Seminario de Modelos y Métodos Cuantitativos
Tarea 2-3

Universidad Técnica Federico Santa María - Campus San Joaquín

9 de diciembre de 2015

Profesor Andrés Moreira

Juan Pablo Escalona G.
juan.escalonag@alumnos.usm.cl
201073515-k

Alonso Lepe M.
alonso.lepe@alumnos.usm.cl
201173593-5

1. Pregunta 1

Se tiene un grafo $G(n = 400, p = 1)$ esto implica que hay 79800 aristas que conectan a los 400 nodos entre todos. i.e. $\frac{399 \cdot 400}{2} = 79800$

Para que una componente conexa sea gigante debe tener un $k = np > 1$, en este caso $k = 400$, como se necesita un $k <= 1$ para que deje de ser gigante, entonces p debería ser a lo mas $\frac{1}{400} = 0{,}0025$

Si se elimina una arista en el segundo 00:00:01 entonces aún quedan 79799 aristas, es decir, una probabilidad de $p = \frac{79799}{79800} = 0{,}9999$. Pero se sabe que se necesita una probabilidad de 0.0025, por lo que la componente dejará de ser conexa cuando solo hayan $\frac{E}{79800} = 0{,}0025 \implies E \sim 200$ aristas.

Luego si cada segundo se elimina 1 arista y es preciso eliminar 79600 aristas entonces se tendría que esperar hasta las 22:06:40 (10 de la noche, 6 minutos y 40 segundos)

Sobre este análisis se realizó el código 1 del anexo en el que se crea una red ER de 400 nodos y probabilidad 1. Luego se comenzó a eliminar nodos hasta lograr un grado promedio ¡1. Se ejecutó el código en un computador con un 2.5 GHz Intel Core i5 y 8 GB ram, tardó 8 minutos 29 segundos en eliminar aristas hasta lograr un grado promedio de 1. Finalmente se eliminaron 79600 aristas por lo que empíricamente también terminó a las 22:06:40

2. Pregunta 2

La Matriz laplaciana de un grafo se puede obtener a partir de la matriz de adyacencia siguiendo el siguiente formato:

$$l_{i,j} = \begin{cases} k_i & \text{si } i = j \\ -1 & \text{si } i \neq j \\ 0 & \text{e.t.o.c.} \end{cases} \tag{1}$$

que es lo mismo que decir $l = K - A$ donde K es la matriz diagonal de los grados de cada nodo y A es la matriz de adyacencia. Luego para la matriz de adyacencia dada se tiene la siguiente matriz laplaciana:

$$l = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{2}$$

l valor de Fiedler es el segundo valor propio mas pequeño, en este caso corresponde a 0.438 ubicado en la tercera columna como se muestra en las ecuaciones (3) y (4).

$$V = \begin{bmatrix} 0{,}185 & -0{,}408 & \mathbf{0.465} & 0{,}378 & -0{,}756 & 0{,}040 \\ 0{,}185 & -0{,}408 & \mathbf{0.465} & -0{,}734 & 0{,}463 & 0{,}238 \\ -0{,}657 & -0{,}408 & \mathbf{0.261} & 0{,}357 & 0{,}292 & -0{,}278 \\ 0{,}657 & -0{,}408 & \mathbf{-0.261} & 0{,}357 & 0{,}292 & -0{,}278 \\ -0{,}185 & -0{,}408 & \mathbf{-0.465} & -0{,}189 & -0{,}148 & -0{,}472 \\ -0{,}185 & -0{,}408 & \mathbf{-0.465} & -0{,}168 & -0{,}145 & 0{,}751 \end{bmatrix} \tag{3}$$

$$E = \begin{bmatrix} 4{,}562 & 0{,}000 & \mathbf{0.438} & 3{,}000 & 3{,}000 & 3{,}000 \end{bmatrix} \tag{4}$$

Es importante ordenarlos para obtener el valor de Fiedler, pues este es el segundo menor valor propio.

$$\text{Valor de Fiedler: } 0{,}4384 \tag{5}$$

$$\text{Vector de Fiedler: } \begin{bmatrix} 0{,}465 & 0{,}465 & 0{,}261 & -0{,}261 & -0{,}465 & -0{,}465 \end{bmatrix} \tag{6}$$

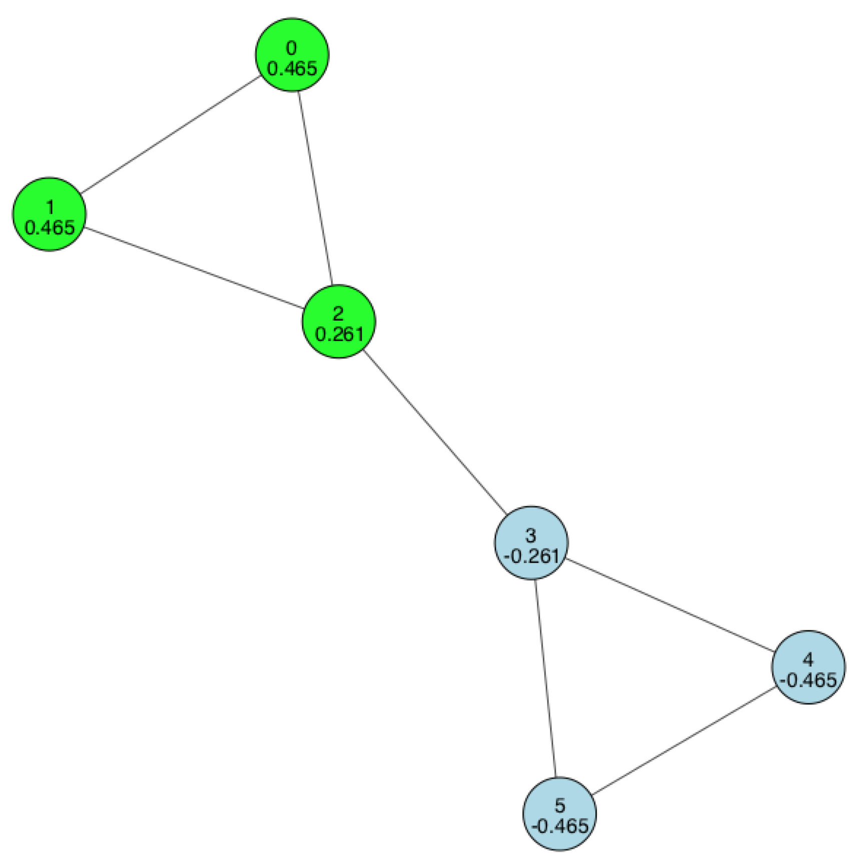

Figura 1: Partición de la red en dos comunidades mostrando sus respectivos valores del vector propio.

Se decidió cortar en el 0, pues los valores se separan uniformemente en torno al este. Se observan claramente dos comunidades en esta red las que interactúan únicamente por el vértice 2-3.

3. Pregunta 3

Dos nodos de un grafo forman un *2-componente* si existen al menos dos caminos independientes entre ellos. Un camino es una secuencia finita de aristas que permiten ir de un nodo inicial a un nodo final dentro de un grafo.

Un *2-componente* o *biconnected component* tienen la particularidad que si se elimina una arista del grafo, aún sería posible llegar de un nodo a otro, pues originalmente existían dos posibles caminos.

Por otro lado un *2-core* es un conjunto de nodos donde cada uno está conectado a los demás con al menos 2 aristas. Un *2-core* solo considera los nodos que estén conectados con una arista dentro del conjunto, si algún nodo tiene una arista a un nodo fuera del conjunto, dicha arista no se cuenta.

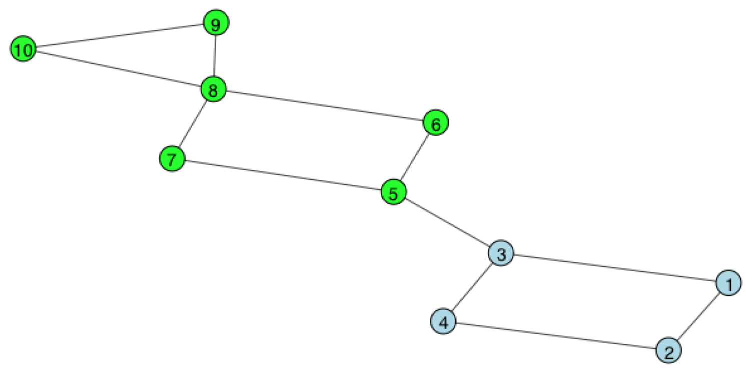

Figura 2: Grafo 2-core con dos 2-componentes.

En la figura 2 se aprecia un grafo con un solo conjunto *2-core* con la totalidad de los nodos, pues todos los nodos del grafo están conectados a los demás con al menos 2 aristas.

Por otra parte existen dos *2-componentes* destacados por los colores. En este caso el *2-componente* $[1, 2, 3, 4]$ y el componente $[5, 6, 7, 8, 9, 10]$. No puede existir un 2-componente entre el nodo 1 y el nodo 8, pues si o si hay que pasar por la arista 3-5 por lo que sería el mismo camino. Si se quita dicha arista no hay otra forma de llegar de 1 a 8.

4. Pregunta 4

4.1. Grafico de la red

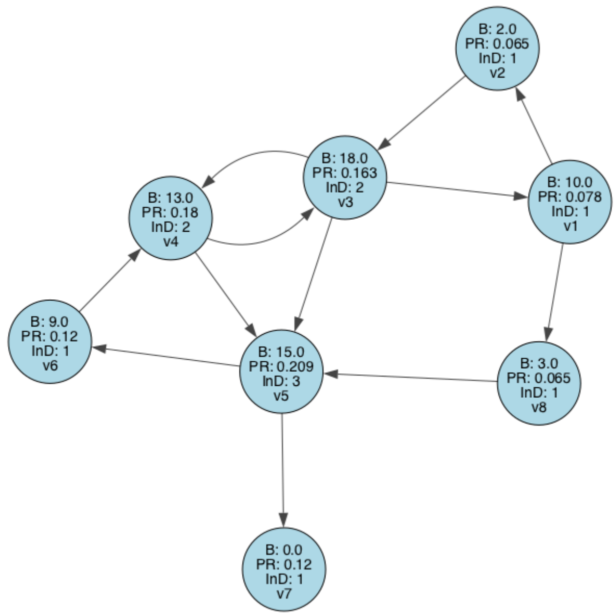

Figura 3: Red chica dirigida. B: Betweenness, PR: PageRank, InD: Grado de entrada.

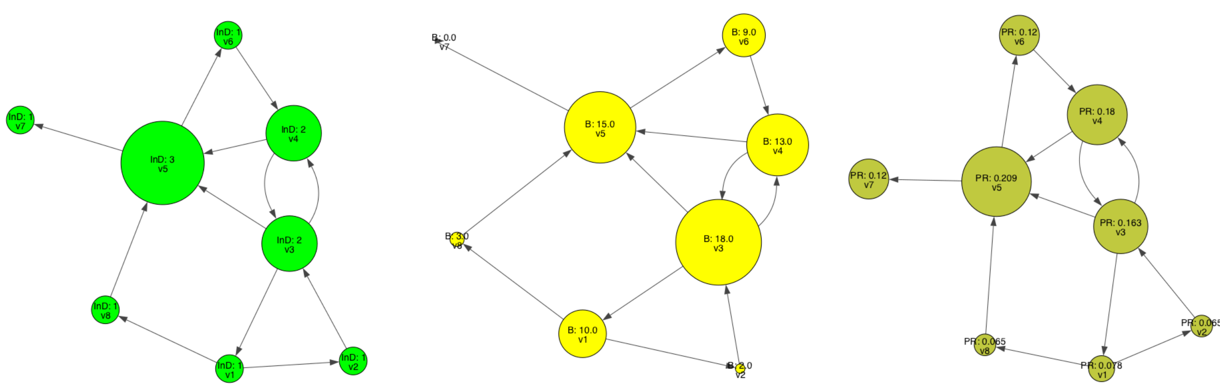

Figura 4: Red chica según el tamaño porcentual para cada indice.

4.2. Ranking de los índices

Grado Entrada			Betweenness			PageRank		
Posición	Nodo	Valor	Posición	Nodo	Valor	Posición	Nodo	Valor
1	v5	3	1	v3	18	1	v5	0.209
2	v4	2	2	v5	15	2	v4	0.180
2	v3	2	3	v4	13	3	v3	0.163
3	v1	1	4	v1	10	4	v6	0.120
3	v2	1	5	v6	9	4	v7	0.120
3	v6	1	6	v8	3	5	v1	0.078
3	v7	1	7	v2	2	6	v2	0.065
3	v8	1	8	v7	0	6	v8	0.065

Cuadro 1: Ranking de cada índice para la red chica.

5. Referencias

6. Apéndice

Listing 1: p1.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import igraph as G
import random

g = G.GraphBase.Erdos_Renyi(400, 1)

deleted = 0
while G.mean(g.degree()) > 1:
    deleted += 1
    g.delete_edges(random.randint(0, g.ecount() - 1))

print "Removed", deleted, "edges"
```

Listing 2: p2.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import igraph as G
import numpy as np

# Build the graph
p2 = G.Graph([(0,1),(0,2),(1,2),(2,3),(3,4),(3,5),(4,5)])

laplacian_matrix = p2.laplacian()
w, v = np.linalg.eig(p2.laplacian()) # Eigen values (w) and eigen vectors (v)
fiedler_value = sorted(w)[1] # Fiedler value is the second smallest eigenvalues
fiedler_vector = v[:, np.where(w == fiedler_value)[0]].T[0] # grab the n-th column of the eigenvectors,
    where n is the index of the Fiedler Value

print "Matriz␣Laplaciana:"
print np.array(laplacian_matrix)

print "Valores␣Propios", [round(x, 3) for x in sorted(w)]
print "Valor␣de␣Fiedler:", round(fiedler_value,4)
```

```
print "Vector␣de␣Fiedler:", [round(i, 3) for i in fiedler_vector]

# Plot the graph
p2.vs["label"] = [str(j) + "\n" + str(round(i, 3)) for j, i in enumerate(fiedler_vector)]
p2.vs["color"] = ["lightblue" if x <= 0 else "green" for x in fiedler_vector]
G.plot(p2, layout=p2.layout("kk"), vertex_size=50, bbox=(600,600), margin=40)
```

Listing 3: p3.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import igraph as G

p1 = G.Graph([(0,1),(0,2),(1,3),(2,3),(2,4),(4,5),(4,6),(5,7),(6,7),(7,8),(7,9),(8,9)])
colors = ["lightblue", "green"]
p1.vs["color"] = [colors[i] for i in [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]]
p1.vs["label"] = range(1,11)
G.plot(p1, layout=p1.layout("kk"), bbox=(600,300))
```

Listing 4: p4.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import igraph as G
import numpy as np

# Build the graph
p2 = G.Graph.Read_GML("../archivos/redchica.gml")
G.summary(p2)
# Plot the graph

betweenness = p2.betweenness()
pageranks = [round(i, 3) for i in p2.pagerank()]
indegree = p2.degree(mode="in")
names = p2.vs["label"]
p2.vs["label"] = ["B:␣" + str(betweenness[i]) + "\nPR:␣" + str(pageranks[i]) + "\nInD:␣" + str(indegree[i]) + "\n" + names[i] for i in range(8)]
p2.es["width"] = 1
p2.vs["color"] = "lightblue"
p2.vs["size"] = 80
print sorted([(i, j) for i, j in enumerate(indegree)], key=lambda x: x[1], reverse=True)
print sorted([(i, j) for i, j in enumerate(betweenness)], key=lambda x: x[1], reverse=True)
print sorted([(i, j) for i, j in enumerate(pageranks)], key=lambda x: x[1], reverse=True)
G.plot(p2, "../img/p4-all.png", margin=50)

# plot indegree
p2.vs["label"] = ["InD:␣" + str(indegree[i]) + "\n" + names[i] for i in range(8)]
p2.vs['size'] = [500.0*i/sum(indegree) for i in indegree]
p2.vs["color"] = "green"
G.plot(p2, "../img/p4-indegree.png", margin=50)

# plot betweenness
p2.vs["label"] = ["B:␣" + str(betweenness[i]) + "\n" + names[i] for i in range(8)]
p2.vs['size'] = [500.0*i/sum(betweenness) for i in betweenness]
p2.vs["color"] = "yellow"
# G.plot(p2, "../img/p4-betweenness.png", margin=50)

# plot PageRank
p2.vs["label"] = ["PR:␣" + str(pageranks[i]) + "\n" + names[i] for i in range(8)]
p2.vs['size'] = [500.0*i/sum(pageranks) for i in pageranks]
p2.vs["color"] = "#C0C93F"
# G.plot(p2, "../img/p4-pageranks.png", margin=50)
```