Optimización de Flujo en Redes: Reporte 2

Mayra Cristina Berrones Reyes

Marzo 2018

Introducción

En este reporte lo que se requiere es, tomar los pasos que realizamos en la práctica anterior y re-acomodarlos en forma de clases. Esto nos dará como resultado la primer fase de la práctica, que consiste en realizar un gráfo simple. A partir de este nuevo formato, se deberán realizar las siguientes funciones para las versiones subsecuentes.

- **Gráfo normal**
- **Gráfo dirigido**
- **Gráfo ponderado**

Fase 1: Gráfo normal

El nombre de la clase es $Grafo$. Se importaron las librerías de **random** y **math**. Las funciones que contiene esta clase se describen de la siguiente manera.

```
// gr.py
   def __init__(self):

   def puntos(self, num):

   def aristas(self, prob):

   def imprimir(self, arch):

   def grafica (self):
```

En la función *init* se guardan las variables globales que se utilizan en las demás funciones, ya que no puedes manipular una variable que esta dentro de una función diferente de la que se este usando actualmente.

Después está la función de *puntos*, en la cual se crean la cantidad de puntos aleatorios indicados en la entrada *num*.

En la función de *aristas* es donde se crean las conexiones entre los puntos que se crearon en *puntos*. En este caso, el dato de entrada que solicita es el de la probabilidad para que dichos puntos se unan. Entre más grande el número, más cantidad de conexiones existirán.

En la función *imprimir* requiere la entrada del nombre del archivo en cual se van a guardar los datos. La terminación en este ejemplo, es de .dat.

Por último, la función de $grafica$ simplemente se encarga de generar la imagen que tenía de salida el programa de la práctica anterior.

Las lineas necesarias para correr este programa son:

```
// ko.py
from rep2 import Grafo
p = Grafo()
p.puntos(10)
p.aristas(0.5)
p.imprimir("nodos.dat")
p.grafica()
```

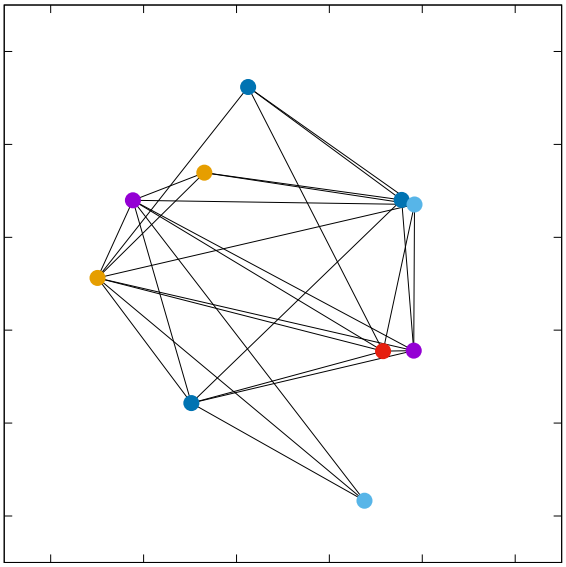

Figure 1: Gráfo normal.

Fase 2: Gráfo dirigido

Para esta sección, se realizó un cambio en la función de *grafica*, en donde se le agregaron lineas de código para especificar la dirección en la que van las aristas hacia los puntos.

```
// gr.py
   def grafica (self, di):
      assert self.archivo is not None
      with open("nodos.plot", 'w') as salida:
         print('set term eps', file = salida)
         print('set output "nodos.eps"', file = salida)
         print('set size square', file = salida)
```

```
print('set key off', file = salida)
print ('set xrange [-.1:1.1]', file = salida)
print ('set yrange [-.1:1.1]', file = salida)
id = 1
for i in range(len(self.Ari)):
   if di is 1:
      print('set arrow', id, 'from', self.Ari[i][0], ',',
          self.Ari[i][1], 'to', self.Ari[i][2], ',',
          self.Ari[i][3], 'head filled lw 1', file = salida)
      id +=1
   else:
      print('set arrow', id, 'from', self.Ari[i][0], ',',
          self.Ari[i][1], 'to', self.Ari[i][2], ',',
          self.Ari[i][3], 'nohead filled lw 1', file = salida)
      id +=1
print('plot "nodos.dat" using 1:2:3 with points pt 7 lc var ps
    2 ', file = salida)
print('quit()', file = salida)
```

En los parámetros de entrada para la función, lo que cambia es que ahora se le pide, 1 en caso de querer un gráfo dirigido, y cualquier otro numero en caso de querer un gráfo normal.

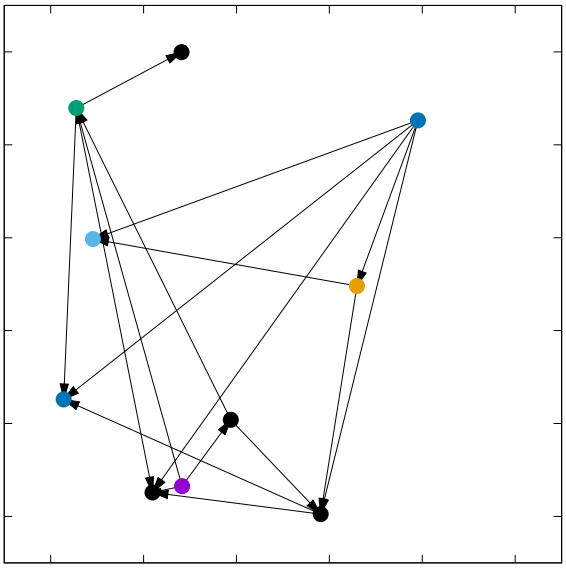

Figure 2: Gráfo dirigido.

Fase 3: Gráfo ponderado

Para esta fase, lo que se requiere es, de alguna manera, darle peso a las aristas que unen los puntos en el gráfo. En este caso, se modificó tanto la función de *grafica* como la de *aristas*. Además se agregó una variable global para guardar estos datos llamada *pesos*.

Lo que se busca con la variable *pesos* es calcular la distancia entre los puntos que si se conectan. Entre mas larga sea esta distancia, el costo de recorrerla será más alto.

```
//rep2.py
   def aristas(self, prob):
      for i in range(self.n - 1):
         self.nodo2.append(self.P[i])
      for i in range(self.n):
         self.nodo3.append(self.P[i])
      for(x1, y1, i) in self.nodo2:
         del self.nodo3[0]
         for(x2, y2, j) in self.nodo3:
            if random() < prob:
               self.Ari.append((x1, y1, x2, y2))
               self.pesos.append((sqrt((x2 - x1) ** 2 + (y2 - y1 )**
                   2)*100, (x1+x2)/2, (y1+y2)/2))
      print(len(self.Ari))

(...)

   def grafica (self, di):
      assert self.archivo is not None
      with open("nodos.plot", 'w') as salida:
         print('set term eps', file = salida)
         print('set output "nodos.eps"', file = salida)
         print('set size square', file = salida)
         print('set key off', file = salida)
         print ('set xrange [-.1:1.1]', file = salida)
         print ('set yrange [-.1:1.1]', file = salida)
         id = 1
         for i in range(len(self.Ari)):
            if di is 2 :
               print('set arrow', id, 'from', self.Ari[i][0], ',',
                   self.Ari[i][1], 'to', self.Ari[i][2], ',',
                   self.Ari[i][3], 'head filled lw 1', file = salida)
               id +=1
            elif di is 1:
               print('set arrow', id, 'from', self.Ari[i][0], ',',
                   self.Ari[i][1], 'to', self.Ari[i][2], ',',
                   self.Ari[i][3], 'nohead filled lw 1', file = salida)
               id +=1
            elif di is 3:
```

```
            print('set arrow', id, 'from', self.Ari[i][0], ',',
                self.Ari[i][1], 'to', self.Ari[i][2], ',',
                self.Ari[i][3], 'nohead filled lw 1', file = salida)
            print('set label', "'", int(self.pesos[i][0]), "'", 'at',
                self.pesos[i][1], ',', self.pesos[i][2], file = salida)
            id +=1

    print('plot "nodos.dat" using 1:2:3 with points pt 7 lc var ps
        2 ', file = salida)
    print('quit()', file = salida)
```

En la entrada de la función *grafica*, lo que se debe tomar en cuenta es que si se le da un 1, es un gráfo simple, si se le da un 2, es un gráfo dirigido, y si es un 3, es un gráfo ponderado.

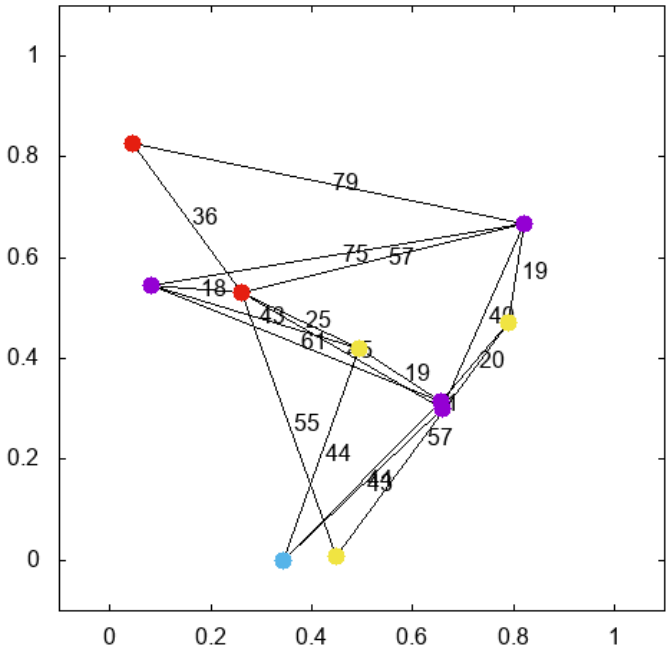

Figure 3: Gráfo ponderado.

Conclusión

Teniendo un buen panorama de estos tres tipos de gráfos, podemos empezar a imaginarlos o trasladarlos a los problemas de optimización que hemos estado aprendiendo en otras clases, como lo es el del problema del agente viajero. Esta

es una de las herramientas que podemos tener para representarlos. Mas adelante, quizá, aprendamos a recorrerlos e intentar encontrar una solución factible.