Tarea 1 Optimización de Flujo de Redes

Evely Gutiérrez Noda

20 de febrero de 2018

1. Introducción

En el siguiente reporte se abordará el tema estudiado en la clase de Optimización de Flujo de Redes. En dicha clase estudiamos sobre la construcción de grafos utilizando lenguaje Python [1] para la programación cuando queremos crear un grafo, además utilizamos la herramienta Gnuplot [2], la cual vinculamos con Python para representar el grafo antes programado.

2. Teoría sobre los Grafos

Primeramente, conocimos un poco sobre los grafos, los cuales son un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no. También sabemos que un grafo se representa mediante una serie de puntos (nodos o vertices) conectados por líneas (aristas)(ver figura 1).

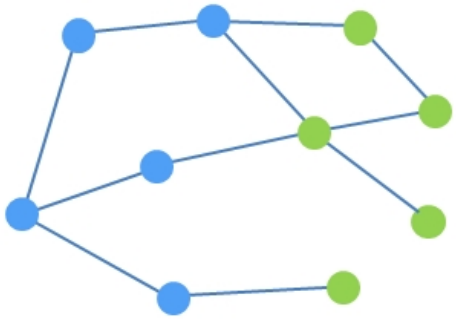

Figura 1: Ejemplo de grafo.

Existen distintos tipos de grafos como por ejemplo:

- Grafo simple: Es un grafo donde una arista cualquiera es la única que une dos vértices específicos.

- Multigrafo: Es uno que acepta más de una arista entre dos vértices. Los grafos simples son una subclase de esta categoría de grafos. También se les llama grafos en general.

- Pseudografo: Son aquellos que incluyen algún lazo. Un lazo en un grafo es una arista que conecta al mismo vértice consigo mismo.

- Grafo orientado: Es un grafo dirigido. Son grafos en los cuales se ha añadido una orientación a las aristas, representada gráficamente por una flecha.

- Grafo etiquetado: Grafos en los cuales se ha añadido un peso a las aristas (número entero generalmente) o un etiquetado a los vértices.

- Grafo aleatorio: Grafo cuyas aristas están asociadas a una probabilidad.

- Hipergrafo: Grafos en los cuales las aristas tienen más de dos extremos, es decir, las aristas son incidentes a tres o más vértices.

- Grafo infinito: Son aquellos grafos con un conjunto de vértices y aristas de cardinal infinito.

- Grafo plano: Los grafos planos son aquellos cuyos vértices y aristas pueden ser representados sin ninguna intersección entre ellos.

- Grafo regular: Un grafo es regular cuando todos sus vértices tienen el mismo grado de valencia.

3. Descripción del trabajo en clase

En la clase aprendimos como programar en Python un grafo a partir de una cantidad de nodos **n**, tomando por ejemplo una cantidad de nodos $\mathbf{n = 10}$, luego utilizando un ciclo **for** creamos un conjunto de nodos con valores aleatorios, los imprimimos en la consola y luego los almacenamos en un archivo de texto llamado **nodos.dat**. Para ello utilizamos las sentencias de código siguientes en la consola de Python:

```
with open("nodos.dat","w") as archivo:          ▷ Trabajo con archivo nodos
from random import random
x = random()                                    ▷ Variables para almacenar los nodos
y = random()
n = 10                                          ▷ Cantidad de nodos
for nodo in range(n):                           ▷ Ciclo para imprimir los 10 nodos
    print(random(),random())
```

Luego de tener los nodos creados, construimos las aristas correspondientes de nodo a nodo. Estas aristas las almacenamos en otro fichero llamado **aristas.dat**. Este trabajo se realizó en la consola de Python, donde utilizamos dos ciclos **for**, uno para recorrer los nodos guardados anteriormente y el otro para agregar un nodo tras del otro siguiendo una condición especificada en un **if**. A continuación esta el código utilizado para este trabajo.

```
with open(.ªristas.dat","w") as aristas:        ▷ Trabajo con archivo aristas
for (x1,y1) in nodos:          ▷ Iteración sobre los primeros pares de coordenadas
for (x2,y2) in nodos:          ▷ Iteración sobre los segundos pares de coordenadas
if random() < 0.1:                                  ▷ Condición para graficar
    print(x1,y1,x2,y2,file= aristas)        ▷ Imprime en consola las coordenadas
```

Luego del trabajo en consola del Python, aprendimos a trabajar vinculando el Python con Gnuplot, por medio de un editor de Python, donde se unen los

códigos de cada programa con una secuencia lógica, para luego abrirlo desde el Gnuplot y de este modo quedaría representado el grafo antes programado.

Esta representación del grafo en el editor de Python se hace utilizando las sentencias de Gnuplot **set arrow ...** para representar las aristas como flechas, en esta sentencia también se definió cuales serian los nodos a unir y graficar; también se puede especificar el estilo con que se va a graficar, en este caso en se hizo con el estilo que trae por defecto.

Para representar el gráfico de puntos desde el archivo **nodos.dat**, por medio de este editor de Gnuplot con sentencias de Python, se usó **plot 'nodos.dat' using 1:2**. A continuación se muestra una parte del código que utiliza estas sentencias.

```
with open("grafo.plt" , "w") as aristas:
    print("set xrange [-10.0:10.0]", file = aristas)
    print("set yrange [-10.0:10.0]", file = aristas)
    num = 1
    for (x1,y1) in nodos:
        for (x2,y2) in nodos:
            if random() < 0.1:
                print("set arrow {:d} from {:f}, {:f} to {:f}, {:f} lw 2
                    lt 5 lc rgb 'red'
                    nohead".format(num,x1,y1,x2,y2),file= aristas)
                num += 1

    print('set size square', file = aristas)
    print('set key off', file = aristas)
    print('set xrange[0:10]', file = aristas)
    print('set yrange[0:10]', file = aristas)
    print("plot 'nodos.dat' using 1:2 with points pt 5", file = aristas)
```

Como los valores de los nodos los creamos aleatorios cada vez que se corre el programa en el editor de Python, se crean nodos diferentes, que arrojaron la creación de diferentes grafos, a partir de varios valores de $\mathbf{n = 5, 6, 10, 100}$ que definen la cantidad de nodos(ver figura 2).

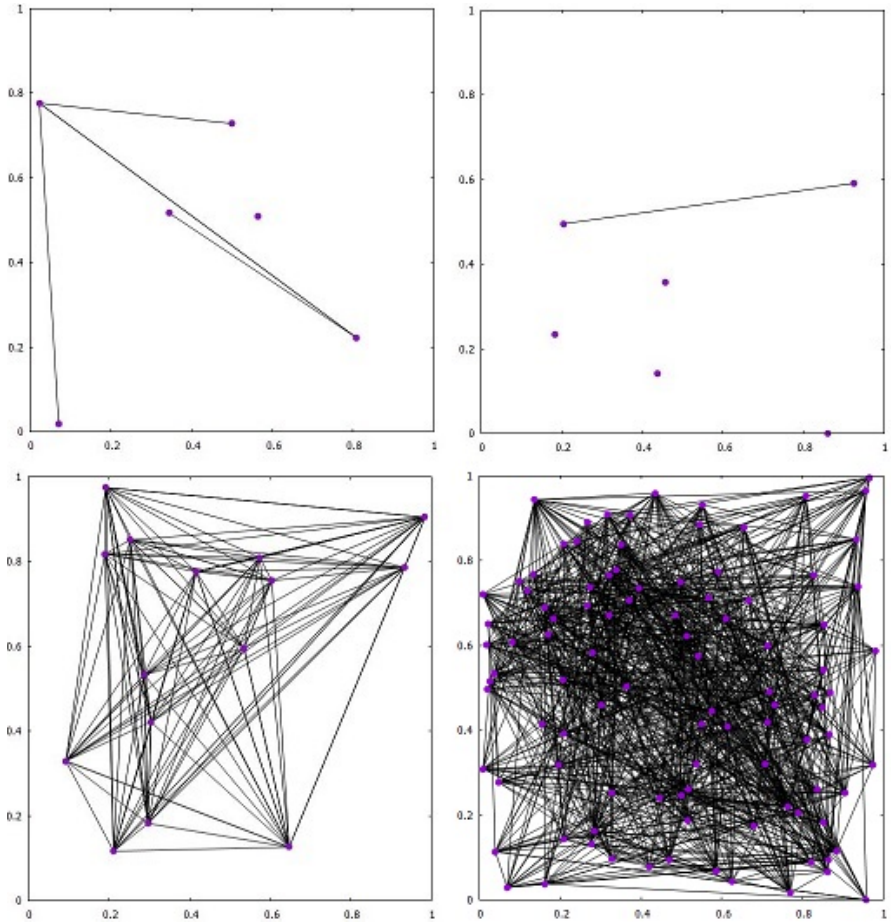

Figura 2: Grafos resultantes de lo aprendido en clase.

4. Tarea Extraclase

Hasta aquí fue lo aprendido en clase, luego de forma investigativa individual, se hicieron varias modificaciones al grafo programado, cambiando la forma de los nodos, ya sean triángulos, cuadrados y círculos, y cambiando de color y grosor las aristas que unen los nodos. Tambien se le puso nombres a los ejes del plano XY y nombre al grafo a representar (ver figura 3).

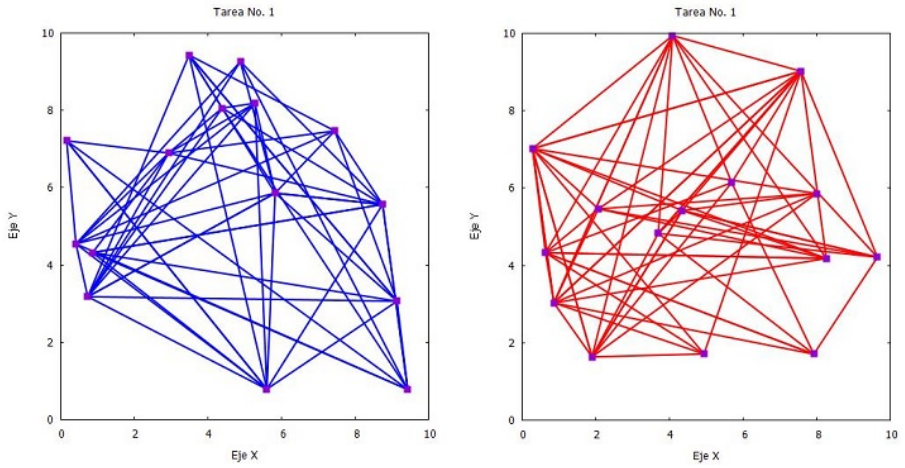

Figura 3: Cambios de grosor y color de aristas.

Para colocar el nombre al eje X y Y, se utilizó el comando **set xlabel "eje x"** y **set ylabel "eje y"** respectivamente, para el título del grafo se utilizó el comando **set title (t) "Título"**.

El cambio de color y de grosor de las aristas fue utilizando las sentencias **linetype n (lt n)** y **linewidth g (lw g)** respectivamente.

Con esto se puede ver la aplicación del uso de grafos para poder resolver o representar problemas de flujo en la vida real, ya sea de transporte, servicio de red o cualquier problema en el cual se desee conocer la vía más económica para resolverlo.

Un ejemplo hipotético podría ser para describir o representar las líneas del metro que existen en Monterrey, utilizando lo aprendido, representar de diferentes colores el flujo de cada línea del metro, y los nodos serían las estaciones correspondientes. Quedarían las líneas rojas para representar la línea del metro 1, y la azul, para la línea del metro 2(ver figura 4).

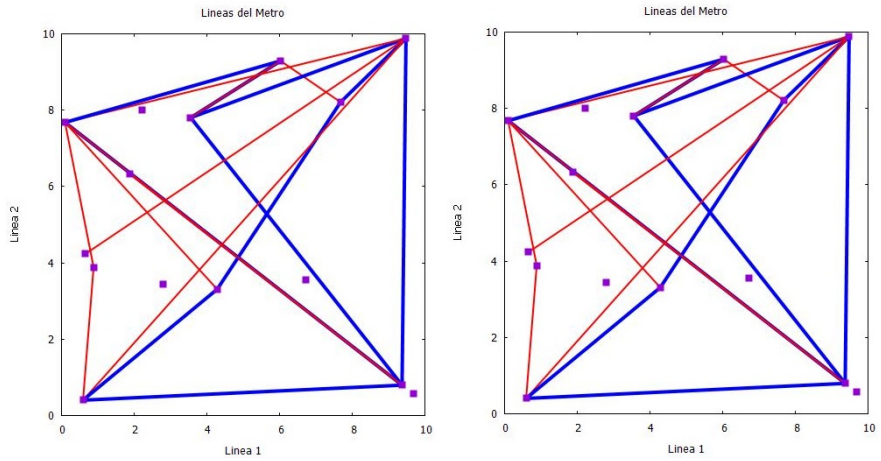

Figura 4: Líneas del Metro

Referencias

[1] Python, **http://www.python.org/**, 19 de Febrero de 2018

[2] Gnuplot, **http://www.gnuplot.info/**, 19 de Febrero de 2018