Tarea - Optimización de redes

Carolina Monzó Cataluña

Partiendo de un grafo aleatorio de 50 nodos y una media de 4 conexiones entre los nodos, debemos optimizar la eficiencia. Para ello, se ha escrito un script en python2.7 que sigue la implementación del procedimiento de optimización indicado en la práctica.

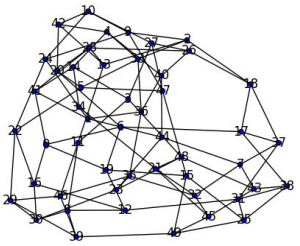
El script consta de cuatro funciones de uso y una función gestora del programa.

- La función “random_network” se encarga de generar el grafo inicial de 50 nodos y 4 conexiones entre los nodos
- A continuación, “efficiency” calcula la media del tamaño de los caminos y el número de caminos, y los utiliza para calcular la eficiencia del grafo siguiendo la fórmula indicada en la práctica: $E(\lambda) = \lambda d + (1 - \lambda)m$
- Una vez tenemos el grafo y la eficiencia inicial, se seleccionan dos nodos al azar y utilizando la función “modify_connection”, se comprueba si los nodos seleccionados están conectados, en caso de estarlo, se elimina su conexión, y si no lo están, genera una conexión entre los nodos
- Nuevamente se utiliza la función “efficiency” para calcular la eficiencia del grafo una vez ha sido modificado según la conexión de sus nodos
- Tras comprobar si la eficiencia del grafo mejora con el cambio realizado, en caso de no mejorarla, se utiliza la función “return_to_ori” que deshace el cambio realizado entre los nodos seleccionados y devuelve el grafo a su estado anterior, de forma que siempre se mantiene el grafo con mejor eficiencia

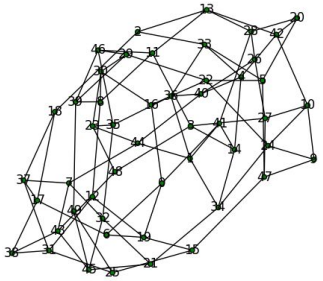
Tras lanzar el script, realizando 100 modificaciones y comprobaciones de eficiencia, obtenemos los siguientes resultados:

```
carolina@local:~/workspace/master/networkx_sisbio
$ python2.7 network.py
Original efficiency for lambda = 0.01 : 99.0305306122
Final efficiency for lambda = 0.01 : 92.1020653061
Original efficiency for lambda = 0.08 : 85.816522449
Final efficiency for lambda = 0.08 : 76.6489795918
Original efficiency for lambda = 0.99 : 4.40612244898
Final efficiency for lambda = 0.99 : 3.69751836735
```

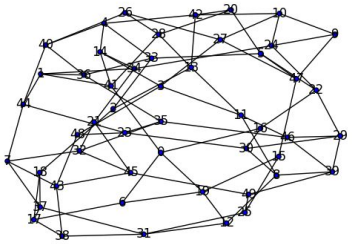
Original Random network with lambda = 0.01 and efficiency = 99.030530612



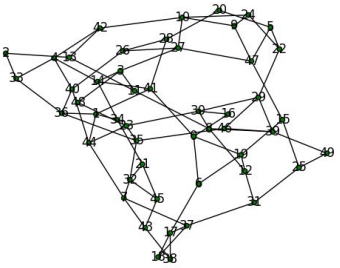
Final optimized network with lambda = 0.01 and efficiency = 92.1020653061



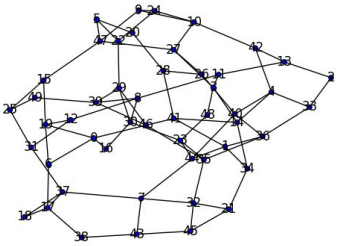
Original Random network with lambda = 0.08 and efficiency = 85.8165224



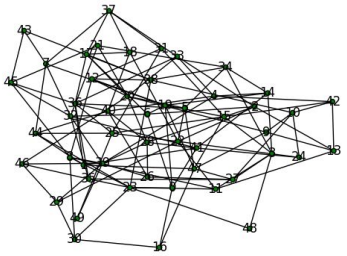
Final optimized network with lambda = 0.08 and efficiency = 76.6489795918



Original Random network with lambda = 0.99 and efficiency = 4.4061224489



Final optimized network with lambda = 0.99 and efficiency = 3.69751836735



Como podemos observar en los resultados que obtenemos en esta actividad, para un mismo número de iteraciones, la eficiencia disminuye a medida que aumentamos el valor de lambda, que corresponde a la importancia relativa entre el camino más corto medio y el número de caminos de un grafo. Teóricamente alcanzaremos un modelo scale free cuando tengamos unos pocos nodos altamente conectados y el resto tengan un número bajo de conexiones, por tanto podremos obtener un modelo scale free a partir de las simulaciones que hemos realizado, si utilizamos un número pequeño de nodos y una lambda también de valor bajo.

La razón de elegir Python para implementar este estudio de grafos, es que es el lenguaje que más hemos trabajado en clase y tiene paquetes como Networkx que facilitan la realización de estudios como el que hemos realizado en esta práctica.