INTRODUCTION TO WEB SCIENCES: Assignment 6

Babitha Bokka

1 November 2014

Contents

1 Question 1: Karate group split into 2 groups

Prove or disprove the result of the split could have been predicted by the weighted graph of social interactions. How well the mathematical model would represent the reality.

1.1 Approach

karate club splits into two after the dispute. So, we now compute a mathematical model using Girvan-Newman Algorithm which can split the groups into two by using the edge betweeness. In order to acheive this we take karate.GraphML file observe the node and edge weights. Based on the edge betweeness identify the edges to be removed.
According to Girvan-Newman Algorithm karateClubGraph.py calculate the edge betweness, finds the edge with maximum betweeness and deletes the edge.

1.2 Description of karateClubGraph.py

1. Load the karate.GraphML.
2. Label all the nodes.
3. Plot the graph to check how the graph is connected before the split.
4. Calculate the betweeness of the all the edges in the graph.
5. Find the edge with maximum betweeness.
6. Remove the edge with maximum betweeness.
7. Repeat the steps in a loop until the single cluster becomes two clusters.
8. Plot the graph with clusters.
9. Verify the output with the reality. You can manually draw the the graph for the karate.GraphML file and check whether the predicted graphs will represent the real graphs .

1.3 Source Code

1.3.1 karateClubGraph.py

```python
#!/usr/bin/env python

import sys
from igraph import *

def main():
    # Load the karate.GraphML and store that into a g
    g =load('karate.GraphML')
    # label the nodes (Key-value pairs)
    g.vs["label"] = g.vs["name"]
    #print len(g.clusters())
    # setting up the layout for the graph
    layout = g.layout("kk")
    # using plot() plot the graph and save that to pdf
    plot(g, 'graph.pdf', layout = layout)
    # break the single cluster into two cluster based on the betweeness
    while len(g.clusters()) ==1 :
        # get the betweeness and store
      ebs = g.edge_betweenness()
        # among all the betweness for  each node take the maximum select max
      max_eb = max(ebs)
        # get the node which has the maximum betweeness
      max_idx = max(xrange(len(ebs)), key = ebs.__getitem__)
        # delete the edges with max betweeness
      g.delete_edges(max_idx)
        # loop untill single cluster forms two clusters
    layout = g.layout("kk")
    # plot the graph with two clusters
    plot(g, 'graph.pdf' , layout = layout)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

1.4 Input

1.4.1 karate.GraphML

```
<?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. Journal of Anthropological Research Vol. 33, No. 4
    452-473</data>
    <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>
  <edge source="n0" target="n1">
      <data key="weight">4</data>
    </edge>
    <edge source="n0" target="n2">
      <data key="weight">5</data>
    </edge>
  <edge source="n0" target="n3">
      <data key="weight">3</data>
    </edge>
  </graph>
</graphml>
```

1.5 Output Files

Figure 1 is the initial graph generated from the karate.GraphML. If we observe the graph each node is connected to some other nodes but there are strong and weekly connected edges. Now, we can split the graph manually and know which person is in support of whom. Figure 2, Figure 3 represent the actual split of the group in reality. Figure 4 has two clusters formed by predicting the weighted graph.

1.5.1 Intial Graph

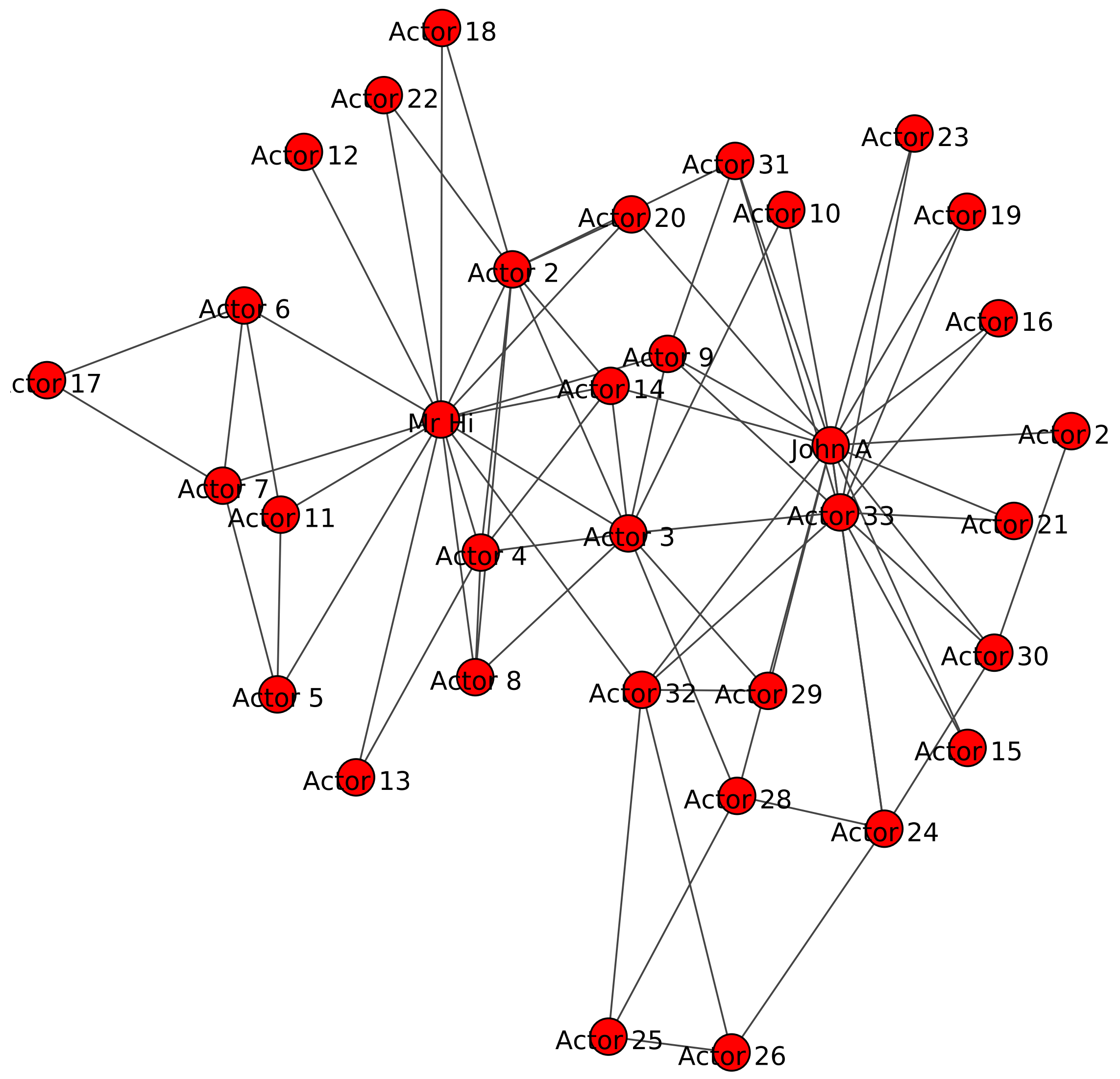

Figure 1: initial-graph

1.6 Graphs after the split

1.6.1 Graph split Reality

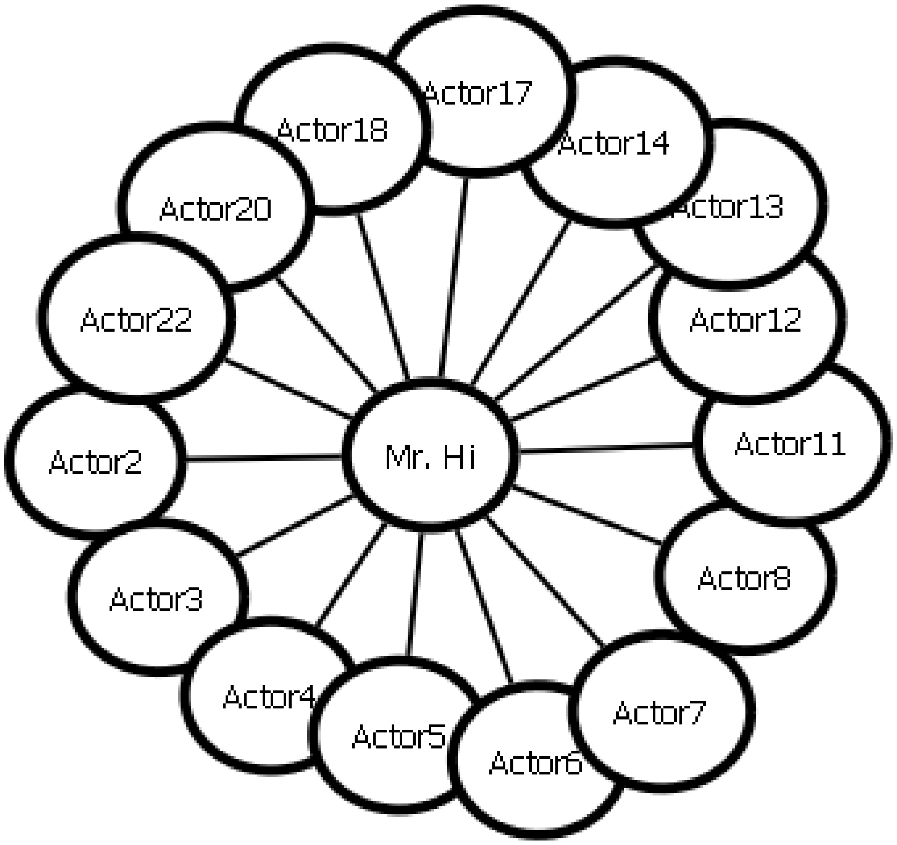

Figure 2: Reality 1

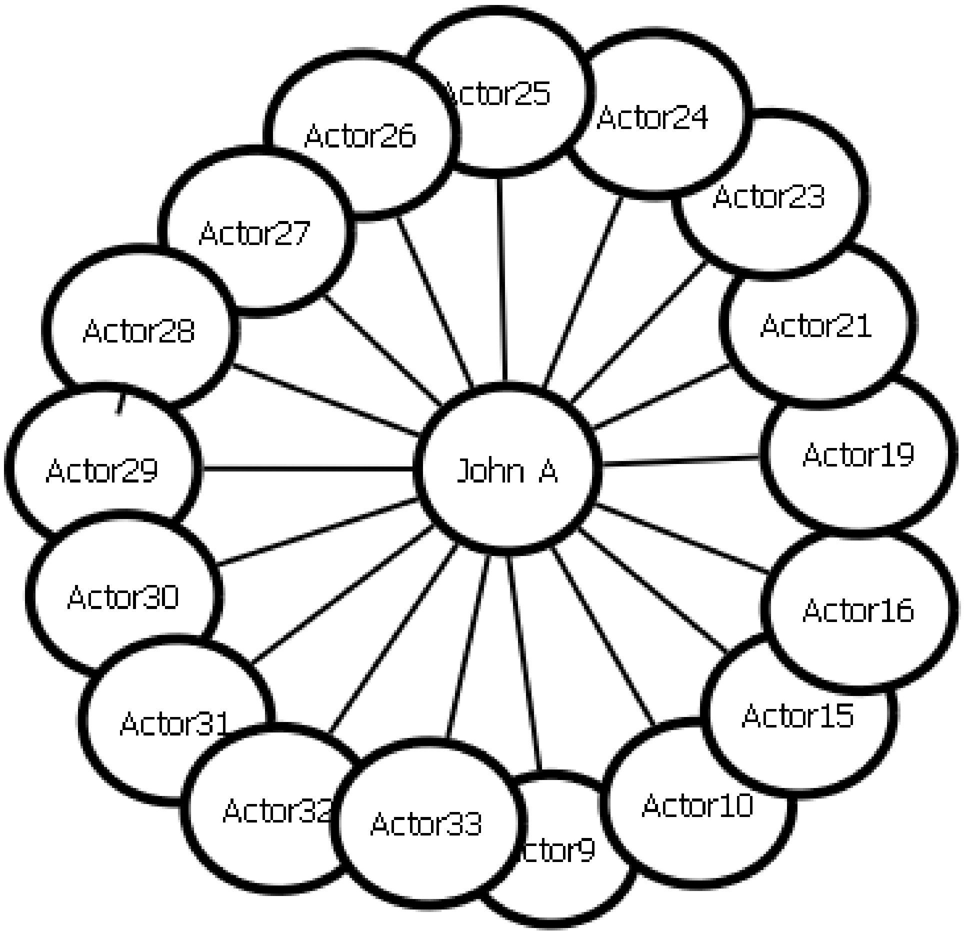

Figure 3: Reality 2

1.6.2 Predicted graph after split

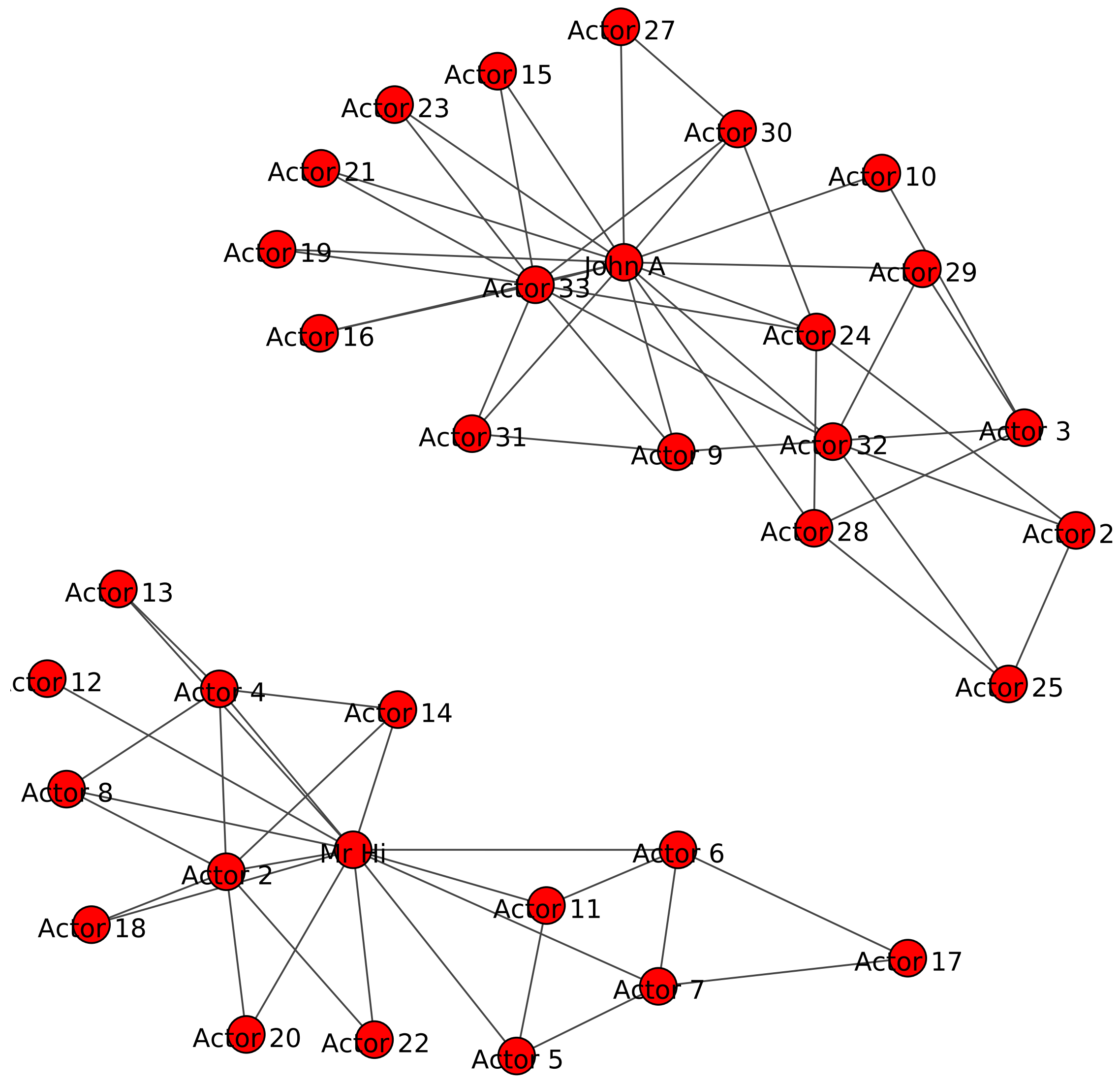

Figure 4: Graph after split

2 Question 2: Karate group split into 3, 4, and 5

We know that the karate club has been split into 2 groups but assume that there were many arguments in the group and it has split up into a numbers of 3, 4, and 5.

2.1 Approach

Approch to solve the problem is same as question 1 the only difference is changing the loop condition or writing a bunch of conditional statements.

2.2 Source Code

2.2.1 karateClubGraphSplit.py

```python
#!/usr/bin/env python

import sys
from igraph import *

def main():
    # Load the karate.GraphML and store that into a g
    g =load('karate.GraphML')
    # label the nodes (Key-value pairs)
    g.vs["label"] = g.vs["name"]
    #print len(g.clusters())
    # setting up the layout for the graph
    layout = g.layout("kk")
    # using plot() plot the graph and save that to pdf
    plot(g, 'graph.pdf', layout = layout)
    # break the single cluster into two cluster based on the betweeness
    # this can work for any number of split based on condition < 4 < 5 :
    while len(g.clusters()) < 3 :
        # get the betweeness and store
        ebs = g.edge_betweenness()
        # among all the betweness for  each node take the maximum select max
        max_eb = max(ebs)
        # get the node which has the maximum betweeness
        max_idx = max(xrange(len(ebs)), key = ebs.__getitem__)
        # delete the edges with max betweeness
        g.delete_edges(max_idx)
        # loop untill single cluster forms two clusters
    layout = g.layout("kk")
    # plot the graph with two clusters
    plot(g, 'graph.pdf' , layout = layout)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

2.3 Input

2.3.1 karate.GraphML

```
<?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. Journal of Anthropological Research Vol. 33, No. 4
       452-473</data>
    <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>
  <edge source="n0" target="n1">
      <data key="weight">4</data>
    </edge>
    <edge source="n0" target="n2">
      <data key="weight">5</data>
    </edge>
  <edge source="n0" target="n3">
      <data key="weight">3</data>
    </edge>
  </graph>
</graphml>
```

2.4 Output Files

2.4.1 Intial-graph

Figure 5 is the initial graph. According to the requirement the graph can be split into any number of groups just by changing a looping condition in the mathematical model. Figure 6, has three clusters, Figure 7 has four clusters, Figure 8 has four clusters formed after splitting the graph with one cluster.

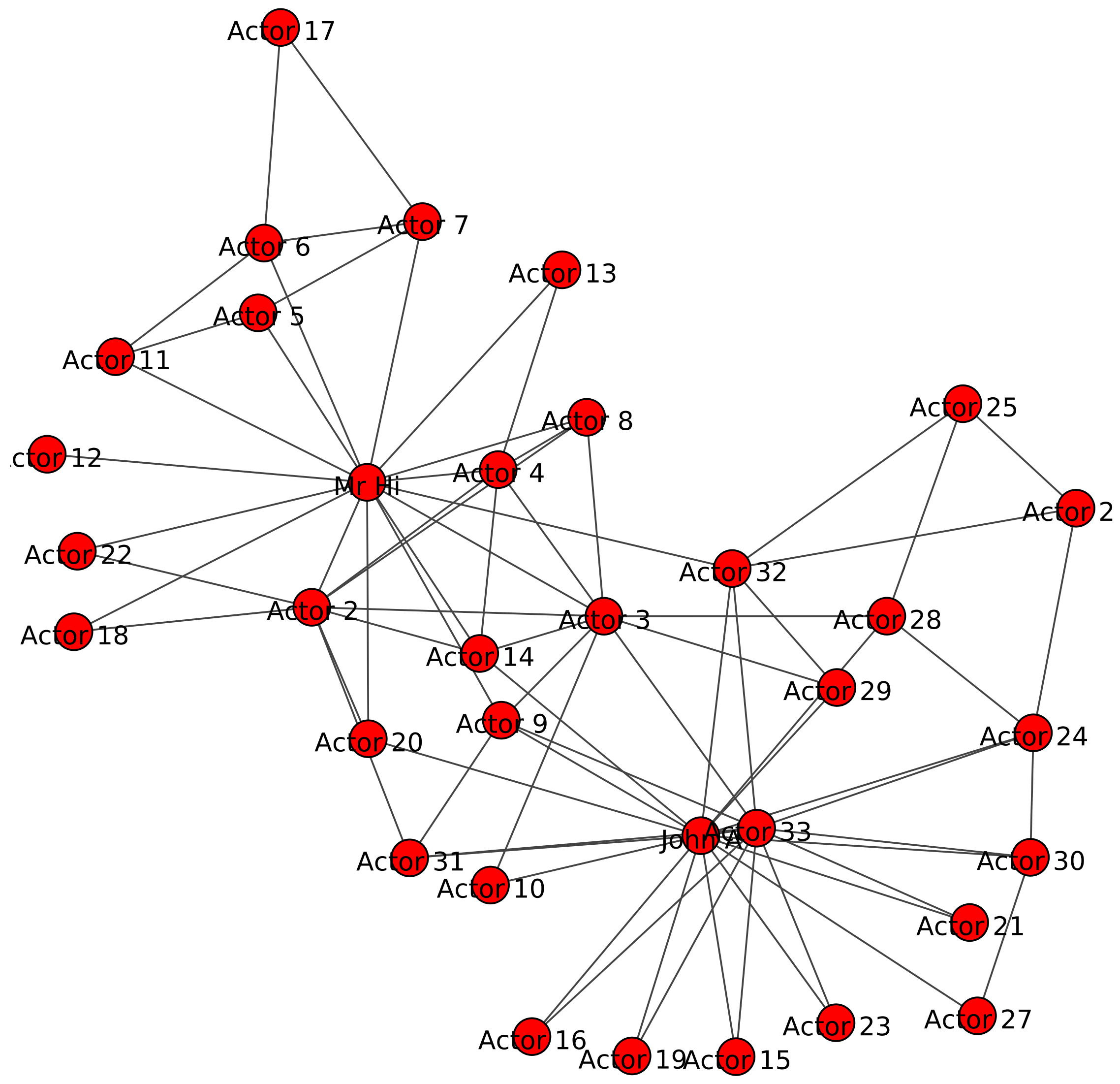

Figure 5: Initial-graph

2.4.2 Graph after the split

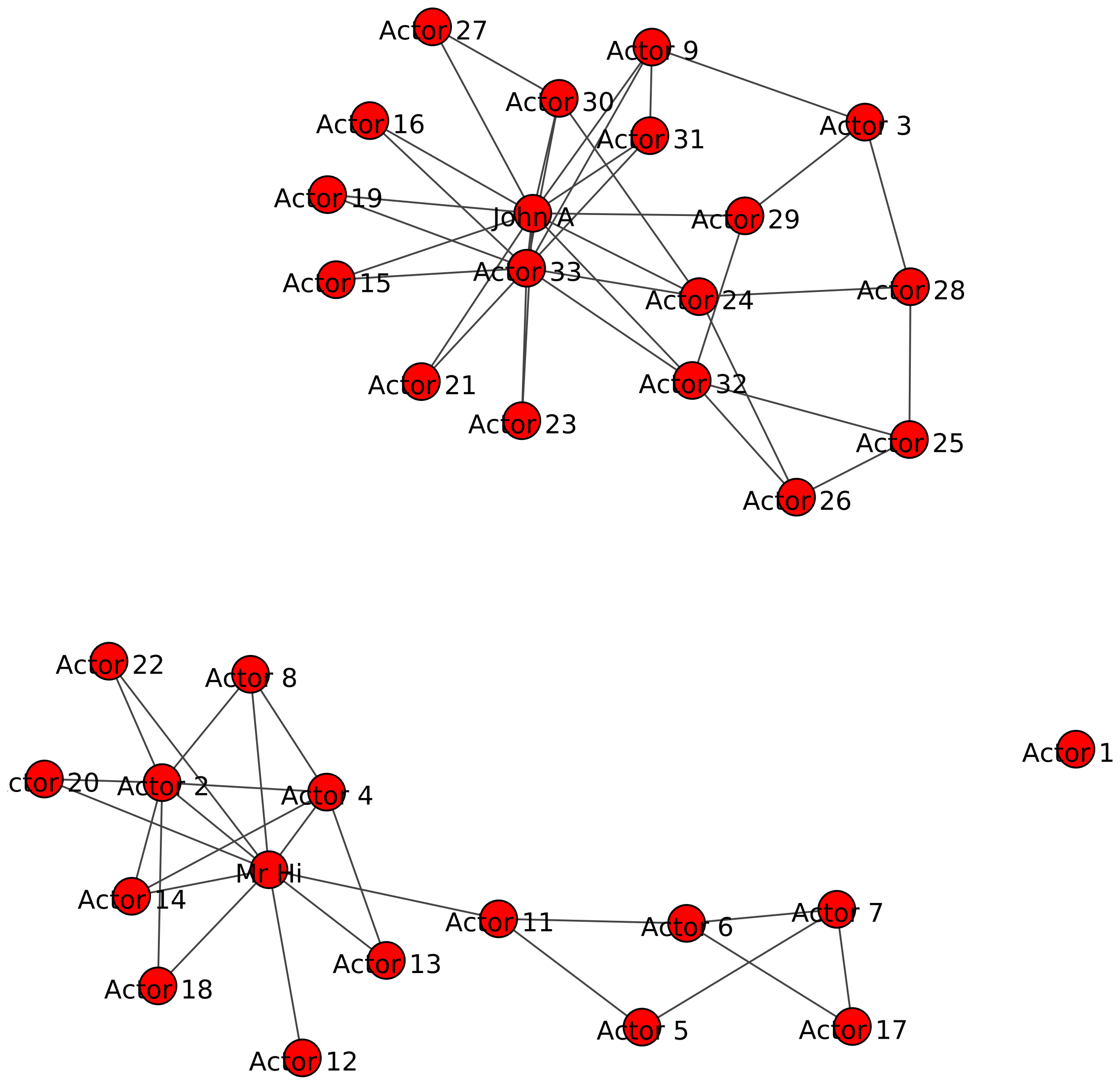

Figure 6: Initial graph split into 3 groups

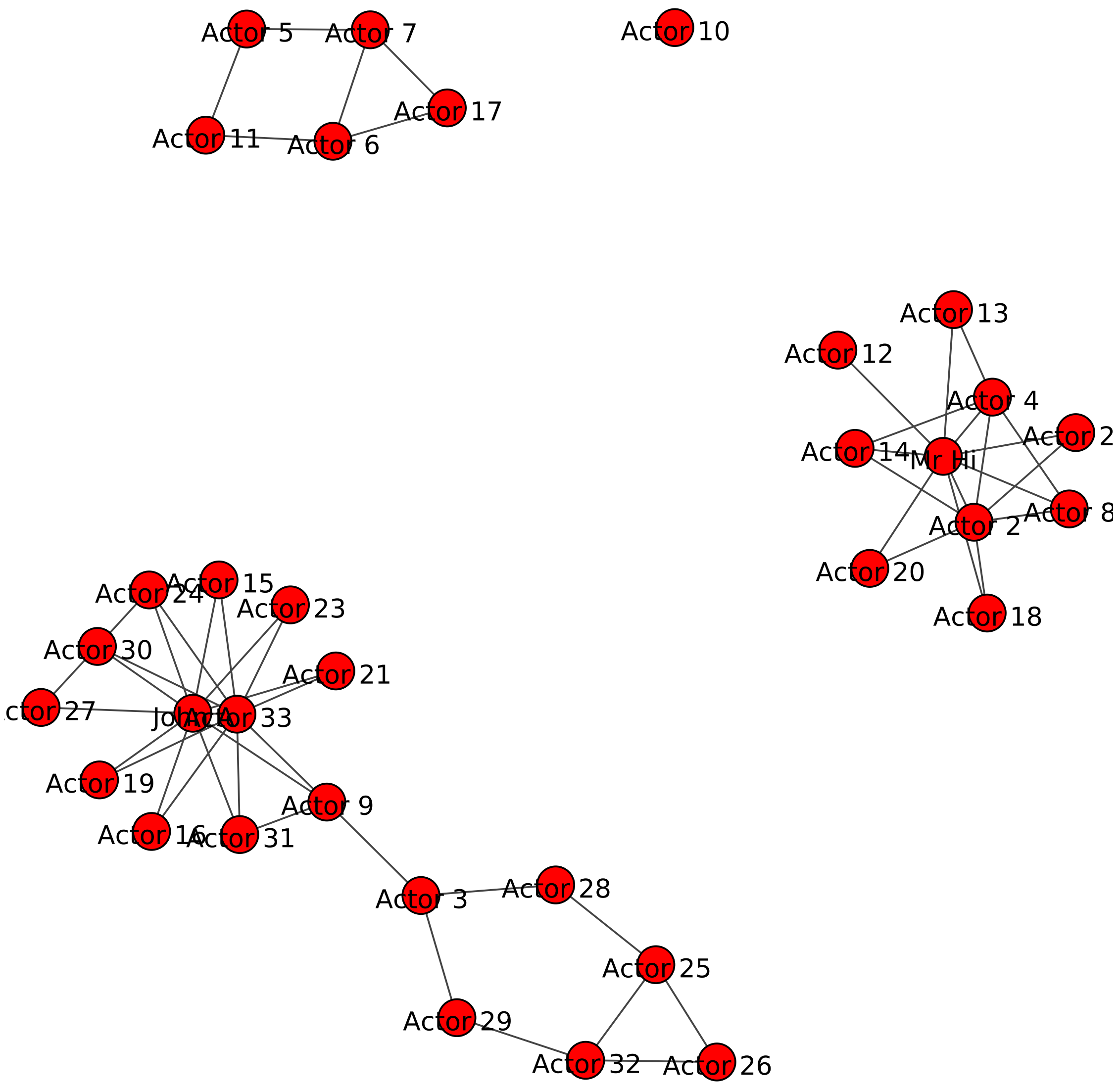

Figure 7: Initial graph split into 4 groups

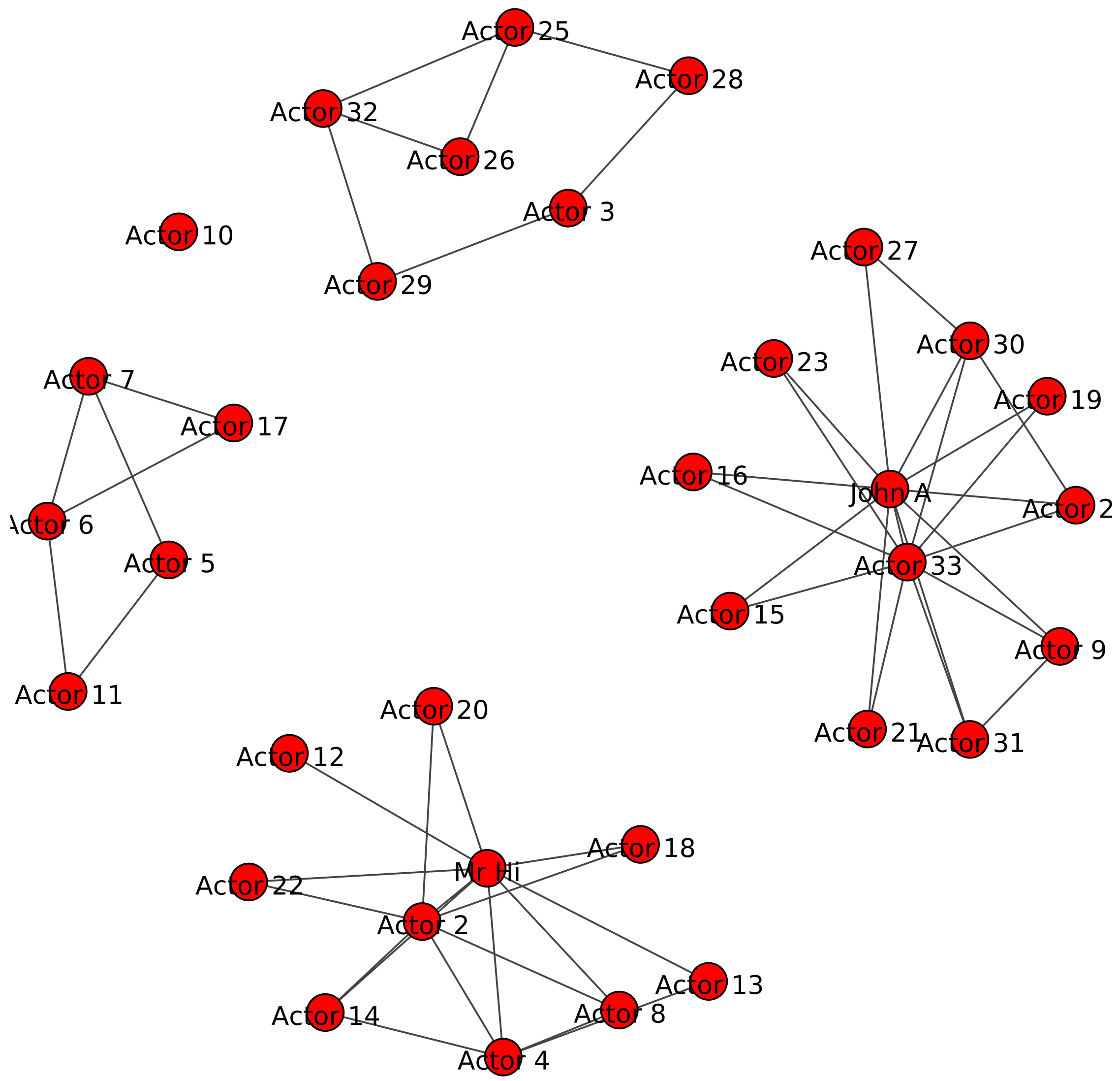

Figure 8: Initial graph split into 5 groups

3 Conclusions

The dispute in the karate club splits the club into two. There are 60 people in the club but we represent a graph with 33 people who are friends and meet outside the club. So, now we know that the club has split into two we try to create a mathematical model which can split the given social network graph into groups in order to achieve this we use Girvan-Newman Algorithm and split the graph and check whether the graphs we obtain match the reality.

When we observe the results from the reality to predicted results they are almost similar except one person. So, the results are not 100% true but the prediction almost resembles the reality with 96% (this is value is detremined by using the accuracy formula) similarity. So, the mathematical model best represents the reality from all the above observation. For a real world social network if you want to predict the number of groups in a network this model can be predict the result well.

Hence, we can prove that split results of a group can be predicted by weighted graph using mathematical model.

References

[1] Helpfultutorials. http://lists.nongnu.org/archive/html/igraph-help/2008-10/msg00024.html.

[2] igraph. http://igraph.org/python/.

[3] igraphtutorial. http://igraph.org/python/doc/tutorial/tutorial.html.

[4] otherresources. http://www.cs.rhul.ac.uk/home/tamas/development/igraph/tutorial/tutorial.html.

[5] W. ZACHARY. An information flow model for conflict and fission in small groups. http://aris.ss.uci.edu/l̃in/76.pdf.

[]