Medición de tiempo de ejecución

Matrícula: 1985281

Resumen

Este trabajo busca resolver grafos con la implementación de los algoritmos de NetworkX de `Python`. Se seleccionan algunos grafos de la tarea 2 y se generan otros, para así poder resolver cinco grafos distintos para cada uno de los cinco algoritmos seleccionados.

Introducción

La librería NetworkX de `Python` proporciona algoritmos para resolver diversos problemas, de los cuales se escogieron los siguientes cinco:

- Centralidad intermedia: Calcula la centralidad de intermediación de rutas más cortas para todos los nodos.
- Árbol de expansión mínimo: Calcula un árbol de expansión mínimo del grafo.
- Flujo máximo: Calcula el flujo máximo de un nodo a otro.
- Ruta mas corta: Calcula las rutas más cortas entre los nodos del grafo.
- Coloración glotona: Colorea una gráfica usando varias estrategias de coloración codiciosa.

Metodología y Resultados

Se seleccionan algunos grafos utilizados en la tarea 2 y se generan otros grafos que son compatibles con los requerimientos que el algoritmo tiene sobre sus datos de entrada, se repite cada ejecución por la cantidad suficiente de veces para que el tiempo total de ejecución del conjunto de réplicas sea mayor a cinco segundos (se sube el número de réplicas hasta que esto se logre). Luego se repite la medición del conjunto de réplicas hasta treinta veces en total, sin exceder quince minutos de tiempo total de ejecución.

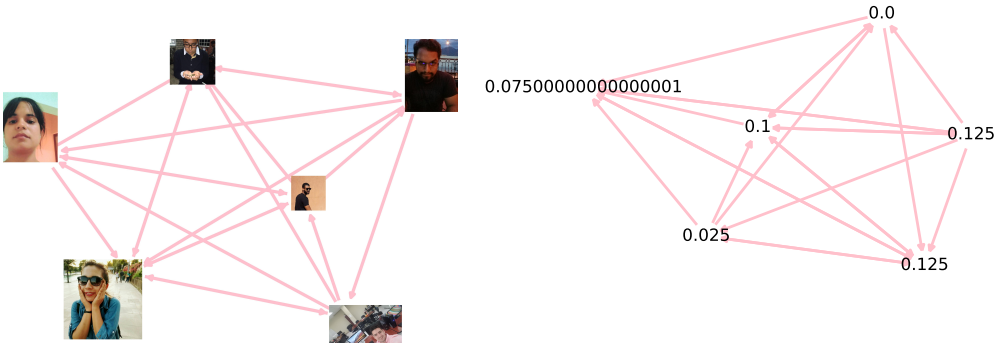
Después con la ayuda de librerías adicionales tales como `NumPy` y `SciPy`, se calcula para cada algoritmo el promedio (μ) y desviación estándar (σ) de la ejecución del conjunto de réplicas y se grafica un histograma de las mediciones individuales para cada algoritmo.

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import time
4 import numpy as np
5 from scipy import stats
```

Centralidad intermedia

La figura [1.b] muestra la centralidad intermedia para cada nodo del grafo ejemplo de redes sociales de la tarea 2.

```
b_w = nx.betweenness_centrality(G1)
```



(a) Versión primaria (b) `betweenness_centrality`

Figura 1: Ejemplo de redes sociales

También, otro grafo utilizado para aplicar este algoritmo es el famoso grafo de la red social *Krackhardt Kite*, una red social de diez actores presentada por David Krackhardt para ilustrar centralidad intermedia. Véase figura [2].

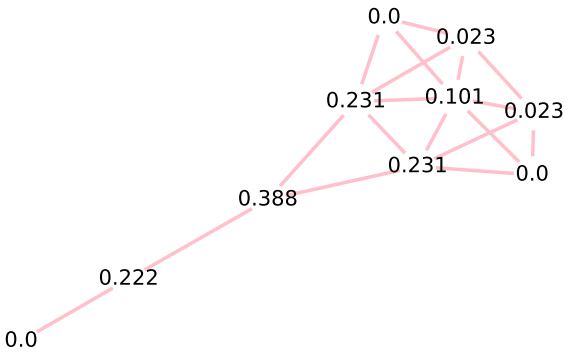
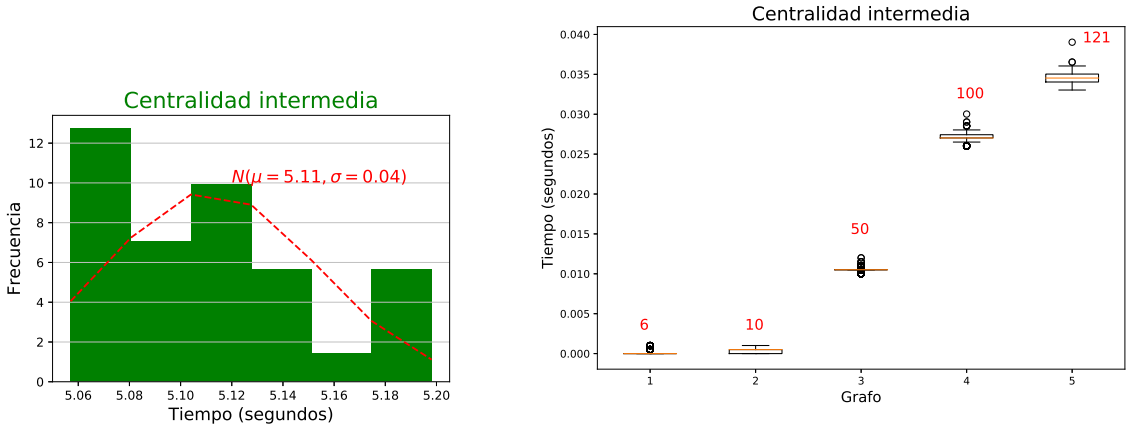


Figura 2: Red social de Krackhardt Kite

Ahora se calcula para el algoritmo, el promedio y desviación estándar de la ejecución del conjunto de réplicas y se grafica un histograma.



(a) Histograma

(b) Diagrama de caja y bigotes

Figura 3: Histograma y diagrama de caja y bigotes

En la figura [3.a] se muestra el histograma de los tiempos promedio del algoritmo y se calcula el promedio (μ) y la desviación estándar (σ) para graficar la línea punteada que corresponde a la distribución normal $N \sim (\mu = \frac{\sum_{i=1}^{n} x_i}{n} = 5.11, \sigma = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n} = 0.04)$ y se puede ver que la línea punteada no se ajusta al histograma por lo que se realiza la prueba estadística *Shapiro-Wilk* para checar la normalidad de los datos.

```
1 stats.shapiro(tiempos_algoritmo_1)
```

In: `stats.shapiro` (tiempos_algoritmo_1) Out: $(W = 0.6412, p = 2.4210 \cdot 10^{-7})$

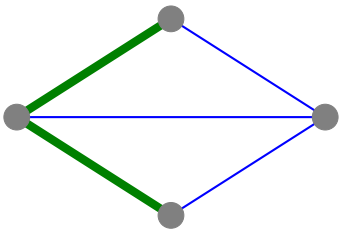
La prueba muestra que el p-valor es menor a 0.05 (nivel de significancia), entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal).

Por otro lado en la figura [3.b] se muestran los diagramas de caja y bigotes para los tiempos individuales de los cinco grafos utilizados en el algoritmo, el número rojo representa la cantidad de nodos del grafo y se puede apreciar que conforme aumenta la cantidad de nodos en el grafo, los tiempos computacionales también crecen.

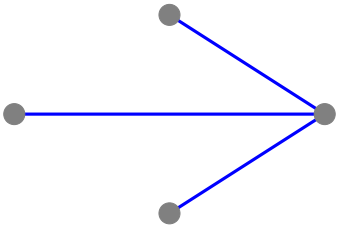
Árbol de expansión mínimo

La figura [4] muestra ejemplos de grafos y sus respectivos grafos de árboles de expansión mínima.

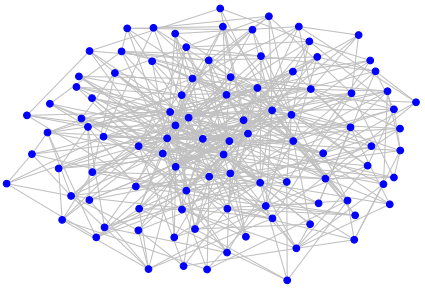
```
T1 = nx.minimum_spanning_tree(G6)
```



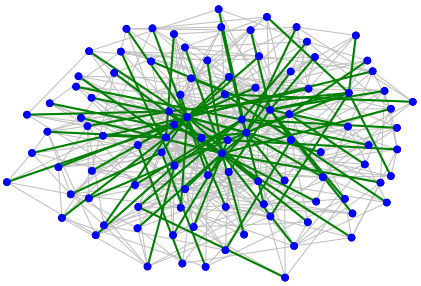
(a) Puentes de *Könisberg*



(b) Árbol de expansión mínima de (a)



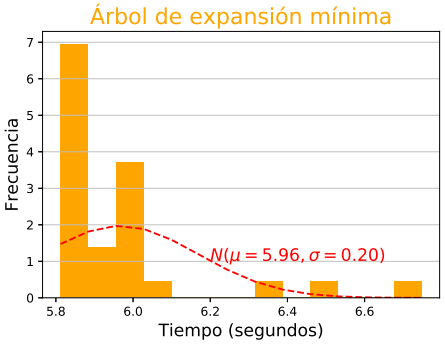
(c) Grafo de 100 nodos



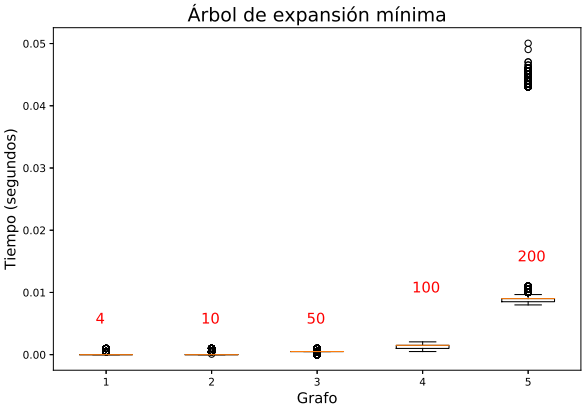
(d) Árbol de expansión mínima de (c)

Figura 4: Ejemplos de árbol de expansión mínima

Ahora se calcula para el algoritmo, el promedio y desviación estándar de la ejecución del conjunto de réplicas y se grafica un histograma.



(a) Histograma



(b) Diagrama de caja y bigotes

Figura 5: Histograma y diagrama de caja y bigotes

En la figura [5.a] se muestra el histograma de los tiempos promedio del algoritmo y se calcula el promedio (μ) y la desviación estándar (σ) para graficar la línea punteada que corresponde a la distribución normal $N \sim (\mu = 5.96, \sigma = 0.20)$ y se puede ver que la línea punteada no se ajusta al histograma por lo que se realiza la prueba estadística *Shapiro-Wilk* para checar la normalidad de los datos.

In: `stats.shapiro` (tiempos_algoritmo_2)	Out: $(W = 0.5320, p = 1.1895 \cdot 10^{-8})$

La prueba muestra que el p-valor es menor a 0.05 (nivel de significancia), entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal).

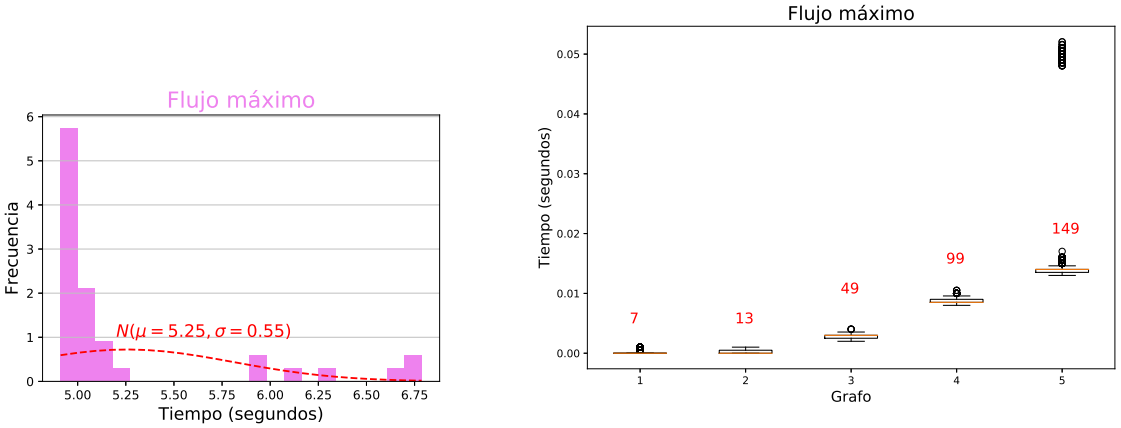
Por otro lado en la figura [5.b] se muestran los diagramas de caja y bigotes para los tiempos individuales de los cinco grafos utilizados en el algoritmo, el número rojo representa la cantidad de nodos del grafo y se puede apreciar que conforme aumenta la cantidad de nodos en el grafo, los tiempos computacionales también crecen.

Flujo máximo

En este ejemplo se le pide al algoritmo de flujo máximo encontrar el flujo máximo del nodo uno al nodo diez del grafo llamado G11.

```
1 MF_1 = nx.maximum_flow(G11, 1, 10, capacity='capacidad', flow_func=None)
```

Ahora se calcula para el algoritmo, el promedio y desviación estándar de la ejecución del conjunto de réplicas y se grafica un histograma.



(a) Histograma

(b) Diagrama de caja y bigotes

Figura 6: Histograma y diagrama de caja y bigotes

En la figura [6.a] se muestra el histograma de los tiempos promedio del algoritmo y se calcula el promedio (μ) y la desviación estándar (σ) para graficar la línea punteada que corresponde a la distribución normal $N \sim (\mu = 5.25, \sigma = 0.55)$ y se puede ver que la línea punteada no se ajusta al histograma por lo que se realiza la prueba estadística *Shapiro-Wilk* para checar la normalidad de los datos.

In: `stats.shapiro` (tiempos_algoritmo_3) Out: $(W = 0.8749, p = 0.0021)$

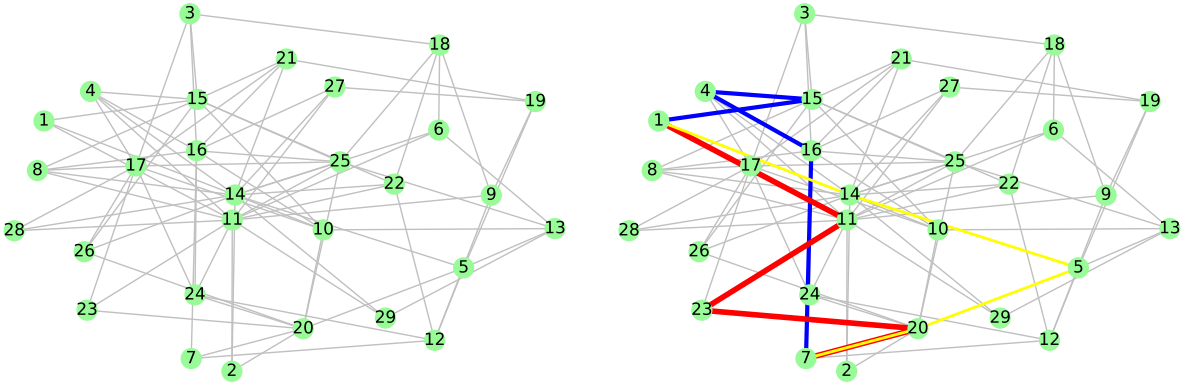
La prueba muestra que el p-valor es menor a 0.05 (nivel de significancia), entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal).

Por otro lado en la figura [6.b] se muestran los diagramas de caja y bigotes para los tiempos individuales de los cinco grafos utilizados en el algoritmo, el número rojo representa la cantidad de nodos del grafo y se puede apreciar que conforme aumenta la cantidad de nodos en el grafo, los tiempos computacionales también crecen.

Ruta mas corta

La figura [7] muestra un ejemplo de un grafo y sus respectivas rutas mas cortas del nodo uno al nodo siete del grafo llamado G16.

```
rutas1 = [p for p in nx.all_shortest_paths(G16, source=1, target=7, weight=None,
    method='dijkstra')]
```

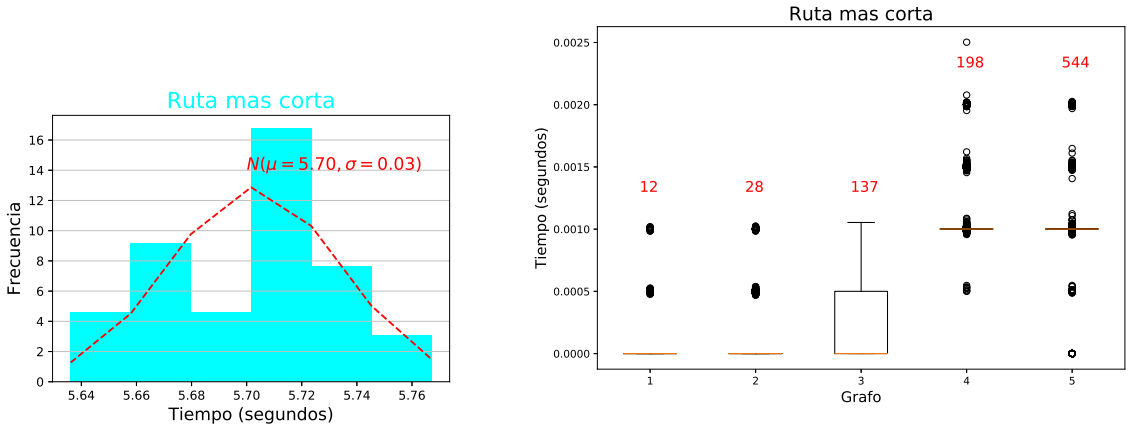


(a) Grafo de 30 nodos

(b) Rutas mas cortas de (a)

Figura 7: Ejemplo de rutas mas cortas

Ahora se calcula para el algoritmo, el promedio y desviación estándar de la ejecución del conjunto de réplicas y se grafica un histograma.



(a) Histograma

(b) Diagrama de caja y bigotes

Figura 8: Histograma y diagrama de caja y bigotes

En la figura [8.a] se muestra el histograma de los tiempos promedio del algoritmo y se calcula el promedio (μ) y la desviación estándar (σ) para graficar la línea punteada que corresponde a la distribución normal $N \sim (\mu = 5.70, \sigma = 0.03)$ y se puede ver que la línea punteada no se ajusta al histograma por lo que se realiza la prueba estadística *Shapiro-Wilk* para checar la normalidad de los datos.

In: `stats.shapiro` (tiempos_algoritmo_4) Out: $(W = 0.7345, p = 5.1768 \cdot 10^{-6})$

La prueba muestra que el p-valor es menor a 0.05 (nivel de significancia), entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal).

Por otro lado en la figura [8.b] se muestran los diagramas de caja y bigotes para los tiempos individuales de los cinco grafos utilizados en el algoritmo, el número rojo representa la cantidad de nodos del grafo y se puede apreciar que conforme aumenta la cantidad de nodos en el grafo, los tiempos computacionales también crecen.

Coloración glotona

La figura [9] muestra un ejemplo de un grafo y sus respectiva coloración propuesta por el algoritmo ejemplo.

```
1 greedy1 = nx.coloring.greedy_color(G21, strategy='random_sequential')
```

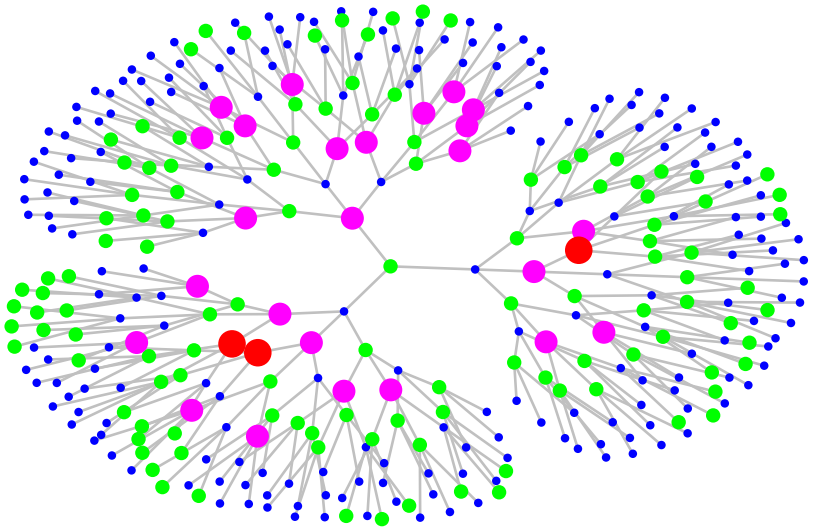
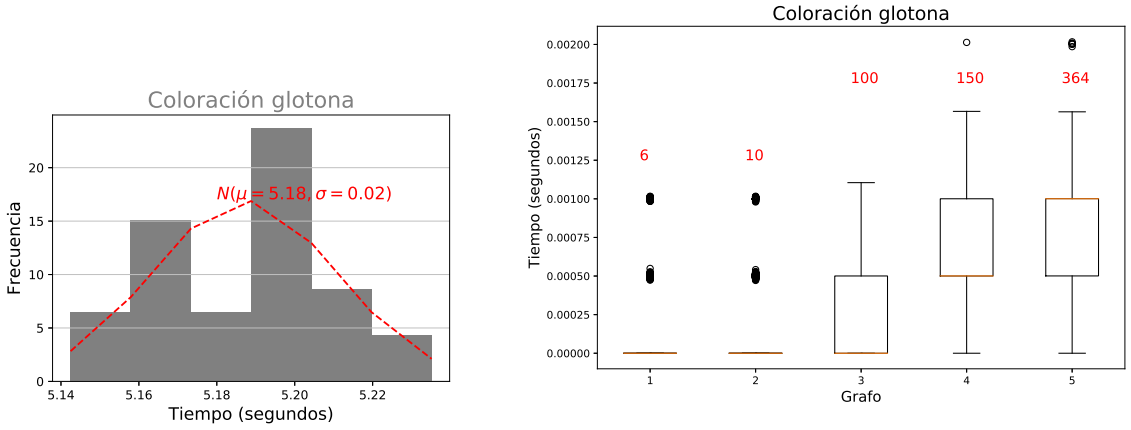


Figura 9: Ejemplo de coloración random

Ahora se calcula para el algoritmo, el promedio y desviación estándar de la ejecución del conjunto de réplicas y se grafica un histograma.



(a) Histograma

(b) Diagrama de caja y bigotes

Figura 10: Histograma y diagrama de caja y bigotes

En la figura [10.a] se muestra el histograma de los tiempos promedio del algoritmo y se calcula el promedio (μ) y la desviación estándar (σ) para graficar la línea punteada que corresponde a la distribución normal $N \sim (\mu = 5.18, \sigma = 0.02)$ y se puede ver que la línea punteada no se ajusta al histograma por lo que se realiza la prueba estadística *Shapiro-Wilk* para checar la normalidad de los datos.

In: `stats.shapiro` (tiempos_algoritmo_5)	Out: $(W = 0.5863, p = 4.9988 \cdot 10^{-8})$

La prueba muestra que el p-valor es menor a 0.05 (nivel de significancia), entonces la hipótesis nula es rechazada (se concluye que los datos no vienen de una distribución normal).

Por otro lado en la figura [10.b] se muestran los diagramas de caja y bigotes para los tiempos individuales de los cinco grafos utilizados en el algoritmo, el número rojo representa la cantidad de nodos del grafo y se puede apreciar que conforme aumenta la cantidad de nodos en el grafo, los tiempos computacionales también crecen.

Además se incluyen dos gráficas de dispersión, en ambas el eje horizontal corresponde al tiempo promedio de ejecución de un algoritmo, contando con barras horizontales que representan la desviación estándar alrededor del punto que indica el promedio, mientras el eje vertical es el número de vértices del grafo en la primera gráfica, ver figura [11] y el número de aristas en la segunda, ver figura [12]; cada combinación de algoritmo-grafo se visualiza con un punto en la gráfica de dispersión (se distinguen entre algoritmos por colores y entre grafos por formas).

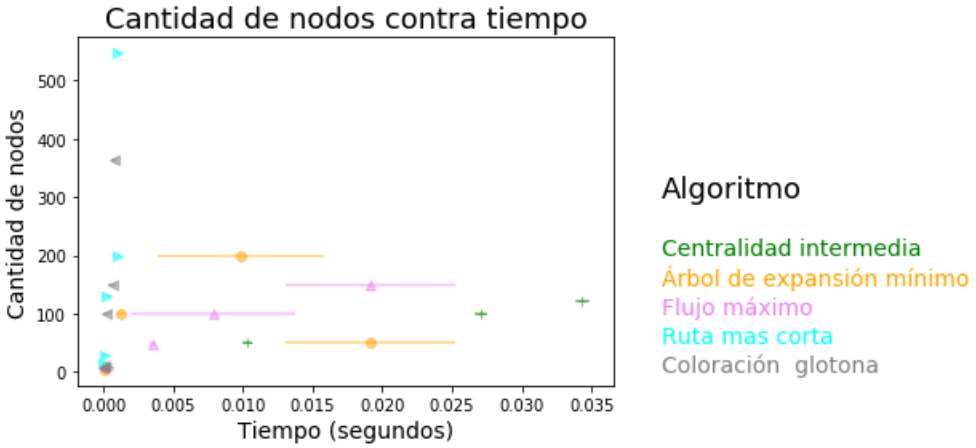


Figura 11: Cantidad de nodos contra tiempos

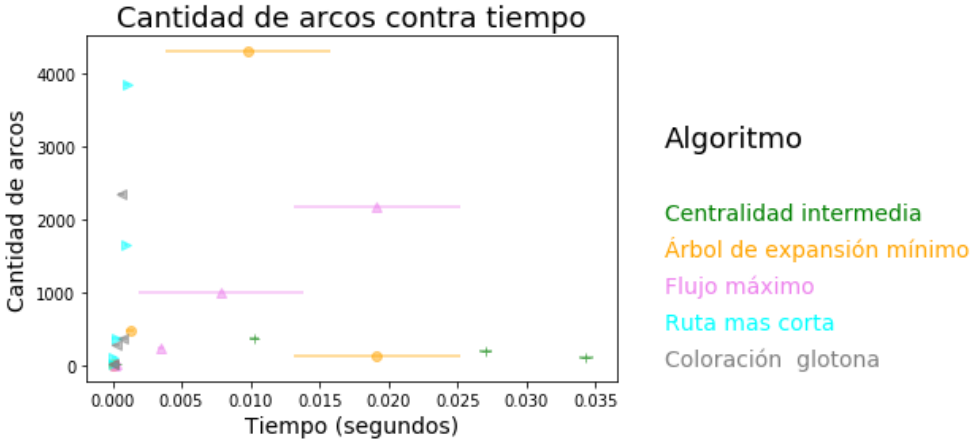


Figura 12: Cantidad de arcos contra tiempos

Referencias

[1] Python. https://www.python.org/.

[2] E. Schaeffer. https://elisa.dyndns-web.com/teaching/opt/flow/.

[3] A. Serna. https://github.com/sernarmando.