# CS896 Introduction to Web Science
# Fall 2013
# Report for Assignment 7

Corren G. McCoy

November 7, 2013

# Contents

# List of Figures

# 1 Question 1

## 1.1 Problem

Using D3, create a graph of the karate club before and after the split. Weight the edges with the data from: `http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat`. Have the transition from before/after the split occur on a mouse click.

## 1.2 Response

The input data for Zachary's karate club was obtained from the Nexus Network Repository (`http://www.nexus.igraph.org/api/dataset_info?id=1&format=html`) in GraphML format. In addition to the nodes and links, their file also contains attributes for edge weight and associated faction (i.e., club president (John A.), karate instructor (Mr. Hi)) of the club member after the split. A snippet of the XML from the graph file is shown below:

```
[nodes] => Array
    (
        [0] => stdClass Object
            (
                [Faction] => 1
                [id] => n12
                [name] => Actor 13
            )

[links] => Array
    (
        [0] => stdClass Object
            (
                [source] => 0
                [target] => 26
                [weight] => 1
            )
```

Since GraphML is not a file format that is natively handled by D3, we used Python to convert the GraphML to a CSV file. The Python source, shown in Appendix A, performs the following tasks:

- Read the GraphML file;
- Convert the file to JSON using the utilities in the NetworkX library;
- Read and parse the JSON file;
- Save the nodes and links in two Python dictionary objects;
- The *source* and *target* in the links array are actually index pointers into the nodes array. Therefore, we must correlate the pointers to retrieve the correct node ID and name attribute. It should be noted that despite the prefix of *Actor*, the node number corresponds to Zachary's nomenclature for identifying each student in the karate club;
- Write the name attribute of the source and target nodes and the faction to a CSV file.

The CSV file was used as input data for processing using the D3[1] JavaScript library. The data includes the source node, target node, faction of the source and faction of the target. A snippet of the generated data file is shown below:

```
source,target,sfaction,tfaction
Actor 13,Mr Hi,1,1
Actor 13,Actor 4,1,1
Actor 14,Mr Hi,1,1
Actor 14,Actor 2,1,1
Actor 14,Actor 3,1,1
Actor 14,Actor 4,1,1
```

We chose to visualize the karate club as a Force Layout diagram. This type of diagram is well-suited for networks since it allows us to easily represent the nodes and links between nodes (edges) as a graph. The D3 source, shown in Appendix B, performs the following tasks:

- Read and parse the CSV file;
- Create a *nodes* and *links* array using the source and target data from the CSV file;
- Assign a color to each node based on its associated faction;
- Establish the configuration parameters for the visualization (e.g., height, width, radius);
- Invoke the D3 construct which creates the force layout on our screen canvas;
- We experimented with combinations of values for the gravity, link distance, and charge parameters until we obtained a layout which seemed visually appealing. These parameters essentially control the initial placement of nodes on the diagram;
- Modify each node so its *name* is displayed in the center of the node;
- Add a button so we can toggle between the *before* and *after* structure of the karate club; and
- Include a drag function so we can manually re-position the nodes on the diagram to minimize overlapping edges and increase the spacing between nodes for enhanced readability.

The initial invocation of the D3 source produces the original layout, Figure 1, for the karate club with the links representing the number of interactions between members. When the *toggle* button is clicked, each node is then filled with a color that shows how the members were clustered into communities after the split, Figure 2. The D3 source for this assignment can be executed from this web site: `http://www.cs.odu.edu/~cmccoy/karate/`.

# 2 Question 2 Extra Credit

## 2.1 Problem

Use D3 to create a who-follows-whom graph of your Twitter account. Use my twitter account ("@phonedude_mln") if you do not have an interesting number of followers.

## 2.2 Response

Did not attempt.

[1] `http://d3js.org/`

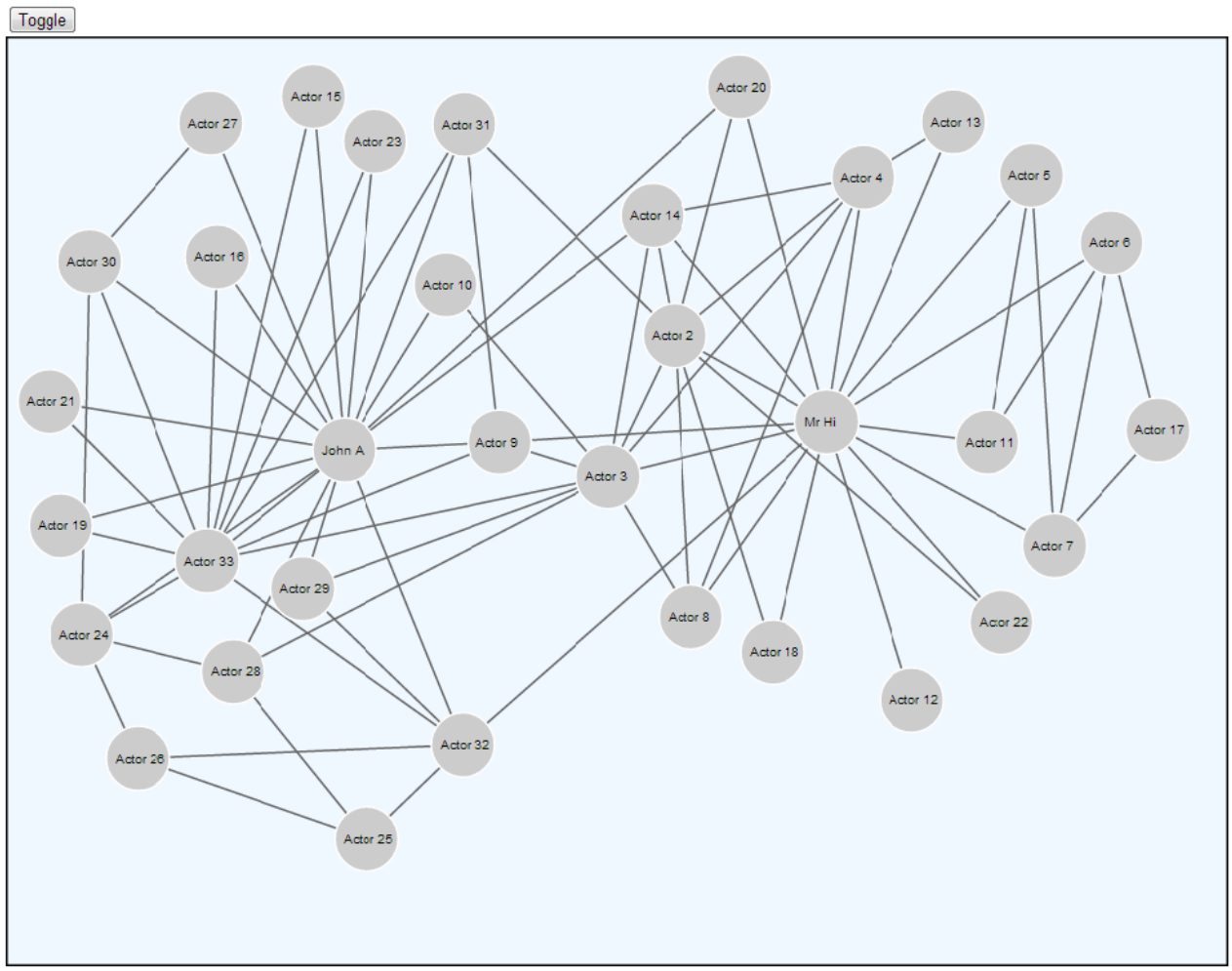


Figure 1: Zachary's Karate Club (Before)

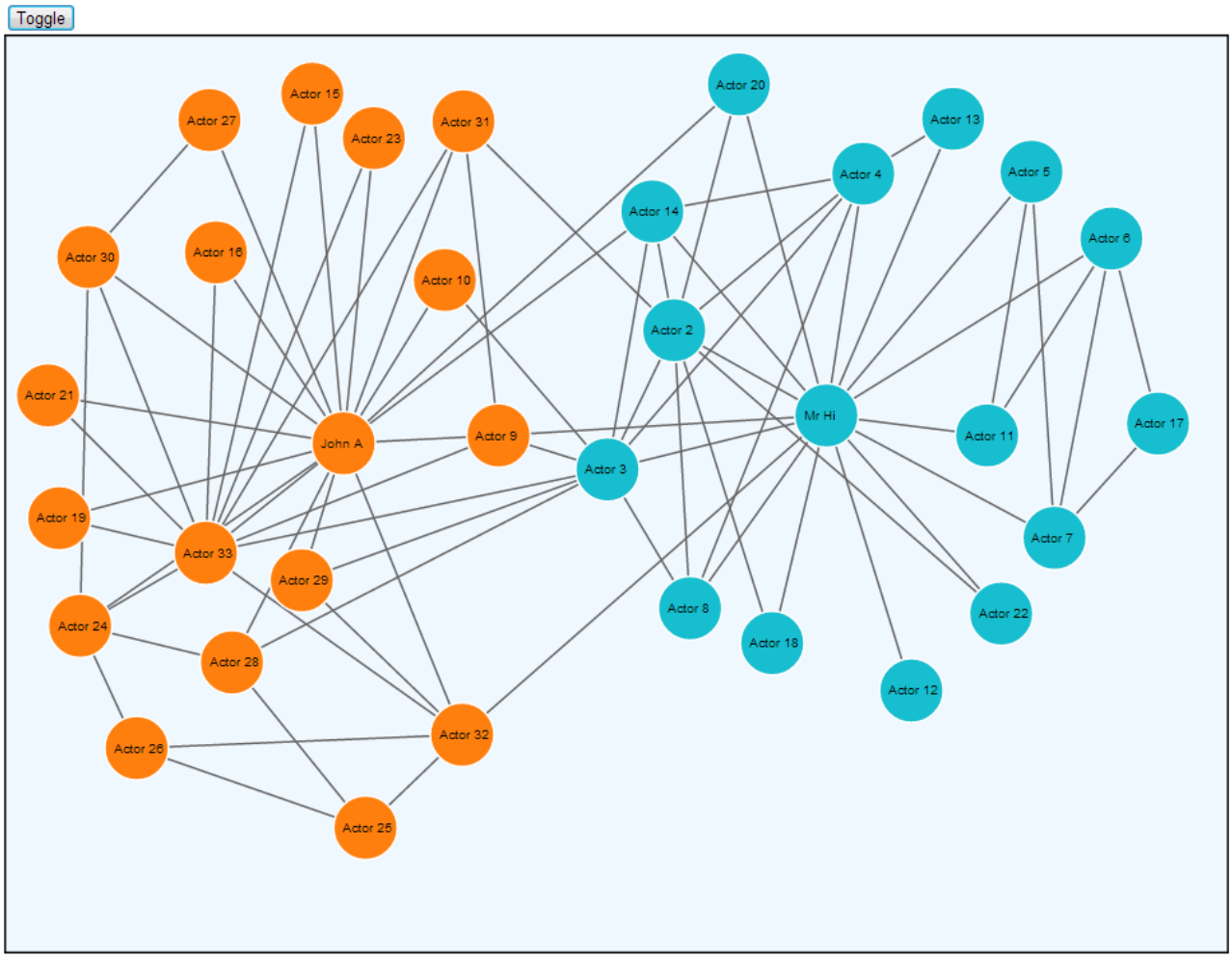


Figure 2: Zachary's Karate Club (After)

# Appendix A

# Python Source Code

```
import json
import csv
import unicodecsv
import networkx as nx
from networkx.readwrite import json_graph

'''
We are given Zachary's karate club network in a GraphML file. The file
metadata describes:
Edge attribute 'weight' based on the number of common interactions.
Vertex attribute 'Faction' either 1 or 2, giving the faction of the
student after the split of the club.
Vertex attribute 'name' for Author name.

This function will convert the file to two JSON files for the nodes and links. The data from
the two files is combined into a single CSV file that will be the source for D3 graphing.

Author:  Corren McCoy, 2013
'''


# GraphML format retrieved from the Nexus repository
# http://www.nexus.igraph.org/api/dataset_info?id=1&format=html
G=nx.read_graphml("C:/Python27/myFiles/Assignment 7/karate.GraphML")
# write json formatted data
d = json_graph.node_link_data(G) # node-link format to serialize
# write json
json.dump(d, open('C:/Python27/myFiles/Assignment 7/karate.json','w'))
jsonFile = open('C:/Python27/myFiles/Assignment 7/karate.json')
data = json.load(jsonFile)

# elements of interest for D3
nodes = data['nodes']
links = data['links']

# print json-formatted string
```

```
print json.dumps(nodes, sort_keys=True, indent=4)

# CSV file
nodeFile =open('C:/Python27/myFiles/Assignment 7/karateNodes.csv','wb+')
nodeFileCSV=csv.writer(nodeFile, delimiter=',')
index=0
node_dict={}
node_faction_dict={}
for x in nodes:
    print x['Faction'], x['id'], x['name']
    nodeFileCSV.writerow([index, x['Faction'], x['id'], x['name']])
    node_dict[index]=x['name']
    node_faction_dict[index]=x['Faction']
    index = index+1
nodeFile.close()

# print json-formatted string
print json.dumps(links, sort_keys=True, indent=4)
# CSV file
linkFile =open('C:/Python27/myFiles/Assignment 7/karateLinks.csv','wb+')
linkFileCSV=csv.writer(linkFile, delimiter=',')
for x in links:
    #print x['source'], x['target']
    source = int(x['source'])
    target=int(x['target'])
    linkFileCSV.writerow([node_dict[source], node_dict[target],
    node_faction_dict[source], node_faction_dict[target]])

linkFile.close()
```

# Appendix B

# D3 Source Code

```
<!DOCTYPE html>
<meta charset="utf-8">
<head>
   <title>Corren McCoy, CS895, Assignment 7</title>
<!--[if IE]><script src="../karate/js/excanvas.js"></script><![endif]-->
<script src="http://d3js.org/d3.v3.js"></script>
<style>
.link {
  fill: none;
  stroke: #666;
  stroke-width: 1.5px;
}

circle {
  fill: #ccc;
  stroke: #fff;
  stroke-width: 1.5px;
}

text {
  fill: #000;
  font: 10px sans-serif;
  pointer-events: none;
}
.node.fixed {
  fill: #f00;
}
rect {
  fill: aliceblue;
  stroke: #000;
  stroke-width: 3px;
}
.faction {
  fill: green;
  stroke: #000;
  stroke-width: 3px;
```

```
}

</style>
<body>
<button>Toggle</button>
<script>
// Sample D3: http://www.d3noob.org/2013/03/d3js-force-directed-graph-example-basic.html

// Get the data generated by the Python conversion function
d3.csv("data/karateLinks.csv", function(error, links) {

// empty container for nodes
var nodes = [];

// Compute the distinct nodes from the links.
// This block of code looks through all of out data from our csv file and for
// each link adds it as a node if it hasn't seen it before.
// Shorthand: *If* `link.source` *does not equal any of the `nodes`
// values then create a new element in
// the `nodes` object with the name of the `link.source` value being considered.*.
// Then the block of code goes on to test the `link.target` value in the s
// ame way. Then the `faction`  variable is converted to a number from
// a string if necessary (`link.faction = +link.faction;`).

links.forEach(function(link) {
// Use the node's faction setting of 1 or 2 as an index to select the color.
// Note: array indexing is zero based
var factionColor = ["red", "#17becf", "ff7f0e"];

    link.source = nodes[link.source] ||
        (nodes[link.source] = { name: link.source,
faction: link.sfaction,
color: factionColor[link.sfaction]});
    link.target = nodes[link.target] ||
        (nodes[link.target] = { name: link.target,
faction: link.tfaction,
color: factionColor[link.tfaction]});
    link.faction = +link.faction;
});

// sets the size of our svg area that we'll be using.
var width = 960,
    height = 720;

// .nodes(d3.values(nodes)) sets our layout to the array of `nodes`
// as returned by the function `d3.values`
// (https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_values).
// Put simply, it sets the nodes to the `nodes`
// we have previously set in our object.
```

```
// .links(links) does for links what ‘.nodes‘ did for nodes.
var force = d3.layout.force()
    .nodes(d3.values(nodes))
    .links(links)
    .size([width, height])
    .linkDistance(200)
    .charge(-300)
    .gravity(0.06)
    .on("tick", tick)
    .start();

// Sticky mode so node doesn't move after re-positioning
var drag = force.drag()
    .on("dragstart", dragstart);

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height)

// boundary of screen canvas
svg.append("svg:rect")
    .attr("width", width)
    .attr("height", height)
.attr("class","rect");

// add the links and the arrows
var link = svg.selectAll(".link")
    .data(force.links())
  .enter().append("line")
    .attr("class", "link");


// define the nodes
var node = svg.selectAll(".node")
    .data(force.nodes())
  .enter().append("g")
    .attr("class", "node")
    .call(force.drag);

// add the nodes
node.append("circle")
    .attr("r", 25);

// add the text for each node
node.append("text")
    .attr("dx", -18)
    .attr("dy", ".35em")
    .text(function(d) { return d.name; });
```

```
d3.select('button').on('click', function(d) {
// Make the nodes change color to show the split
d3.selectAll("circle")
.style("fill", function(d) {return  d.color;}) ;
})

// add the link lines
function tick() {
  link
      .attr("x1", function(d) { return d.source.x; })
      .attr("y1", function(d) { return d.source.y; })
      .attr("x2", function(d) { return d.target.x; })
      .attr("y2", function(d) { return d.target.y; });

node
      .attr("transform", function(d) { return "translate(" + d.x + "," + d.y + ")"; });
}

// keep nodes in position after dragging
function dragstart(d) {
  d.fixed = true;
  d3.select(this).classed("fixed", true);
} // dragstart


}); // end go d3.csv

</script>

</body>
</html>
```