Complejidad asintótica experimental

Matrícula: 1985281

Resumen

Este trabajo busca resolver grafos con la implementación de los algoritmos de NetworkX de `Python`. Primero se seleccionan tres métodos de generación de grafos, luego con cada generador, se generan grafos de cuatro diferentes órdenes en escala logarítmica (16, 32, 64 y 128 en base 2), después se generan diez grafos distintos de cada orden.

Introducción

La librería NetworkX de `Python` proporciona algoritmos y generadores de grafos, si los grafos generados son ponderados o multigrafos, se convierten a grafos simples. Además se asignan pesos no-negativos normalmente distribuidos (con media $\mu = 1$ y desviación $\sigma = 0$) a las aristas para que se puedan utilizar como instancias del problema de flujo máximo.

Luego se eligen tres implementaciones de NetworkX de los algoritmos de flujo máximo, se ejecutan los algoritmos seleccionados con cinco diferentes pares de fuente-sumidero por lo menos cinco veces cada par $s - t$, cada grafo de cada tamaño con cada algoritmo.

Generadores:

- `dense_gnm_random_graph` (Generador 1): Devuelve un $G_{n,m}$ grafo aleatorio, es decir, se elige de manera uniformemente al azar del conjunto de todos los grafos con n nodos y m arcos.
- `gnp_random_graph` (Generador 2): Devuelve un $G_{n,p}$ grafo aleatorio, también conocido como un grafo de *Erdös-Rényi* o un grafo binomial, es decir, se elige cada uno de los arcos posibles con probabilidad p.
- `gnm_random_graph` (Generador 3): Devuelve un $G_{n,m}$ grafo aleatorio.

Algoritmos:

- `maximum_flow` (Algoritmo 1): Encuentra el flujo máximo de un nodo fuente a un nodo sumidero.
- `algorithms.flow.edmonds_karp` (Algoritmo 2): Encuentra el flujo máximo de un nodo fuente a un nodo sumidero utilizando el algoritmo *Edmonds-Karp*. Este algoritmo tiene un tiempo de ejecución de $O(nm^2)$ para n nodos y m arcos.
- `algorithms.flow.boycov_kolmogorov` (Algoritmo 3): Encuentra el flujo máximo de un nodo fuente a un nodo sumidero utilizando el algoritmo *Boykov-Kolmogorov*. Este algoritmo tiene complejidad $O(n^2m|C|)$ para n nodos m bordes, y $|C|$ es el coste del corte mínimo.

Metodología y Resultados

Efecto que el generador de grafo usado tiene en el tiempo de ejecución

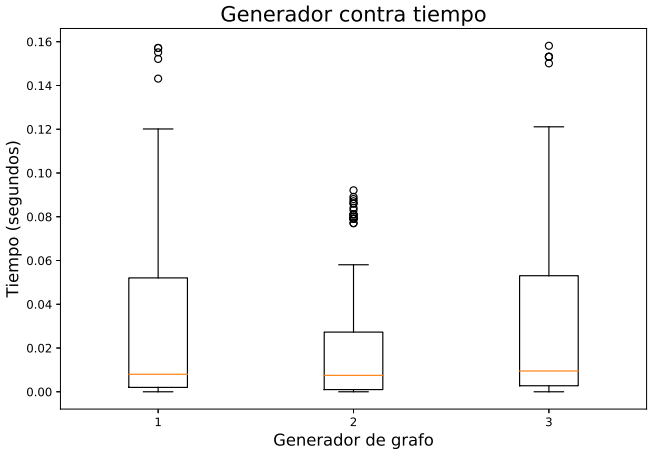


Figura 1: Tipo de generador de grafo contra tiempo

En la figura 1 se puede apreciar que el generador más rápido de los tres seleccionados es el Generador 2, mientras que los otros dos parecen ser igual de tardados.

Efecto que el algoritmo usado tiene en el tiempo de ejecución

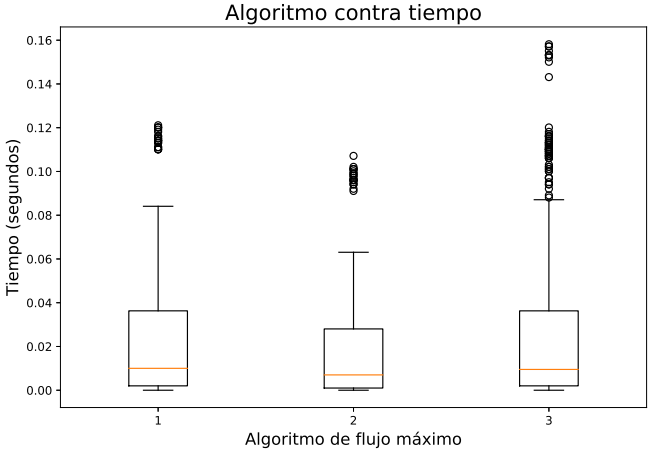


Figura 2: Algoritmo de flujo máximo contra tiempo

En la figura 2 se puede apreciar que el algoritmo más rápido de los tres seleccionados es el Algoritmo 2, mientras que el Algoritmo 3 parece ser el más tardado de los tres.

Efecto que el orden del grafo tiene en el tiempo de ejecución

Figura 3: Orden del grafo contra tiempo

En la figura 3 se puede apreciar que conforme el orden del grafo va creciendo $(2^4, 2^5, 2^6, 2^7)$ respectivamente, el tiempo de ejecución también va creciendo.

Efecto que la densidad del grafo (como tasa de aristas presentes entre aristas posibles) tiene en el tiempo de ejecución

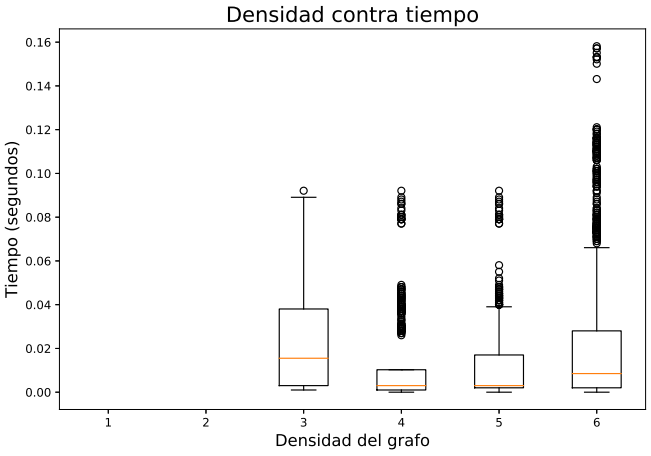


Figura 4: Densidad del grafo contra tiempo

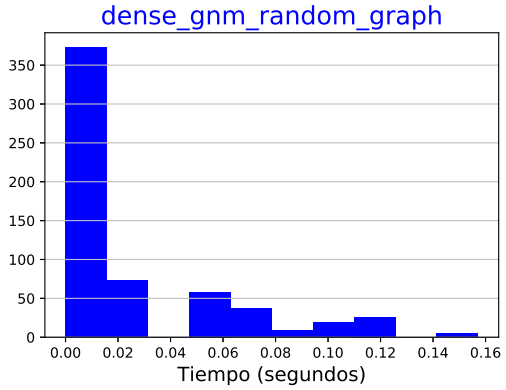
En la figura 4 se puede apreciar que conforme la densidad del grafo va creciendo ([0, 0.5], (0.5, 0.6], (0.6, 0.7], (0.7, 0.8], (0.8, 0.9], (0.9, 1]) respectivamente, el tiempo de ejecución va disminuyendo.

Se puede observar que los diagramas de caja y bigotes de las figuras 1, 2, 3 y 4, no muestran tan profundamente si sí o no estos efectos tienen interacciones, por lo que se sigue a realizar un análisis de varianza para cada una de las variables.

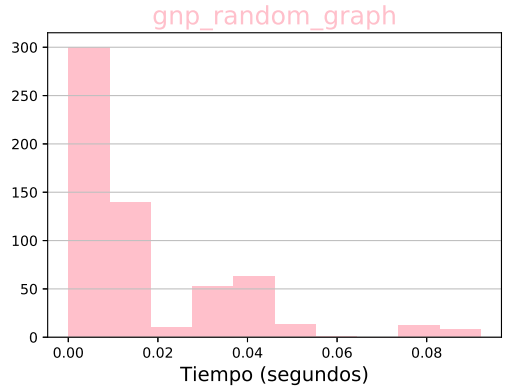
ANOVA.txt

	sum_sq	df	F	PR>F
Algoritmo	0.001083	1.0	4.666000	3.089750e-02
Generador	0.002269	1.0	9.775689	1.796692e-03
Orden	1.218398	1.0	5249.074786	0.000000e+00
Densidad	0.046203	1.0	199.052050	6.059348e-43
Residual	0.416650	1795.0	NaN	NaN
c@FancyVerbLinee650	1795.0	NaN	NaN	NaN

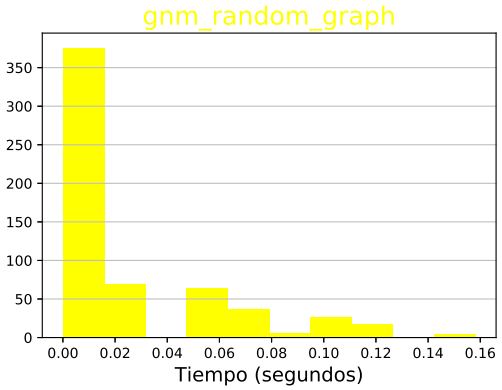
Tenemos que el p-valor obtenido por la prueba ANOVA es muy pequeño, es decir, que las medias de las variables con respecto al tiempo son distintas, entonces se puede decir que las variables están relacionadas con los tiempos de ejecución. Una vez hecha la ANOVA se debe verificar que todos los datos deben distribuirse normalmente para que la estadística F sea confiable.



(a) Generador 1



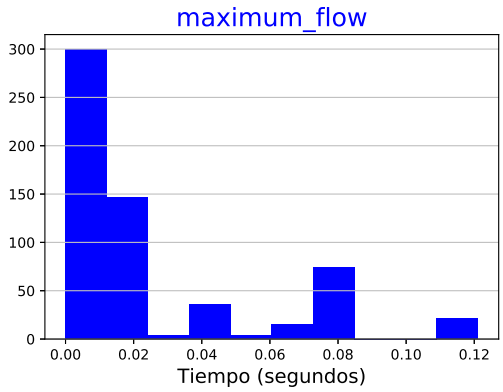
(b) Generador 2



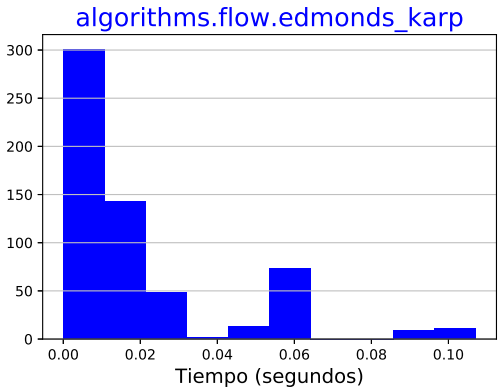
(c) Generador 3

Figura 5: Histogramas de los tiempos por tipo de generador de grafo

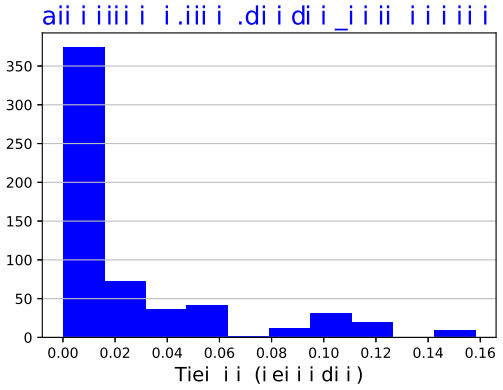
De la figura 5 es fácil ver que los datos de los tiempos de los generados de grafos no son distribuidos normalmente.



(a) Algoritmo 1



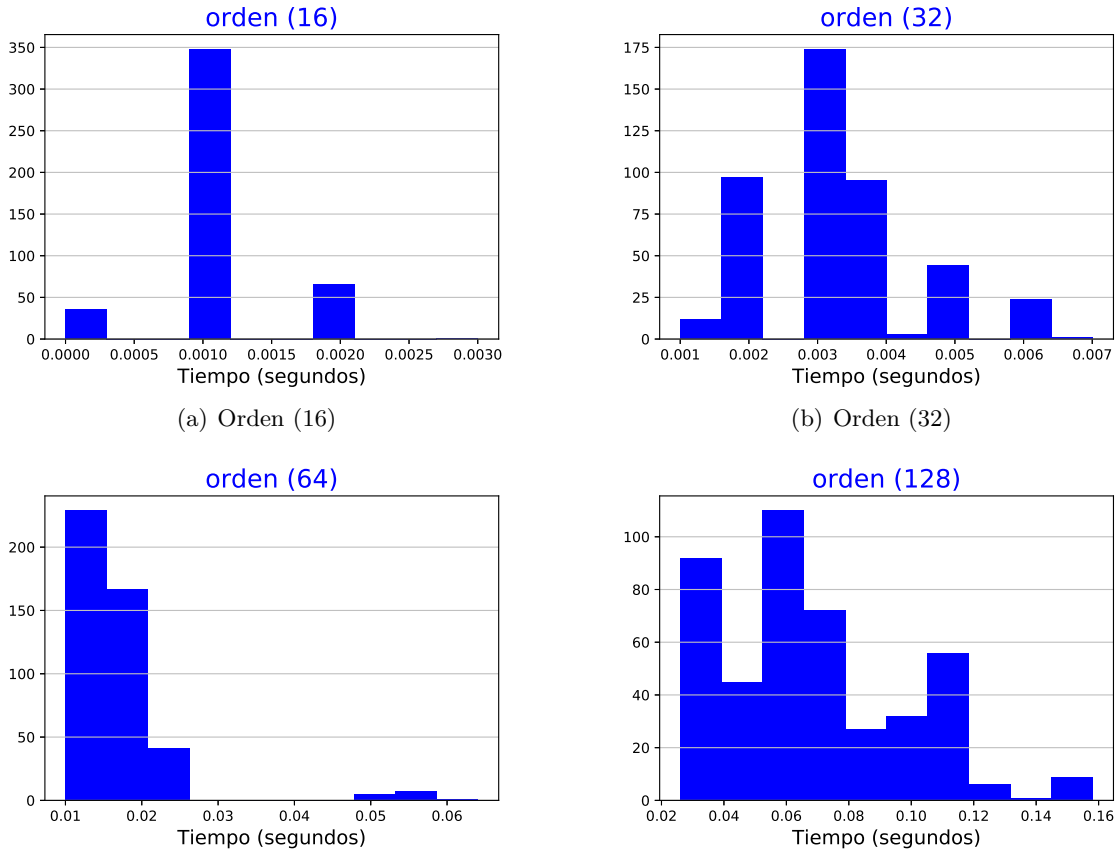
(b) Algoritmo 2



(c) Algoritmo 3

Figura 6: Histogramas de los tiempos por tipo de algoritmo de flujo máximo

De la figura 6 es fácil ver que los datos de los tiempos de los algoritmos de flujo máximo no son distribuidos normalmente.



(a) Orden (16)

(b) Orden (32)

(c) Orden (64)

(d) Orden (128)

Figura 7: Histogramas de los tiempos por tamaño de orden del grafo

De la figura 7 es fácil ver que los datos de los tiempos por tamaño del orden de los grafos no son distribuidos normalmente.

De las figuras 5, 6, y 7, podemos ver que los tiempos de ejecución no distribuyen normalmente para las variables por lo que se propone para estudiar más a fondo la relación entre las variables una prueba de Mínimos Cuadrados (OLS).

OLS.txt

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Tiempo   R-squared:                       0.758
Model:                            OLS   Adj. R-squared:                  0.758
Method:                 Least Squares   F-statistic:                     1408.
Date:                Mon, 01 Apr 2019   Prob F-statistic:               0.00
Time:                        22:50:42   Log-Likelihood:                 4979.9
No. Observations:                1800   AIC:                            -9950.
Df Residuals:                    1795   BIC:                            -9922.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0421      0.002    -18.436      0.000      -0.047      -0.038
Algoritmo      0.0010      0.000      2.160      0.031    8.74e-05       0.002
Generador     -0.0014      0.000     -3.127      0.002      -0.002      -0.001
Orden          0.0006   8.39e-06     72.450      0.000       0.001       0.001
Densidad       0.0329      0.002     14.109      0.000       0.028       0.037
==============================================================================
Omnibus:                      705.514   Durbin-Watson:                   1.147
ProbOmnibus:                    0.000   Jarque-Bera JB:               3663.472
Skew:                           1.785   ProbJB:                          0.00
Kurtosis:                       9.009   Cond. No.                         635.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
c@FancyVerbLineeors assume that the covariance matrix of the errors is correctly specified.
```

La prueba de Mínimos Cuadros muestra con la $r - cuadrada$ que se puede eliminar hasta el 75 % de errores para determinar el tiempo, por otro lado, el $p - valor$ muestra que la medias de las variables son distintas respecto al tiempo y por último podemos ver que la variable Orden está más relacionada con los tiempos de ejecución, luego le sigue el Densidad del grafo y déspues el Algoritmo de flujo máximo. Para rectificar la prueba de Mínimos Cuadrados se hace una Matriz de Correlaciones. Ver figura 8.

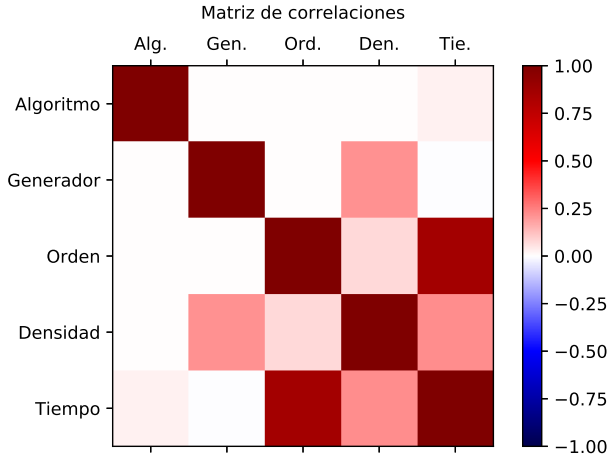


Figura 8: Matriz de correlaciones

Por último se representan en la figura 9 todos los grafos y se identifican según su generador por color y su algorimo de solución por forma.

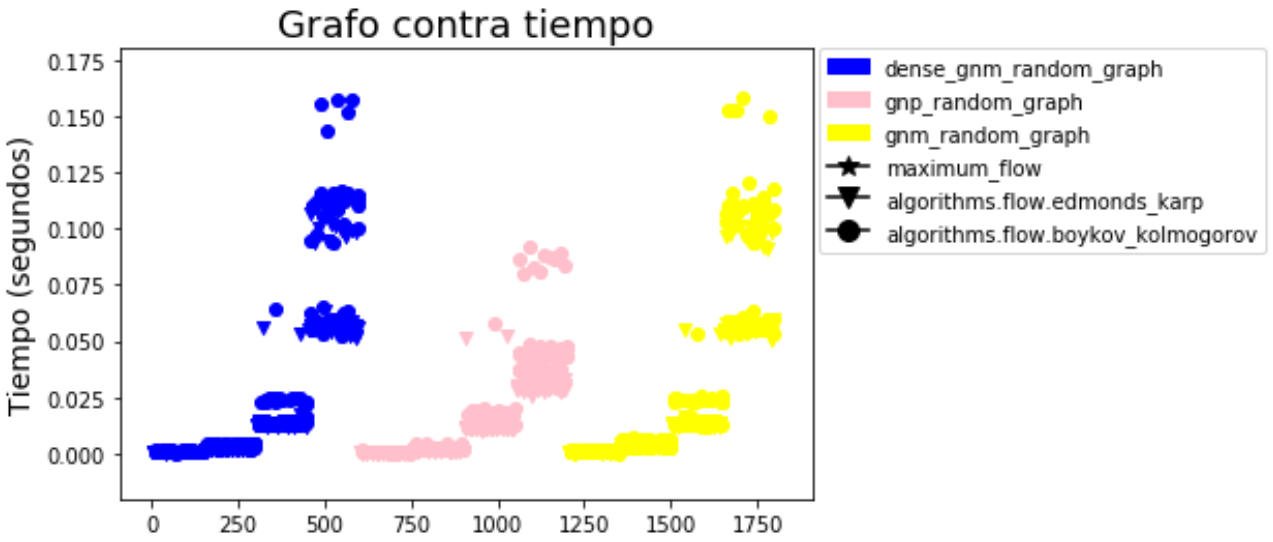


Figura 9: Grafos contra tiempo

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.