Optimización de Flujo en Redes: Reporte 1.

Mayra Cristina Berrones Reyes.

Matricula: 1646291

Febrero 2018

Introducción

En este reporte se describirá el proceso que se llevó a cabo para realizar la primer práctica de la clase de optimización de flujo en redes. Las herramientas que se utilizaron para llevarla a cabo fueron:

- **Gnuplot.**
- **Python 3.**

Fase 1: Localizar los puntos.

En esta sección veremos lo primer para el desarrollo de nuestro programa, lo cual consiste en localizar coordenadas en una gráfica por medio de puntos. En este caso, lo que buscamos inicialmente fue hacer un archivo en el cual pudiéramos guardar una serie de números aleatorios, con los cuales formaríamos nuestras coordenadas x y y, con las cuales, finalmente, utilizaremos la herramienta de gnuplot para realizar una gráfica, en donde estos números aleatorios representaran puntos en ella.

```
// Puntos.py
from random import random #importar libreria para generar tus numeros
    aleatorios.
nodos = [] #matriz donde vas a ir creando num
n = 50 #cantidad de numeros que voy a generar de manera aleatoria

with open ("nodos.dat", 'w') as salida: #generamos un archivo de salida
    para mis datos.
   for nod in range (n): #Este for me genera dos numeros aleatorios n
       veces.
      x = random()
      y = random()
      nodos.append((x,y))
      print(x, y, file = salida)

with open ("nodos.plot", 'w') as salida: #Archivo para poder generar mi
    grafica de salida.
   #NOTA: Recuerda que siempre tienes que llamar al dato de salida para
       cada uno de los comandos de gnplot
   print ('set term png', file = salida)
   #png para formato de salida de mi grafica
   print ('set output "nodos.png"', file = salida)
   print ('set size square', file = salida)
   #Keyoff es para que la marca de agua que sale en la parte superior
       der. no se vea.
   print ('set key off', file = salida)
   #El rango x y y son para el tamanio en que va a salir mi grafica.
```

```
print ('set xrange [-.1:1.1]', file = salida)
print ('set yrange [-.1:1.1]', file = salida)
#Por ultimo, se manda a llamar el archivo en el cual colocamos los
    numeros aleatorios.
print ('plot "nodos.dat" using 1:2 with points pt 7', file = salida)
#En el 1:2 nos referimos a las columnas que tiene mi archivo, x y y.
print ('quit()', file = salida)
```

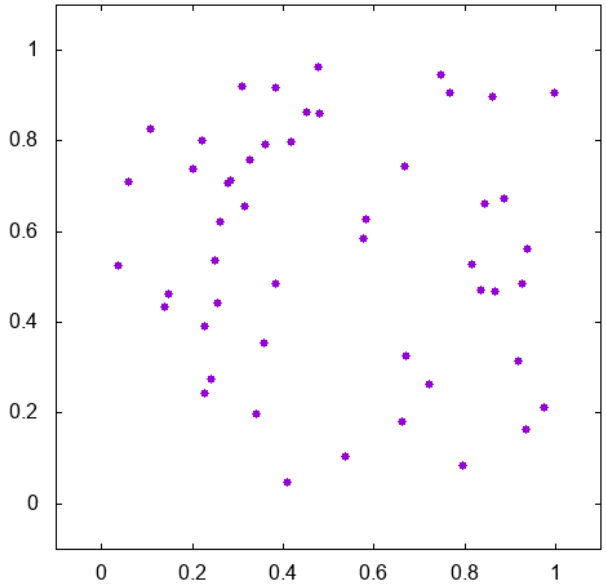


Figure 1: Gráfica de los puntos en un plano de 0 a 1.

Fase 2: Unir las coordenadas a distintos puntos.

El siguiente paso consistía en unir algunos de los puntos de las coordenadas con otros puntos. Para esto, se necesitaba agrupar de manera diferente los puntos x y y en el archivo de datos, y además agregar una variable mas que nos ayudara a saber con que probabilidad iba a unir cada uno de los puntos.

```
//cord.py

from random import random #importar libreria para generar tus numeros
    aleatorios.
nodos = [] #matriz donde vas a ir creando num
```

```
n = 30 #cantidad de numeros que voy a generar de manera aleatoria
p = 0.09 #Variable de probabilidad para que una coordenada se una con
    otra.
id = 1 #Esto lo necesito para nombrar a las aristas mas adelante.

with open ("nodos.dat", 'w') as salida: #generamos un archivo de salida
    para mis datos.
   for nod in range (n): #Este for me genera dos numeros aleatorios n
       veces.
      x = random()
      y = random()
      nodos.append((x,y))
      print(x, y, file = salida)
with open ("nodos.plot", 'w') as salida: #Archivo para poder generar mi
    grafica de salida.
   #NOTA: Recuerda que siempre tienes que llamar al dato de salida para
       cada uno de los comandos de gnplot
   print ('set term png', file = salida)
   #png para formato de salida de mi grafica
   print ('set output "nodos.png"', file = salida)
   print ('set size square', file = salida)
   #Keyoff es para que la marca de agua que sale en la parte superior
       der. no se vea.
   print ('set key off', file = salida)
   #El rango x y y son para el tamanio en que va a salir mi grafica.
   print ('set xrange [-.1:1.1]', file = salida)
   print ('set yrange [-.1:1.1]', file = salida)

#Para el siguiente FOR, llame a los datos ya agrupados, en donde toma el
    primero, y lo compara con todos los demas.
   for(x1, y1) in nodos:
      for (x2, y2) in nodos:
      #El siguente IF es donde entra en juego la probabilidad p, en la
          cual un numero aleatorio decidira si la va a unirse
      #con otra coordenada o no.
         if random () < p:
            #El sig. print es el comando de gnuplot para unir un punto
                con otro.
            print ('set arrow', id, 'from', x1, ',',y1,'to', x2, ',', y2,
                'nohead', file = salida)
            id += 1

   print ('plot "nodos.dat" using 1:2 with points pt 7', file = salida)
   print ('quit()', file = salida)
```

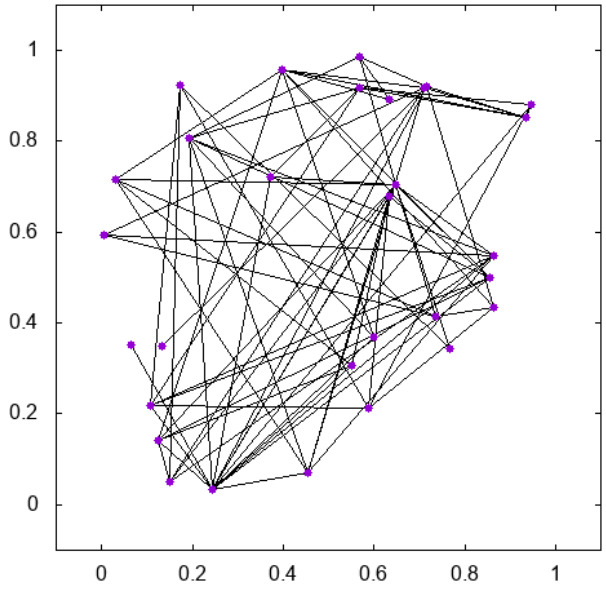


Figure 2: Gráfica de aristas.

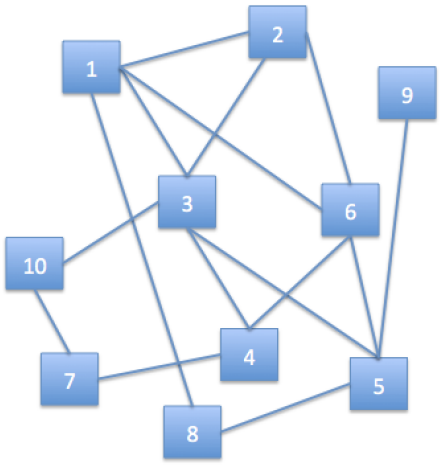
Fase 3: Agregar elementos.

En esta última fase, lo que se busca hacer es encontrar alguna forma de darle utilidad a una gráfica de este tipo. Por ejemplo, en clase hemos visto los diferentes tipos de utilidad que puede tener este tipo de gráfos.

En este caso, se querían hacer mas notorios los nodos que eran mas visitados por aristas. Primero, se buscó hacer algo simple para diferenciar un nodo de otro, y por esto se le dio una variable extra que ayudara a cambiar los colores de ellos.

Después, se formuló la manera de hacer un conteo de aristas por nodos en la gráfica. Al igual que con los colores, se agregó una nueva variable, y se cambiaron un poco la forma de representar las x y las y de las coordenadas, de manera que no hubiera error en el conteo (repetir un nodo con arista). La manera de explicar la forma de conteo para esto, se puede ver representado mejor en una tabla.

Por último, lo que queda por hacer en el código es jugar un poco con los parámetros que se le dan a la variable p, que es la que controla la cantidad de aristas por nodos, al igual que la cantidad de nodos que se desean agregar a la gráfica.



K	Conteo
1	4
2	3
3	5
4	3
5	4
6	4
7	2
8	2
9	1
10	2

Figure 3: Ejemplo de forma de conteo y uso de variable K. (NOTA: Esta imagen no fue realizada en Gnuplot, y solo sirve de ejemplo gráfico para el uso de la nueva variable.)

\\arrrow.py

```
from random import random #importar libreria para generar tus numeros
    aleatorios.
nodos = [] #matriz donde vas a ir creando num
n = 50 #antidacd de numeros que voy a generar de manera aleatoria
p = 0.1 #Variable de probabilidad para que una coordenada se una con
    otra.
id = 1 #Esto lo necesito para nombrar a las aristas mas adelante.
k=[0 for i in range(n)]

for nod in range (n): #Este for me genera dos numeros aleatorios n veces.
   x = random()
   y = random()
   nodos.append((x,y,nod))

#Las siguientes tres variables son las que me van a ayudar a formar la
    lista en la que
#voy a hacer el conteo de aristas por nodo.
q=0
nodos2=[]
nodos3=[]
#Para este primer FOR, lo que estoy buscando es que se posisione en el
    primer nodo a evaluar,
```

```
#y llegue hasta el penultimo, ya que si solo pongo n, me marca error.
for i in range(n-1):
   nodos2.append(nodos[i])
#En el sig. FOR voy desde el segundo elemento hasta el ultimo.
for i in range(1,n):
   nodos3.append(nodos[i])
aristas=[] #Al igual que en la matriz de aristas, aqui voy a guardar las
    uniones mas adelante.

for(x1, y1,i) in nodos2:
   del nodos3[0]
   for (x2, y2,j) in nodos3:
      if random () < p:
#Al igual que en la version anterior de este programa, el IF es el que
    controla cuantas aristas van a formarse.
         aristas.append((x1,y1,x2,y2))
         k[i]=k[i]+1
         k[j]=k[j]+1
         q+=1
#En estos ultimos K, es donde estoy asignandole la cantidad de aristas
    que lleva cada nodo.

with open ("nodos.dat", 'w') as salida: #generamos un archivo de salida
    para mis datos.
   for nod in range (n):
      #Esta C es la que me va a ayudar a cambiar el color de los nodos
          para diferenciarlos tanto
      #por color como por tamanio.
      c = (nodos[nod][0] + nodos[nod][1]) /2
      print(nodos[nod][0], nodos[nod][1], c, k[nod], file = salida)

with open ("nodos.plot", 'w') as salida: #Archivo para poder generar mi
    grafica de salida.
   #NOTA: Recuerda que siempre tienes que llamar al dato de salida para
       cada uno de los comandos de gnplot
   print ('set term png', file = salida)
   #png para formato de salida de mi grafica
   print ('set output "nodos.png"', file = salida)
   print ('set size square', file = salida)
   #Keyoff es para que la marca de agua que sale en la parte superior
       der. no se vea.
   print ('set key off', file = salida)
   #El rango x y y son para el tamanio en que va a salir mi grafica.
   print ('set xrange [-.1:1.1]', file = salida)
   print ('set yrange [-.1:1.1]', file = salida)
   for i in range(q):
      #El sig. print es el comando de gnuplot para unir un punto con
          otro.
      print ('set arrow', id, 'from', aristas[i][0],
          ',',aristas[i][1],'to', aristas[i][2], ',', aristas[i][3],
```

```
        'nohead', file = salida)
    id += 1
#Por ultimo, la cuarta columna del using, es 3*256 por la paleta de
    colores que tiene gnuplot.
print ('plot "nodos.dat" using 1:2:4:($3*256) with points pt 7 ps var
    lc var', file = salida)
print ('quit()', file = salida)
```

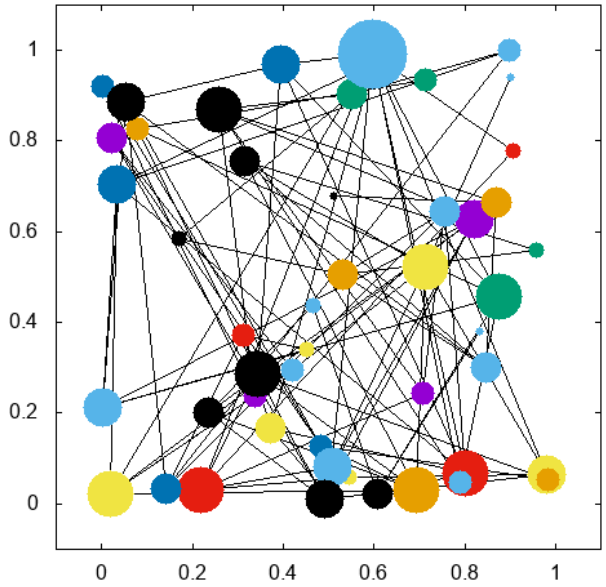


Figure 4: Resultado final en donde se aprecian nodos de distintos colores, con tamaños diferentes, dependiendo de la cantidad de aristas que recibe.

Conclusión

En clase, se formuló un ejemplo para facilitar a las personas encargadas de hacer los horarios de cada semestre. En este caso, se podría como ejemplo el problema que se presento al inicio del semestre, en el cual, algunas de nuestras clases quedaron unas sobre otras. La forma en que veo que este tipo de modelación puede ayudar, sería representando las aristas como los alumnos, y los nodos como las clases. Si un alumno esta interesado en la clase, se conecta con el nodo. Entre más grande sea este nodo, es mayor el interés por llevar esa clase, por lo que las personas encargadas de hacer los horarios podrían evaluar esta información y colocar esas clases en un horario en el que la mayor cantidad de alumnos puedan asistir, sin que se empalme con otra materia.