Medición experimental de la complejidad asintótica con Python y Gnuplot

Astrid González

Optimización de flujo en redes

9 de abril de 2018

Introducción

Un algoritmo corresponde al procedimiento matemático utilizado para la computación de alguna función, el cuál es llevado a cabo de forma mecánica [1].

En casi la totalidad de las ramas de ciencia tradicional y contemporánea es necesario poder realizar la medición de la complejidad de un problema, algoritmo o estructura.

Tenemos entonces que la complejidad computacional se mide en base a la cantidad de recursos computacionales utilizados para dicha tarea, la limitación de los recursos disponibles genera diferentes clases computacionales [1].

En el caso de los algoritmos de flujo [3], el algoritmo Ford-Fulkerson se encuentra basado en caminos aumentantes, por lo que su complejidad computacional dependerá del número de conexiones presentes en el grafo $G(E)$ as como de su respectivo flujo máximo, f. Entonces se tiene que su complejidad corresponde a $O(E * f)$

Por otro lado, existen variantes el algoritmo utilizado para encontrar el camino más corto [3], las cules pueden calcular la distancia entre un nodo inicial y el resto de los nodos, o calcular la distancia entre el nodo inicial y un nodo final, denotados nodo fuente y nodo sumidero respectivamente. En este trabajo es de interés el caso que analiza la distancia entre un nodo fuente y el nodo sumidero, cuya complejidad corresponde al cuadrado de la cantidad de nodos en el grafo $O(V^2)$.

Metodología

Para el diseño de los grafos, fue empleada la versión 3.6.4 del lenguaje de programación Python [5] acorde al procedimiento obtenido previamente para la generación de grafos simples, dirigidos y ponderados [2]. A dicho código se incorporaron algoritmos [4] para el cálculo del flujo máximo y del camino más corto.

Entonces con la finalidad de obtener mediciones certeras de los tiempos de desempeño de los algoritmos incorporados, para cada tipo de grafo a analizar y para cada tamaño k, se generaron n cantidad de grafos diferentes del mismo tamaño k. Por lo que se obtuvieron n repeticiones de la medición del tiempo de desempeño para cada tamaño de cada tipo de grafo.

Es decir que en cada repetición y para cada tamañ k se genera un solo grafo al cual se le incorporan las propiedades de dirección y pon-

deración. Obteniendo de esta manera grafos simples, dirigidos sin ponderación, ponderados no dirigidos, y ponderados dirigidos.

Posteriormente, para cada tipo de grafo se produjo un marco de información que incluye los datos de todos los tamaños analizados, y en donde para cada tamaño se describen las propiedades estadísticas de las mediciones obtenidas de ese tipo de grafo. El pseudocódigo 1 corresponde a dicho procedimiento.

Pseudocódigo 1: Procedimiento de medición de los tiempos y generación del script.

```
Función loopyLoop(grafo, archivo,
 repeticiones, tamaños):
    para iteracion en rango(repeticiones)
     hacer
        para tamaño en tamaños hacer
            para i en rango(tamaño) hacer
                generar datos de los nodos
            para k en tiposGrafo hacer
                x ← grafo tipo k
                tA ← tiempo de medición del
                 algoritmo camino corto
                tB ← tiempo de medición del
                 algoritmo flujo máximo
                tipoGrafo[tamaño].append(
                (tA, tB) )
    regresar archivo
Función getStats(archivo):
    alg1 ← estadísticos del algoritmo camino
     corto
    alg2 ← estadísticos del algoritmo flujo
     máximo
    regresar alg1 alg2
Función plotStats(tamaños, alg, archivo):
    regresar script
```

Es entonces que a partir de dichas mediciones se procedió a generar diagramas de caja y bigote para representar la relación entre el tiempo de desempeño y la cantidad de nodos o tamaño del grafo. Para la visualización de dicha relación se utilizó la versión 4.6 de Gnuplot [6].

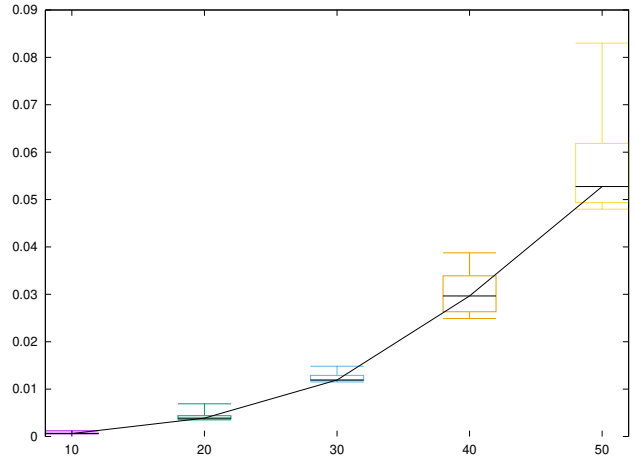
Resultados

En la figura 1 se observan los tiempos de desempeño del algoritmo de camino más corto para el grafo de tipo simple. Se puede apreciar un incremento en los tiempos con respecto a la cantidad de nodos de tipo exponencial, cuyo comportamiento es más acentuado en la figura 1(d).

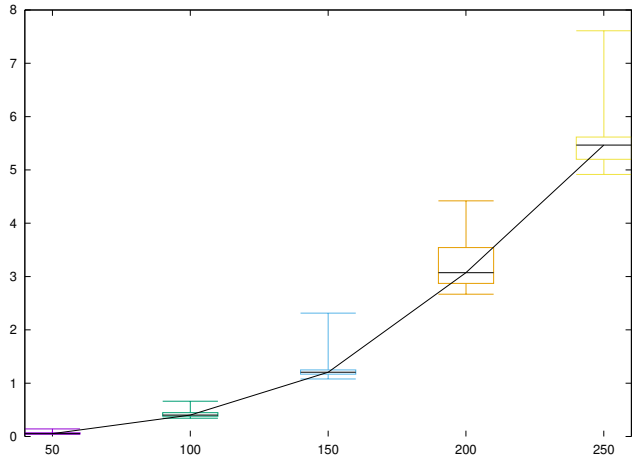
La línea negra corresponde al comportamiento de los valores medios de las mediciones del tiempo de ejecución tras 10 iteraciones en cada tamaño k. Así mismo, para cada tamaño se muestran cajas de diferente color donde el límite inferior y superior de la caja corresponden al primer y tercer cuartil respectivamente, mientras que la línea horizontal representa a la mediana y los bigotes de la caja correspondiente a los valores máximos y mínimos de los datos.

En la figura 2 se presentan los tiempos de desempeño del algoritmo de camino más corto para cada tipo de grafo, donde se puede observar que presentan un comportamiento similar entre sí.

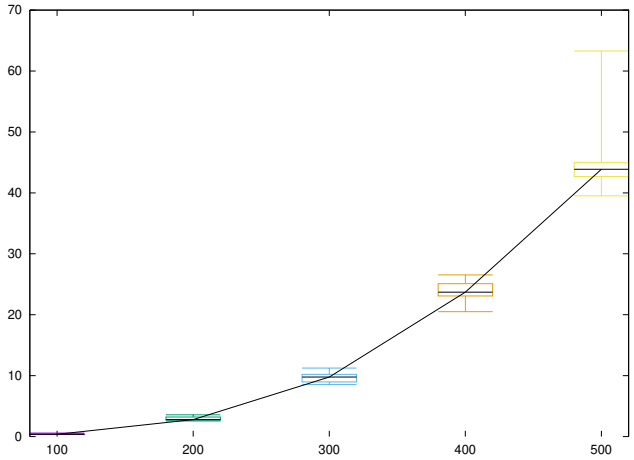
De la misma manera, la figura 3 corresponde a los tiempos de ejecución del algoritmo de flujo máximo para cada tipo de grafo, en donde se presenta un comportamiento linear.



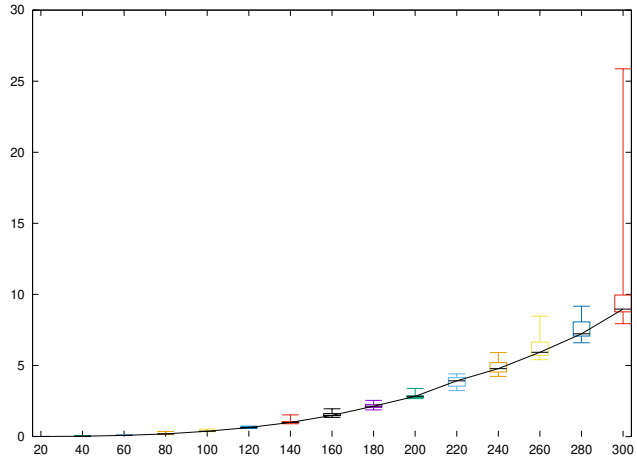
(a) Incrementos de 10 nodos.



(b) Incrementos de 50 nodos.

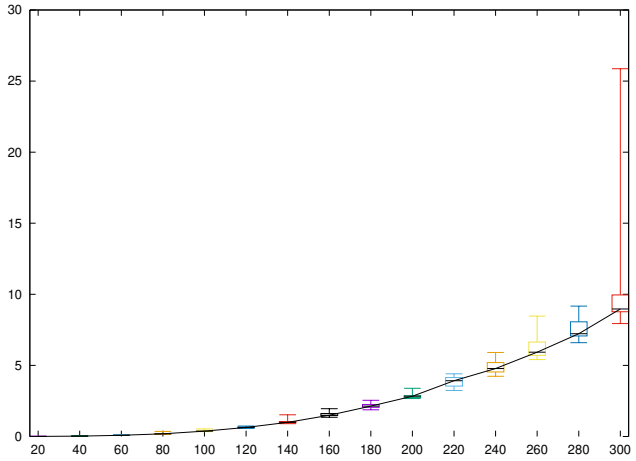


(c) Incrementos de 100 nodos.

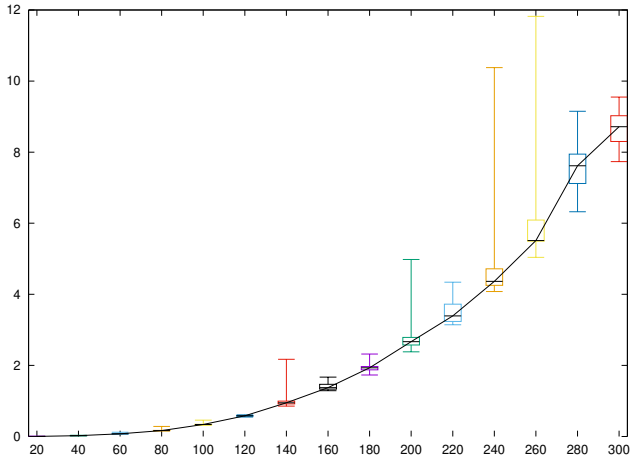


(d) Incrementos de 20 nodos.

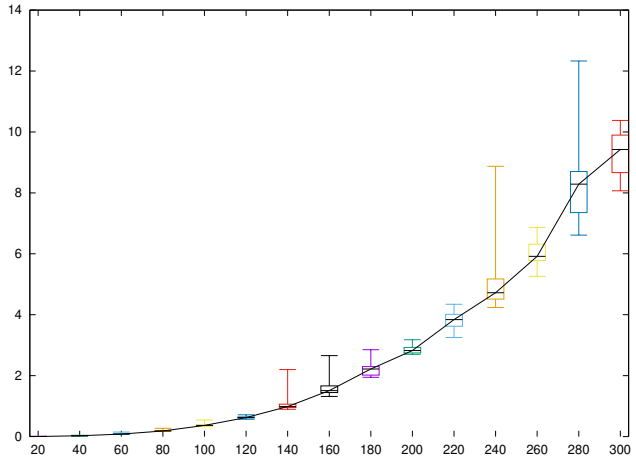
Figura 1: Visualización del incremento en los tiempos de desempeño del algoritmo de camino más corto para el grafo de tipo simple. Se contrasta el tiempo de ejecución en segundos en el eje vertical contra el eje horizontal que representa la cantidad de nodos presentes en el grafo.



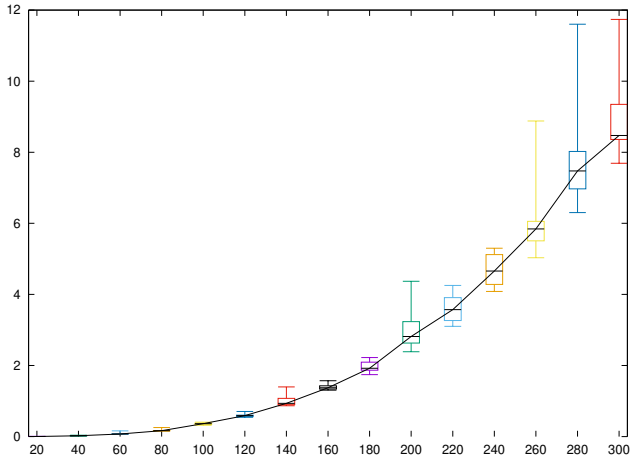
(a) Grafo simple.



(b) Grado dirigido no ponderado.

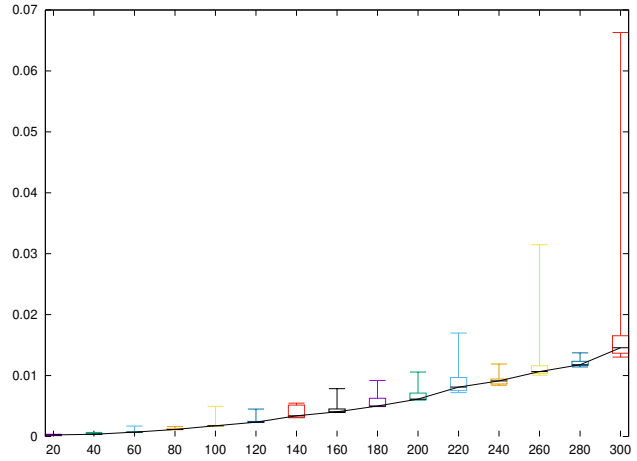


(c) Grafo ponderado no dirigido.

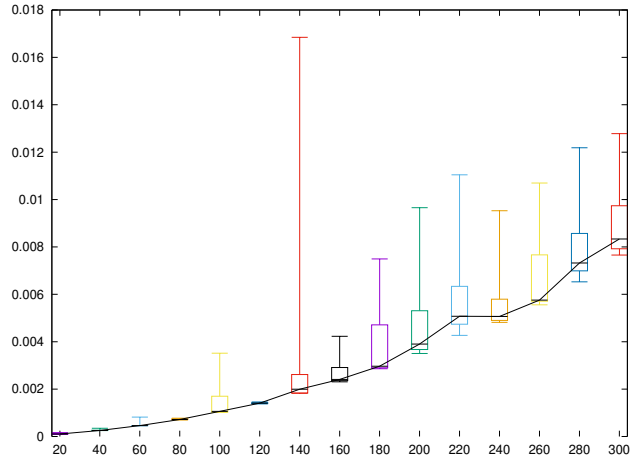


(d) Grafo dirigido ponderado.

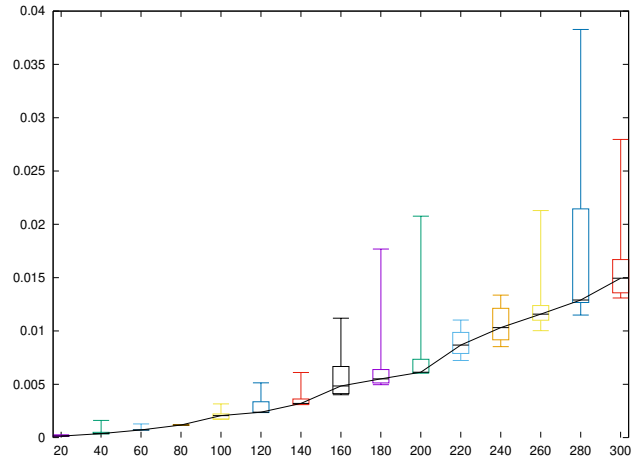
Figura 2: Visualización del incremento en los tiempos de desempeño del algoritmo de camino más corto para cada tipo de grafo.



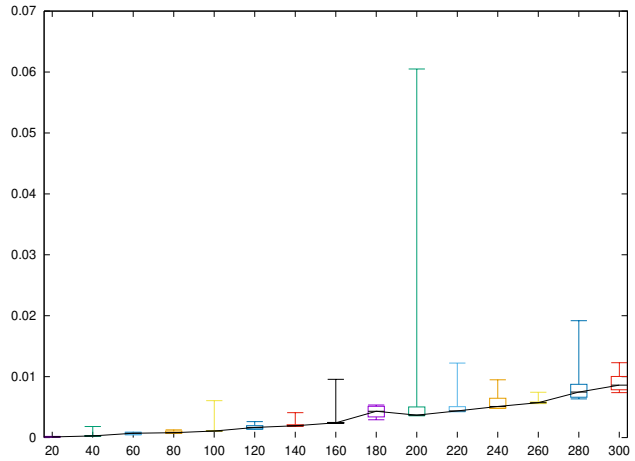
(a) Grafo simple.



(b) Grado dirigido no ponderado.



(c) Grafo ponderado no dirigido.



(d) Grafo dirigido ponderado.

Figura 3: Visualización del incremento en los tiempos de desempeño del algoritmo de flujo máximo para cada tipo de grafo.

Discusión

A pesar que en todos los tipos de grafo con el algoritmo de camino más corto se aprecia un crecimiento de tipo exponencial, en el grafo simple se puede observar un comportamiento exponencial débil en comparación con los demás grafos.

Sin embargo cuando se agregan las propiedades de dirección o ponderación al grafo, este crecimiento es más marcado, sin mostrar diferencia significativa entre si presenta una, otra, o ambas características.

El aumento en el tiempo pudiera deberse al surgimiento de elecciones debido a la jerarquía de la ponderación, así como del aumento en la complejidad al limitar el camino.

Cabe mencionar que en el caso de grafos dirigidos, en el código se retiraron las conexiones de los nodos en sentido opuesto a la dirección deseada.

Por otro lado, en el caso de los tiempos de ejecución del algoritmo de flujo máximo, se observó que se presenta un comportamiento lineal. La pendiente de dicho comportamiento es similar y

más pronunciada en grafos con una sola característica, mientras que en los grafos que presentan ambas o ninguna característica se observó que la pendiente es menor.

Dicha variación entre los tipos de grafo de la pendiente de los tiempos de ejecución, pudiera deberse de la misma manera al surgimiento de elecciones por la ponderación así como la limitación del camino.

Sin embargo, los tiempos obtenidos para el grafo dirigido ponderado en la figura 3(d) no concuerdan con los comportamientos observados en donde se añade solo una característica representados en la figura 3(b) y la figura 3(c) ya que se esperaría que los tiempos se comportaran de manera similar a los grafos antes mencionados. Esto pudiera ser debido a los tiempos máximos obtenidos de la medición de 200 nodos, en la cual se presenta un incremento significativo del valor de la medición máxima lo que pudiera influenciar la visualización del comportamiento debido a la escala del eje horizontal que se necesita utilizar para representar a dicho valor.

Referencias

[1] Gács, P. and László, L. (1999). Complexity of algorithms. `http://web.cs.elte.hu/~lovasz/complexity.pdf{}`. Consultado en Abril 2018.

[2] González, A. (2018). Visualización de grafos simples, ponderados y dirigidos con gnuplot. *Universidad Autónoma de Nuevo León*, Optimización de flujo en redes.

[3] Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation. ISBN: 0-486-40258-4.

[4] Schaeffer, S. E. (2018). Curso en línea: Matemáticas discretas. `https://elisa.dyndns-web.com/teaching/mat/discretas/md.html`. Consultado en Abril 2018.

[5] Van Rossum, G. and the Python Development Team (2018). The python language reference. release 3.6.4. `https://docs.python.org/3.6/`. Consultado en Febrero 2018.

[6] Williams, T. and Kelley, C. (2013). Gnuplot 4.6: an interactive plotting program. `http://gnuplot.info/`. Consultado en Febrero 2018.