Transition Paths of Karate Club Network

In [1]:
```
import scipy.io as sio
```

In [2]:
```
!wget "http://math.stanford.edu/~yuany/course/data/karate.mat" -O 'dat
```

```
--2017-11-11 01:36:20--  http://math.stanford.edu/~yuany/course/data
/karate.mat (http://math.stanford.edu/~yuany/course/data/karate.mat)
Resolving math.stanford.edu... 171.64.38.20
Connecting to math.stanford.edu|171.64.38.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 377 [text/plain]
Saving to: ‘data/karate.mat’

data/karate.mat     100%[===================>]     377  --.-KB/s
in 0s

2017-11-11 01:36:21 (39.9 MB/s) - ‘data/karate.mat’ saved [377/377]
```

In [3]:
```
data = sio.loadmat('data/karate.mat')
```

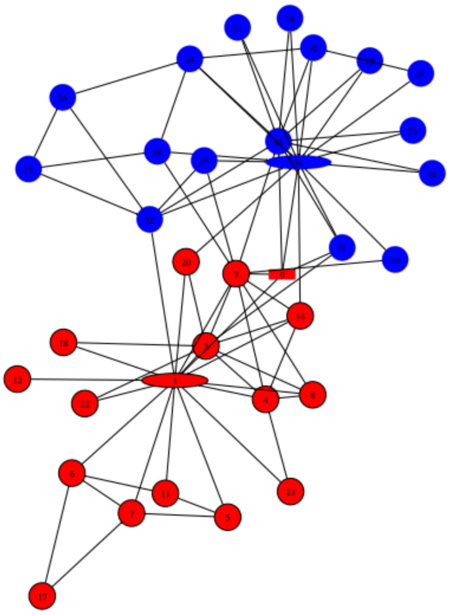
```
In [4]: data
```

```
Out[4]: {'A': array([[0, 1, 1, ..., 1, 0, 0],
                [1, 0, 1, ..., 0, 0, 0],
                [1, 1, 0, ..., 0, 1, 0],
                ...,
                [1, 0, 0, ..., 0, 1, 1],
                [0, 0, 1, ..., 1, 0, 1],
                [0, 0, 0, ..., 1, 1, 0]], dtype=uint8),
          '__globals__': [],
          '__header__': 'MATLAB 5.0 MAT-file, Platform: MACI64, Created on: T
         hu Oct 20 20:00:58 2011',
          '__version__': '1.0',
          'c0': array([[0],
                [0],
                [0],
                [0],
                [0],
                [0],
                [0],
                [0],
                [0],
                [1],
                [0],
                [0],
                [0],
                [0],
                [1],
                [1],
                [0],
                [0],
                [1],
                [0],
                [1],
                [0],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1],
                [1]], dtype=uint8)}
```

Dataset

The following dataset contains a 34-by-34 adjacency matrix A of Zachery's Karate Club Network.

As shown in Figure 1, node 1 represents the coach of the club and node 34 is the owner (president) of the club. The undirected, unweighted edges between nodes represent the affinity relation between club members. The story behind the network is this: the coach would like to raise the instruction fee while the president does not allow this; the conflicts finally result in a fission of the club – the coach leaves the club with his funs and sets up his own club marked in red, and the blue nodes remain in the old club with the president.



In [5]:
```
import numpy as np
matrix = np.array(data['A'])
print matrix.shape
matrix
```

```
(34, 34)
```

Out[5]:
```
array([[0, 1, 1, ..., 1, 0, 0],
       [1, 0, 1, ..., 0, 0, 0],
       [1, 1, 0, ..., 0, 1, 0],
       ...,
       [1, 0, 0, ..., 0, 1, 1],
       [0, 0, 1, ..., 1, 0, 1],
       [0, 0, 0, ..., 1, 1, 0]], dtype=uint8)
```

Adjency Matrix View

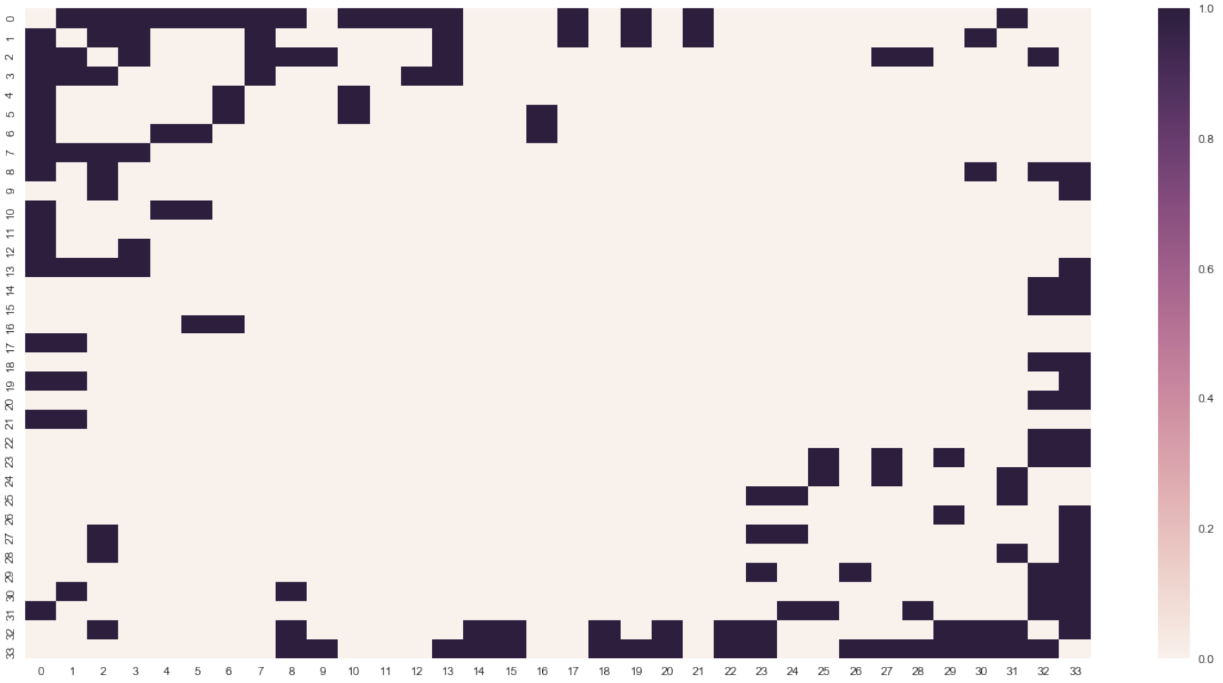
In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [7]:
```
for a in matrix:
    print a
```

```
[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
]
[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 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
]
[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
]
[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 1 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 1 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 0 0 1 1 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 1 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 1 1 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 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 1 0 0 1 0 0 0 0 0 1 1
]
[0 1 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 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 1 1 0 0 1 0 0 0 1 1
]
[0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 1
]
[0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0
]
```

In [8]:
```
plt.figure(figsize=(20,10))
sns.heatmap(matrix)
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x113e28fd0>
```



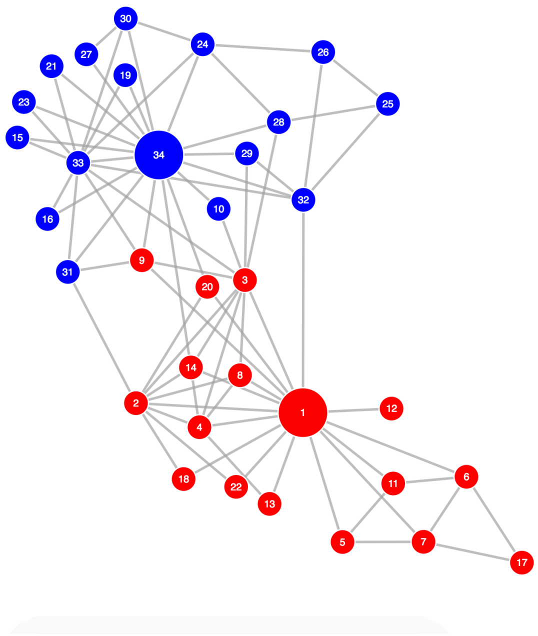
In [9]:
```
RED_LIST = sorted([17,7,5,11,6,13,22,12,1,4,8,18,2,3,20,14,9])
BLUE_LIST= sorted([32,31,10,34,33,29,25,28,16,23,26,24,21,19,30,15,27]
print RED_LIST
print BLUE_LIST
print len(RED_LIST) + len(BLUE_LIST)
```

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

Force Layout

In [10]:
```
nodes = [{'id':a, 'group':'red'} for a in RED_LIST] + [{'id':a, 'group
links = []
for index, a in enumerate(matrix):
    for index2, b in enumerate(a):
        if b and index > index2:
            links.append({
                'source':index+1,
                'target':index2+1,
                'value':1
            })
graph = {
    'nodes':nodes,
    'links':links
}
```

In [11]:
```
import json
with open('karate.json', 'w') as outfile:
    json.dump(graph, outfile)
```



Save

In [12]:
```
all_data = {
    'graph':graph,
    'matrix':matrix,
    'red_list':RED_LIST,
    'blue_list':BLUE_LIST
}
```

In [13]:

```
import pickle
pickle.dump(all_data, open( "data/karate_cleaned.p", "wb" ))
```

In [ ]: