Stat6289 Homework 3

Chenrui Xu

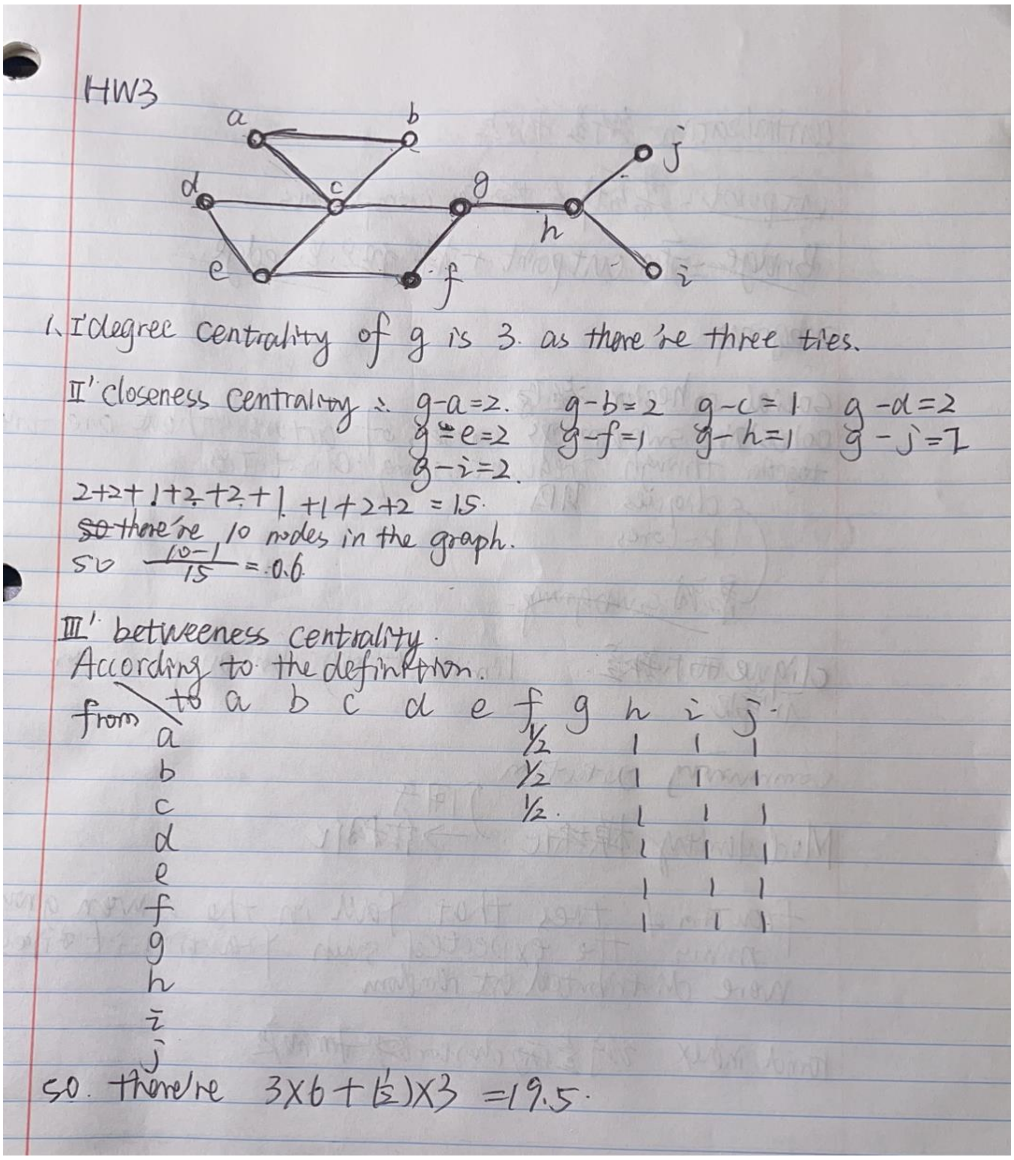
Background

This week, we want to explore the inner relationship of network. We talked about prominence, a few methods of centrality, cliques, k-cores and the most important one, modularity.

K-cores means that we use iteration to delete nodes that has degree less than k.

Modularity is an important network characteristic used in many community detection algorithms. It is a measure of network structure. The extent to which nodes show clustering patterns with greater density within each cluster and less density among different clusters. It is also a chance-corrected statistic while it is defined as the fraction of ties that fall within the given group minus the expected such fraction if ties were distributed at random. It ranges from -1 to 1 as when it close to 1, it means that the more the network shows clustering with respect to the given node grouping.

Part I and Part II


HW3

1. I' degree centrality of g is 3. as there're three ties.

II' closeness centrality ∴ g-a=2. g-b=2 g-c=1 g-d=2
g-e=2 g-f=1 g-h=1 g-j=2
g-i=2.

2+2+1+2+2+1 +1+2+2 = 15.
~~so~~ there're 10 nodes in the graph.
so $\frac{10-1}{15}$ = 0.6

III' betweeness centrality.
According to the definition.

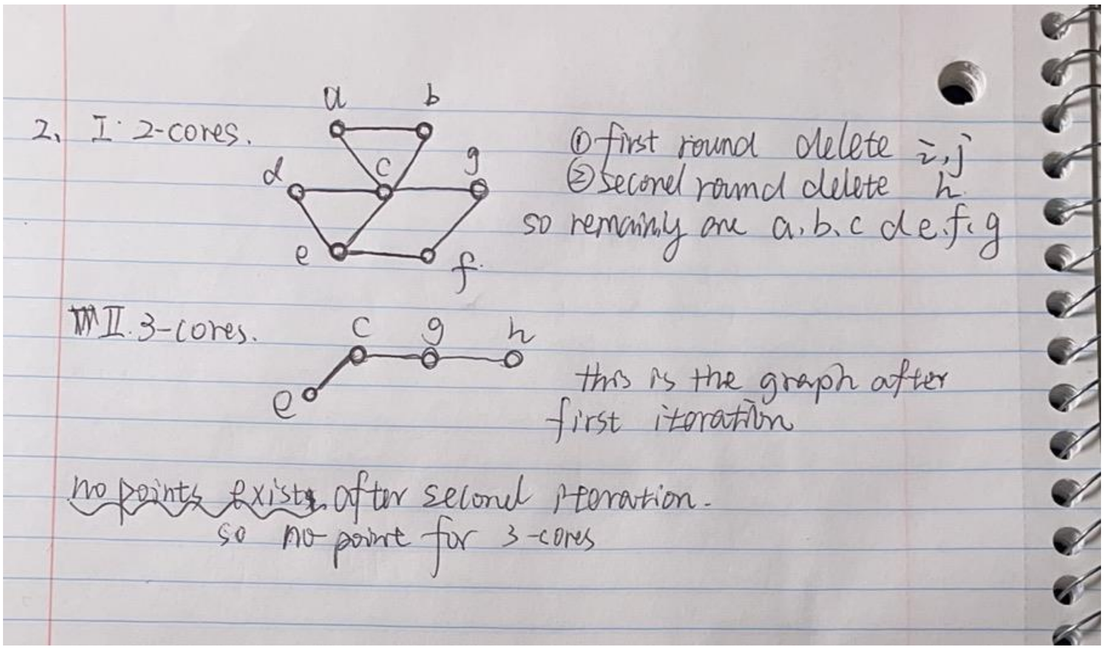
from \ to	a	b	c	d	e	f	g	h	i	j
a						½		1	1	1
b						½		1	1	1
c						½		1	1	1
d								1	1	1
e								1	1	1
f								1	1	1
g										
h										
i										
j										

so. there're 3×6+(½)×3 = 19.5.

The picture shown above is Question 1. The first one is very simple, what we need to do is counting how many edges connecting to vertex g and the answer is three.

In terms of closeness centrality, we add distinces together. As we have 10 nodes, we will use 10-1 and divide by 15 (0.6).

For first six nodes, there’s only one way to access node h/i/j with g on the way. That will count 18. Node a/b/c have two ways to get node f, one passing g. So the answer should be 18+(0.5)*3=19.5



Part two is shown above as a,b,c,d,e,f,g are the nodes contributing to the subgroup network in terms of 2-cores. And nothing for 3-cores.

Part III

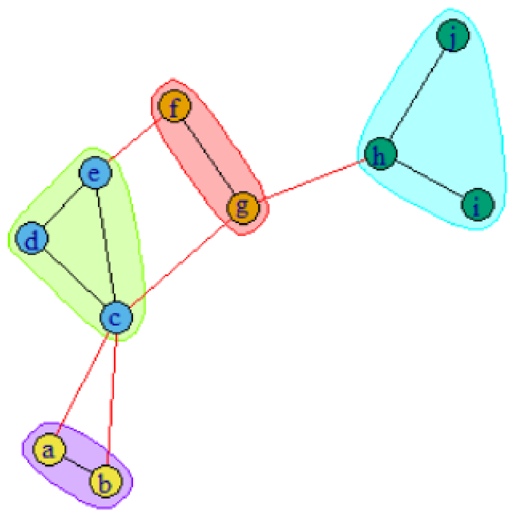
We applied kinds of methods of modularity to fit the network. Here is the result statistics

-no_cluster: -0.12
-cluster_walktrap: 0.2951389
-cluster_edge_betweenness: 0.3229167
-cluster_spinglass: 0.3125
-cluster_fast_greedy: 0.3229167
-cluster_label_prop: 0.2465278
-cluster_leading_eigen: 0.3229167
-cluster_louvain: 0.3229167
-cluster_optimal: 0.3298611
-cluster_infomap: 0.2465278

As when we first calculate modularity directly, it showed that V(inet) have ten levels as there are ten nodes, which means one node corresponding to one level. So, there is no doubt that the network is not likely to show cluster pattern at first. Then after we calculating the statistics, it is -0.12, which is same with the idea I mentioned above. Then I used different methods to cluster the network and found that the statistics are from 0.24 to 0.33, which means there's some kind of possibility for the network to have cluster pattern. It showed that the optimal method is the best one as the statistic is the largest one (0.3298). Also, what we can see from the membership is that, most of clustering methods divide network into three parts, and two of them have two clusters and four is the largest number of clustering.

Part IV

We apply R-function cluster_walktrap() to this network and the plot is shown below.



Appendix

````
---
title: "6289_HW4"
author: "Chenrui Xu"
date: "2021/2/21"
output: html_document
---

```{r}
library(statnet)
library(UserNetR)
library(igraph)
library(intergraph)
```

```{r}
netmat=rbind(
  # a b c d e f g h i j
  c(0,1,1,0,0,0,0,0,0,0),#a
  c(1,0,1,0,0,0,0,0,0,0),#b
  c(1,1,0,1,1,0,1,0,0,0),#c
  c(0,0,1,0,1,0,0,0,0,0),#d
  c(0,0,1,1,0,1,0,0,0,0),#e
  c(0,0,0,0,1,0,1,0,0,0),#f
  c(0,0,1,0,0,1,0,1,0,0),#g
  c(0,0,0,0,0,0,1,0,1,1),#h
  c(0,0,0,0,0,0,0,1,0,0),#i
  c(0,0,0,0,0,0,0,1,0,0)
)
```

```{r}
rownames(netmat)<-c('a','b','c','d','e','f','g','h','i','j')
colnames(netmat)<-c('a','b','c','d','e','f','g','h','i','j')
net=network(netmat,matrix.tpye="adjacency",directed = F)
#set.vertex.attribute(net,'name',c('a','b','c','d','e','f','g','h','i','j'))
```

```{r}
gplot(net,displaylabels = T)
```

Part 1
```{r}
sna::degree(net,gmode = "graph")
```

```{r}
````
````

sna::closeness(net,gmode="graph")
```

```{r}
sna::betweenness(net,gmode="graph")
```

Part 2
```{r}
inet=asIgraph(net)
coreness=graph.coreness(inet)
coreness
```

```{r}
inet2 <- induced.subgraph(inet, vids=which(coreness > 1))
plot(inet2)
```

Part 3
```{r}
table(V(inet))
V(inet)[1:10]
modularity(inet,V(inet))
```

```{r}
cw=cluster_walktrap(inet)
membership(cw)
modularity(cw)
```

```{r}
ceb=cluster_edge_betweenness(inet)
modularity(ceb)
membership(ceb)
```

```{r}
cs=cluster_spinglass(inet)
modularity(cs)
membership(cs)
```

```{r}
cfg=cluster_fast_greedy(inet)
modularity(cfg)
membership(cfg)
```
```

```{r}
clp=cluster_label_prop(inet)
modularity(clp)
membership(clp)
```

```{r}
cle=cluster_leading_eigen(inet)
modularity(cle)
membership(cle)
```

```{r}
cl=cluster_louvain(inet)
modularity(cl)
membership(cl)
```

```{r}
co=cluster_optimal(inet)
modularity(co)
membership(co)
```

```{r}
im=cluster_infomap(inet)
modularity(im)
membership(im)
```

```{r}
table(V(inet),membership(cw))
```

Part 4

```{r}
plot(cw,inet, vertex.label = net%v%'vertex.names')
```