Visualización de grafos simples, ponderados y dirigidos con Gnuplot

Astrid González

Optimización de flujo en redes

4 de marzo de 2018

Introducción

La visualización gráfica de un conjunto de datos nos permite obtener una mayor comprehensión del comportamiento general de sus componentes. Un grafo[1] corresponde a la representación matemática de una red.

Entonces, se tiene que un grafo G consiste de un set no vacío y finito denotado $V(G)$, cuyos elementos son llamados vértices o nodos, y de una familia finita $E(G)$ de pares no ordenados de elementos de $V(G)$ llamados conexiones o aristas[5]. Bajo este contexto, una conexión (v, w) representa una relación entre los nodos v y w.

Dicha conexión entre los elementos de un grafo, puede denotar un camino o dirección entre los nodos, en cuyo caso se dice que es una conexión dirigida. Además, las conexiones pueden presentar cierto peso o valor relacionado a ellas, lo que se denota como una conexión ponderada. Si ésta no presenta dirección ni ponderación, se trata de una conexión simple.

En este trabajo se presentan ejemplos de grafos con dichas características. Para ello, la información de los nodos y conexiones fue generada en la versión 3.6.4 del lenguaje Python[3], mientras que la versión 4.6 de Gnuplot[4] fue utilizada para visualizar los grafos.

Metodología

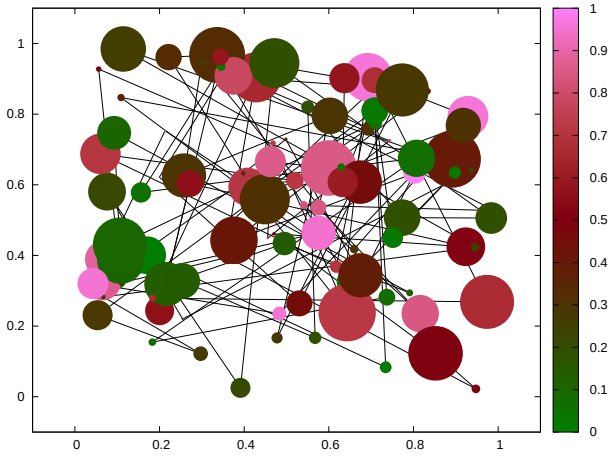
Fue diseñada una librería que permitiera mantener control sobre los nodos, conexiones, y propiedades del grafo. Esta librería habilita la posibilidad de obtener cuatro tipos de grafos: no dirigidos con o sin ponderación, así como dirigidos con o sin ponderación.

Se genera un grafo de cada tipo, donde se crean cien nodos cuyo tamaño, posición y color fueron obtenidos de manera aleatoria dentro del intervalo [0,1] de los números reales. La información para cada nodo es entonces almacenada en un archivo con extensión .dat.

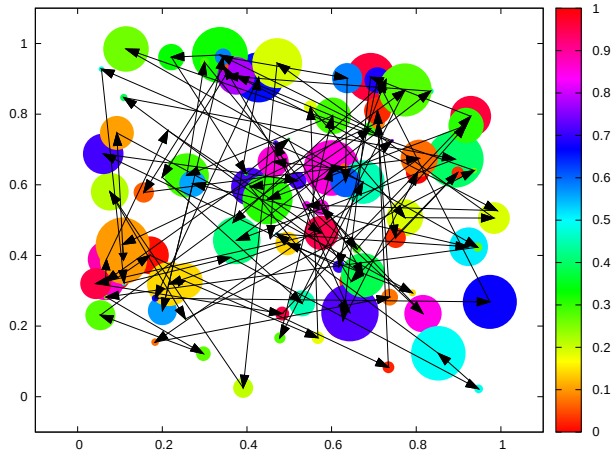
Se procede a realizar la conexión en base al valor de alguna característica de los nodos, para lo cual se seleccionó al color. Al realizar la conexión de los nodos se utiliza la librería Pandas[2] para diseñar un marco de información y poder manipular de forma conjunta los datos para cada nodo. En dicho paso es posible obtener grafos con conexiones ponderadas y grafos con conexiones sin ponderación.

Finalmente, una función genera un script a partir del archivo de información obtenido previamente. Es posible elegir si las conexiones serán dirigidas o simples. La salida de esta función es un archivo .dat el cual es posteriormente leído en Gnuplot mediante la instrucción load.

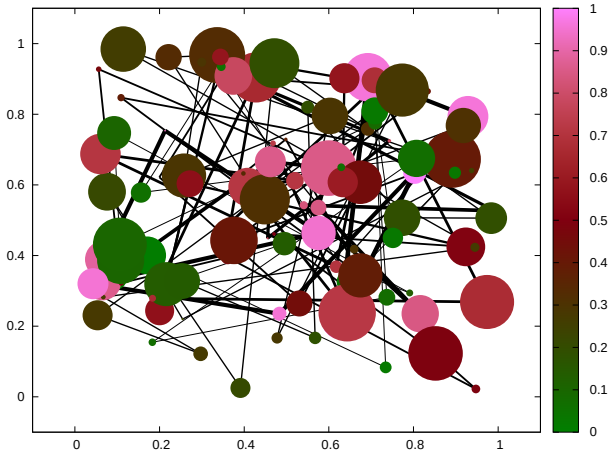
Resultados



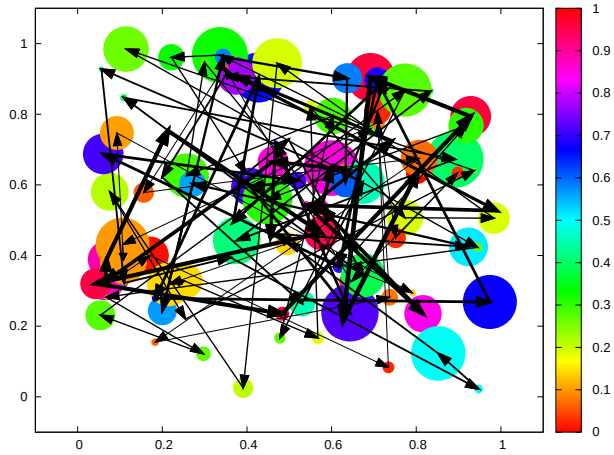
(a) Grafo simple.



(b) Grafo dirigido no ponderado.



(c) Grafo no dirigido ponderado.



(d) Grafo dirigido ponderado.

Figura 1: Representación de los cuatro tipo de grafos.

A partir del archivo de datos, se obtuvieron cuatro grafos diferentes, con la misma información para las variables de posición, tamaño y color de los nodos, y cuyos valores pertenecen a los números reales.

Las cuatro imágenes de la figura 1 representan las posibles configuraciones que puede tener un grafo:

- El grafo simple presenta conexiones sin dirección y sin ponderación, es decir que éstas son líneas con el mismo grosor.
- El grafo dirigido no ponderado nos muestra conexiones con dirección y sin ponderación en forma de flechas del mismo grosor.
- Por otro lado, el grafo no dirigido ponderado que cuenta con conexiones sin dirección pero con ponderación, tiene líneas de diferentes grosores.
- Y el grafo dirigido ponderado que presen-

ta conexiones con dirección y ponderación, cuenta con flechas de diferentes grosores.

El grosor de los grafos ponderados fue determinado en base a los valores designados a la variable de color. Iniciando con un grosor de 0.50 unidades, para todos los nodos i, j tal que $\{i, j \in V(G)\}$ si la diferencia del valor de color del nodo j con respecto al nodo i es menor o igual a 0.03 el grosor de su respectiva conexión (i, j) tal que $\{(i, j) \in E(G)\}$ en forma de línea o flecha se mantendrá constante. Sin embargo si esta diferencia se encuentra dentro del intervalo (0.03,0.07] se produce un aumento al grosor de 0.50, mientras que si es superior a 0.07 el aumento es de 1 unidad.

Discusión

Las combinaciones de conexiones que se obtuvieron fueron posibles gracias a una función donde el usuario es capaz de determinar el peso o grosor al momento de conectar los nodos, es decir si designar o no ponderación a las conexiones. Más adelante, cuando se genera el script del grafo se incluye una opción donde se selecciona si las conexiones llevaran o no dirección, es decir si las uniones se realizaran por medio de líneas o flechas. De esta manera es posible obtener las cuatro combinaciones de grafos: simples, dirigidos no ponderados, no dirigidos ponderados, y dirigidos ponderados. En el pseudocódigo 1 es posible observar estos pasos.

Pseudocódigo 1: Obtención de los diferentes tipos de grafo.

```
Función conectarG(grafo, elección, modo):
    y ← lista de nodos ordenados por elección
    k ← 0.05
    para n en rango(len(y) − 1) hacer
        si modo es 's' entonces
            conectar los nodos y[n] y y[n + 1] con grosor k
        si modo es 'p' entonces
            difNodos ← (grafo.color[y[n + 1]] − grafo.color[y[n]])
            si (difNodos ≤ 0.03) entonces
                conectar los nodos y[n] y y[n + 1] con grosor k
            si (0.03 < difNodos ≤ 0.07) entonces
                k ← k + 0.5
                conectar los nodos y[n] y y[n + 1] con grosor k
            si (difNodos > 0.07) entonces
                k ← k + 1
                conectar los nodos y[n] y y[n + 1] con grosor k
    regresar y
Función plotG(grafo, y, datos, archivo, modo):
    si modo es 's' entonces
        para n en rango(len(y) − 1) hacer
            script para graficar con líneas las conexiones de los datos del grafo
            guardar script en archivo
    si modo es 'd' entonces
        para n en rango(len(y) − 1) hacer
            script para graficar con flechas las conexiones de los datos del grafo
            guardar script en archivo
```

Dado un conjunto de números naturales en el intervalo $[0, 1]$, en el caso de los grafos ponderados, para poder determinar el incremento del grosor se seleccionó una cota inferior en base a prueba y error. A través de repeticiones se aseguró que la cota pudiera mantener constante el grosor de las conexiones de los grupos de nodos con valores similares de la variable color. El valor de dicha cota inferior corresponde a 0.03.

Posteriormente, habiendo intentando con valores cercanos a 0.1 y 0.5 se decidió realizar la búsqueda de la cota superior en el rango de valores cercanos al doble del valor de la cota inferior. Fue entonces confirmado la cota superior resultante de 0.07 es capaz de detectar cuando existen una diferencia significativa entre nodos, por lo que controla el aumento del grosor de sus conexiones entre 0.5 y 1 unidades.

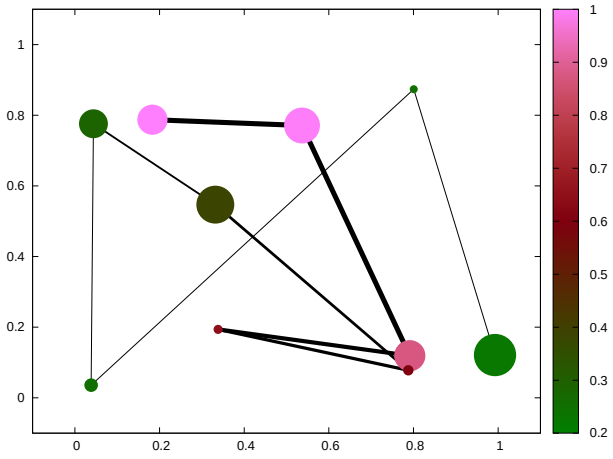


Figura 2: Visualización del incremento del grosor en las conexiones del grafo.

Las conexiones que reciben un incremento de 1 unidad a su grosor, corresponden a un pico detectado ya que tras numerosas repeticiones fue observado que habiendo obtenido los valores de forma aleatoria, con la cantidad de nodos que se utilizaron en este proyecto no se producen valores distantes entre sí lo que reduce la cantidad de picos. Sin embargo, en caso de presentarse un nodo que no perteneciera a ese patrón, sería efectivamente detectado por la cota superior. Ambas cotas fueron probadas en un grafo con menor cantidad de nodos, en cuyo caso es más evidente la existencia de picos y por ende se aprecia un incremento más prominente en el grosor de las conexiones, lo cual se puede observar en la figura 2.

Por último cabe mencionar que a diferencia de los grafos sin dirección, en el caso de los grafos dirigidos fue necesario colocar las conexiones por encima de los nodos debido a que la cabeza de las flechas, y por ende la dirección, no era visible en los nodos con tamaño relativamente grande.

Referencias

[1] Barabási, A.-L. (2016). *Network science*. Cambridge University press. ISBN: 9781107076266.

[2] McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy and IPython*. O'Reilly Media, second edition. ISBN: 9781491957615.

[3] Van Rossum, G. and the Python Development Team (2018). The python language reference. release 3.6.4. `https://docs.python.org/3.6/`. Consultado en Febrero 2018.

[4] Williams, T. and Kelley, C. (2013). Gnuplot 4.6: an interactive plotting program. `http://gnuplot.info/`. Consultado en Febrero 2018.

[5] Wilson, R. J. (1996). *Introduction to Graph Theory*. Prentice Hall, fourth edition. ISBN: 0582249937.