From the data provided for Zachary’s Karate Club we know that the initial size of the data was 34 members and the volume was 78 edges. However, when the split occurred it resulted into two groups Group 1 which had 16 members and group 2 which had 18 members.

The groups and members is as following:

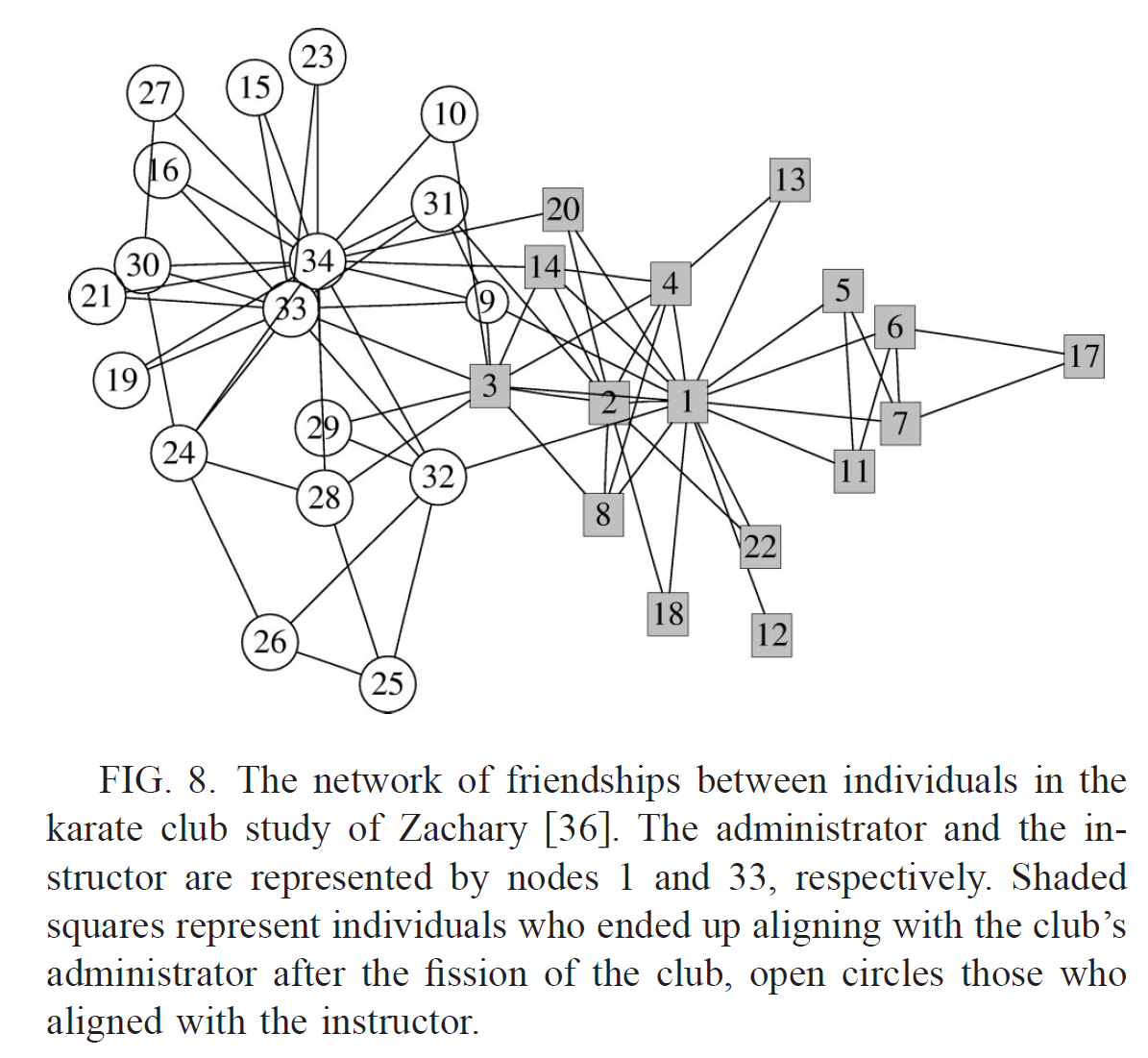
Group 1: (16 members)

1,2,3,4,5,6,7,8,11,12,13,14,17,18,20,22

Group2: (18 members)

9,10,15,16,19,21,23,24,25,26,27,28,29,30,31,32,33,34

Zachary’s resulting split graph:



As a solution to determain if the split was predictable or not I used “Igraph” ; which is a collection of software packages for graph theory and network analysis. I downloaded the karate club data from the source file <http://igraph.sourceforge.net/karate.net>, and I took two approaches:

Method One:

Algorithm name: Girvan–Newman algorithm

It is one of the methods used to detect communities. The method focuses on the edges that are the least central. This method is preformed by removing edges from the graph until the graph is divided into different graphs.

The algorithm's steps:

1-The betweenness of all existing edges in the network is calculated first.(the number of shortest paths between pairs of nodes that run along it)

2-The edge with the highest betweenness is removed.

3-The betweenness of all edges affected by the removal is recalculated.

4-Steps 2 and 3 are repeated until no edges remain.

Function used in R:

edge.betweenness.community

R code:

# parts of the code was found at:

#<http://igraph.wikidot.com/community-detection-in-r>

#using igraph library

library(igraph)

#downloading the data

g <- read.graph("~/Desktop/Assignment6/karate.net", format="pajek")

ebc <- edge.betweenness.community(g)

mods <- sapply(0:ecount(g), function(i) {

g2 <- delete.edges(g, ebc$removed.edges[seq(length=i)])

cl <- clusters(g2)

if(no.clusters(g2)==2){

plot(g2)

print(cl)

}

})

The resulting output (from printing cl):

$membership

[1] 1 1 2 1 1 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2

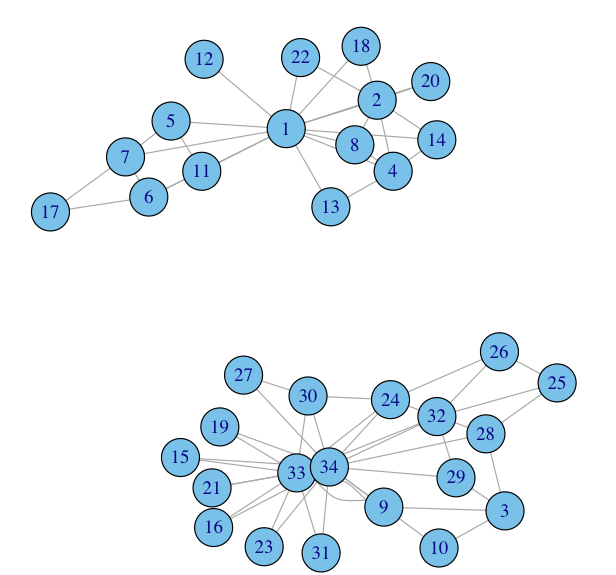
$csize

[1] 15 19

$no

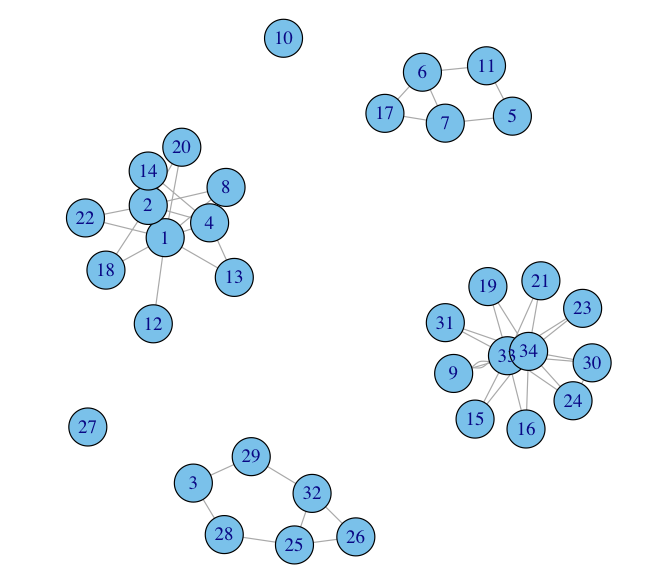
[1] 2

The resulting graph:



Result:

The actual betweenness result is 6 different communities; however in the code I limited it to 2 clusters and compared the resulting graph with Zacharys split.



> ebc

Graph community structure calculated with the edge betwenness algorithm

Number of communities (best split): 6

Modularity (best split): 0.3850263

Membership vector:

[1] 1 1 2 1 3 3 3 1 4 5 3 1 1 1 4 4 3 1 4 1 4 1 4 4 2 2 6 2 2 4 4 2 4 4

> modularity(g, membership(ebc))

[1] 0.3850263

By comparing the existing split and the split we got from the code. We notice that node 3 was in group 1 in Zacharys split but it was in group2 in our predicted split.

We compare the number of edges between the graph before and after the split (note that I split the graph to show each group separately)

> summary(g)

IGRAPH U--- 34 78 --

> summary(g2)

IGRAPH U--- 34 52 --

However, this method is slow there are *26* edges to be removed so total runtime is O(*262n*).

Method Two:

Algorithm name: leading eigenvector algorithm developed by Mark Newman.

This function tries to find densely connected subgraphs in a graph by calculating the leading non-negative eigenvector of the modularity matrix of the graph.

Algorithm Steps:

Calculating the eigenvector of the modularity matrix for the largest positive eigenvalue and then separating vertices into two community based on the sign of the corresponding element in the eigenvector. If all elements in the eigenvector are of the same sign that means that the network has no underlying community structure.

Function used in R:

leading.eigenvector.community

R code:

#The code found at:

#<http://igraph.sourceforge.net/screenshots2.html>

#using igraph library

library(igraph)

#downloading the data

g <- read.graph("("~/Desktop/Assignment6/karate.net ", format="pajek")

cs <- leading.eigenvector.community(g, steps=1)

V(g)$color <- ifelse(cs$membership==1, "lightblue", "green")

scale <- function(v, a, b) {

v <- v-min(v) ; v <- v/max(v) ; v <- v \* (b-a) ; v+a

}

#customizing the color

V(g)$size <- scale(abs(cs$eigenvectors[[1]]), 10, 20)

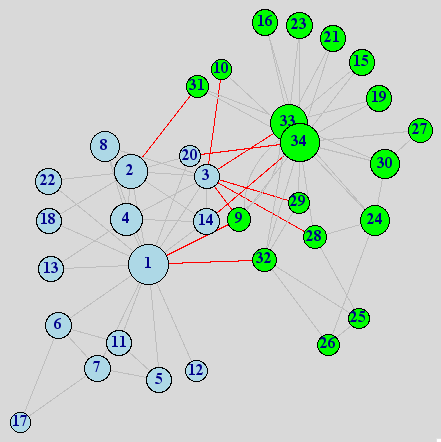
E(g)$color <- "grey"

E(g)[ V(g)[ color=="lightblue" ] %--% V(g)[ color=="green" ] ]$color <- "red"

#Drawing the graph using tkplot

tkplot(g, layout=layout.kamada.kawai, vertex.label.font=2)

The resulting graph:



Further Analysis:

> cs

Graph community structure calculated with the leading eigenvector algorithm

Number of communities (best split): 2

Modularity (best split): 0.3714661

Membership vector:

[1] 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2

> sizes(cs)

Community sizes

1 2

16 18

> modularity(g, membership(cs))

[1] 0.3714661

Result:

When analyzing this graph and comparing it to the split that occurred, first I compared the to group members I found its exactly the same.

Also, I have noticed that the tkplot editor of the R package is better than the X11 plot since you can move the nodes around to have a better view, it used for small graphs. The editor tkplot is downloaded from: <http://cran.r-project.org/bin/macosx/tools/>

Other community detection algorithms:

* walktrap.community

> walktrap.community(g, modularity=TRUE)

Graph community structure calculated with the walktrap algorithm

Number of communities (best split): 5

Modularity (best split): 0.3614399

Membership vector:

[1] 1 1 3 1 5 5 5 1 3 3 5 1 1 3 2 2 5 1 2 1 2 1 2 4 4 4 2 4 3 2 2 3 2 2

* spinglass.community

> spinglass.community(g, spins=10)

Graph community structure calculated with the spinglass algorithm

Number of communities: 4

Modularity: 0.4160092

Membership vector:

[1] 2 2 2 2 3 3 3 2 1 1 3 2 2 2 1 1 3 2 1 2 1 2 1 4 4 4 1 4 4 1 1 4 1 1

* label.propagation.community

> label.propagation.community(g)

Graph community structure calculated with the label propagation algorithm

Number of communities: 3

Modularity: 0.3744247

Membership vector:

[1] 1 1 2 1 3 3 3 1 2 2 3 1 1 1 2 2 3 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2

* fastgreedy.community

> fastgreedy.community(g)

Error in fastgreedy.community(g) :

At fast\_community.c:552 : fast-greedy community finding works only on graphs without multiple edges, Invalid value

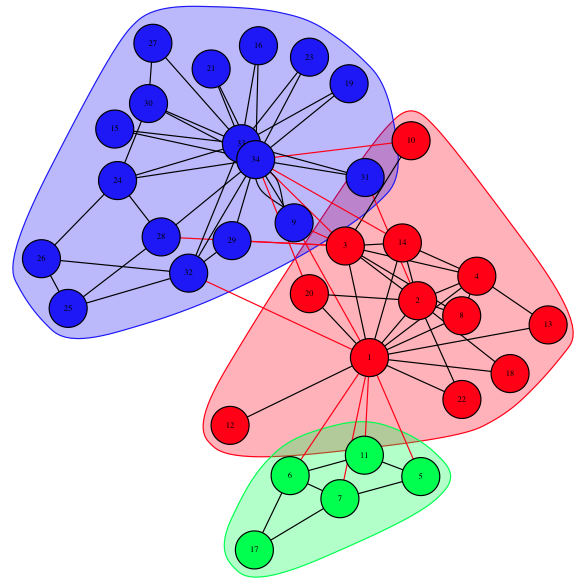
Q2-

Different community algorithms split the initial graph to different number of community. And by knowing the resulting number of communities for each (shown at end of question one), I have selected the algorithms that result in 3, 4, and 5 number of groups.

Three groups:

> G3 <- label.propagation.community(g)

> plot(G3,g,vertex.size=15,vertex.label.color= "black",vertex.label.cex=0.45,layout=layout.fruchterman.reingold)



The splits can be dont as well by the code in question1, but change the number of clustering needed.

Three:

library(igraph)

#downloading the data

g <- read.graph("~/Desktop/Assignment6/karate.net", format="pajek")

ebc <- edge.betweenness.community(g)

mods <- sapply(0:ecount(g), function(i) {

g2 <- delete.edges(g, ebc$removed.edges[seq(length=i)])

cl <- clusters(g2)

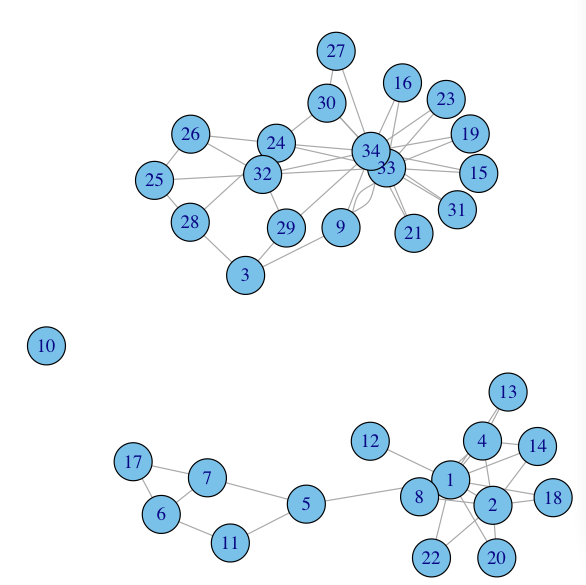
if(no.clusters(g2)==3){

plot(g2)

print(cl)

}

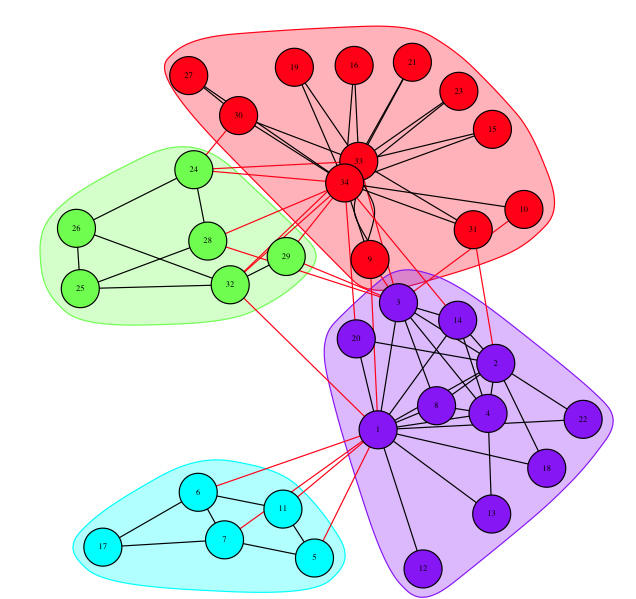
})



Four groups:

> G4 <- spinglass.community(g, spins=10)

> plot(G4,g,vertex.size=15,vertex.label.color= "black",vertex.label.cex=0.45,layout=layout.fruchterman.reingold)



Four:

library(igraph)

#downloading the data

g <- read.graph("~/Desktop/Assignment6/karate.net", format="pajek")

ebc <- edge.betweenness.community(g)

mods <- sapply(0:ecount(g), function(i) {

g2 <- delete.edges(g, ebc$removed.edges[seq(length=i)])

cl <- clusters(g2)

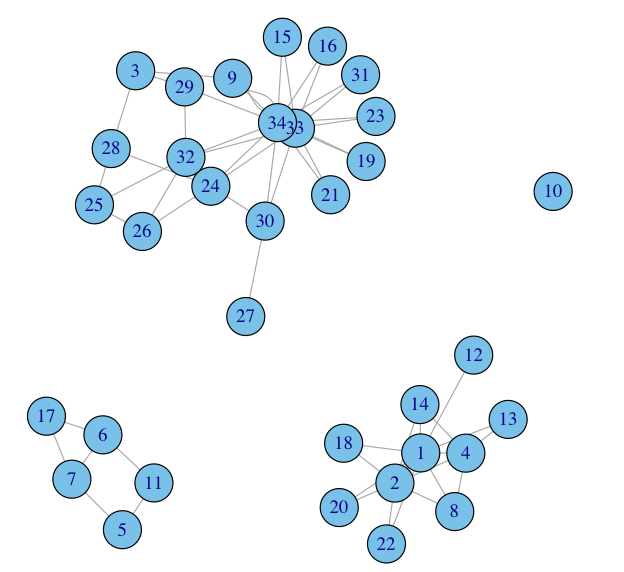
if(no.clusters(g2)==4){

plot(g2)

print(cl)

}

})



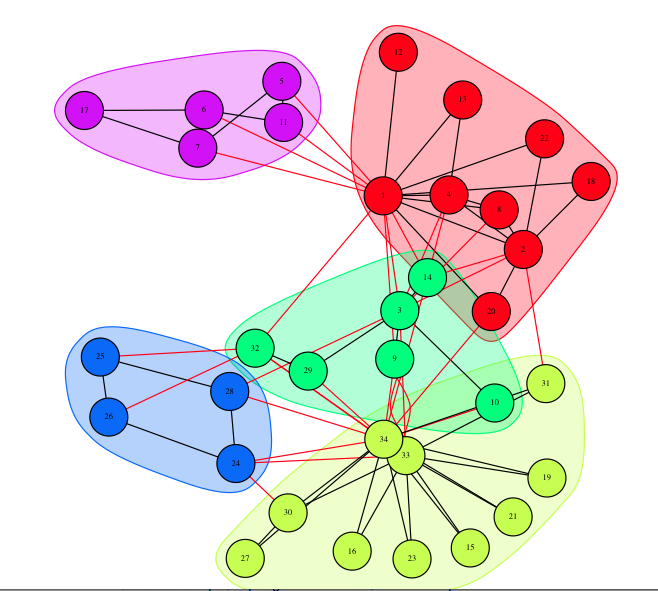
Five groups:

> G5 <- walktrap.community(g, modularity=TRUE)

> plot(G5,g,vertex.size=15,vertex.label.color= "black",vertex.label.cex=0.45,layout=layout.fruchterman.reingold)

>mycolors <- heat.colors(length(G5))

tkplot(g, vertex.color=mycolors[membership(G5)])



Five:

library(igraph)

#downloading the data

g <- read.graph("~/Desktop/Assignment6/karate.net", format="pajek")

ebc <- edge.betweenness.community(g)

mods <- sapply(0:ecount(g), function(i) {

g2 <- delete.edges(g, ebc$removed.edges[seq(length=i)])

cl <- clusters(g2)

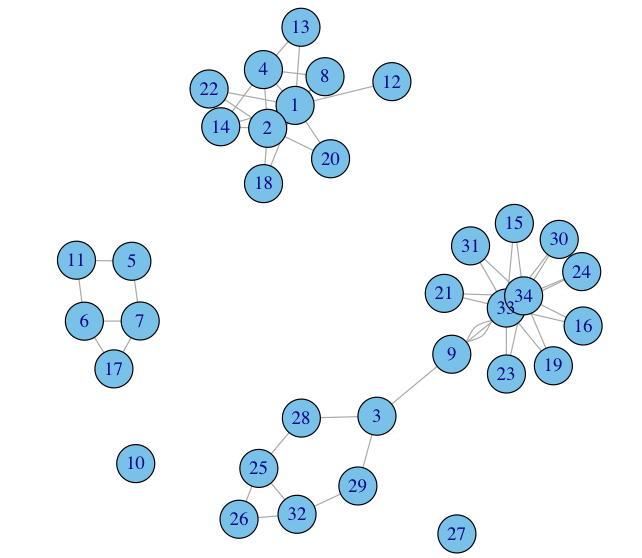
if(no.clusters(g2)==5){

plot(g2)

print(cl)

}

})

Resources

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

http://www-personal.umich.edu/~ladamic/courses/networks/si614w06/ppt/lecture18.ppt

http://clair.si.umich.edu/si767/papers/Week03/Community/CommunityDetection.pptx

http://networkx.github.io/documentation/latest/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

http://igraph.wikidot.com/community-detection-in-r

https://github.com/igraph/igraph/tree/master/interfaces/R/igraph/

http://books.google.com/books?id=atfCl2agdi8C&pg=PA70&lpg=PA70&dq=zachary+karate+club+predicted+split&source=bl&ots=LyTV0moGuw&sig=iOWdV68vQWRteDqJhCCuTO0usos&hl=en&sa=X&ei=GGtsUvG\_NO-WyAHFm4GIDw&ved=0CDEQ6AEwAA#v=onepage&q&f=false

http://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf

http://theory.ic.ac.uk/%7Etime/networks/communitystructure.html#louvain

http://igraph.sourceforge.net/doc-0.6/html/ch22s05.html

http://en.wikipedia.org/wiki/Girvan–Newman\_algorithm

https://github.com/kjahan/community

http://www.cse.iitk.ac.in/users/sigml/slides/s01e02.pdf

<http://igraph.sourceforge.net/doc/R/community.edge.betweenness.html>

<http://stackoverflow.com/questions/15787145/saving-the-plots-in-igraph-as-igraph-object-or-gml-format>