Web Science Assignment 5 Report

Dr. Nelson

James Pindell

3/5/2018

Content

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.

Clues:

1. Draw original Karate club graph (two connected components) after split (Week 6 lecture, slide 98).

2. Run multiple iterations of graph partioning algorithm (e.g., Girvan-Newman Algorithm) on experimental Karate club graph until the graph splits into two connected components.

3. Compare the connected components of the experimental graph (in 2.) with the original connected components of the split Karate club graph (in 1.). Are they similar?

Useful sources include:

* Original Paper:

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

* Week 6 Slides:

https://docs.google.com/presentation/d/1ihf6N8bHgzM5VLAyHkmF_i5JGUBVpCSdsvYpk8XgHwo/edit?usp=sharing

* Other 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:

https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.social.karate_club_graph.html

https://networkx.github.io/documentation/networkx-1.9/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

Solution

The solution for this problem is outlined by the following steps:

1. **Draw Experimental Karate Club Graph (Before Split)**: In order to draw the experimental Karate Club graph, numerous pieces of code from numerous sources were needed. These sources are listed within the code, but they are also listed below:

   Sources:
   https://gist.github.com/millionsmile/3682029

   https://networkx.github.io/documentation/networkx-1.9/examples/drawing/labels_and_colors.html

   https://gawron.sdsu.edu/python_for_ss/course_core/book_draft/Social_Networks/Networkx.html

You can view the complete code, called KarateClub.py, from within the package, but a snippet of the code is shown below:

```
def updateGraph(G):
    ebc = NC.edge_betweenness(G)
    maxs = 0
    medge = None
    for k, v in ebc.items():
        if maxs < v:
            medge, maxs = k, v
    G.remove_edge(*medge)
    #return G

# Draws the current state of the Karate Club graph
def drawGraph(G, pos, output):

    # Draws the two factions of nodes, denoted by different colors (Mr. Hi: John A: )
    NX.draw_networkx_nodes(G,pos,nodelist=[0,1,3,4,5,6,7,10,11,12,13,16,17,19,21],node_color='r',node_size=500,alpha=0.8)
    NX.draw_networkx_nodes(G,pos,nodelist=[2,8,9,14,15,18,20,22,23,24,25,26,27,28,29,30,31,32,33],node_color='g',node_size=500,alpha=0.8)

    # Draws the edges between the corresponding nodes
    NX.draw_networkx_edges(G,pos,width=1.0,alpha=0.5)

    # Draws the labels for each node (e.g., 1, 2, 3, ... n)
    new_labels = dict((x,x + 1) for x in G.nodes())
    NX.draw_networkx_labels(G,pos,new_labels,font_size=14,font_color='black')

    # Turns the axis off, and saves the graph to the corresponding 'output' file
    P.axis("off")
    P.savefig(output)
    P.draw()
    P.close()
```

In addition, the source code for the Karate Club graph as defined by networkx, called KarateClubSource.py, can be viewed from within the package, but a snippet of the source code is also shown below:

```
# Create the set of all members, and the members of each club.
all_members = set(range(34))
club1 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16, 17, 19, 21}
# club2 = all_members - club1

G = nx.Graph()
G.add_nodes_from(all_members)
G.name = "Zachary's Karate Club"

zacharydat = """\
0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0
1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0
1 1 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0
1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
```

After running the KarateClub.py file, the experimental Karate Club graph is drawn as a png file, called karate00.png, and placed inside a folder, called img. You can view this graph from within the folder, that's within the package, but you can also view this graph at the bottom of the report (Graph 0').

Note: This graph is called Graph 0' because this is what the experimental graph looks like before any Graph Partitioning Algorithms (e.g., Girvan-Newman Algorithm) was used to split the graph. This is to simply keep the graph numbering and iteration numbering the same, to help avoid any confusion (e.g., Graph 0' = Iteration 0')

Note: In the experimental graph, the red nodes indicate those who belong to the Mr. Hi faction, and the green nodes indicate who belong to the John A ("Officer") faction

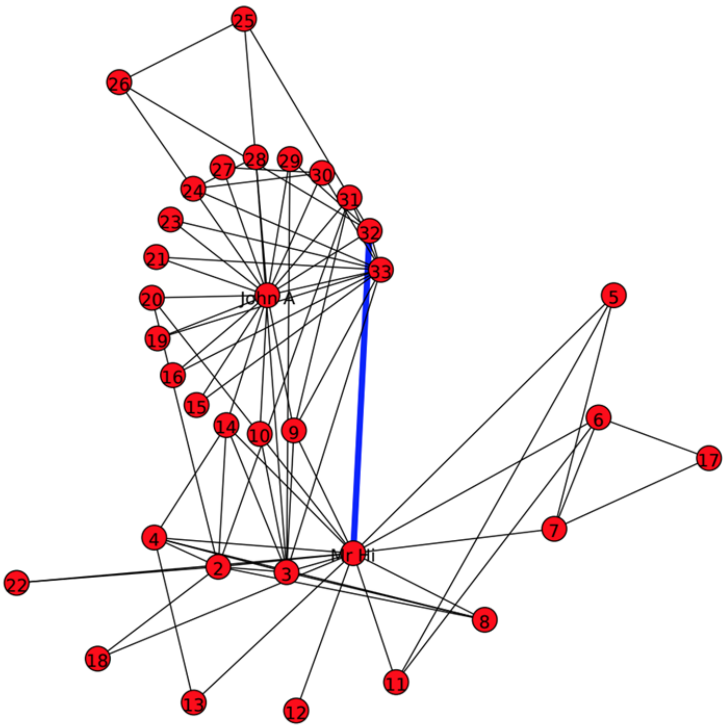
2. **Use Graph Partitioning Algorithm to Split Experimental Graph**: Truthfully, the code shown above (Step 1, KarateClub.py) doesn't just draw the experimental Karate Club graph, but in addition, it uses the Graph Partitioning Algorithm (Girvan-Newman Algorithm) to remove the edge with the highest edge betweenness, draws a new graph, (without erasing the previous graph) and repeats this process until all edges have been removed.

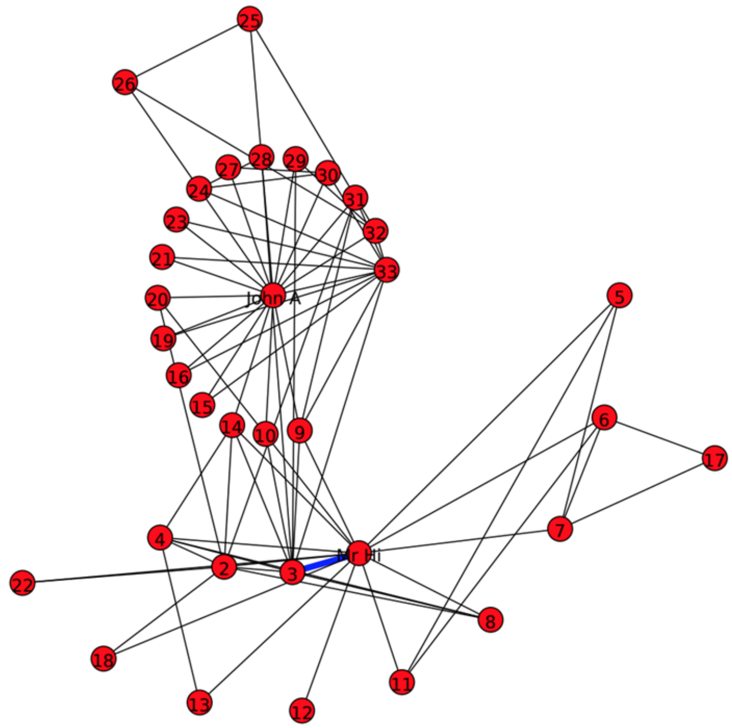
   You can view all of these experimental graphs from within the folder, called img, that is within the package, but only Graphs 1' – 11', along with Graph 0', are going to be shown at the bottom of this report, since these graphs are the only graphs that are needed to compare and contrast the original Karate Club graphs (Graphs 0 – 11), as shown in the Week 6 Slides (and the bottom of the report), and the experimental Karate Club graphs (Graphs 0' – 11').

3. **Compare and Contrast (Experimental vs. Original)**: After carefully analyzing both the original graphs (Graphs 0 – 11) and the experimental graphs (Graphs 0' – 11'), it can be concluded that both sets of graphs are indeed similar. This is because both sets of graphs split at the 11th iteration, creating two clusters. Therefore, it is reasonable to also conclude that it is proven that the result of the split can be predicted by using the weighted graph of social interactions. In addition, it is reasonable to argue that this mathematical model represents reality fairly well.

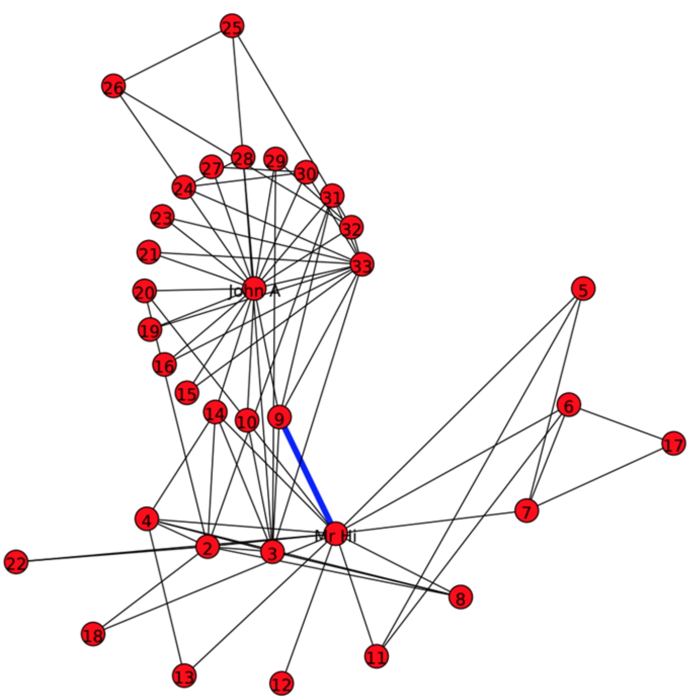
   Note: In the experimental graphs (Graphs 0' – 11'), the number of nodes in each faction is slightly different than the number of nodes in the original graphs (Graph 0 – 11). Also, in the experimental graphs (Graphs 0' – 11'), the node placement is not quite as organized compared to the node placement in the original graphs (Graphs 0 – 11). Regardless, both sets of graphs are still fundamentally the exact same.

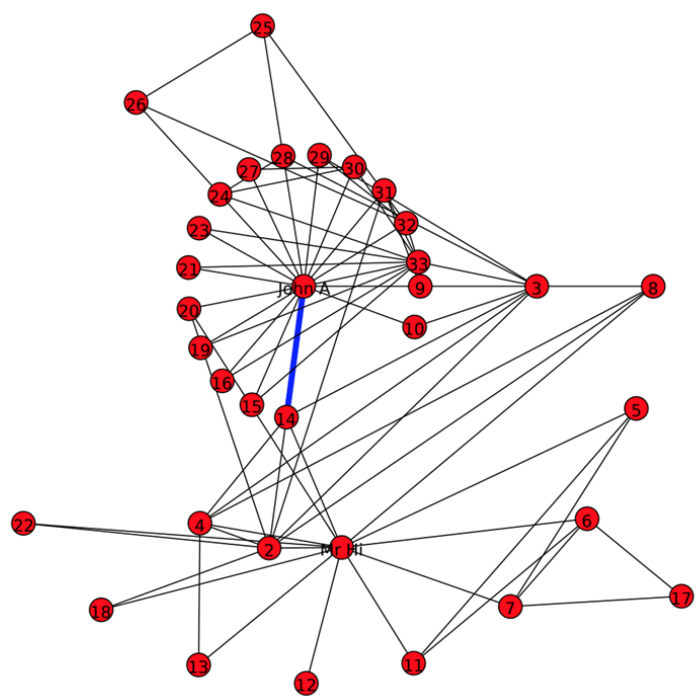
(Graph 0, Left) and (Graph 1, Right)



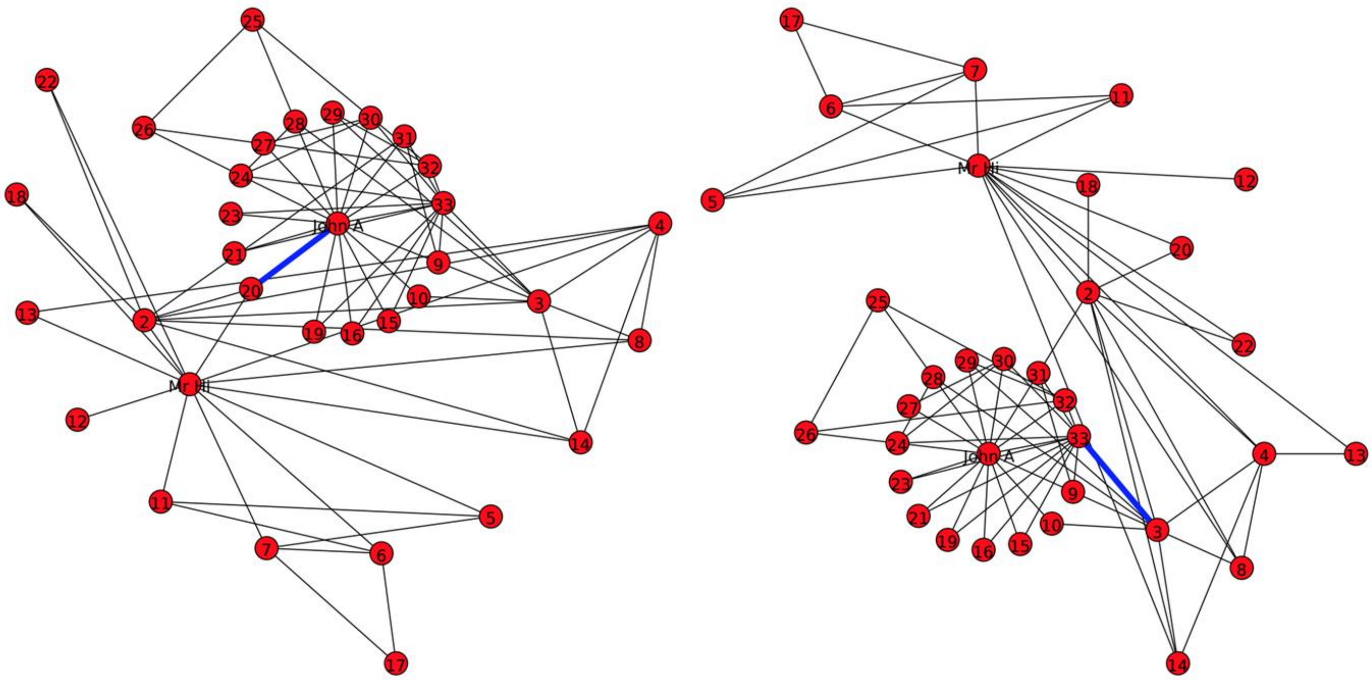


(Graph 2, Left) and (Graph 3, Right)

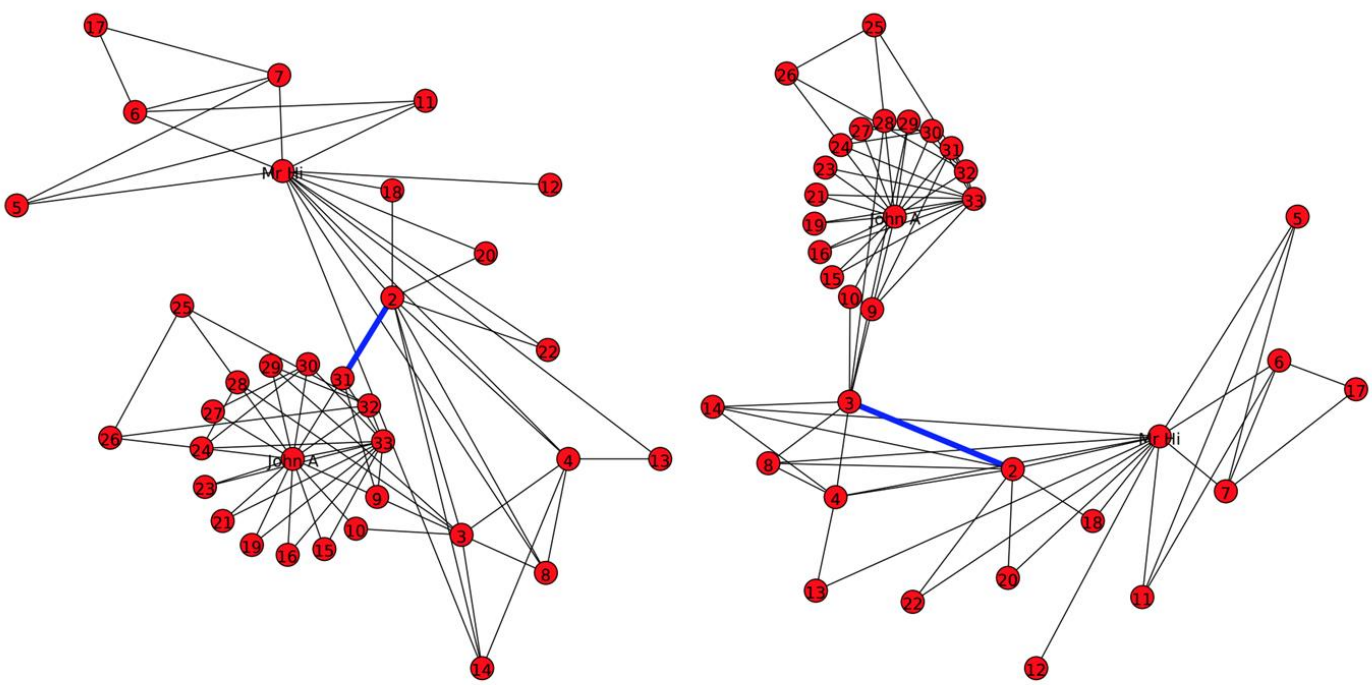




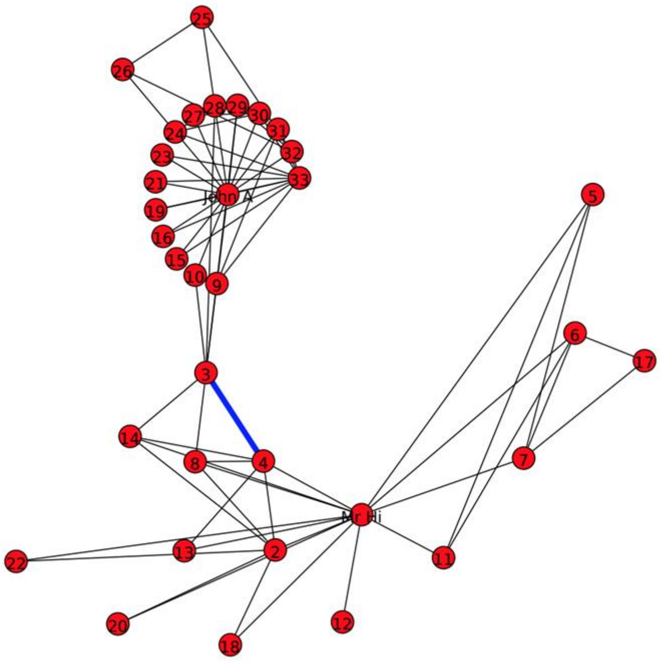
(Graph 4, Left) and (Graph 5, Right)

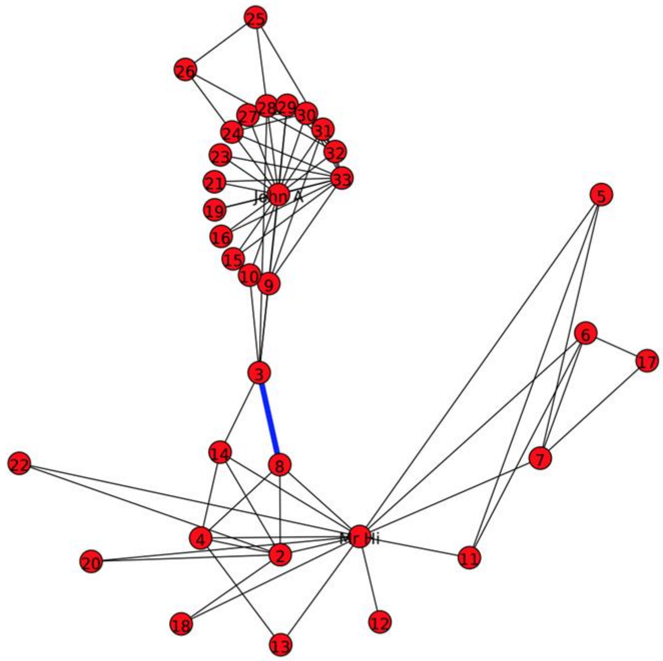


(Graph 6, Left) and (Graph 7, Right)

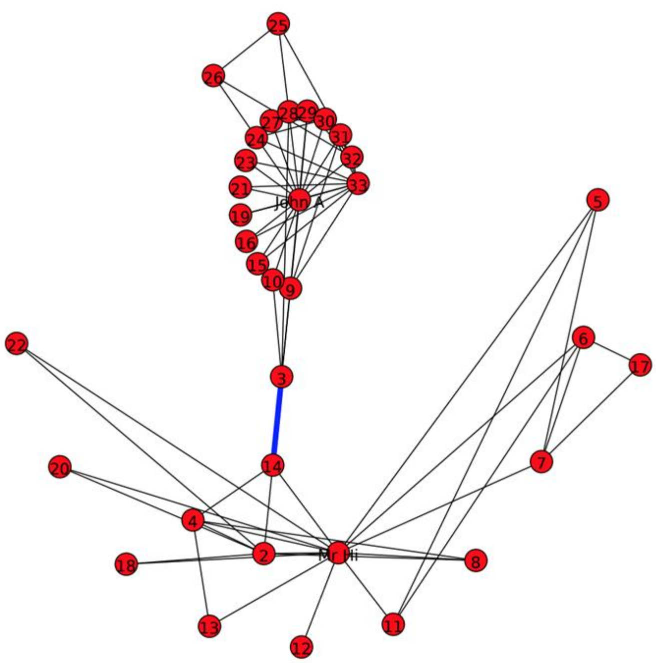


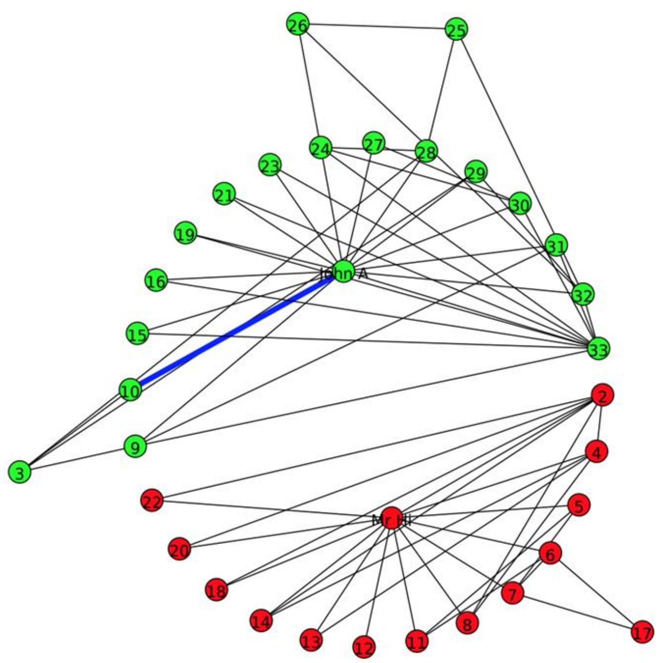
(Graph 8, Left) and (Graph 9, Right)



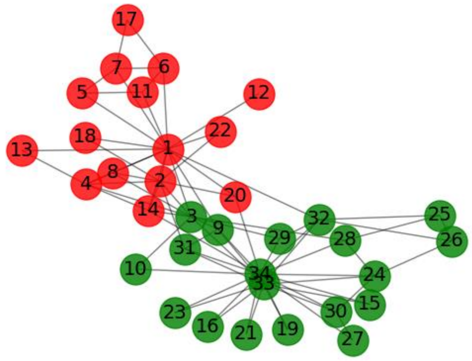


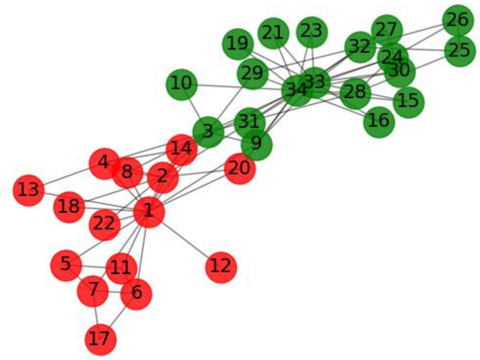
(Graph 10, Left) and (Graph 11, Right)



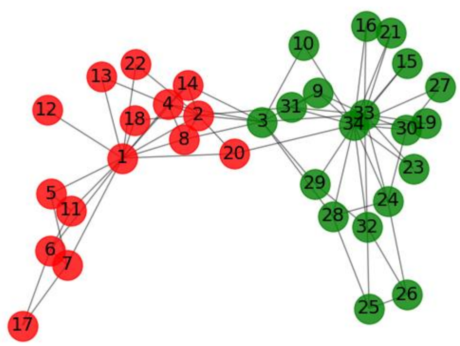


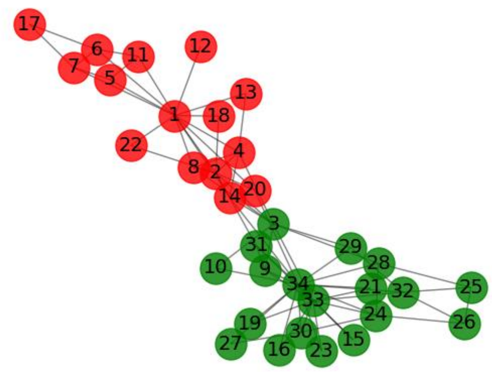
(Graph 0’, Left) and (Graph 1’, Right)



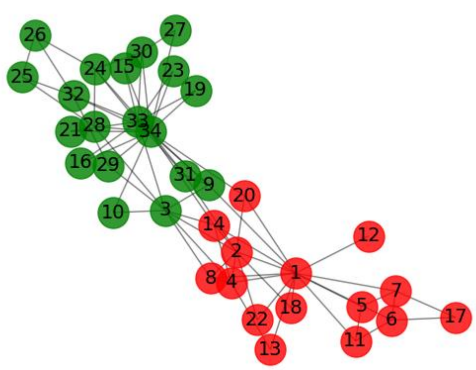


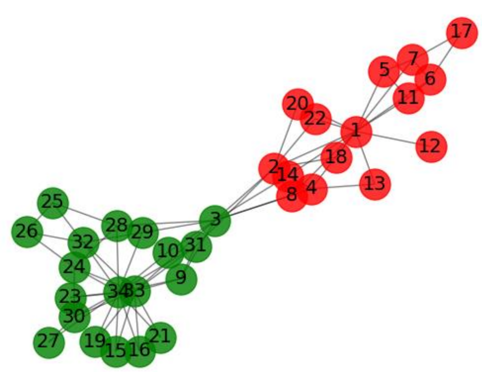
(Graph 2’, Left) and (Graph 3’, Right)



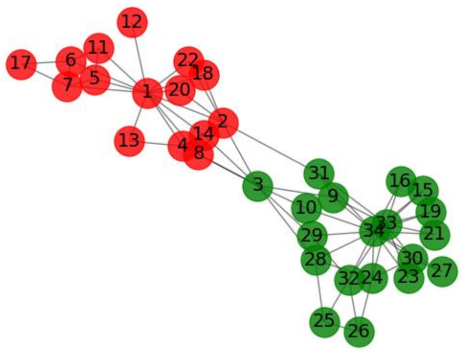


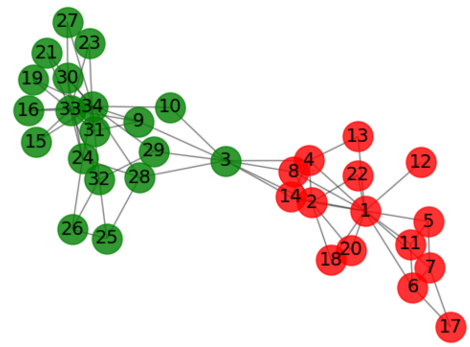
(Graph 4’, Left) and (Graph 5’, Right)



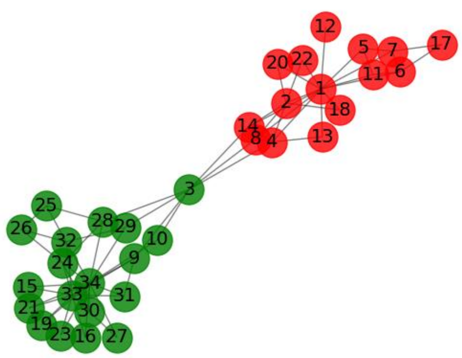


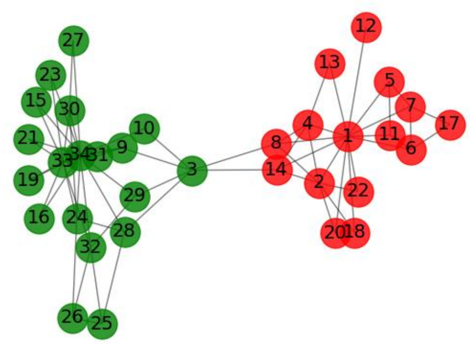
(Graph 6’, Left) and (Graph 7’, Right)





(Graph 8’, Left) and (Graph 9’, Right)





(Graph 10’, Left) and (Graph 11’, Right)