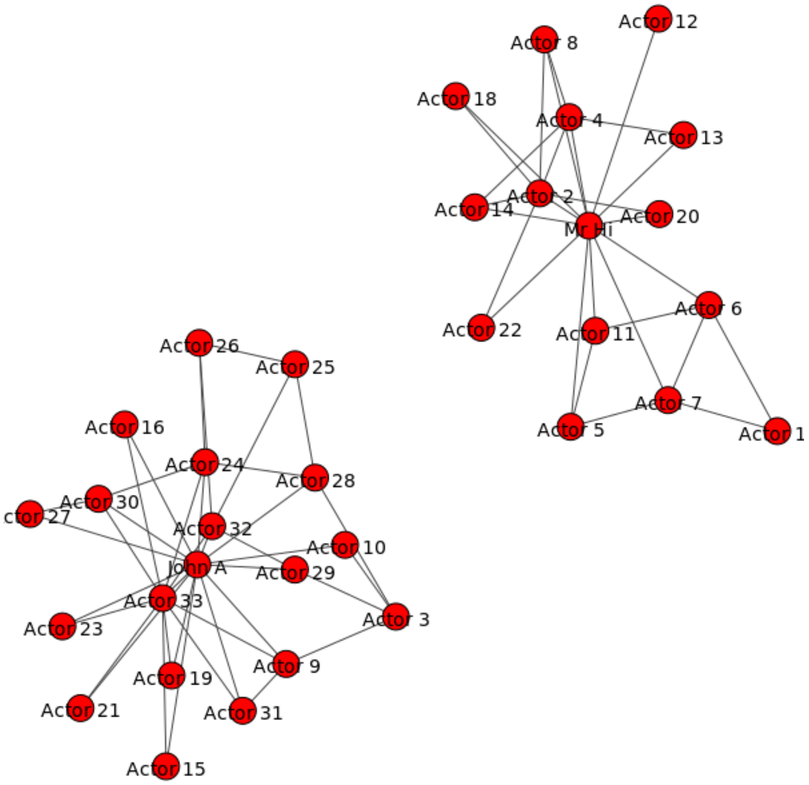


Figure 4: Predicted Graph

1.4 Conclusion

1. The initial graph which can be seen in figure2 is represented in 2 colors.
2. All the actors in green belong to Mr. Hi group and all the actors in orange belong to John's group.
3. This graph is drawn based on the "Faction" key in the graphML which groups all the actors into Mr. Hi's group or John's group.
4. Output graph of 2 clusters can be seen here in figure4.
5. Now when I compare both of these, we can observe that "Actor 3" is in Mr. Hi's group, this is according to the Faction key which is the original graph.
6. Whereas, In the predicted graph that we got by implementing the Girvan-Newman Algorithm, "Actor 3" is in John's group.
7. So, Except Actor 3 all other predictions are correct and this proves that the predicted graph is highly accurate.
8. Finally, this solution tells us that Girvan-Newman Algorithm produces predicted graphs accurately.

2 Problem 2

```
(extra credit, 3 points)

We know the group split in two different groups.  Suppose the
disagreements in the group were more nuanced -- what would the clubs
look like if they split into groups of 3, 4, and 5?
```

2.1 Solution

1. This is almost similar to the solution of question 1, the only difference is to get 3, 4 and 5 clusters rather than 2.
2. The initial graph can be seen in figure6 and the code can be seen in listing2
3. The graphs with 3, 4 and 5 clusters can be seen in figure8, figure9, figure10

2.2 Code Listing

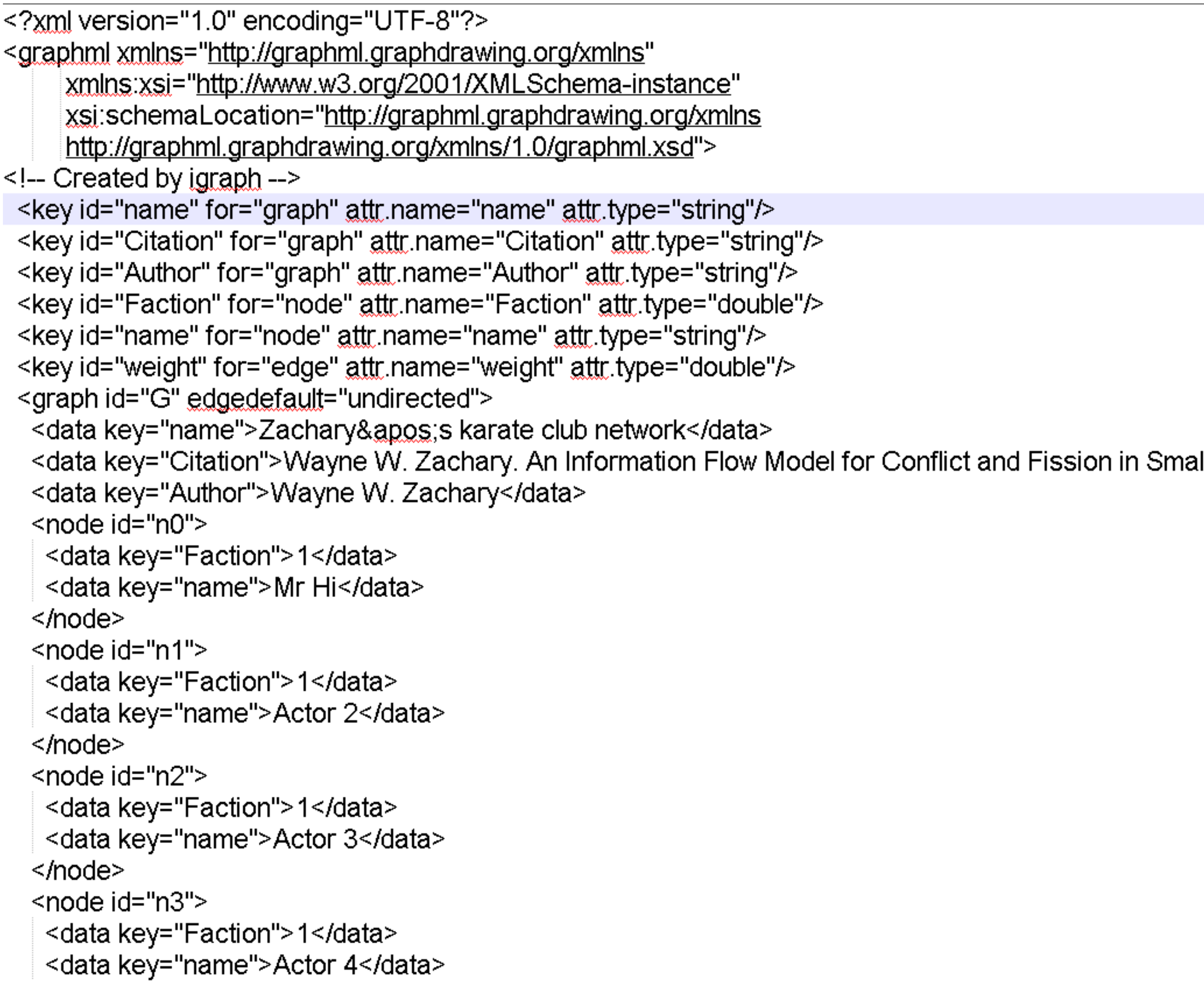
Here is the Python program for dividing 1 cluster into 3,4,5 clusters consecutively

```
from igraph import *
#visual_style = {}
#with open('karate.GraphML','r') as my_data :
#        data=my_data.read()
data=Graph.Read_GraphML('karate.GraphML') #reading graphml data into variable data
#print data
data.vs["label"]=data.vs["name"] #giving names to all the nodes
layout = data.layout("grid_fr") #providing "grid_fr" layout for the graph
plot(data, 'group2.png', layout = layout) #plotting initial graph
length =len(data.clusters())
count=0
flag1=0
flag = 0
#print length

while length <5 :
        layout = data.layout("grid_fr")
        if (length == 3):
                flag=flag +1
                plot(data, 'output2.png', layout = layout)
                if flag == 4 :
                        print data.clusters()
                count1=count
        elif(length == 4) :
                count2=count
                flag1=flag1+1
                plot(data, 'output3.png', layout = layout)
                if flag1 == 5 :
                        print data.clusters()
        find_eb= data.edge_betweenness()
        #print find_eb
        #max_eb=max(find_eb)
        #print max_eb
        #a=xrange(len(find_eb))
        #print a
        #key = find_eb.__getitem__
        #print key
        maximum_node = max(xrange(len(find_eb)), key = find_eb.__getitem__)
        #print maximum_node
        count=count+1

        data.delete_edges(maximum_node)
        length =len(data.clusters())
plot(data, 'output4.png', layout = layout)
print data.clusters()
print "Number of edges deleted for getting 3 clusters are : ",+count1
print "Number of edsge deleted for getting 4 clusters are : ",+count2
print "Number of edges deleted for getting 5 clusters are : ",+count
```

Listing 2: Python code

2.3 Results

Sample Karate GraphML file given as input to the program



```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<!-- Created by igraph -->
  <key id="name" for="graph" attr.name="name" attr.type="string"/>
  <key id="Citation" for="graph" attr.name="Citation" attr.type="string"/>
  <key id="Author" for="graph" attr.name="Author" attr.type="string"/>
  <key id="Faction" for="node" attr.name="Faction" attr.type="double"/>
  <key id="name" for="node" attr.name="name" attr.type="string"/>
  <key id="weight" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <data key="name">Zachary&apos;s karate club network</data>
    <data key="Citation">Wayne W. Zachary. An Information Flow Model for Conflict and Fission in Smal
    <data key="Author">Wayne W. Zachary</data>
    <node id="n0">
      <data key="Faction">1</data>
      <data key="name">Mr Hi</data>
    </node>
    <node id="n1">
      <data key="Faction">1</data>
      <data key="name">Actor 2</data>
    </node>
    <node id="n2">
      <data key="Faction">1</data>
      <data key="name">Actor 3</data>
    </node>
    <node id="n3">
      <data key="Faction">1</data>
      <data key="name">Actor 4</data>
```

Figure 5: Sample GraphML file

Graph before splitting into 3/4/5 clusters

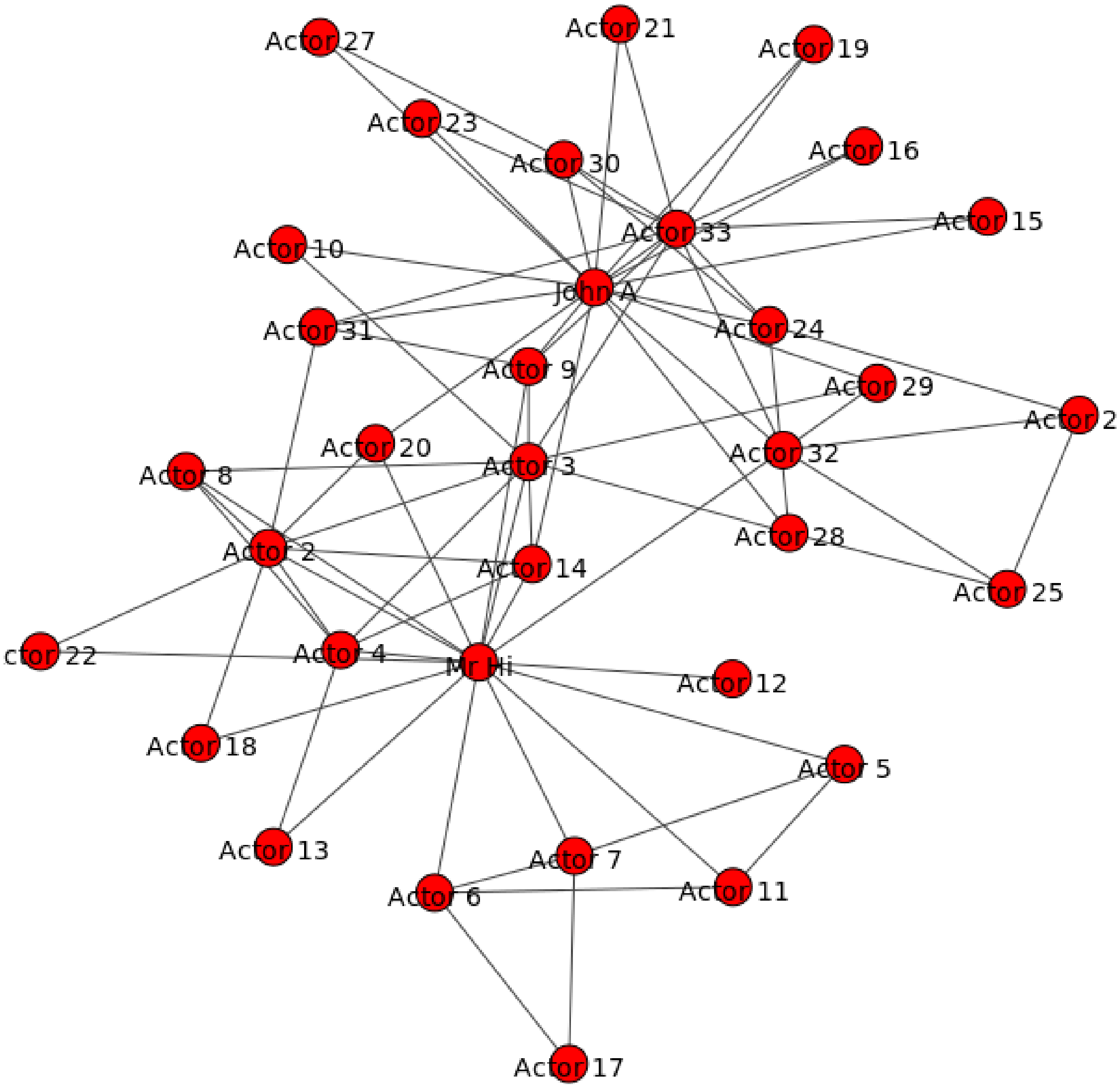
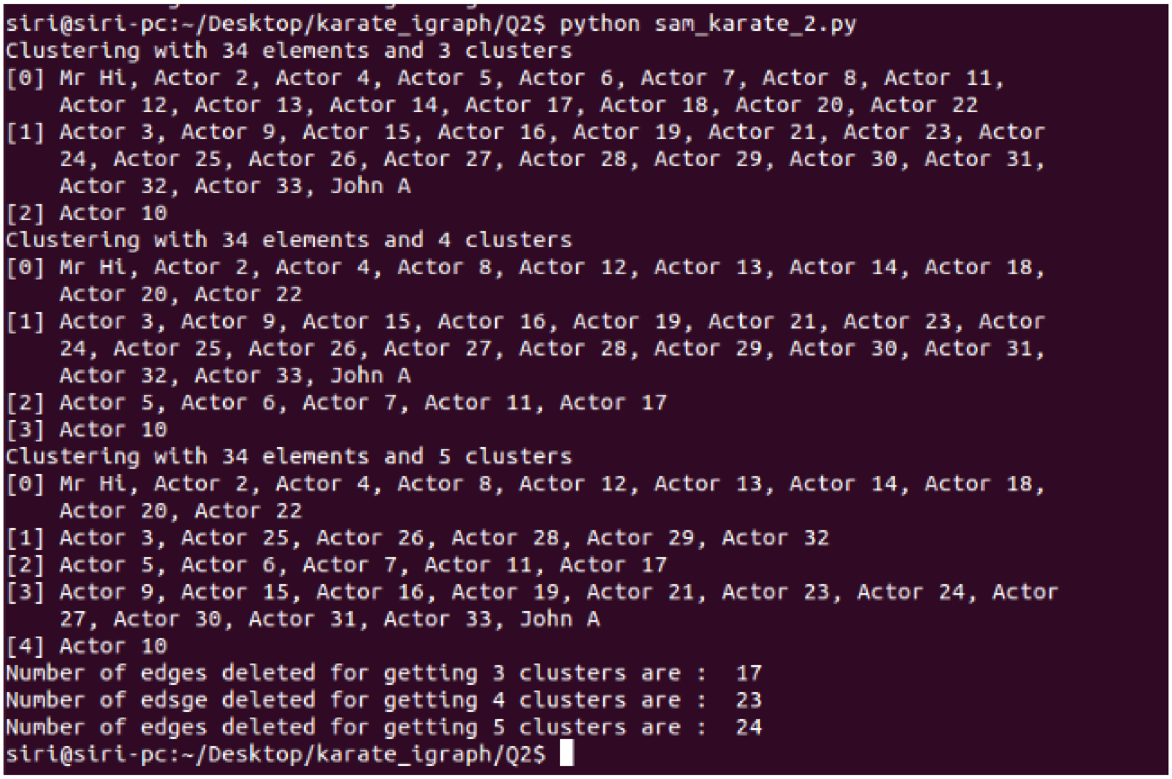


Figure 6: Original Graph

Command line output of nodes in each cluster and number of edges deleted to get 2 clusters



```
siri@siri-pc:~/Desktop/karate_igraph/Q2$ python sam_karate_2.py
Clustering with 34 elements and 3 clusters
[0] Mr Hi, Actor 2, Actor 4, Actor 5, Actor 6, Actor 7, Actor 8, Actor 11,
    Actor 12, Actor 13, Actor 14, Actor 17, Actor 18, Actor 20, Actor 22
[1] Actor 3, Actor 9, Actor 15, Actor 16, Actor 19, Actor 21, Actor 23, Actor
    24, Actor 25, Actor 26, Actor 27, Actor 28, Actor 29, Actor 30, Actor 31,
    Actor 32, Actor 33, John A
[2] Actor 10
Clustering with 34 elements and 4 clusters
[0] Mr Hi, Actor 2, Actor 4, Actor 8, Actor 12, Actor 13, Actor 14, Actor 18,
    Actor 20, Actor 22
[1] Actor 3, Actor 9, Actor 15, Actor 16, Actor 19, Actor 21, Actor 23, Actor
    24, Actor 25, Actor 26, Actor 27, Actor 28, Actor 29, Actor 30, Actor 31,
    Actor 32, Actor 33, John A
[2] Actor 5, Actor 6, Actor 7, Actor 11, Actor 17
[3] Actor 10
Clustering with 34 elements and 5 clusters
[0] Mr Hi, Actor 2, Actor 4, Actor 8, Actor 12, Actor 13, Actor 14, Actor 18,
    Actor 20, Actor 22
[1] Actor 3, Actor 25, Actor 26, Actor 28, Actor 29, Actor 32
[2] Actor 5, Actor 6, Actor 7, Actor 11, Actor 17
[3] Actor 9, Actor 15, Actor 16, Actor 19, Actor 21, Actor 23, Actor 24, Actor
    27, Actor 30, Actor 31, Actor 33, John A
[4] Actor 10
Number of edges deleted for getting 3 clusters are :  17
Number of edsge deleted for getting 4 clusters are :  23
Number of edges deleted for getting 5 clusters are :  24
siri@siri-pc:~/Desktop/karate_igraph/Q2$ 
```

Figure 7: Command line output

Graph after splitting into 3 clusters

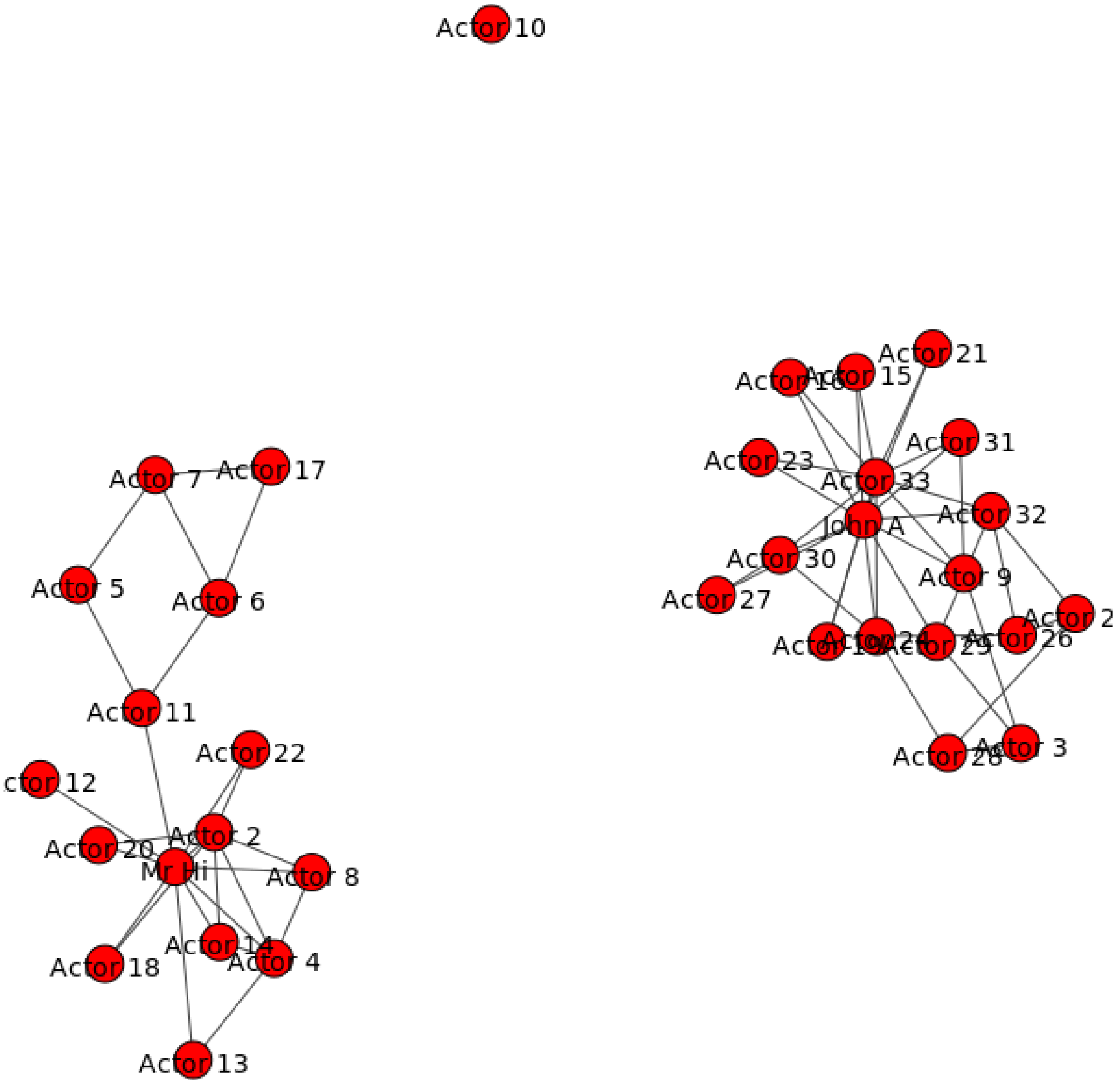


Figure 8: 3 clusters

Graph after splitting into 4 clusters

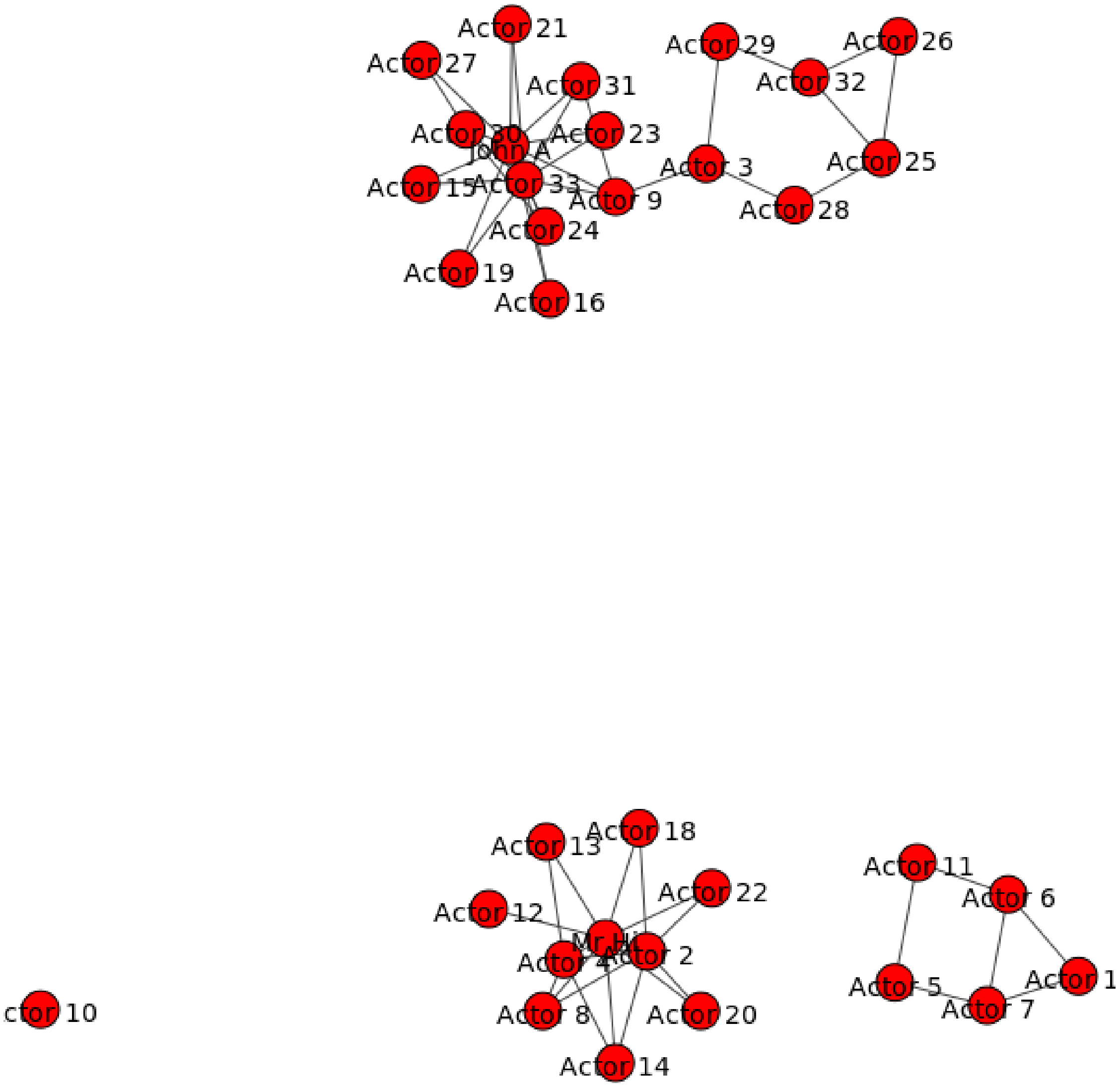


Figure 9: 4 clusters

Graph after splitting into 5 clusters

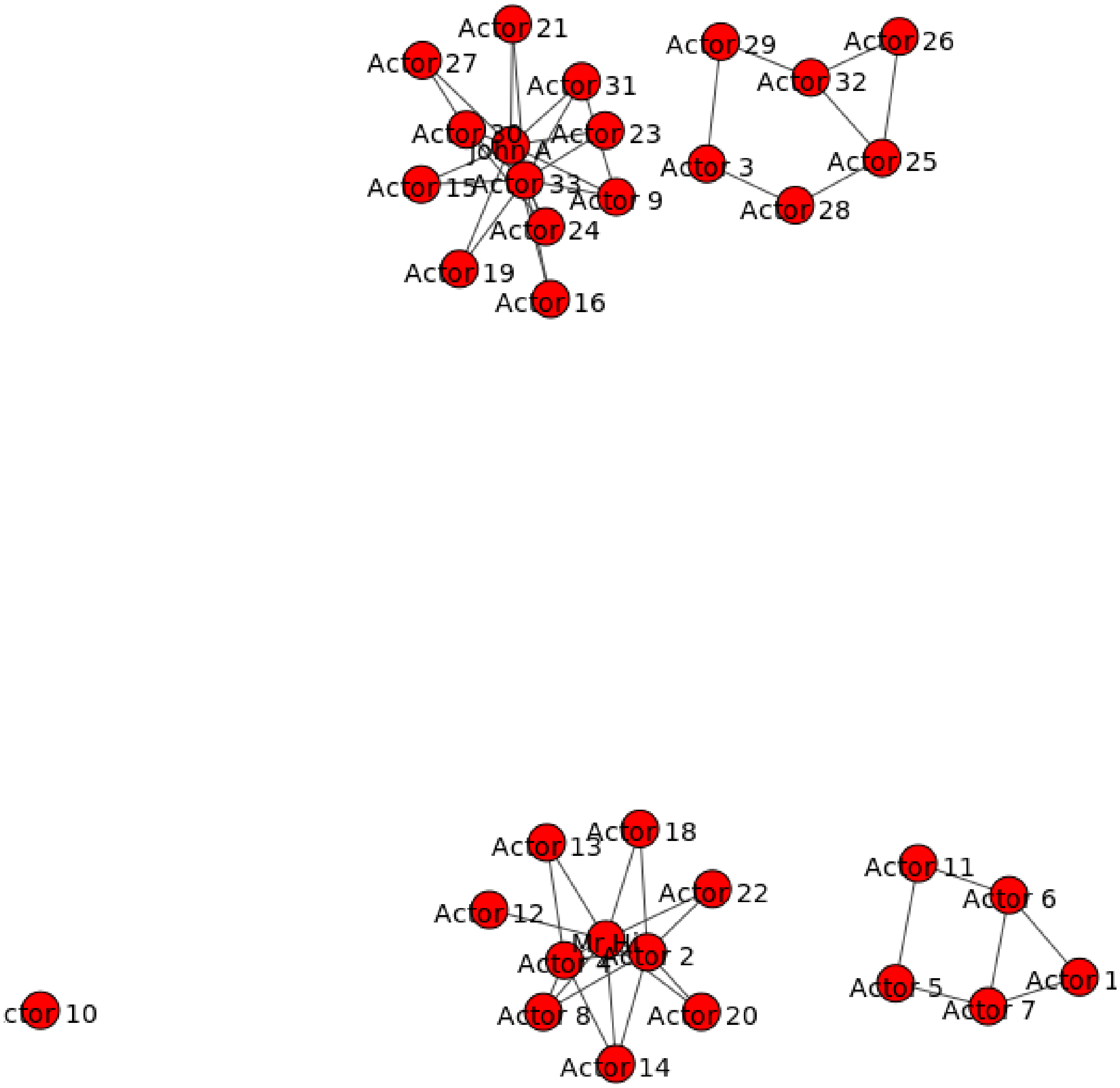


Figure 10: 5 clusters

Bibliography

[1] Python Software Foundation. Python built-in functions. https://docs.python.org/2/library/functions.html, 2016.

[2] The igraph core team. python-igraph manual. http://igraph.org/python/doc/tutorial/tutorial.html, 2015.

[3] The Nexus team. Nexus - zacharys karate club. http://nexus.igraph.org/api/dataset_info?id=1&format=html, 2011.

[4] Wayne W. Zachary. An information flow model for conflict and fission in small groups1. http://aris.ss.uci.edu/~lin/76.pdf, 1977.