Tarea 4
Optimización de flujo en redes

Victor Aram Dominguez Ramirez

23 de Abril de 2018

1. Introducción

En este reporte tenemos un numero determinado de nodos situados en una circunferencia conectados entre si. Mediante el algoritmo de Floyd-Warshall calculamos un resultado útil para una distancia promedio de aristas entre las conexiones de nodos. Tenemos en cuenta también el coeficiente de agrupamiento de cada nodo, este considera que cada nodo del grafo midiendo la densidad de su vecindad de sus nodos vecinos. Las propiedades van a variar dado que existe un par de nodos aleatorios no asociados que tienen una probabilidad de de que se conecten. Teniendo las propiedades de colectividad y el coeficiente de agrupamiento tenemos haremos pruebas variando estas propiedades donde van variando mediante la probabilidad de conectar un par de nodos en el grafo.

2. Programación del programa

Creamos un grafo con n cantidad de nodos situados al rededor de una circunferencia con un centro c y radio que sera donde estén situados los nodos. Para que los nodos tengan una congruencia entre si sobre la circunferencia donde tendremos a $x = rad * cos(ang * n) + c[0]$ y $y = rad * sin(ang * n) + c[1]$ donde tenemos en consideración al radio y centro de la circunferencia.

```
def agrega(self, v, c, rad, ang):
    with open("Nodos.dat", "w") as crear:
        for n in range(1, v+1):
            x = rad*cos(ang*n) + c[0]
            y = rad*sin(ang*n) + c[1]
            self.V[v]=(x,y)
            self.nodos.append((x,y))
            print(x, y, file = crear)
            if not (x, y) in self.vecinos:
                self.vecinos[(x,y)] = []
```

Tendremos que conectar cada nodo con $2 * k$ nodos. Los k nodos sucesores y los k nodos anteriores en la circunferencia.

```
def conecta(self, k):
    for j in range(0,v):
        x1 = self.nodos[j][0]
        y1 = self.nodos[j][1]
        for r in range(1,k+1):
            x2 = self.nodos[j-r][0]
            y2 = self.nodos[j-r][1]
            #conexión
            self.E[(a, b)] = self.E[(b, a)]=r
            self.aristas.append((x1,y1,x2,y2))
            self.aristas[(a,b)] = self.aristas[(b,a)] = r
            self.vecinos[a].append(b)
            self.vecinos[b].append(a)
```

Y por ultimo tendremos que conectar nodos aleatorios no previamente conectados con una probabilidad asignada

```
def Inter(self, h, m, v):
    prob=random()
    maxi=0.2
    if prob > maxi:
        for g in range(v):
            t1=self.nodos[h][0]
            n1=self.nodos[h][1]
            t2=self.nodos[m][0]
            n2=self.nodos[m][1]
            self.aristas.append((t1,n1,t2,n2))
```

Dada la figura 1 se ve los nodos situados con una distancia similar entre cada uno y dentro de una circunferencia. Se ven las aristas que conectan en cada par de nodos.

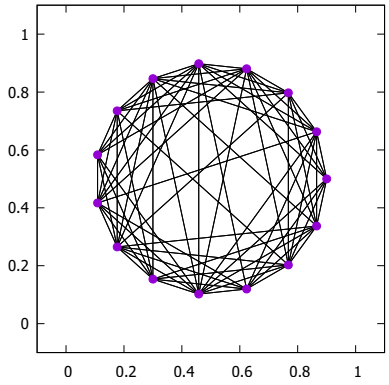

Figura 1: Ejemplo de 15 nodos con $k = 4$.

Haciendo variaciones obtenemos la densidad de las vecindades de cada nodo vecino al cual se tienen, obtuvimos también el promedio de las distancias entre los nodos (figura 2).

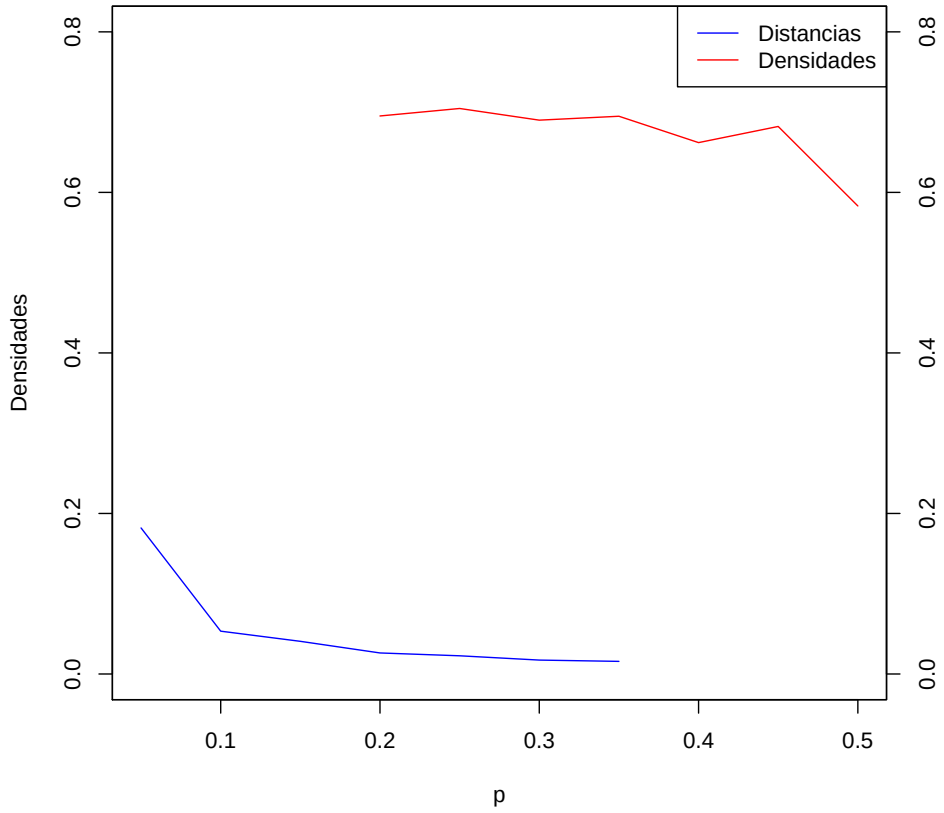

Figura 2: Densidad y distancia.
También tenemos las gráficas con los tiempos de ejecución aumentando los nodos para saber su aumento en el tiempo (figura 3).

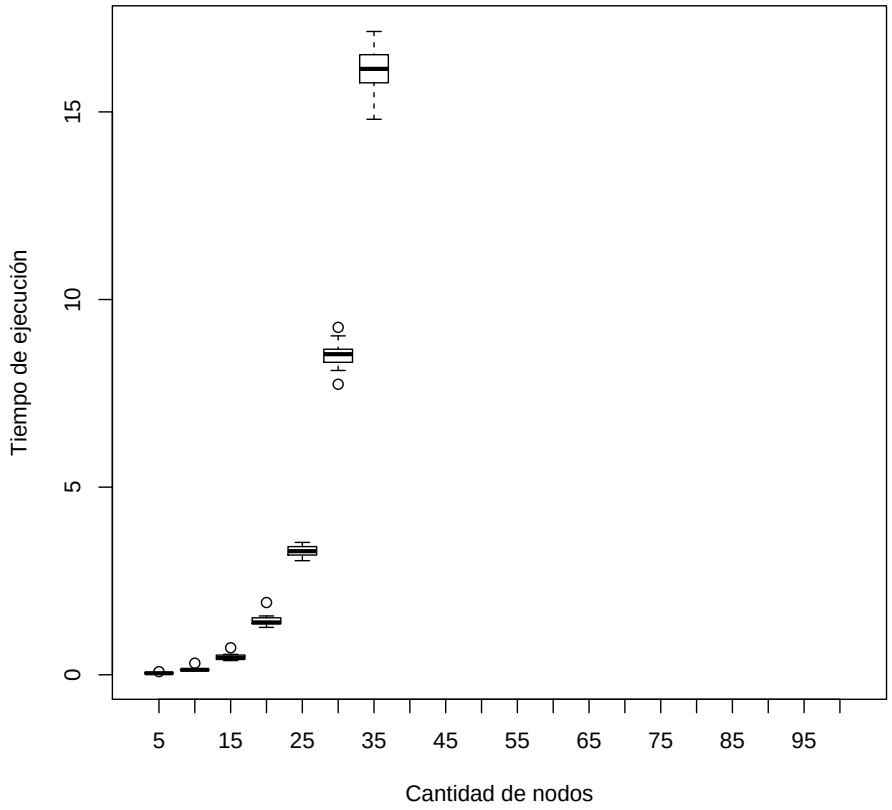

Figura 3: Tiempo de ejecución.