Visualización de grafos simples, ponderados y dirigidos con `gnuplot`

Francisco Gerardo Meza Fierro

1. Introducción

En esta práctica se trabaja con el código que se encuentra en el mismo repositorio que este reporte [1], éste se reestructura usando clases y se modifica para que los grafos que antes se creaban tengan ahora sus aristas dirigidas y ponderadas.

Igual que en la previa práctica, el código creado para esta práctica se escribió en `python` el cual genera un archivo que al ser leído en `gnuplot` genera una imagen que contiene el nuevo grafo creado.

2. Clases

Lo primero que se hizo fue la reestructuración del previo código para usar clases. Dentro de la clase `Grafo` se encuentran:

- `init` que contiene los datos de entrada para crear los grafos
- `CreaNodos` y `CreaAristas` crean los nodos y aristas respectivamente
- `ArchivoGnu` genera el archivo que será leído en `gnuplot`

Además, a este nuevo código se le agregó que las aristas sean del mismo color del *nodo madre* que conectan al resto de los nodos, como se puede ver en la figura 1.

3. Grafo simple

Desde la práctica anterior, el código ya generaba grafos simples por lo que solo se agrega una visualización de un grafo simple que el código genera, mostrada en la figura 1.

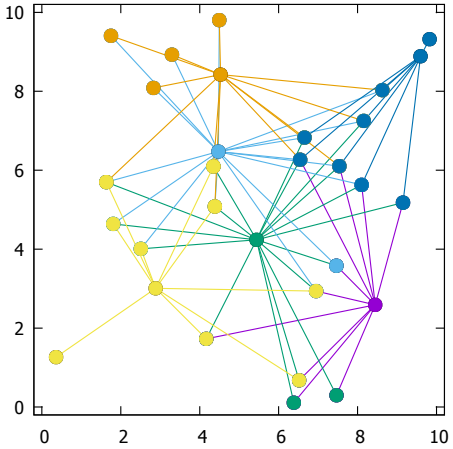


Figura 1: Un ejemplo de un grafo simple con 30 nodos.

4. Grafo ponderado

Para incluir el peso de las aristas en la imagen, primero se creó una función que calcula el punto medio de alguna arista que conecte un *nodo madre* con algún otro nodo, ya que se decidió escribir el peso de la arista en ese punto.

```
def pmedio(p,q):
    return ((p[0]+q[0])/2),((p[1]+q[1])/2)
```

Se decidió que la ponderación de las aristas sería el techo de la distancia entre el *nodo madre* y el nodo que se conecta. Y al igual que las aristas, los pesos están pintados del mismo color que el del *nodo madre*, tal y como se aprecia en la figura 2. Recordemos que los nodos de color no fueron conectados por ningún *nodo madre*.

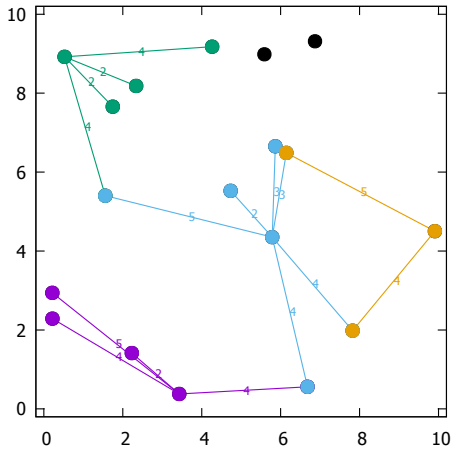


Figura 2: Ejemplo de un grafo ponderado con 15 nodos.

5. Grafo dirigido

Crear grafos dirigidos no fue tarea complicada, tan solo se modificó la instrucción `set arrow` dentro de `ArchivoGnu`, que le da dirección a la arista. Se decidió que la dirección de las aristas sería del *nodo madre* a los nodos que están dentro del radio de alcance del *nodo madre*, tal como se aprecia en la figura 3

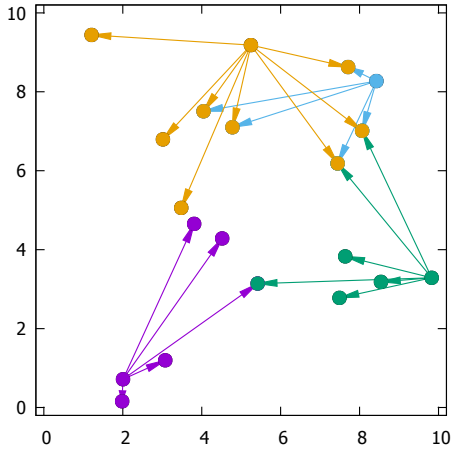


Figura 3: Ejemplo de un grafo dirigido con 20 nodos.

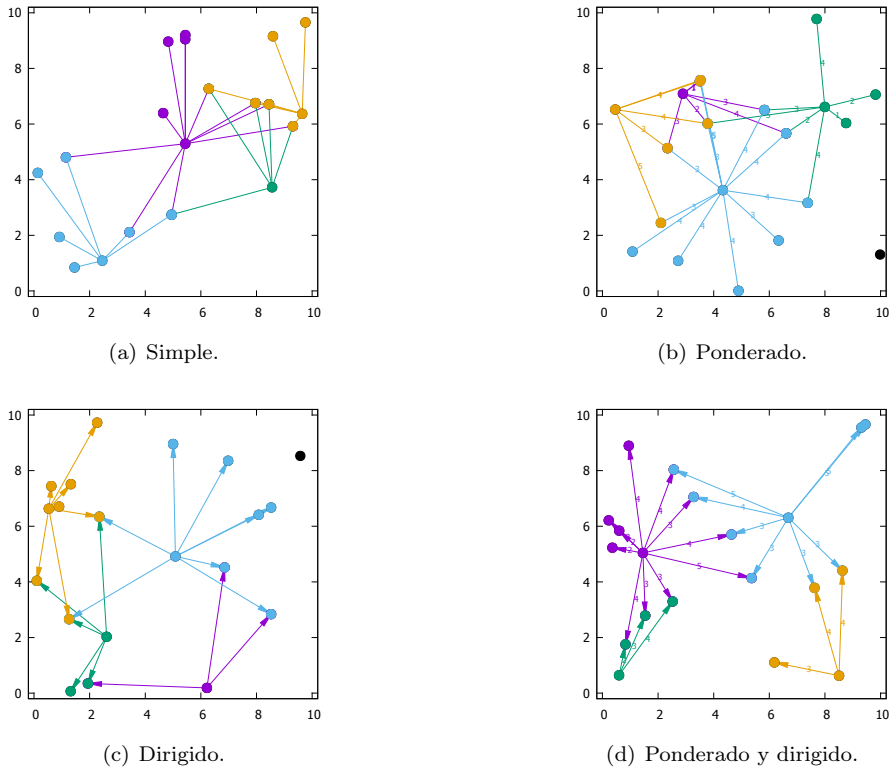
6. Grafo ponderado y dirigido

Por último, se decidió hacer flexible el código permitiéndole al usuario decidir en un inicio si desea generar un grafo dirigido, ponderado o alguna combinación entre ellos, así como la cantidad de nodos que el grafo tendrá, esto se hace dentro de una función `for` que recorre todos los nodos que se conectan.

```
Nnodos = 20
dirigido = 1 # no dirigido = 0 , dirigido = 1
pesos = 0    # sin pesos   = 0 , con pesos =1

num=1
    for a in range (len(self.aristas)):
        (x1,y1,x2,y2) = self.aristas[a]
        (xm,ym) = self.pm[a]
        w = self.pesos[a]
        col = self.colores[a]
        if dirigido is 1:
            print("set arrow {:d} from {:f},{:f} to {:f},{:f} head filled size
                .5,10 lt {:f} lw 1".format(num,x1,y1,x2,y2,col),file=archivo)
            if pesos is 1:
                print("set label font ',9' '{:d}' at {:f},{:f} tc lt {:f}".format(
                    w, xm, ym, col), file = archivo)
        else:
            print("set arrow {:d} from {:f},{:f} to {:f},{:f} nohead lt {:f} lw 1"
                .format(num,x1,y1,x2,y2,col),file=archivo)
            if pesos is 1:
                print("set label font ',9' '{:d}' at {:f},{:f} tc lt {:f}".format(
                    w, xm, ym, col), file = archivo)
    num+=1
```

La figura 4 muestra algunos de los resultados que se obtienen al crear varios grafos distintos.



(a) Simple.

(b) Ponderado.

(c) Dirigido.

(d) Ponderado y dirigido.

Figura 4: Diferentes grafos con 20 nodos.

Referencias

[1] Francisco Meza. *Representación de redes a través de la teoría de grafos en* `python` https://github.com/Cicciofano/FlujoEnRedes/blob/master/Tarea1, 2018.