INTRODUCTION TO WEB SCIENCE- FALL 2014

Assignment Six

Sybil Melton

October 30, 2014

Contents

List of Figures

Listings

1 Karate Club Split into Two

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?

1.1 Solution

This problem was approached by learning the NetworkX graph programming. The given data, a matrix of individuals and their connection strength to each other, was saved as a CSV file [5]. Row and column one, designates Mr. Hi and 34 designates John A., the two faction leaders. [7] The python csv reader read each line as string list, which resulted in a list of lists and made it easier to parse into a list of weighted edges. [2] If the weight was greater than 0, it was added to the edge list as a tuple of individual, connected individual, and weight. Then a weighted NetworkX graph was created by adding the weighted edge list. The original Karate Club graph was saved as a dot file using Pydot, and used to create Figure 1.1 using Gephi. [6]

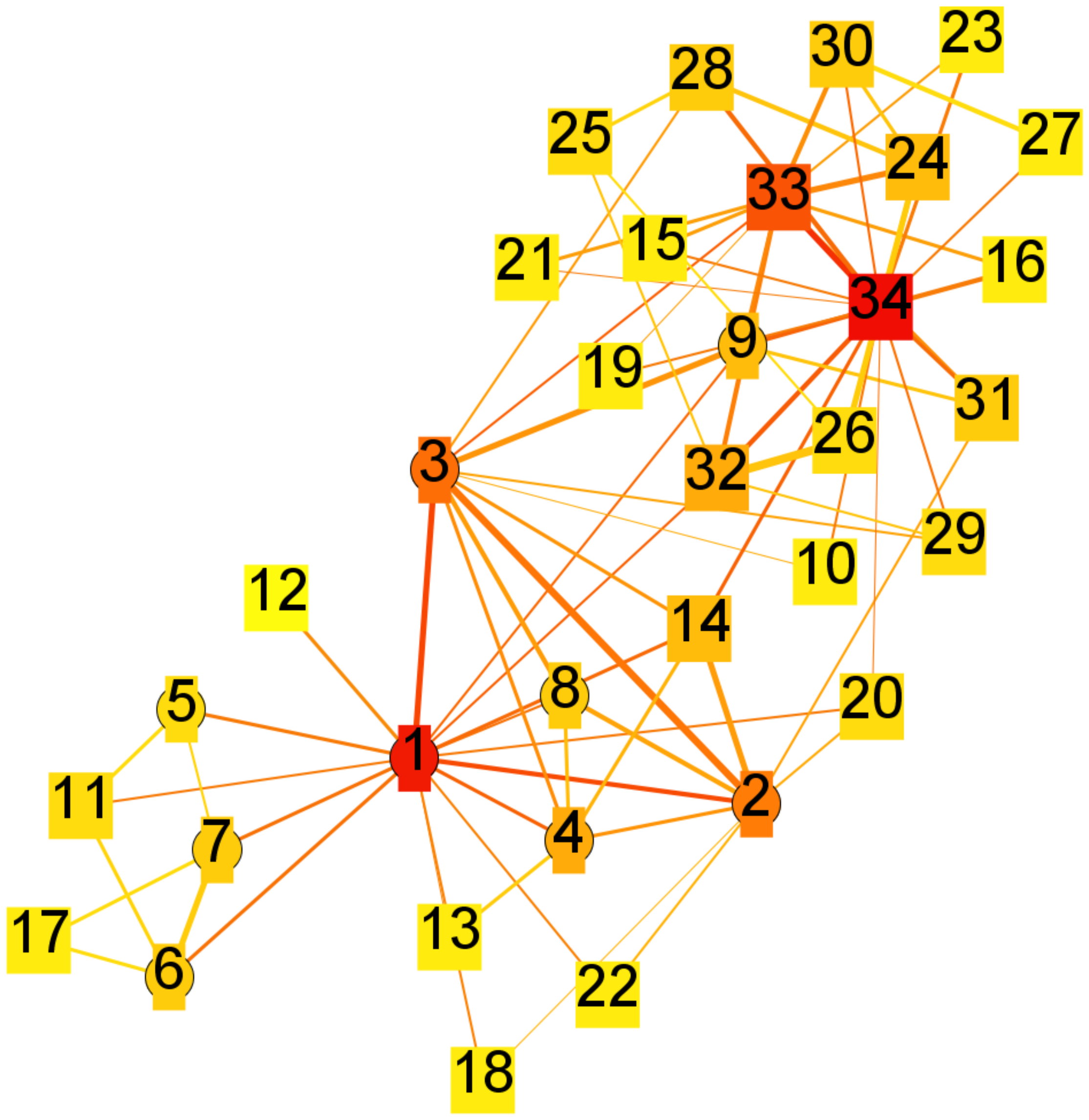


Figure 1.1: Original Karate Club

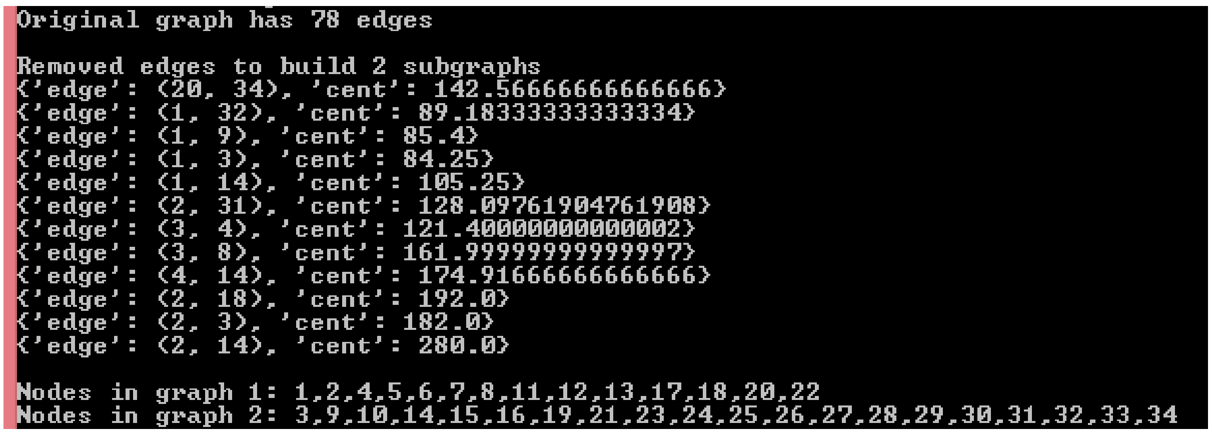
NetworkX has the ability to create subgraphs by grouping the connected components together. The result is a subgraph list. [1] Using *edge betweenness* and the algorithm of Girvan and Newman, the Karate Club graph was split into two. The steps of the algorithm are:

1. Compute edge centrality.
2. Remove edge with largest centrality, choose randomly for a tie.
3. Recalculate edge centralities on the running graph.
4. Repeat from step 2. [4]

"Betweenness centrality of an edge e is the sum of the fraction of all-pairs shortest paths that pass through e:

$$\sum_{x,t\in V} \frac{\sigma(s,t|e)}{\sigma(s,t)}$$

where V is the set of nodes, $\sigma(s,t)$ is the number of shortest (s,t)-paths, and $\sigma(s,t|e)$ is the number of those paths passing through edge e." [3] A loop kept track of the length of the subgraph list and terminated when it reach two. Each iteration recalculated the edge betweenness centrality. It kept a running list of the removed edges and printed them out upon completion with sorted node lists for each subgraph. Figure 1.2 shows the output upon completion of creating the 2 subgraphs.



```
Original graph has 78 edges

Removed edges to build 2 subgraphs
{'edge': (20, 34), 'cent': 142.56666666666666}
{'edge': (1, 32), 'cent': 89.18333333333334}
{'edge': (1, 9), 'cent': 85.4}
{'edge': (1, 3), 'cent': 84.25}
{'edge': (1, 14), 'cent': 105.25}
{'edge': (2, 31), 'cent': 128.09761904761908}
{'edge': (3, 4), 'cent': 121.40000000000002}
{'edge': (3, 8), 'cent': 161.99999999999997}
{'edge': (4, 14), 'cent': 174.91666666666666}
{'edge': (2, 18), 'cent': 192.0}
{'edge': (2, 3), 'cent': 182.0}
{'edge': (2, 14), 'cent': 280.0}

Nodes in graph 1: 1,2,4,5,6,7,8,11,12,13,17,18,20,22
Nodes in graph 2: 3,9,10,14,15,16,19,21,23,24,25,26,27,28,29,30,31,32,33,34
```

Figure 1.2: Program Output 2 Clubs

The resulting subgraphs were also saved as dot files, in order to create Figures 1.3 and 1.4.

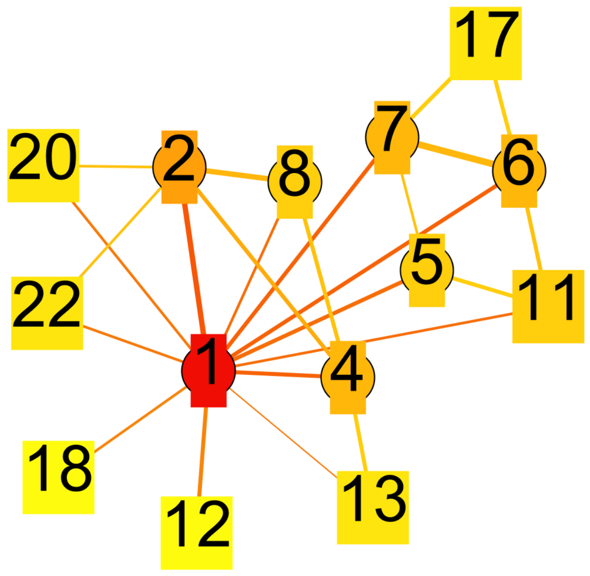


Figure 1.3: 2-way Split Club 1

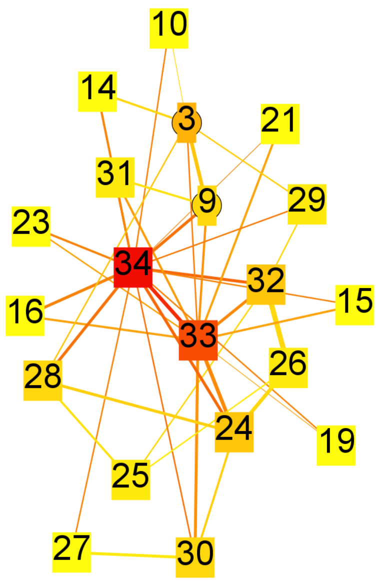


Figure 1.4: 2-way Split Club 2

The entire python program is included in Section 1.3, in Listing 1. The code for calculating edge betweenness centrality, including the shortest path algorithm is included with this report as betweenness.py. The connected component subgraph code is included in connected.py. Both files are unedited versions from the Pythonxy distribution.

1.2 Result

In my result, there were three individuals; 3, 9, and 14; who should have been in the graph with Mr. Hi (individual 1). Mathematical models using only edge betweenness cannot always be 100% correct. For example, individual nine had joined Mr. Hi's club, although he supported John A., for reasons unrelated to the split. [7] In order to deduce a reason three and 14 ended up in the wrong graph, I looked at their positions in the original club. Both were individuals had ties to both sides, so as more edges were removed, their edge betweenness centrality increased. I believe the edge weights kept their edge betweenness centrality higher eventually led to being disconnected from Mr. Hi's graph. Perhaps if the strength of their ties to Mr. Hi's supporters had also been taken into account, it would have resulted closer to reality. The rest of the individuals were placed into the correct graph, so the Girvan and Newman algorithm was over 91% correct.

1.3 Python Code

```python
import connected
import networkx as nx
import betweenness
import csv
import pydot
import argparse


def buildClubGraphs(inFile, n):
  club = []
  #weighted edge data retrieved from http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat
  #open the csv and put the edge matrix into a list of lists
  with open(inFile) as i:
    reader = csv.reader(i)
    for row in reader:
      club.append(row)
  #create a list of tuples, with edges and weights
  eList = []
  for i in range(0,len(club[0])):
    for j in range(0, len(club[0])):
      if int(club[i][j]) > 0:
        eList.append((i+1,j+1,int(club[i][j])))
  #Use list of edges to create the nx graph
  K = nx.Graph()
  K.add_weighted_edges_from(eList)
  # Create the .dot file for the whole karate club
  nx.write_dot(K, 'karateclub.dot')
  print 'Original graph has ' + str(K.number_of_edges()) + ' edges\n'

  graphs = list(nx.connected_component_subgraphs(K))
  removed = []
  while len(graphs) < n:
    b = nx.edge_betweenness_centrality(K, weight='weight', normalized=False)
    e = (0, 0)
    centrality = 0.0
    for i in b:
      if b[i] > centrality:
        centrality = b[i]
        e = i
      # check
      #b.get(item)
    edges = [e] #put returned tuple into a list, to be used to remove from graph
    #keep track of removed edges and their weighted edge betweenness centrality
    removed.append({'edge': e, 'cent': centrality})
    K.remove_edges_from(edges)
    graphs = list(nx.connected_component_subgraphs(K))

  print 'Removed edges to build '+str(n)+' subgraphs\n'+'\n'.join(str(r) for r in removed)+'\n'
  #http://stackoverflow.com/questions/12633024/python-concatenate-string-list
  # write the .dot files for the split clubs
  for i in range(0,len(graphs)):
    print 'Nodes in graph ' + str(i+1) + ': ' + ','.join(str(g) for g in sorted(graphs[i].nodes()))
    nx.write_dot(graphs[i], str(n)+'-'+str(i)+'-split.dot')


if __name__ == '__main__':
  parser = argparse.ArgumentParser(description='Build club graph and split into communities')
  parser.add_argument('file', help='input edge matrix file, csv format')
  parser.add_argument('communities', type=int, help='integer, number of communities to split into')
  args = parser.parse_args()
  f = args.file
  n = args.communities
  buildClubGraphs(f, n)
```

Listing 1: Python graph code

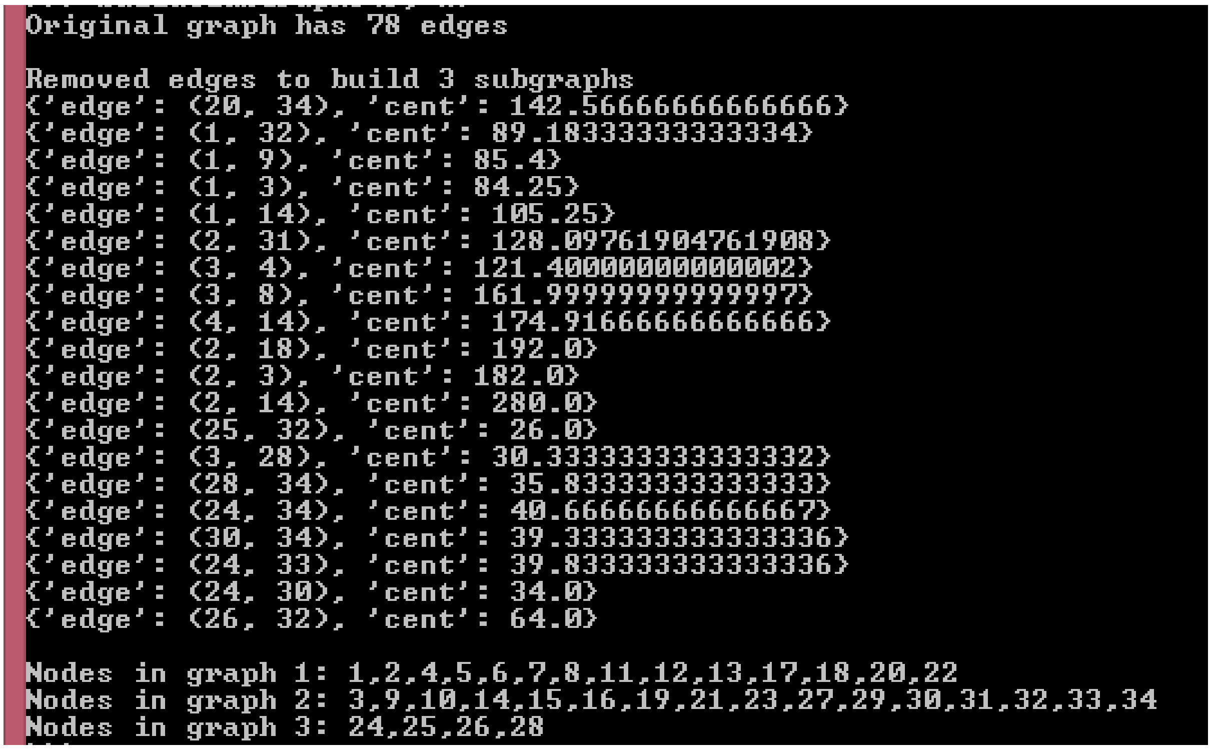
2 Karate Club Multiple Splits

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 Result

The graphs program was written with the desired number of subgraphs as an argument. The loop kept removing edges until the graph was split into the desired number of subgraphs. The resulting dot files are included with this report and were used to create the resulting graphs. As the group split into more subgroups, the next subgroup split from one of the two larger subgroups. So the third group in the 4- and 5-way splits remained, as well as the fourth group remained in the 5-way split. The screen output and each resulting graph are illustrated in the following sections.

2.1.1 Three Clubs



```
Original graph has 78 edges

Removed edges to build 3 subgraphs
{'edge': (20, 34), 'cent': 142.56666666666666}
{'edge': (1, 32), 'cent': 89.18333333333334}
{'edge': (1, 9), 'cent': 85.4}
{'edge': (1, 3), 'cent': 84.25}
{'edge': (1, 14), 'cent': 105.25}
{'edge': (2, 31), 'cent': 128.09761904761908}
{'edge': (3, 4), 'cent': 121.40000000000002}
{'edge': (3, 8), 'cent': 161.99999999999997}
{'edge': (4, 14), 'cent': 174.91666666666666}
{'edge': (2, 18), 'cent': 192.0}
{'edge': (2, 3), 'cent': 182.0}
{'edge': (2, 14), 'cent': 280.0}
{'edge': (25, 32), 'cent': 26.0}
{'edge': (3, 28), 'cent': 30.333333333333332}
{'edge': (28, 34), 'cent': 35.833333333333333}
{'edge': (24, 34), 'cent': 40.666666666666667}
{'edge': (30, 34), 'cent': 39.333333333333336}
{'edge': (24, 33), 'cent': 39.833333333333336}
{'edge': (24, 30), 'cent': 34.0}
{'edge': (26, 32), 'cent': 64.0}

Nodes in graph 1: 1,2,4,5,6,7,8,11,12,13,17,18,20,22
Nodes in graph 2: 3,9,10,14,15,16,19,21,23,27,29,30,31,32,33,34
Nodes in graph 3: 24,25,26,28
```

Figure 2.1: Program Output 3 Clubs

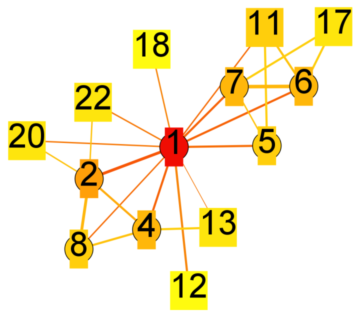


Figure 2.2: 3-way Split Club 1

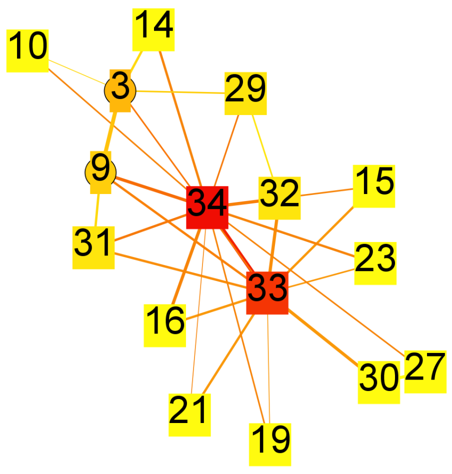


Figure 2.3: 3-way Split Club 2

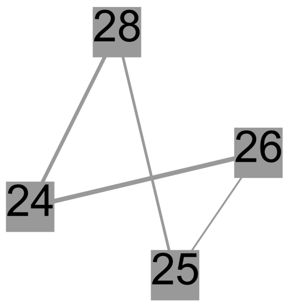
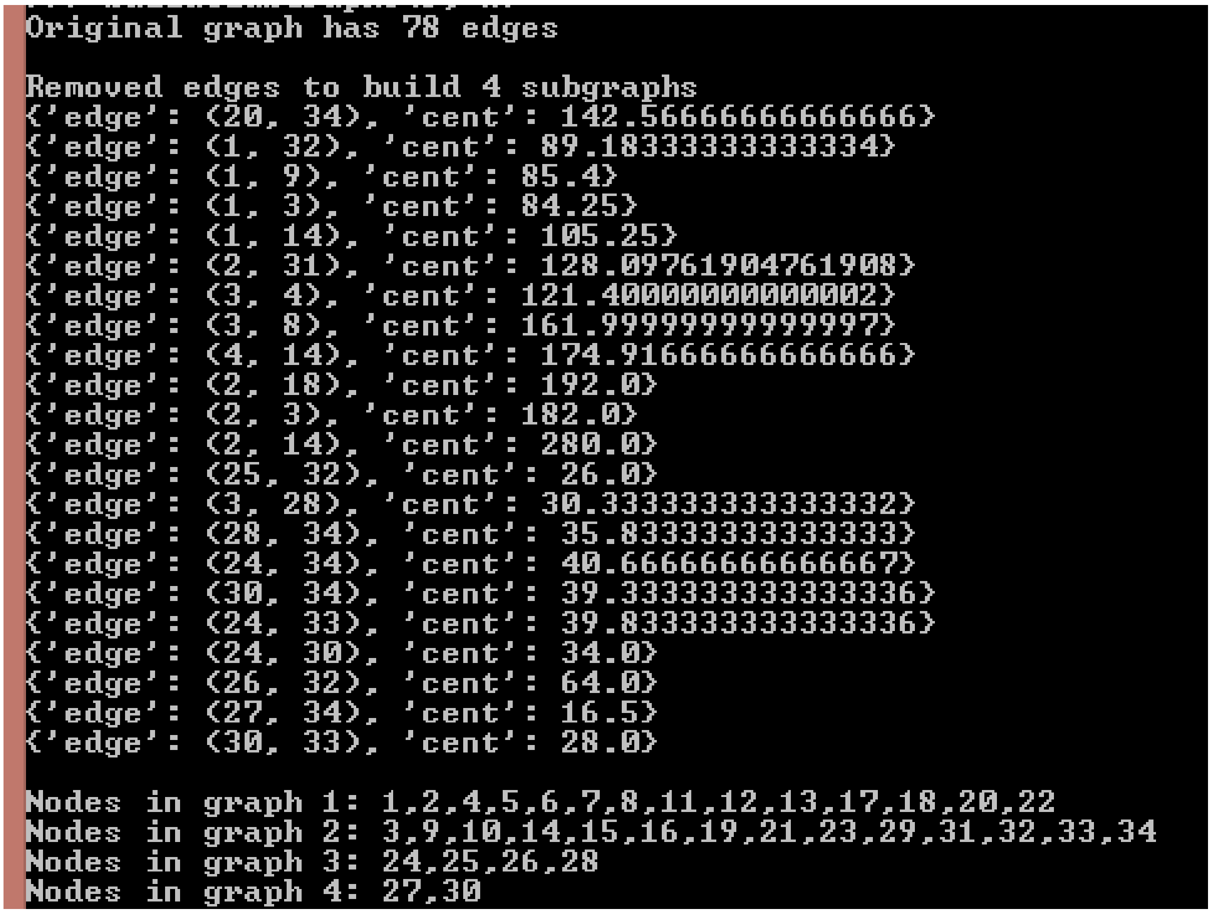


Figure 2.4: 3-way Split Club 3

2.1.2 Four Clubs



```
Original graph has 78 edges

Removed edges to build 4 subgraphs
{'edge': (20, 34), 'cent': 142.56666666666666}
{'edge': (1, 32), 'cent': 89.18333333333334}
{'edge': (1, 9), 'cent': 85.4}
{'edge': (1, 3), 'cent': 84.25}
{'edge': (1, 14), 'cent': 105.25}
{'edge': (2, 31), 'cent': 128.09761904761908}
{'edge': (3, 4), 'cent': 121.40000000000002}
{'edge': (3, 8), 'cent': 161.99999999999997}
{'edge': (4, 14), 'cent': 174.91666666666666}
{'edge': (2, 18), 'cent': 192.0}
{'edge': (2, 3), 'cent': 182.0}
{'edge': (2, 14), 'cent': 280.0}
{'edge': (25, 32), 'cent': 26.0}
{'edge': (3, 28), 'cent': 30.333333333333332}
{'edge': (28, 34), 'cent': 35.833333333333333}
{'edge': (24, 34), 'cent': 40.666666666666667}
{'edge': (30, 34), 'cent': 39.333333333333336}
{'edge': (24, 33), 'cent': 39.833333333333336}
{'edge': (24, 30), 'cent': 34.0}
{'edge': (26, 32), 'cent': 64.0}
{'edge': (27, 34), 'cent': 16.5}
{'edge': (30, 33), 'cent': 28.0}

Nodes in graph 1: 1,2,4,5,6,7,8,11,12,13,17,18,20,22
Nodes in graph 2: 3,9,10,14,15,16,19,21,23,29,31,32,33,34
Nodes in graph 3: 24,25,26,28
Nodes in graph 4: 27,30
```

Figure 2.5: Program Output 4 Clubs

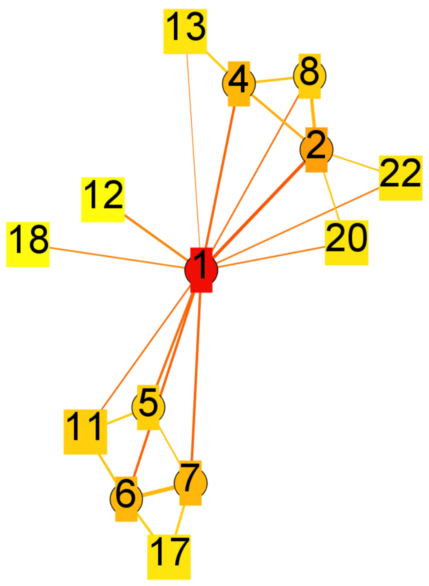


Figure 2.6: 4-way Split Club 1

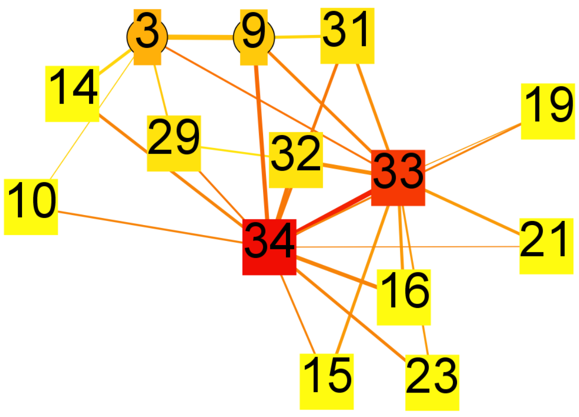


Figure 2.7: 4-way Split Club 2

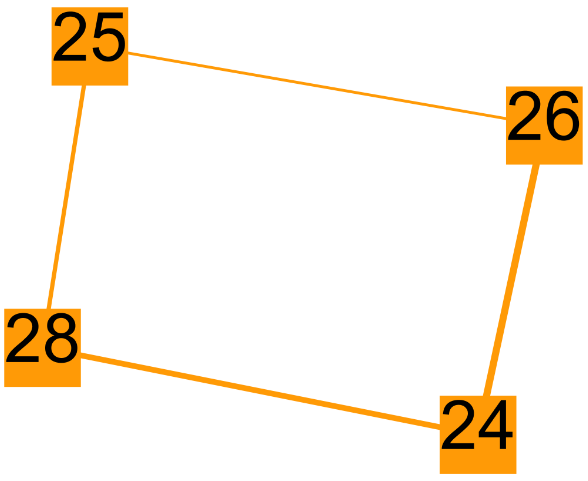


Figure 2.8: 4-way Split Club 3

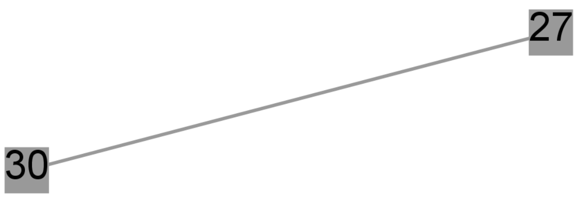
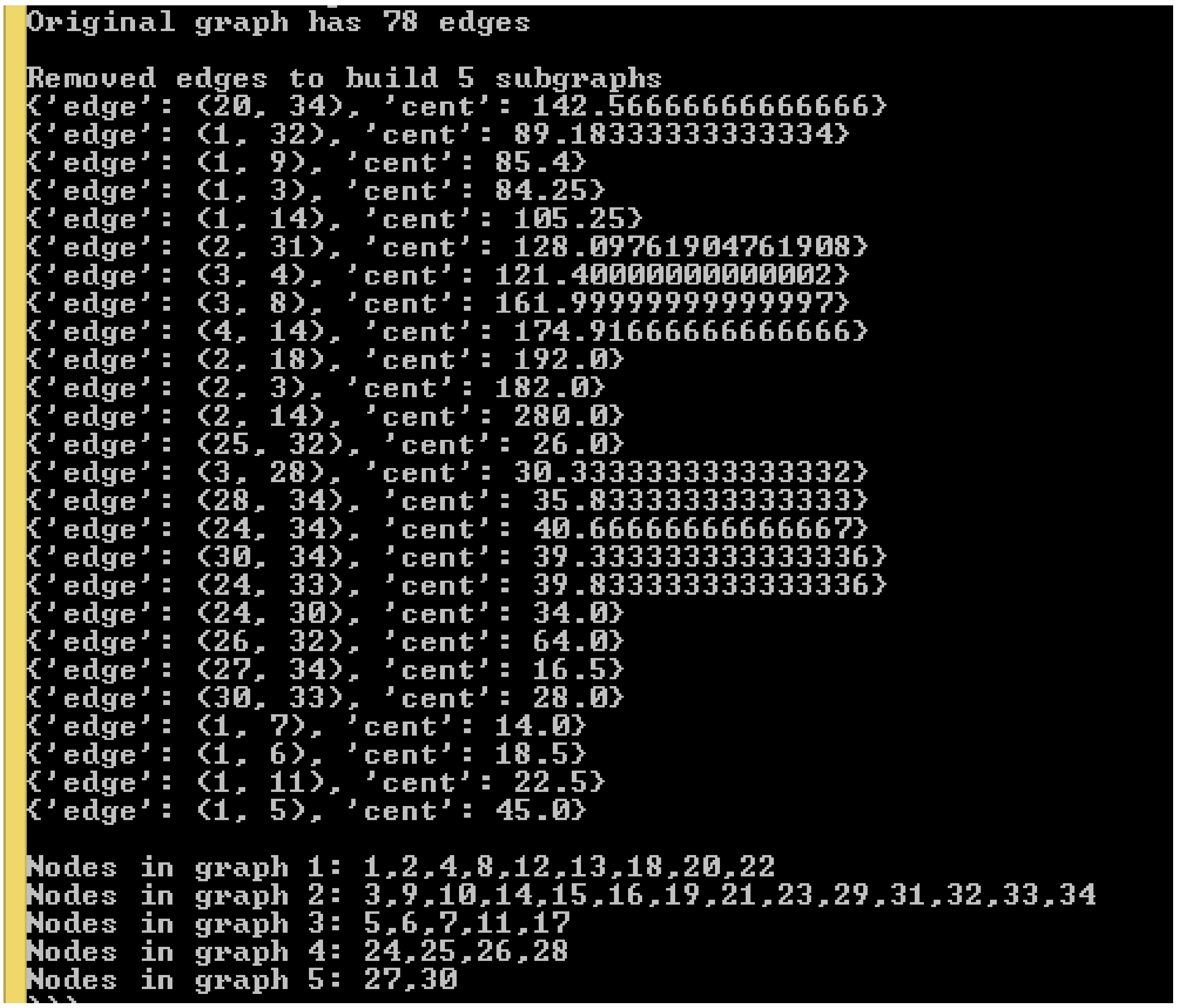


Figure 2.9: 4-way Split Club 4

2.1.3 Five Clubs



```
Original graph has 78 edges

Removed edges to build 5 subgraphs
{'edge': (20, 34), 'cent': 142.56666666666666}
{'edge': (1, 32), 'cent': 89.18333333333334}
{'edge': (1, 9), 'cent': 85.4}
{'edge': (1, 3), 'cent': 84.25}
{'edge': (1, 14), 'cent': 105.25}
{'edge': (2, 31), 'cent': 128.09761904761908}
{'edge': (3, 4), 'cent': 121.40000000000002}
{'edge': (3, 8), 'cent': 161.99999999999997}
{'edge': (4, 14), 'cent': 174.91666666666666}
{'edge': (2, 18), 'cent': 192.0}
{'edge': (2, 3), 'cent': 182.0}
{'edge': (2, 14), 'cent': 280.0}
{'edge': (25, 32), 'cent': 26.0}
{'edge': (3, 28), 'cent': 30.333333333333332}
{'edge': (28, 34), 'cent': 35.833333333333333}
{'edge': (24, 34), 'cent': 40.666666666666667}
{'edge': (30, 34), 'cent': 39.333333333333336}
{'edge': (24, 33), 'cent': 39.833333333333336}
{'edge': (24, 30), 'cent': 34.0}
{'edge': (26, 32), 'cent': 64.0}
{'edge': (27, 34), 'cent': 16.5}
{'edge': (30, 33), 'cent': 28.0}
{'edge': (1, 7), 'cent': 14.0}
{'edge': (1, 6), 'cent': 18.5}
{'edge': (1, 11), 'cent': 22.5}
{'edge': (1, 5), 'cent': 45.0}

Nodes in graph 1: 1,2,4,8,12,13,18,20,22
Nodes in graph 2: 3,9,10,14,15,16,19,21,23,29,31,32,33,34
Nodes in graph 3: 5,6,7,11,17
Nodes in graph 4: 24,25,26,28
Nodes in graph 5: 27,30
```

Figure 2.10: Program Output 5 Clubs

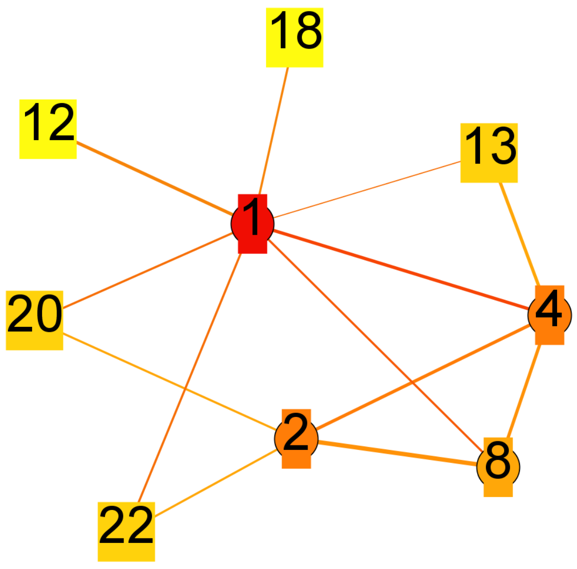


Figure 2.11: 5-way Split Club 1

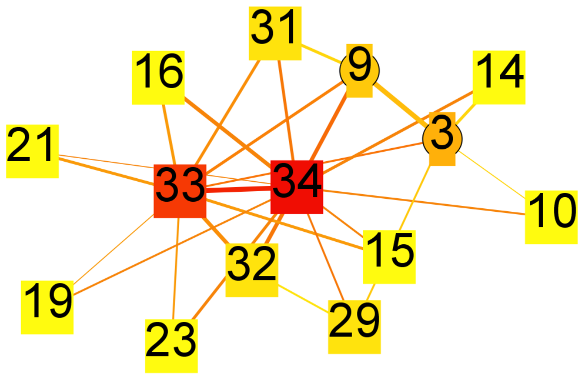


Figure 2.12: 5-way Split Club 2

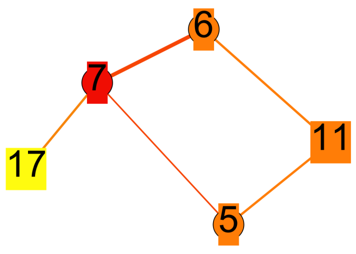


Figure 2.13: 5-way Split Club 3

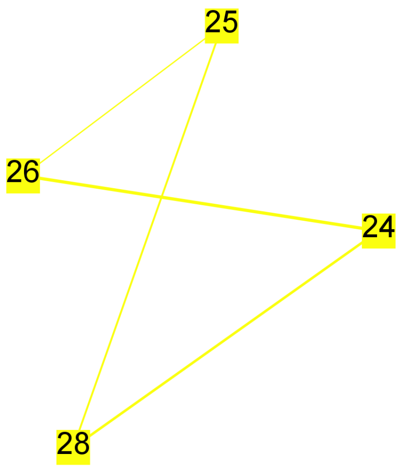


Figure 2.14: 5-way Split Club 4

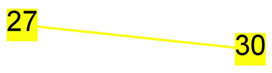


Figure 2.15: 5-way Split Club 5

References

[1] Networkx connected component subgraphs. `http://networkx.lanl.gov/reference/generated/networkx.algorithms.components.connected.connected_component_subgraphs.html#networkx.algorithms.components.connected.connected_component_subgraphs`. Accessed: 2014-10-22.

[2] CSV file reading and writing. `https://docs.python.org/2/library/csv.html`. Accessed: 2014-10-14.

[3] Networkx edge betweenness centrality. `http://networkx.lanl.gov/reference/generated/networkx.algorithms.centrality.edge_betweenness_centrality.html#networkx.algorithms.centrality.edge_betweenness_centrality`. Accessed: 2014-10-22.

[4] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010. Accessed: 2014-10-22.

[5] Zachary karate club network dataset – KONECT, October 2014. Accessed: 2014-10-22.

[6] Networkx tutorial. `http://networkx.github.io/documentation/networkx-1.9.1/tutorial/tutorial.html`. Accessed: 2014-10-22.

[7] Wayne Zachary. An information flow model for conflict and fission in small groups. *J. of Anthropological Research*, 33:452–473, 1977. Accessed: 2014-10-22.