Tarea 4

Samuel Alejandro Sánchez Vázquez

23 de abril de 2018

Objetivo

El siguiente trabajo tiene como objetivo el utilizar el código de Floyd-Warshall[1], anteriormente utilizado en la tarea tres, realizando ciertas modificaciones tales como convertir los grafos de salida en circulares, y el crear funciones de medición (d como distancia promedio, c como conexiones con los vecinos promedio, s como cota superior de la cantidad de conexiones) para poder observar el comportamiento y poder llegar a un conclusión.

Código

A continuación se mencionarán las modificaciones y funciones que se realizaron.

Grafo circular

Durante clases se realizo la modificiación al código para que los nodos estuvieran situados de tal manera que al realizar el grafo, formarán un circulo perfecto.

```
def generar(self, orden):
    with open("Nodos.dat", "w") as archivo:
        centro = (0.5, 0.5)
        radio = 0.5
        angulo = (360/orden)*(pi/180)
        for n in range(1, orden+1):
            x = radio*cos(angulo*n) + centro[0]
            y = radio*sin(angulo*n) + centro[1]
            self.n.append((x,y))
            print(x, y, file = archivo)
            if not (x, y) in self.vecinos:
                self.vecinos[(x,y)] = []
```

Lo que se busca en este tipo de gráficos es una misma posición circular, cambiando la cantidad de nodos y aristas que podamos unir (Figura 1).

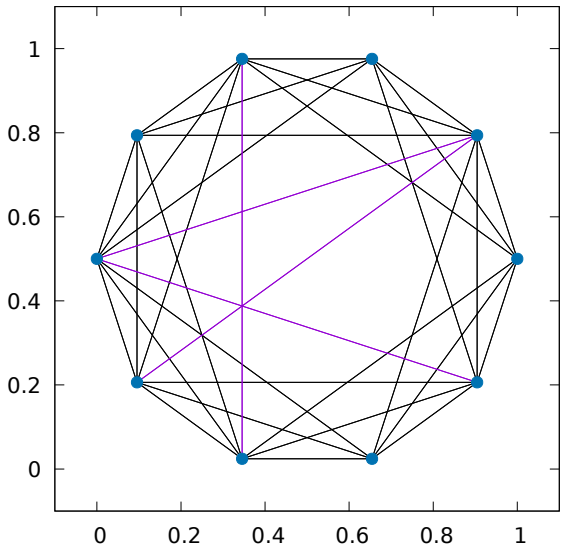

Figura 1

Trabajo con Funciones

A continuación observaremos los resultados que se obtuvieron al realizar las funciones correspondientes:

Se trabajo con diferentes valores de n, que se fueron aumentando por 10, cada vez que se corrían se tomaba el tiempo. Se realizaron corridas hasta con $n = 340$ teniendo un compartamiento lineal hasta los con $n = 300$ pero variando después. (Figura 2)

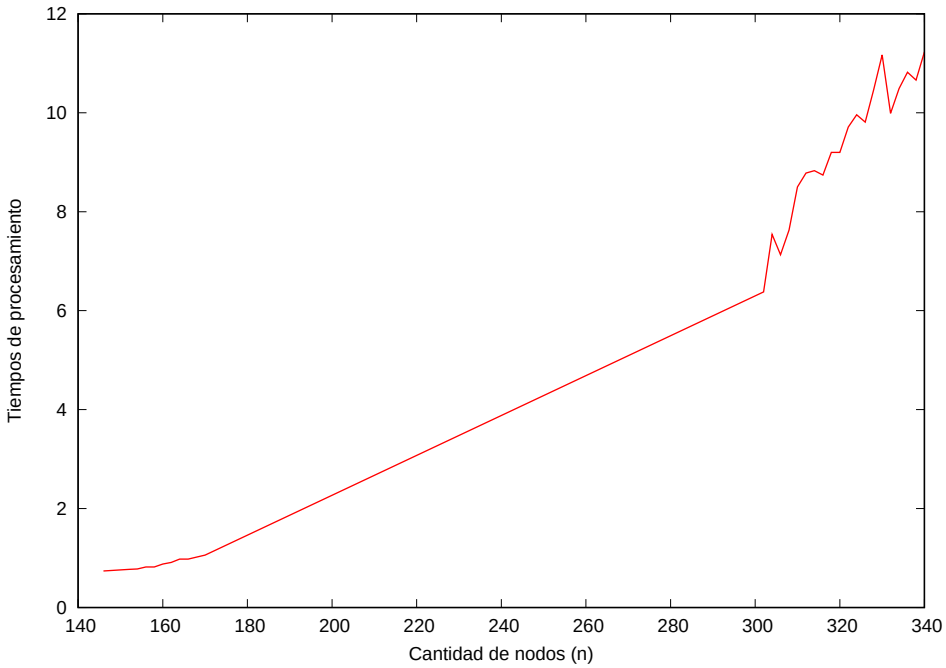

Figura 2: Tiempos de ejecución

En relación a los valores de densidad contra los valores de probabilidad, esto para poder contrastar la relación que se tienen (Figura 3).

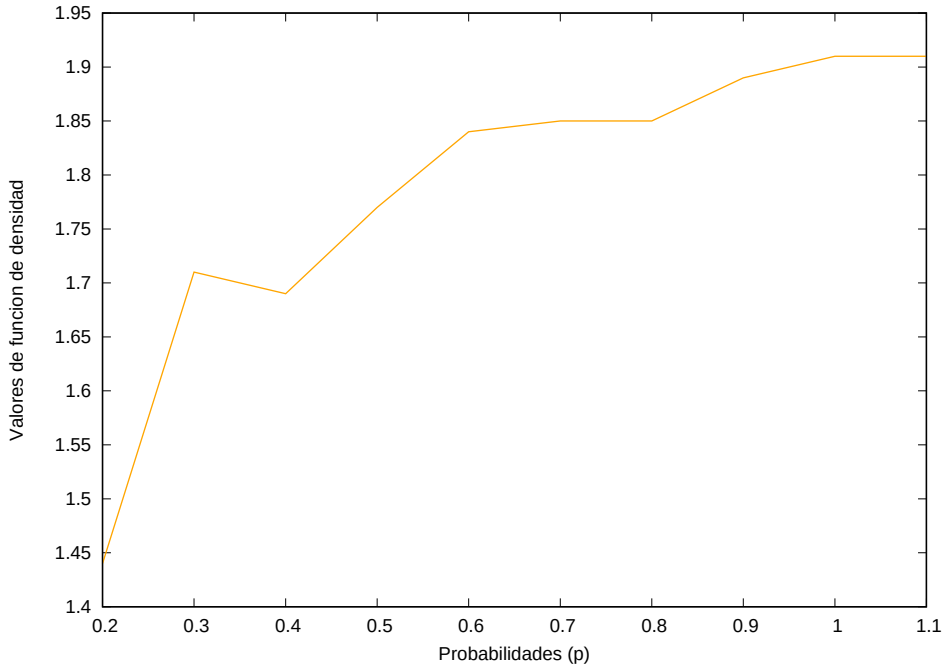

Figura 3: Tiempos contra funciones de densidad

Podemos observar que a mayor probabilidad, mayor la densidad de conexiones entre los nodos. Con la función *distancia* se busco poder obtener los valores promedio de distancias entre los nodos conectados, asÃ poder obtener un archivo con esos valores para poder graficarlos. Se busco normalizar esas distancias, diviendolas entre su cota mayor con la función *floor* obtuvimos el valor mayor de las distancias posibles, con eso se pudp dividir entre el promedio de todas las distancias.

```
def distancia(self):
        self.s = floor((3*Tk)-(i/4))
        with open ("Distancia.txt", "w") as distancia:
            self.sumDis = 0
            for u in self.d:
                self.sumDis = self.sumDis + self.d[u]
            self.avgDis = self.sumDis/len(self.d)
            self.DisNormalizada = self.avgDis / self.s
            print(self.DisNormalizada, file = distancia)
```

En la figura 4 donde se quiso observar la relación que hay entre la densidad de las conexiones y las distancias promedio contra las probabilidades de conexión, se puede entender que las distancias van aumentando y las densidades se mantienen muy similares a lo largo del cambio de probabilidades.

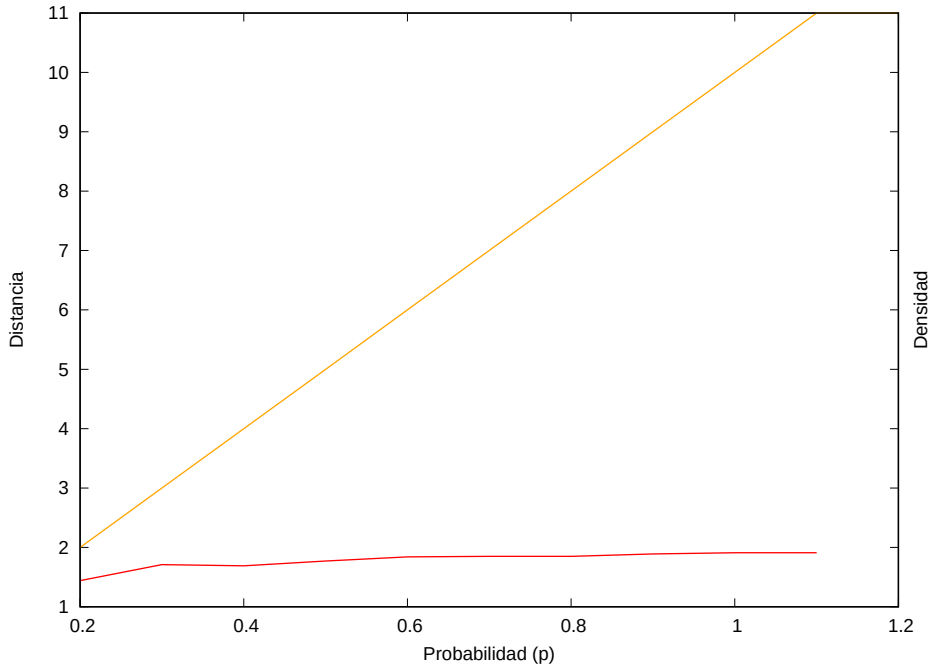

Figura 4: Distancias Promedio Normalizadas y Densidad Promedio con probabilidad de conexión

En conclusión podemos destacar que el compartamiento del algoritmo Floyd-Warshall varÃa en cada caso que se especifique, observandose que va aumentando en sus tiempos de procesamiento, asÃ como en su densidad de conexiones y las distancias promedios, pero manteniendose en sus probabilidades casi sin variación.

Bibliografía

[1] Samuel Sánchez, Tarea 3, `//github.com/samsan91/Opt.-Flujo-en-Redes/tree/master/Tarea3`, 2018.