Tarea 2: Visualización de grafos simples, ponderados y dirigidos

Gabriela Sánchez Y.

Introducción

Existen diferentes tipos de grafos, en la presente práctica veremos tres tipos: grafos simples, ponderados y dirigidos. Un grafo **simple** es un grafo en el cual cada par definido de nodos es unido por una sola arista, un grafo **ponderado** tiene la característica de que sus aristas tienen un valor definido y en el grafo **dirigido** las aristas tienen definida una dirección.

El objetivo de la práctica es utilizar programación orientada a objetos para crear una clase en el lenguaje de programación `Python` [1], que nos permita crear los distintos tipos de grafos. En conjunto se trabaja con `Gnuplot` [2] para la visualización de los graficos.

Clase grafo

La clase cuenta con tres funciones: `nodo`, `arista` y `grafica`. A continuación se explica el funcionamiento de cada una de ellas.

La función `nodo` recibe cuatro parámetros: v la etiqueta para el nodo, coordenadas x, y y r el radio que tendrá el nodo. Esta función guarda las coordenadas y el radio del nodo en el archivo `nodos.dat`.

```
def nodo(self,v,x,y,r):
    self.V[v] = (x,y)
    with open("nodos.dat", 'a') as archivo:
        print(x,y,r, file = archivo)
```

La función `arista` también recibe cuatro parámetros: las etiquetas de los nodos en los que se desea crear una arista, un valor booleano que nos dice si la

arista será dirigida o no y un valor p de ponderación de la arista, y los guarda en una lista.

```
def arista(self,u,v,d,p):
    self.E.append((u,v,d,p))
```

Finalmente la función `grafica` contiene las especificaciones necesarias para graficar el tipo de grafo que se desea. Ésta revisa para cada arista las características que tiene, dependiendo de las mismas, se hace la sentencia adecuada para obtener el grafo deseado.

```
def grafica(self):
    i = 1
    with open("grafo.plot", 'w') as archivo:
        print('set term png', file = archivo)
        print('set output "grafo.png"', file = archivo)
        print('set size square', file = archivo)
        print('set key off', file = archivo)
        print('set xrange [-0.1:1.1]', file = archivo)
        print('set yrange [-0.1:1.1]', file = archivo)
        for v in self.E:
            u = v[0]
            w = v[1]
            (x1, y1) = self.V[u]
            (x2, y2) = self.V[w]
            if v[3] != 0 and v[2] == False: # ponderado
                print('set arrow',i, 'from', x1,',',y1,'to',x2,',',y2,
                    'nohead lw {:f}'.format(v[3]), file = archivo)
                i += 1
            elif v[3] != 0 and v[2] == True: # ponderado y dirigido
                print('set arrow',i, 'from', x1,',',y1,'to',x2,',',y2,
                    'head filled size screen 0.03,15 lw
                    {:f}'.format(v[3]), file = archivo)
                i += 1
            elif v[2] == True: # dirigido
                print('set arrow',i, 'from', x1,',',y1,'to',x2,',',y2,
                    'head filled size screen 0.03,15', file = archivo)
                i += 1
            else: # simple
                print('set arrow',i,'from', x1 , ',', y1, 'to', x2,',',
                    y2,'nohead', file = archivo)
                i += 1
    print('plot "nodos.dat" using 1:2:3 with points pt 7 ps var', file =
        archivo)
```

Visualización

El archivo `grafo_clase.py` utiliza la clase `grafo` para crear los grafos simples, ponderados y dirigidos. En este archivo se crean n nodos y para cada uno de ellos se determina la posibilidad de crear una arista de acuerdo a una probabilidad p. En las siguientes subsecciones se muestran los resultados.

Grafo simple

En el caso del grafo simple se llama a la función arista de la siguiente forma:

```
G.arista(v, w, False, 1)
```

La figura 1 muestra dos ejemplos de grafos simples, uno con nodos uniformes y otro con nodos de distintos tamaños.

Grafo ponderado

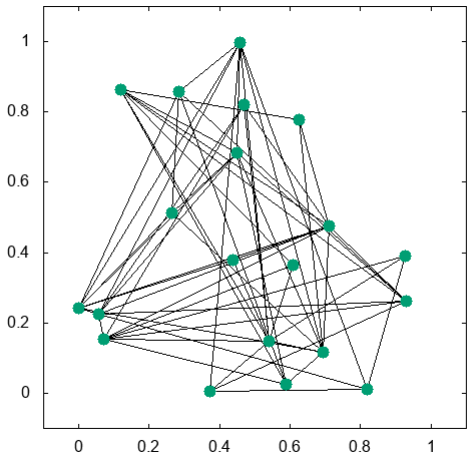
Para crear un grafo ponderado, la función arista recibirá en el cuarto parámetro la ponderación que se desee. Como ejemplo se encuentra el siguiente:

```
for v in G.V:
    for w in G.V:
        if random() < 0.08:
            if random() < 0.5:
                G.arista(v, w, False, randint(2,7)/2.0)
            else:
                G.arista(v, w, False, 0)
```

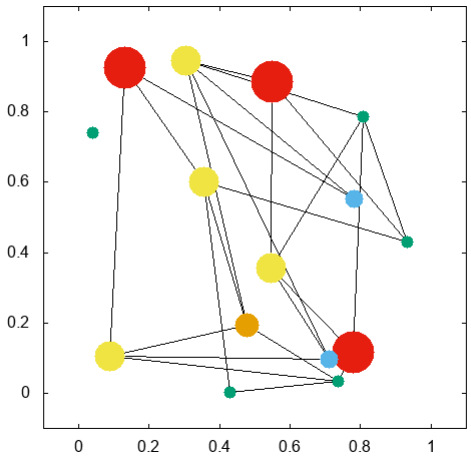
En la figura 2 se observan dos ejemplos de grafos ponderados.

Grafo dirigido

El grafo dirigido se crea si el tercer parámetro de la función arista tiene el valor *True*. La figura 3 muestra un ejemplo.

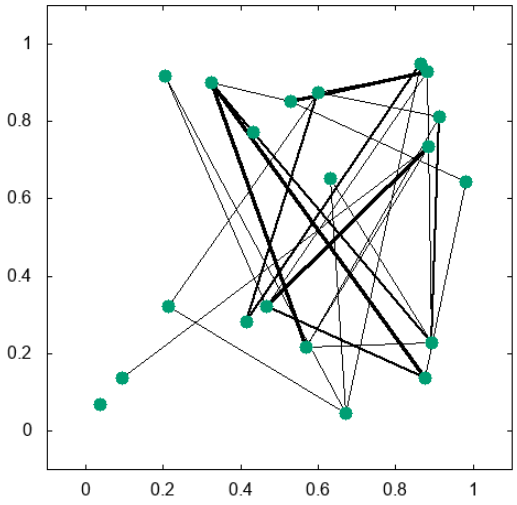


a) grafo simple con nodos uniformes

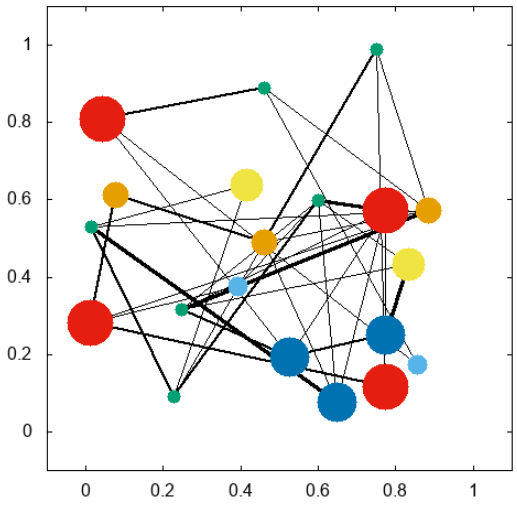


b) grafo simple con nodos variantes

Figura 1: Ejemplos de grafos simples.



a) grafo ponderado con nodos uniformes



b) grafo ponderado con nodos variantes

Figura 2: Ejemplos de grafos ponderados.

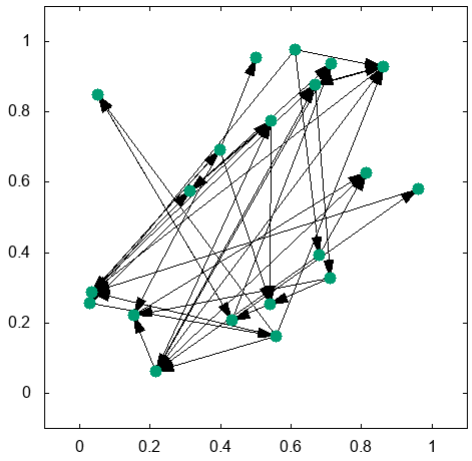


Figura 3: Grafo dirigido.

Mixto

La clase es flexible para crear grafos que sean dirigidos y ponderados. A continuación se muestra un ejemplo de cómo debe llamarse a la función arista para crear un grafo con estas características. En la figura 4 se muestra la visualización del ejemplo.

```
for v in G.V:
    for w in G.V:
        if random() < 0.08:
            if random() < 0.4:
                G.arista(v, w, True, randint(2,7)/2.0)
            else:
                G.arista(v, w, False, randint(0,3))
```

Referencias

[1] *Python 3.2.6* Lenguaje de programación. www.python.org

[2] *Gnuplot 5.0* Programa interactivo para realizar graficos. www.gnuplot.info

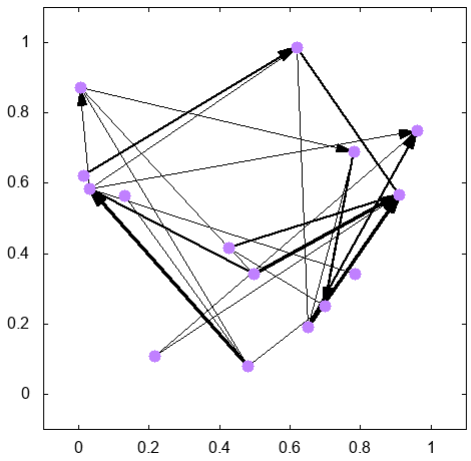


Figura 4: Grafo ponderado y dirigido.