Tarea 3

Jesus Angel Patlán Castillo (5261)

19 de marzo de 2019

En esta tarea se analizan 5 distintos algoritmos en grafos midiendo su tiempo de ejecución en base a grafos realizados tareas anteriores. Los algoritmos se obtuvieron por medio de la librería NetworkX [3] de Python [1], la librería Matplotlib [4] es utilizada para generar las gráficas de comparación entre los distintos algoritmos estudiados. El código empleado se obtuvo consultando la documentación oficial de la librería NetworkX [2]. Las imágenes y el código se encuentran disponibles directamente en mi repositorio [5].

1. Metodología

La librería NetworkX tiene a la mano diversos algoritmos para resolver distintos problemas que se pueden presentar en los grafos. Particularmente para esta investigación, se realiza el análisis de los algoritmos:

- Ruta más corta: El algoritmo de la ruta más corta, como lo dice el nombre del algoritmo, trata de buscar la ruta más corta entre dos nodos dentro de un grafo. En la librería de NetworkX esta disponible por medio de la función "all shortest paths(G,N1,N2)", donde el parámetro G es el grafo, N1 el nodo origen y N2 el nodo destino:

```
1        nx.all_shortest_paths(G1,1,5)
```

- Árbol búsqueda a profundidad: El algoritmo de árbol búsqueda a profundidad recorre el grafo dado para generar un árbol por medio de una búsqueda a profundidad. En la librería de NetworkX esta disponible por medio de la función "dfs tree(G)", donde el parámetro G es el grafo:

```
1        nx.dfs_tree(G6)
```

- Problema de la liga de amigos: El algoritmo para el problema de la liga de amigos trata de encontrar el nodo con mayor clique", es decir, el que posee un mayor de número de nodos conectados. En la librería de NetworkX esta disponible por medio de la función "max clique(G)", donde el parámetro G es el grafo:

```
1        nx.make_max_clique_graph(G11)
```

- Árbol de mínimo grado: El algoritmo de árbol de mínimo grado recorre entre los nodos del grafo para determinar el nodo con grado mínimo. En la librería de NetworkX esta disponible por medio de la función "treewidth min degree(G)", donde el parámetro G es el grafo:

```
        tree.treewidth_min_degree(G11)
```

- Árbol de mínima expansión: El algoritmo de árbol de expansión mínima recorre el grafo proporcionado para obtener el árbol con mínima expansión que se pueda generar en él. En la librería de NetworkX esta disponible por medio de la función "minimum spanning tree(G)", donde el parámetro G es el grafo:

```
        nx.minimum_spanning_tree(G11)
```

Cada uno de estos algoritmos fueron ejecutados sobre 5 grafos diferentes, siendo estos 5 grafos simples con ciclos no dirigidos. Cada uno de los grafos, enumerados del 1 al 5, contiene un número mayor de aristas y nodos conforme incrementa su numeración, es decir, el grafo 1 es el de menor número de aristas y nodos, mientras que el grafo 5 es el de mayor número de aristas y nodos. Para cada grafo cuales se realizo la cantidad de repeticiones posibles para que el tiempo mínimo de ejecución fueran de 5 segundos, y posteriormente a este conjunto de repeticiones por algoritmos se realizaron 30 réplicas distintas. Con estos datos, se obtuvo la media y la desviación estándar de cada una de las muestras obtenidas, además de realizar un histograma y un diagrama caja bigote para el análisis de cada uno de los algoritmos. Finalmente, se analiza el resultado final por medio de una gráfica de dispersión entre los datos de las 5 muestras, comparando el tiempo de ejecución contra la cantidad de nodos y contra la cantidad de aristas de cada grafo. A continuación se muestra el código para la generación del histograma, diagrama caja y bigote, y el histograma con la librería NetworkX, Matplot y Sciplot:

```
n, bins, patches = plt.hist(a1_times, 'auto', density=True,
    facecolor='blue', alpha=0.75)
y = scipy.stats.norm.pdf(bins, a1_mean,a1_standard)
plt.plot(bins, y, 'r--')
plt.xlabel('Tiempo (segundos)')
plt.ylabel('Frecuencia')
plt.title('Ruta m s corta (Media='+str(round(a1_mean,2))+' ,STD='+
    str(round(a1_standard,2))+' )',size=18, color='green')
plt.savefig('H1.eps', format='eps', dpi=1000)
plt.show()

#  Boxplots
to_plot=[a1_t1_times,a1_t2_times,a1_t3_times,a1_t4_times,
    a1_t5_times]
fig=plt.figure(1,figsize=(9,6))
ax=fig.add_subplot(111)
bp=ax.boxplot(to_plot, showfliers=False)
plt.xlabel('Grafo')
plt.ylabel('Tiempo (segundos)')
plt.title('Ruta m s corta')
```

```
plt.savefig('BP1.eps', format='eps', dpi=1000)
```

```
to_plot=[a1_t1_times,a1_t2_times,a1_t3_times,a1_t4_times,
    a1_t5_times]
fig=plt.figure(1,figsize=(9,6))
ax=fig.add_subplot(111)
bp=ax.boxplot(to_plot, showfliers=False)
plt.xlabel('Grafo')
plt.ylabel('Tiempo (segundos)')
plt.title('Ruta m s corta')
plt.savefig('BP1.eps', format='eps', dpi=1000)
plt.show()
```

```
plt.errorbar(y_1,x_1_1, xerr=z_1, fmt='o',color='blue',alpha=0.5)
plt.errorbar(y_2,x_1_2, xerr=z_2, fmt='s',color='yellow',alpha=0.5)
plt.errorbar(y_3,x_1_3, xerr=z_3, fmt='*',color='green',alpha=0.5)
plt.errorbar(y_4,x_1_4, xerr=z_4, fmt='h',color='red',alpha=0.5)
plt.errorbar(y_5,x_1_5, xerr=z_5, fmt='D',color='orange',alpha=0.5)
plt.xlabel('Tiempo (segundos)', size=14)
plt.ylabel('Nodos', size=14)
plt.title('Nodos vs tiempo',size=18)
plt.savefig('S1.eps', format='eps', dpi=1000)
plt.show()
```

Este proceso se realizó en una laptop con las siguientes características:

- Procesador: Intel Core i7-7500U 2.7GHz
- Memoria RAM: 16GB
- Sistema Operativo: Windows 10 64 bits

2. Resultados

2.1. Ruta más corta

A continuación se muestra el histograma en la figura 1 y el diagrama caja y bigote en la figura 2 con los resultados obtenidos:

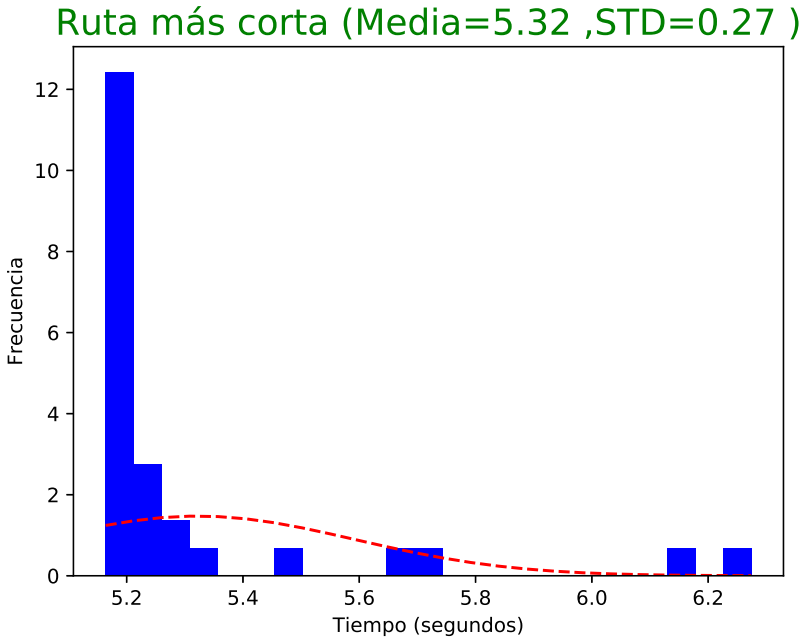

Figura 1: Histograma para la ruta más corta.

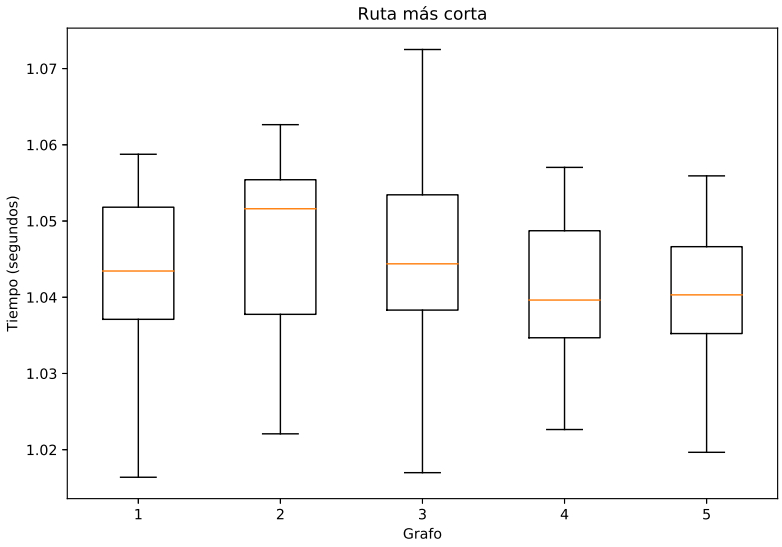

Figura 2: Diagrama caja y bigote para la ruta más corta.

2.2. Árbol búsqueda a profundidad

A continuación se muestra el histograma en la figura 3 y el diagrama caja y bigote en la figura 4 con los resultados obtenidos:

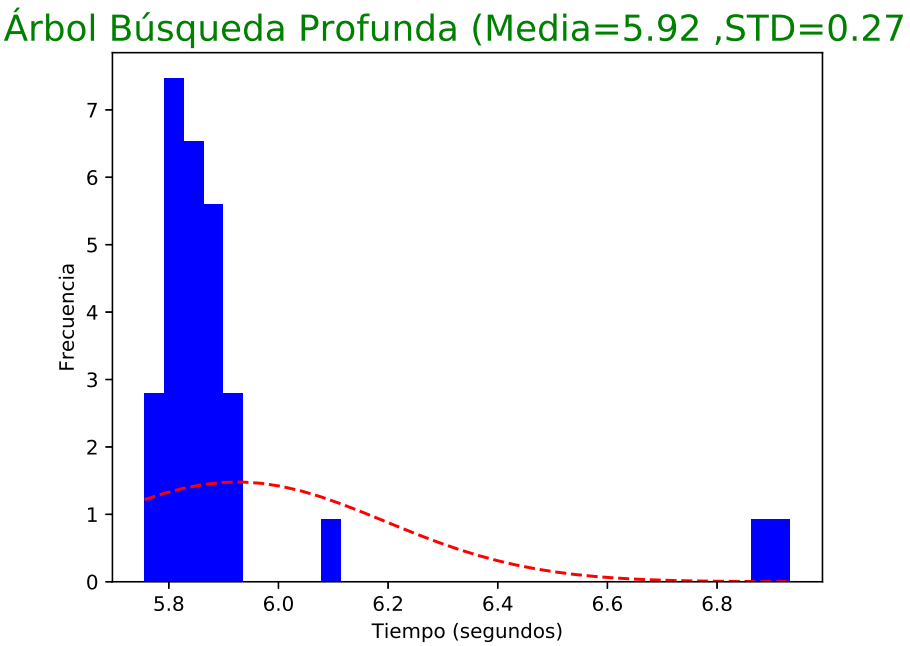

Figura 3: Histograma para el árbol búsqueda a profundidad.

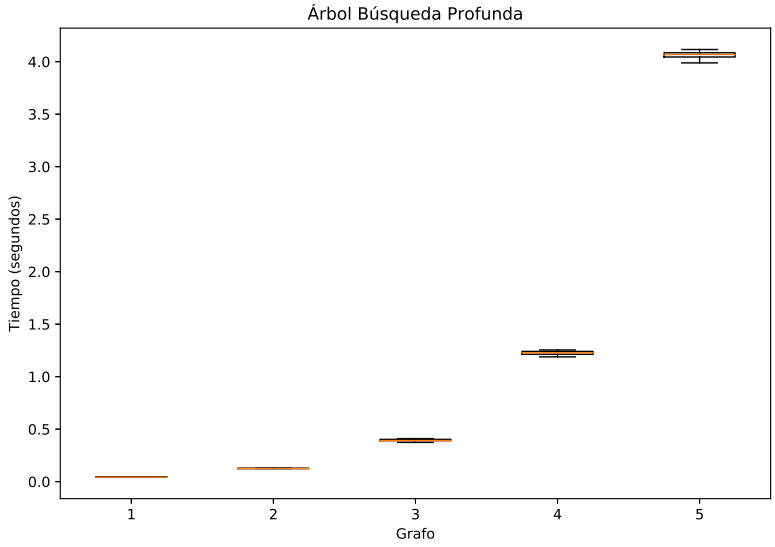

Figura 4: Diagrama caja y bigote para el árbol búsqueda a profundidad.

2.3. Problema de la liga de amigos

A continuación se muestra el histograma en la figura 5 y el diagrama caja y bigote en la figura 6 con los resultados obtenidos:

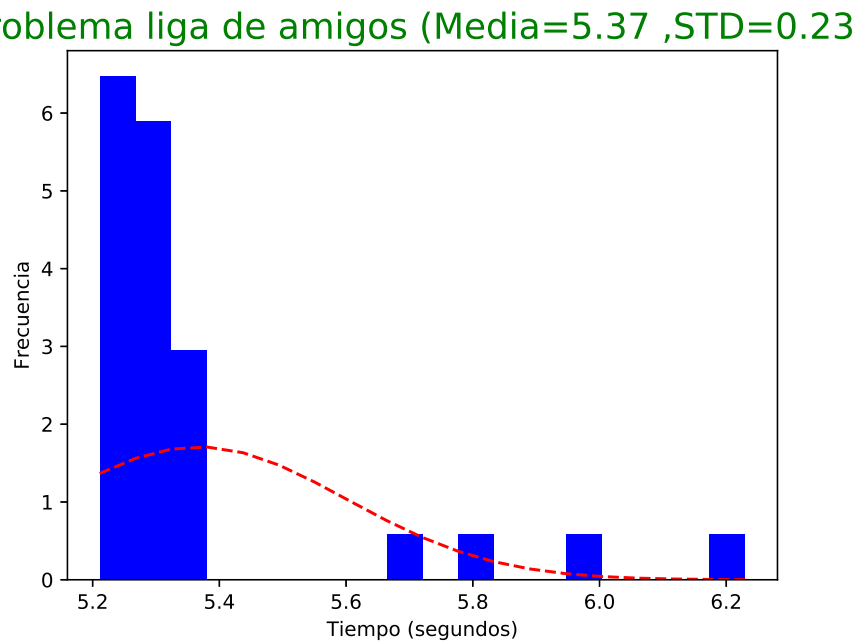

Figura 5: Histograma para el problema de la liga de amigos.

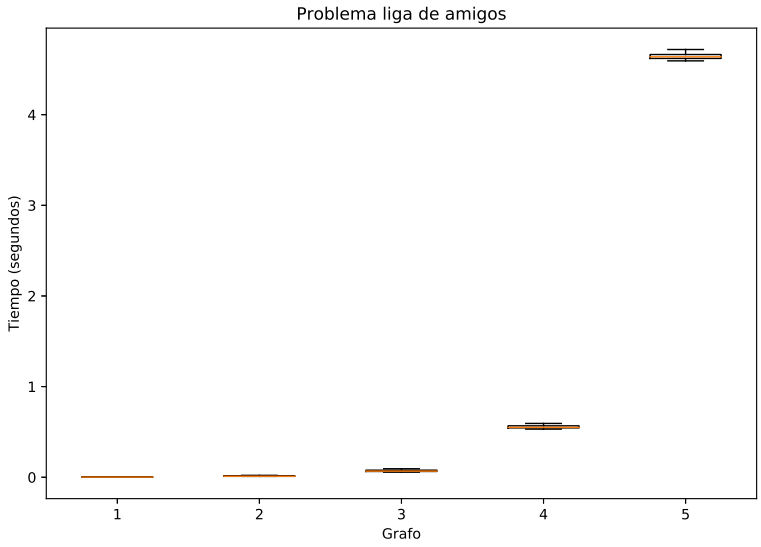

Figura 6: Diagrama caja y bigote para el problema de la liga de amigos.

2.4. Árbol de mínimo grado

A continuación se muestra el histograma en la figura 7 y el diagrama caja y bigote en la figura 8 con los resultados obtenidos:

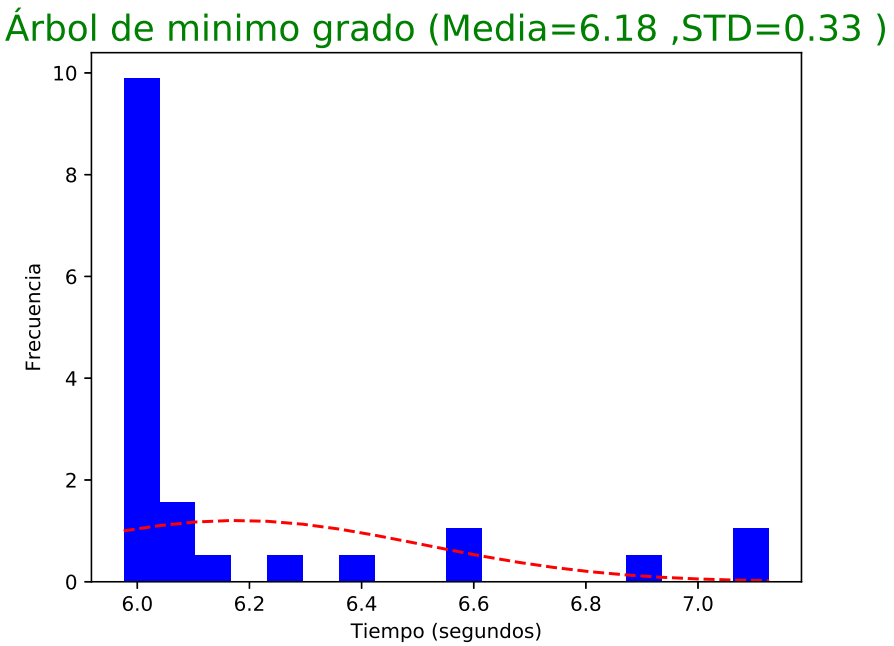

Figura 7: Histograma para el árbol de mínimo grado.

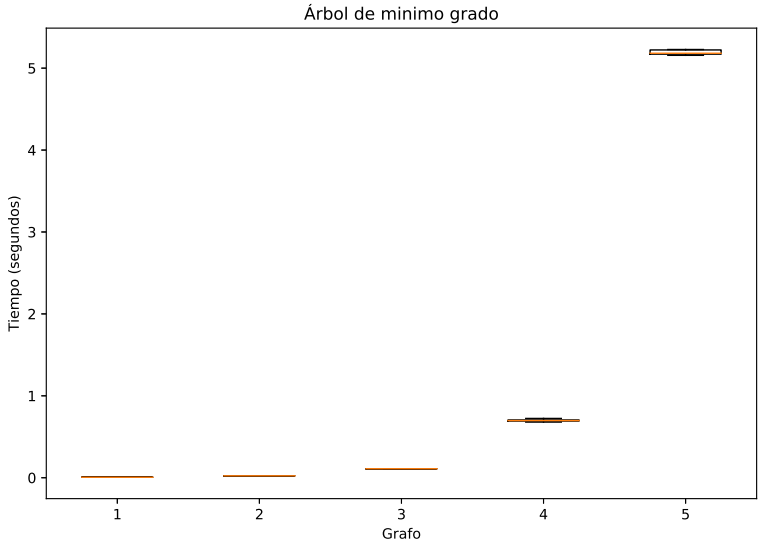

Figura 8: Diagrama caja y bigote para el árbol de mínima grado.

2.5. Árbol de mínima expansión

A continuación se muestra el histograma en la figura 9 y el diagrama caja y bigote en la figura 10 con los resultados obtenidos:

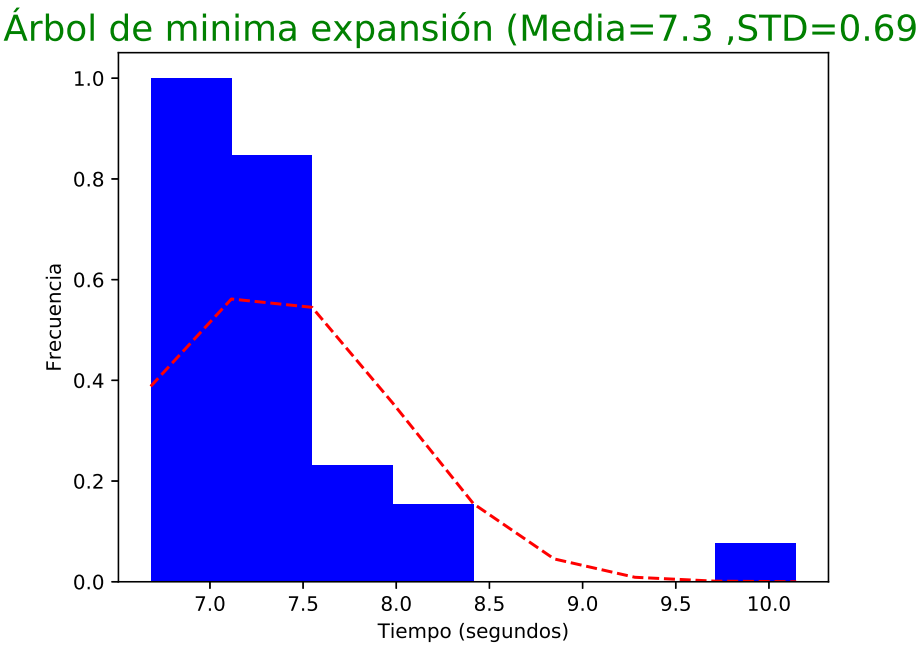

Figura 9: Histograma para el árbol de mínima expansión.

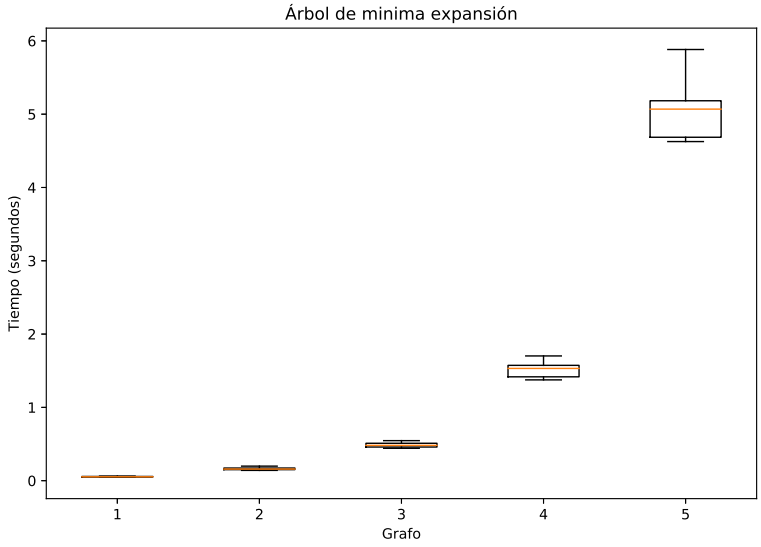

Figura 10: Diagrama caja y bigote para el árbol de mínima expansión.

2.6. Comparación entre algoritmos

La gráfica de dispersión siguiente compara los resultados obtenidos entre los 5 algoritmos, en base al tiempo de ejecución vs el número de nodos (figura 11) y el tiempo de ejecución vs el número de aristas (figura **??**):

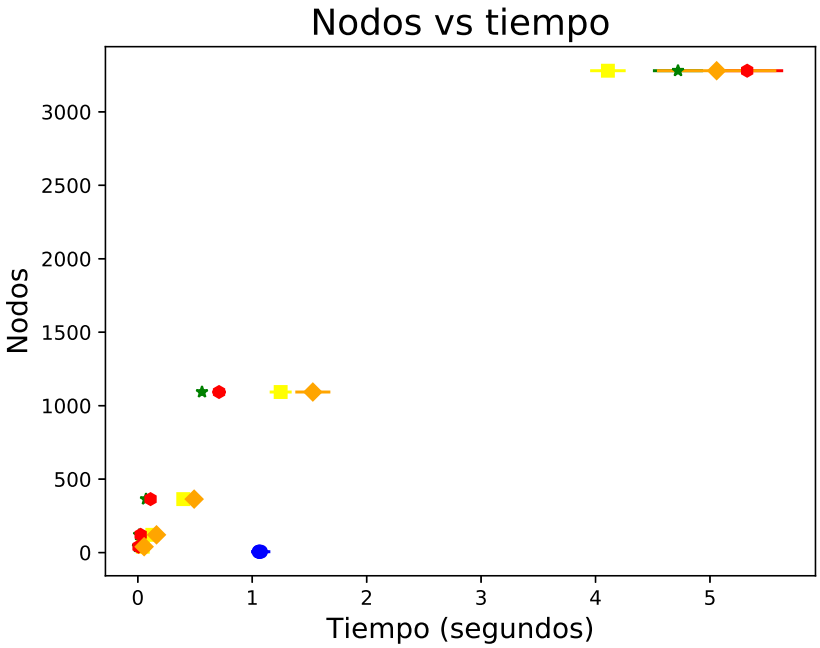

Figura 11: Gráfica de dispersión: tiempo de ejecución vs número de nodos.

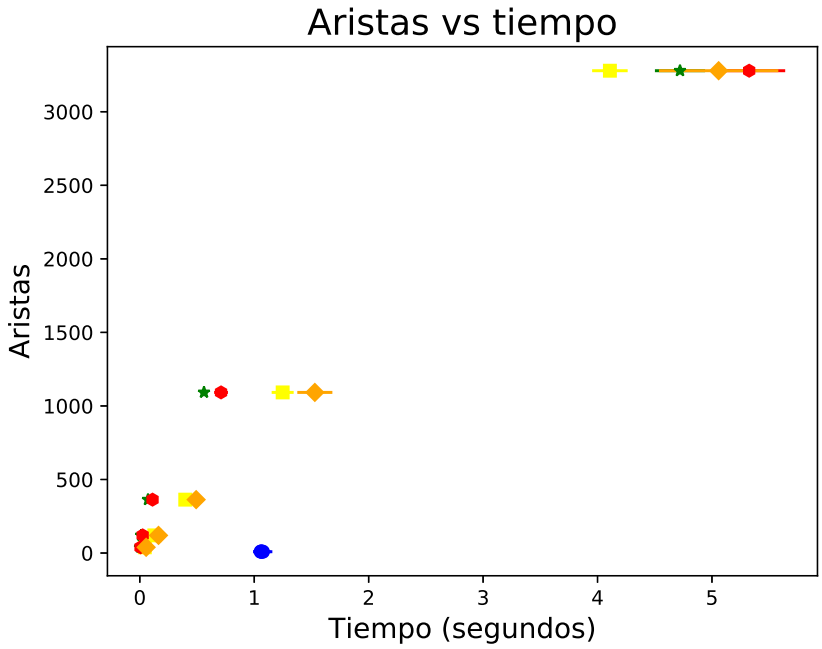

Figura 12: Gráfica de dispersión: tiempo de ejecución vs número de aristas.

3. Conclusiones

Como conclusión de esta investigación, se tiene que el tiempo de ejecución de cualquiera de los 5 algoritmos estudiados es proporcional al número de nodos y de aristas que se tiene en el grafo, por lo que un grafo con un número mayor de nodos incrementa el tiempo de ejecución de los algoritmos.

Referencias

[1] Python Software Foundation Versión 3.7.2. https://www.python.org/.

[2] NetworkX developers con última actualización el 19 de Septiembre 2018. https://networkx.github.io/documentation/stable/index.html.

[3] NetworkX developers Versión 2.0. https://networkx.github.io/.

[4] The Matplotlib development team Versión 3.0.2. https://matplotlib.org/.

[5] Patlán Castillo Jesús Angel. Repositorio Optimización Flujo en Redes. https://github.com/JAPatlanC/Flujo-Redes.