Tarea 3

1985269

18 de marzo de 2019

1. Resumen

Fueron escogidos cinco algoritmos de la librería NetworkX, cada uno de estos resolvió cinco problemas generados de forma aleatoria. Fueron realizadas 30 replicas durante un segundo para cada problema con cada algoritmo. A continuación se muestra la información recolectada a partir del análisis de los tiempos de procesamiento.

2. Deep first search tree

En los datos figura 1,2,3 pueden ser obsevardos la densidad con la que se distribuyen los tiempos medios que demora este algoritmo para resolver cuatro de los cinco problemas generados. Cada uno las gráficas muestran que el comportamiento de los datos no sigue un distribución normal y pesentan datos atípicos y saltos en los intervalos de frecuencia.

A continuación se comparte el código de Python con el que se recopilación la información:

```
Graf=nx.Graph()
rog=0
diccionario_inst_Shortest_path={}
while rog<=4:
    Graf.clear()
    rango=random.randint(random.randint(10,len(matrizadd)),len(matrizadd))
    for i in range(rango):
        for j in range(rango):
            if matrizadd[i,j]!=0:
                matrizadd[j,i]=0
                #Graf.add_weighted_edges_from([(i,j,matrizadd[i,j])])
                Graf.add_edges_from([(i,j)])
    cont=0
    lista_tiempos=[]
    lista_tiempos_completos={}
    tiempo_de_paro=0
    tiempo_inicial=0
    tiempo_final =0
    tiempo_ejecucion=0
    for r in range(30):
        lista_tiempos_completos[r+1]=[]
        tiempo_de_paro=0
        while tiempo_de_paro<1:
            tiempo_inicial = time()
            nx.shortest_path(Graf, source=None, target=None, weight=None,
    method='dijkstra')
            tiempo_final = time()
            tiempo_ejecucion = tiempo_final - tiempo_inicial
            if tiempo_ejecucion>0.0:
                lista_tiempos_completos[r+1].append((tiempo_ejecucion*10000))
            tiempo_de_paro+=tiempo_ejecucion
    guardar_n_e[rog]=[]
    diccionario_inst_Shortest_path[rog]=[]
    for i in lista_tiempos_completos.keys():
         media=np.mean(lista_tiempos_completos[i])
         diccionario_inst_Shortest_path[rog].append(media)
    guardar_n_e['nodos'].append(len(Graf.nodes))
    guardar_n_e['edges'].append(len(Graf.edges))
    guardar_n_e['media'].append(np.mean(diccionario_inst_Shortest_path[rog]))
    guardar_n_e['desv'].append(np.std(diccionario_inst_Shortest_path[rog]))
    rog+=1
```

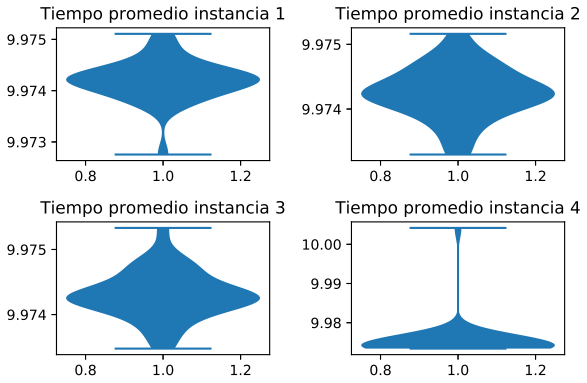

Figura 1: Violinplot dfstree

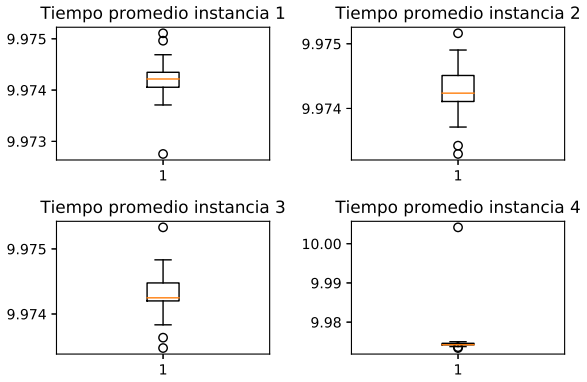

Figura 2: Boxplot dfstree

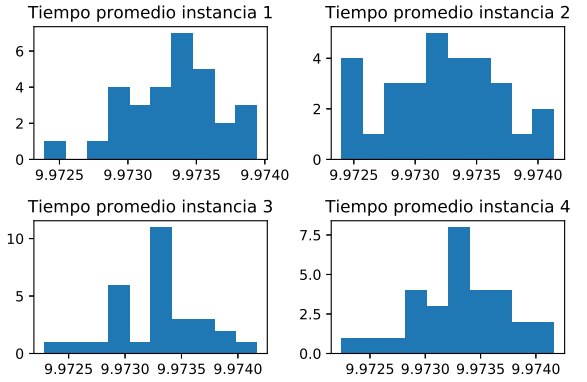

Figura 3: dfstree

3. Greedy color

En este caso la mayoría de los datos figuras 4,5,6 se encuentran en un intervalo con observaciones aisladas en otro intervalo mayor. Tampoco hay normalidad y existe un sesgo a la izquierda.

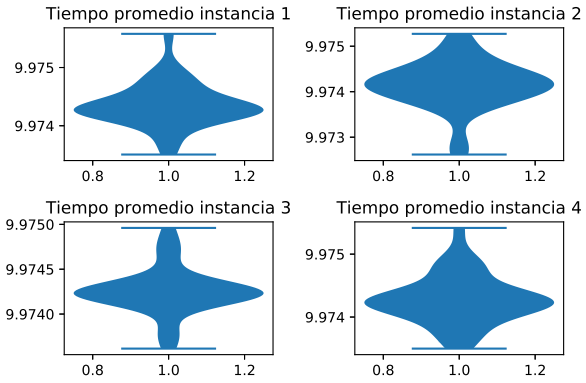

Figura 4: Violinplot greedy color

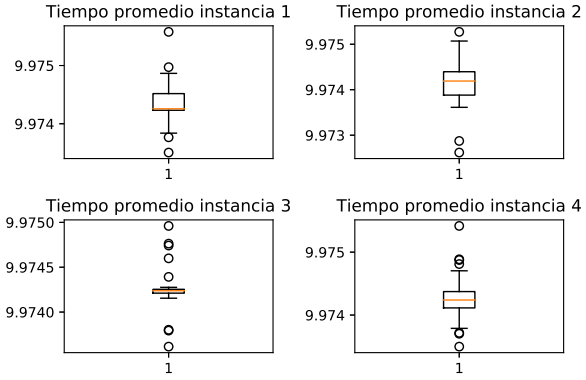

Figura 5: Boxplot greedy color

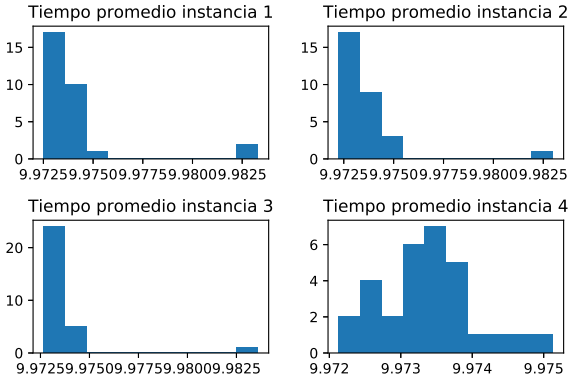

Figura 6: Greedy color

4. Maximun flow

De forma parecida las figuras muestran 7,8,9 como los datos se encuentran en intervalos separados y no siguen una distribucion precisa, no se aprecia ninguna relación con las característica de la instancia.

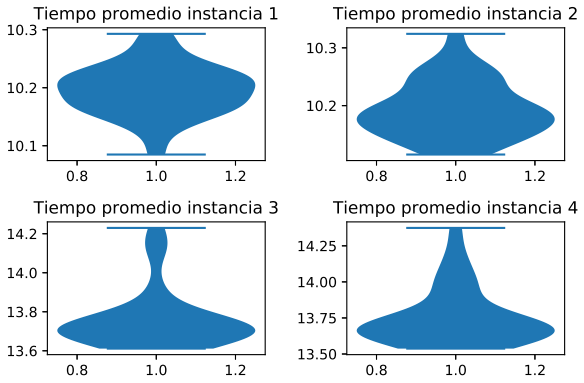

Figura 7: Violinplot maximun flow

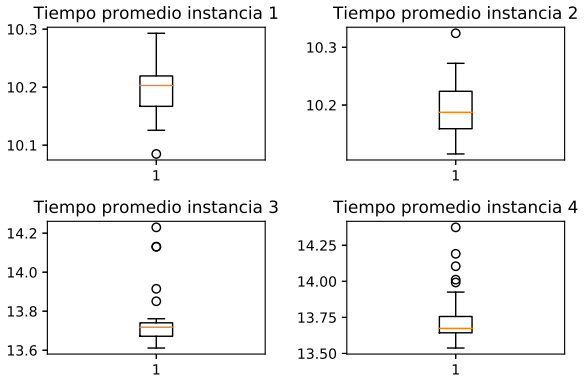

Figura 8: Boxplot maximun flow

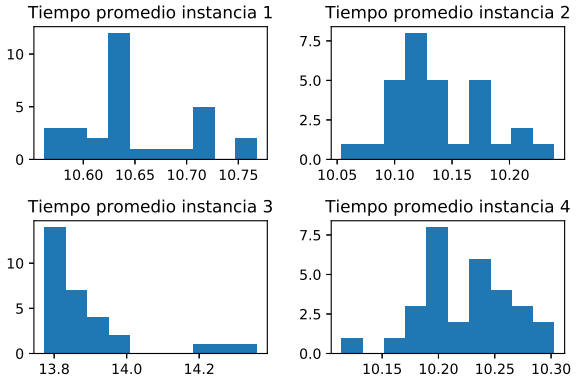

Figura 9: Maximun flow

5. Shortest path

En esta ocación vemos una mayor difenrencia entre los tiempos promedios de ejecucioón aun asi de estos no se infiere ningun comportamiento de probabilidad.

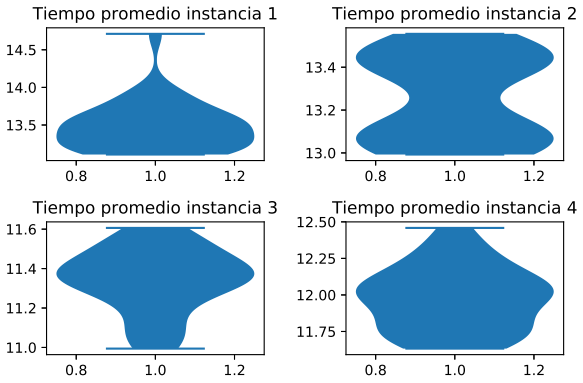

Figura 10: Violinplot Shortest path

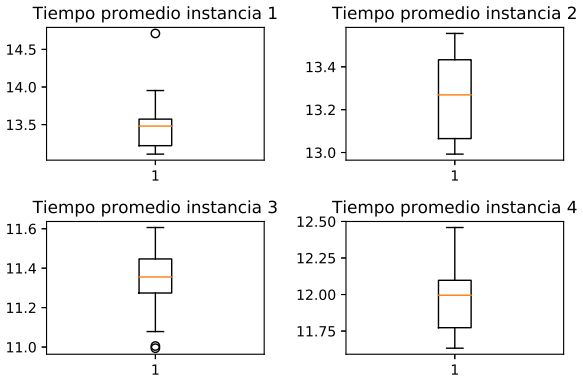

Figura 11: Boxplot Shortest path

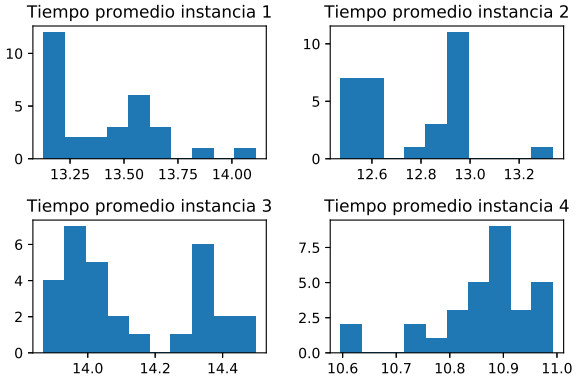

Figura 12: Shortestpath

6. Spanning tree

Las gráficas 16,17,18 19 corresponden a las medias con las desviaciones extandar de cada uno de los problemas resueltos con un algoritmo representado por una figura de un color. Los algoritmos para determinar la distancia más corta entre dos nodos así como el máximo flujo son los más sensibles al tiempo cuando aumenta la cantidad de nodos

y vértices del grafo. En ambos casos vértices vs tiempo y nodos vs tiempo se realizó un acercamiento en la zona donde se concentran la mayoría de los tiempos medios entre $(9.725; 9.9775)$.

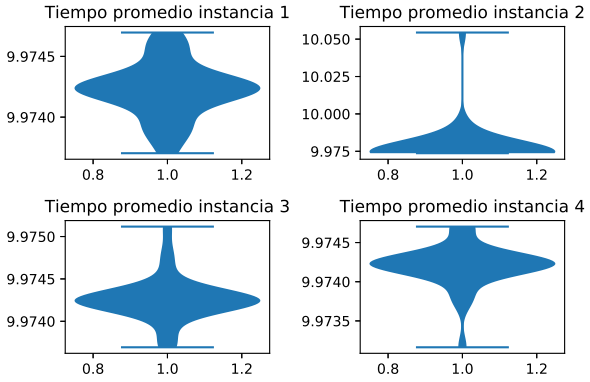

Figura 13: Violinplot spanning tree

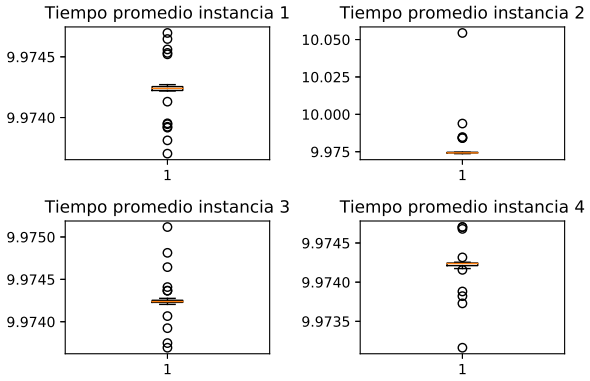

Figura 14: Boxplot spanning tree

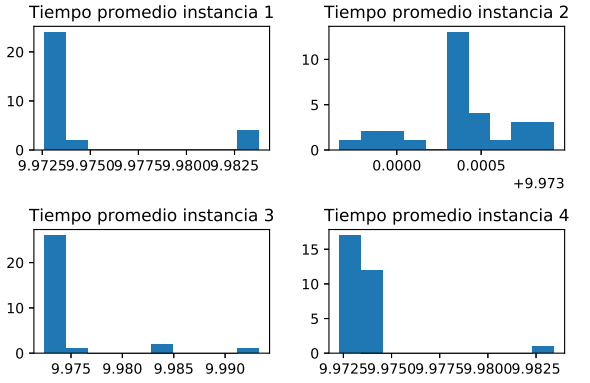

Figura 15: Spanning tree

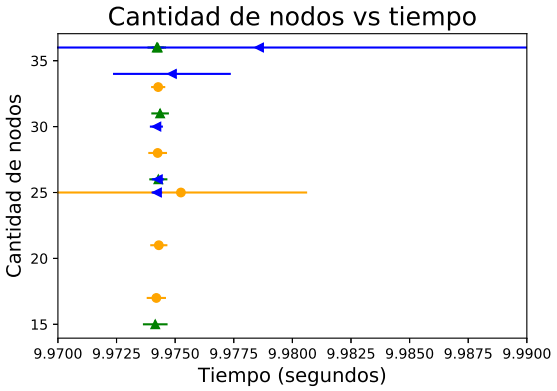

Figura 16: Errors bars nodes

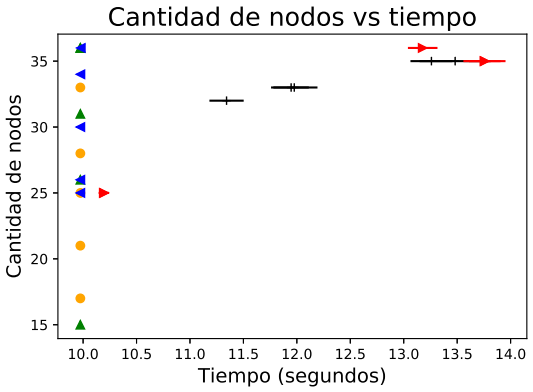

Figura 17: Errors bars nodes

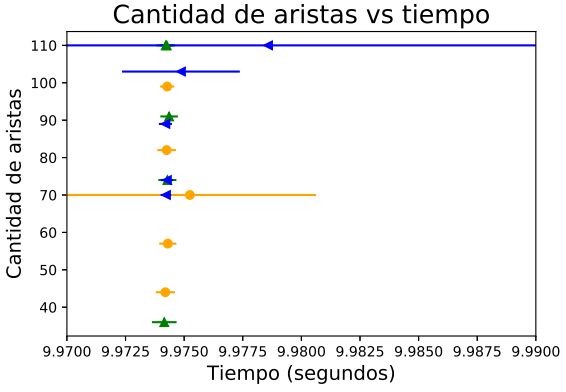

Figura 18: Errors bars edges

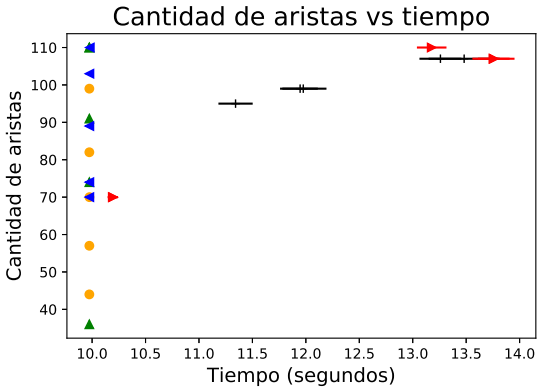

Figura 19: Errors bars edges

Referencias

[1] Janet M Six and Ioannis G Tollis. A framework for circular drawings of networks. In *International Symposium on Graph Drawing*, pages 107–116. Springer, 1999.

[2] Yehuda Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pages 496–508. Springer, 2003.

[3] EW Mayr. Praktikum algorithmen-entwurf (teil 6), nov. 2002, 6–11. *Technische Universität München http://wwwmayr. in. tum. de/lehre/2002WS/algoprak/part6. ps. gz.*

[4] E. Shaeffer. https://elisa.dyndns-web.com/.