Tarea 5

5271

30 de abril de 2019

1. Algoritmo generador de grafos

Gracias a la versatilidad de Los grafos como modelo de representación de datos, los procesos aleatorios de generación de grafos también son relevantes en aplicaciones que van desde la física, biología a la sociología [8].

Otra de las aplicaciones de los grafos de generación aleatoria es en la representación de una red telefónica donde los vértices son centrales telefónicas, las aristas son los enlaces troncales los cuales presentan cierta capacidad de llamadas. De esta red se desea conocer el flujo máximo entre una central telefónicas fuente y una central telefónicas sumidero.

En realización de la tarea anterior se seleccionaron tres algoritmos generadores de grafos de la biblioteca de *networkx*, con el objetivo de crear los grafos con pesos con distribución normal a los que se le aplicaron los tres algoritmos de flujo máximo para realizar los experimentos. Los tres algoritmos generadores antes mencionados son los siguientes.

Los algoritmos son los siguientes:

- *Erdős Rényi graph* (en el modelo *G (n, p)*, el grafo se construye conectando los n vértices al azar. Cada arista se incluye en el grafo con probabilidad p independiente de cualquier otro borde) [3].
- *Fast gnp random graph* (este algoritmo recibe como parámetros n números de vértices y probabilidad p de ocurrencia de aristas, el mismo devuelve un grafo aleatorio) [4].
- *Binomial graph* (este algoritmo recibe como parámetros n números de vértices y probabilidad p de ocurrencia de aristas, el mismo devuelve un grafo aleatorio, para grafos dispersos (para valores pequeños de p), *Fast gnp random graph* es un algoritmo más rápido) [5].

De los algoritmos anteriores el seleccionado para la realización de esta tarea es el *Erdős Rényi graph* dado que la tarea anterior arrojo que ninguno de los tres influían en el tiempo de ejecución de los algoritmo de flujo máximo que se le aplicaron a los grafos generados. como se muestra en el cuadro 1, es decir que los tres generan grafos con características similares.

Cuadro 1: ANOVA, relación del algoritmo generador con el *tiempo de ejecución*

Factor	**SS**	**DF**	**MS**	**F**	**p-unc**	**np2**
generador_grafo	0.280	2	0.140	0.354	0.702	0
Within	712.085	1797	0.396	-	-	-

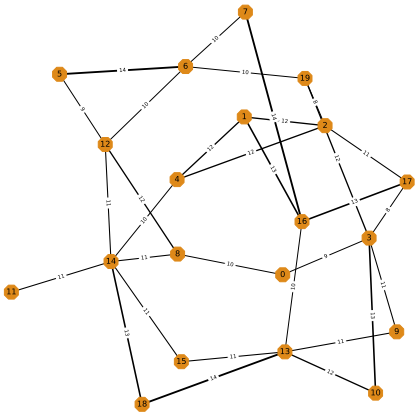
En el cuadro 1 se muestra que no existen diferencia entre las medianas de los grupos de factores ya que el $p - unc$ es mayor que 0,05 por lo que se acepta la hipótesis de que el tipo de generador no influye en el *tiempo de ejecución*.

Para la generación de los grafos con el algoritmo *Erdős Rényi graph* hizo el siguiente código de Python donde se generan cinco grafos con la misma cantidad de vértices pero con diferentes números de aristas y capacidades de las mismas como se muestra en la figura 1 de la página 3.

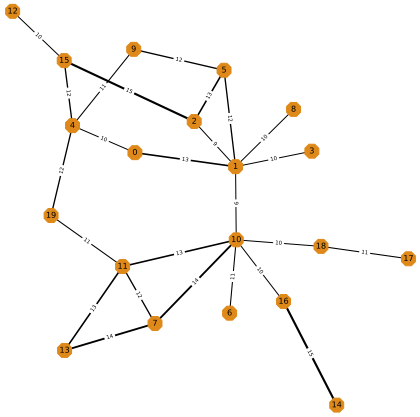
```
def leer_capacity(G):
    capacity=[]
    for (u, v) in G.edges():
      capacity.append(G.edges[u, v]["capacity"])
    return capacity

Grafo = nx.erdos_renyi_graph(20, 0.19, seed=None)
aristas = Grafo.number_of_edges()
pesos_normalmente_distribuidos = np.random.normal(12, 1.5, aristas)
loop = 0
for (u, v) in Grafo.edges():
    Grafo.edges[u, v]["capacity"] = pesos_normalmente_distribuidos[
    loop]
    loop += 1
df = pd.DataFrame()
df = nx.to_pandas_adjacency(Grafo, dtype=int, weight='capacity')
df.to_csv("Grafo5" + ".csv", index=None, header=None)
```

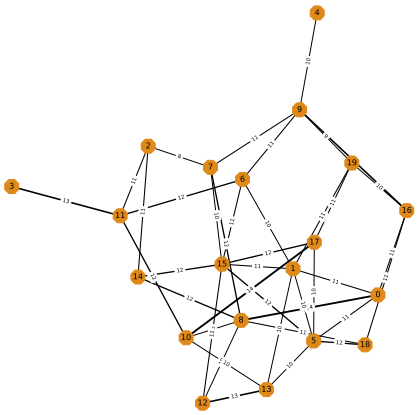
Generar_g.py



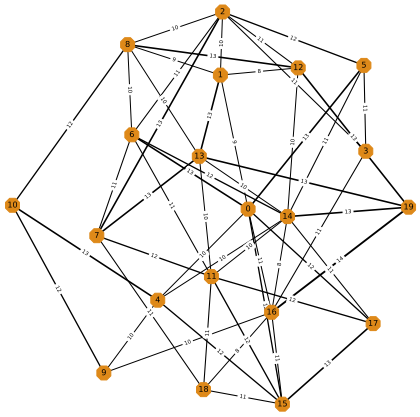
(a) *Grafo1*, con 20 vértices



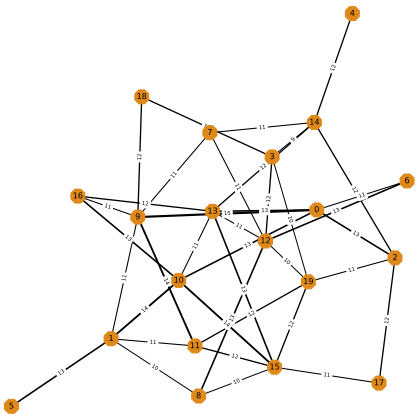
(b) *Grafo2*, con 20 vértices



(c) *Grafo3*, con 20 vértices



(d) *Grafo4*, con 20 vértices



(e) *Grafo5*, con 20 vértices

Figura 1: Grafos generados con el algoritmo seleccionado, donde el grosor de las aristas representan la capacidad de las mismas.

2. Algoritmo de flujo máximo.

En los problemas de flujo en redes, las aristas representan vías por las que puede circular elementos: datos, agua, corriente eléctrica, llamadas telefónicas entre otras. Los pesos de las aristas representan la capacidad máxima de una vía: velocidad de una conexión, volumen máximo de agua, voltaje de una línea eléctrica,cantidad máxima llamadas entre otras; aunque es posible que la cantidad real de flujo sea menor.

El problema del flujo máximo consiste en lo siguiente: dado un grafo con pesos, $G = (V, A, W)$, que representa las capacidades máximas de los canales, un vértice fuente f y otro sumidero s en V, encontrar la cantidad máxima de flujo que puede circular desde f hasta s.

En la biblioteca de *networkx* encontramos varios algoritmos con los que podemos atacar los problemas de flujo máximo, para la realización de esta tarea se escogerá uno de los tres algoritmos utilizados en la tarea anterior pertenecientes a dicha librería. Basándonos en los resultados obtenidos en las pruebas realizadas en la tarea anterior.

Los algoritmos escogidos en la tarea anterior son los siguientes:

- *Shortest augmenting path* es uno de los enfoques más clásicos para la máxima coincidencia y los problemas de flujo máximo. Sorprendentemente, aunque esta idea es una de las técnicas más básicas, está lejos de ser completamente entendida. Es más fácil hablar de ello introduciendo el problema de emparejamiento bipartito en línea [1]. Este algoritmo encuentra el flujo máximo de un solo producto utilizando la ruta de aumento más corto y devuelve la red residual resultante después de calcular el flujo máximo.
- *Maximum flow* encuentra la ruta por la cual pasa la máxima cantidad de flujo, recibe como parámetros un grafo G, una fuente f, un sumidero s y además una capacidad que de no tenerla, se considera que el borde tiene una capacidad infinita. Se puede aplicar en grafos tanto dirigidos como no dirigidos.) [6].
- *Preflow push* encuentra un flujo máximo de un solo producto utilizando el algoritmo de empuje previo al flujo de la etiqueta más alta. Esta función devuelve la red residual resultante después de calcular el flujo máximo. Este algoritmo tiene un tiempo de ejecución de $O(n^2\sqrt{m})$ para n vértices y m aristas. [7].

De los algoritmos anteriores el que se utilisa en esta tarea es el *Shortest augmenting path* por ser el algoritmo que mejores tiempos registró en la tarea anterior.

3. Generación de datos

Con el objetivo de realizar las mediciones de los tiempos de ejecución de los algoritmo de flujo máximo seleccionado asi como el calculo de las seis propiedades de los nodos (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad,*pagerank*) se desarrolló el siguiente código.

En primer lugar, se crea una función (`Maxflow()`) que recibe como parámetros el grafo al que se le aplicara el algoritmo (`G`), la fuente (`a`) y sumidero (`b`). Al llamar esta función se genera con cada ejecución los datos que son guardados en un *data frame* del cual se muestra un fragmento en el cuadro 2 de la página 5.

Cuadro 2: Fragmento del *data frame* que contiene los datos recopilados.

Grafo	Fuente	Sumidero	Media	Mediana	FlujoMax	Grado	CoefAg	CentCer	CentCag	Excent	PageRag
Grafo1	0	1	0.026	0.020	19	2	0.000	0.388	0.048	4	0.035
Grafo1	15	7	0.022	0.020	22	2	0.000	0.413	0.035	4	0.034
Grafo2	0	9	0.029	0.020	23	2	0.000	0.396	0.044	4	0.039
Grafo2	2	4	0.028	0.024	34	3	0.333	0.404	0.123	4	0.056
Grafo3	1	2	0.044	0.052	30	6	0.267	0.543	0.084	3	0.068
Grafo3	1	15	0.042	0.024	63	6	0.267	0.543	0.084	3	0.068
Grafo4	0	13	0.079	0.084	69	7	0.143	0.613	0.096	2	0.064
Grafo4	3	10	0.055	0.032	33	3	0.333	0.475	0.016	3	0.032
Grafo5	3	15	0.020	0.020	43	4	0.167	0.463	0.059	4	0.052
Grafo5	8	2	0.076	0.074	33	3	0.000	0.452	0.034	4	0.039

En este fragmento del código se realiza toda la recopilación de la información necesaria para el análisis de los factores que in fluyen en el tiempo de ejecución y el flujo máximo, también dibuja los grafos a los que se les aplico el algoritmo.

```
def Leer_grafos(nomb):
    dr = pd.read_csv((nomb+".csv"), header=None)
    A = nx.from_pandas_adjacency(dr)
    return A

def leer_capacity(G):
    capacity=[]
    for (u, v) in G.edges():
      capacity.append(G.edges[u, v]["weight"])
    return capacity


def Maxflow(G, a, b):
    tiempo_inicial = dt.datetime.now()
    # d =shortest_augmenting_path(G, a, b, capacity="weight")
    for i in range(40):
        d =shortest_augmenting_path(G, a, b, capacity="weight")
        print( dt.datetime.now())
    tiempo_final = (dt.datetime.now() - tiempo_inicial).
    total_seconds()
    return tiempo_final
```

```
21
22
23 def listnodos(d,f,s):
24     for nodes in d.nodes():
25         if d.nodes[nodes]!=f and d.nodes[nodes]!=s :
26             nodos.append(nodes)
27     return nodos
28 def Tiempos(G,nombre):
29     df = {"Grafo": [], "Fuente": [], "Sumidero": [], "Media": [], "
       Mediana": [], "DesvStd": [], "flujoMax":
30         [], "Grado": [], "CoefAg": [], "CentCer": [],"CentCag":[],
       "Excentricidad": [], "PageRag": []}
31     Nodes = G.nodes
32     print(Nodes)
33     for i in Nodes:
34         for j in Nodes:
35             if i != j:
36                 t = []
37                 for k in range(10):
38                     print(t, end="\n\n")
39                     time = Maxflow(G, i, j)
40                     t.append(time)
41                 PageRag = nx.pagerank(G, weight="capacity")
42                 df["Grafo"].append(nombre)
43                 df["Grado"].append(G.degree(i))
44                 df["CoefAg"].append(round(nx.clustering(G, i), 4))
45                 df["CentCer"].append(round(nx.closeness_centrality
46                                            (G, i), 4))
47                 df["CentCag"].append(round(nx.load_centrality
48                                            (G, i), 4))
49                 df["Excentricidad"].append(nx.eccentricity(G, i))
50                 df["PageRag"].append(round(PageRag[i], 4))
51                 df["Fuente"].append(i)
52                 df["Sumidero"].append(j)
53                 df["Media"].append(round(np.mean(t), 4))
54                 df["Mediana"].append(round(np.median(t), 4))
55                 df["DesvStd"].append(round(np.std(t), 4))
56                 df["flujoMax"].append(nx.maximum_flow_value
57                                       (G, i, j, capacity="weight"))
58
59     dd = pd.DataFrame(df)
60     dd.to_csv(nombre+"D.csv", index=None)
61 G = Leer_grafos("Grafo1")
62 widths = []
63 widths = leer_capacity(G)
64 print(widths)
65 widths[:] = [(x - 7)/2 for x in widths]
66 plt.figure(figsize=(15, 15))
67 labels = {}
68 for u, v, data in G.edges(data=True):
69     labels[(u, v)] = data['weight']
70 position = nx.kamada_kawai_layout(G, dist=None, pos=None, scale
       =0.5, center=None, dim=2)
71 nx.draw_networkx_nodes(G, position, node_size=800, node_color="#
       de8919", node_shape="8")
72 nx.draw_networkx_edges(G, position, width=widths, edge_color='black
       ')
```

```
nx.draw_networkx_edge_labels(G, position, edge_labels=labels,
    font_size=10, label_pos=.5)
nx.draw_networkx_labels(G, position, font_size=15)

df = pd.DataFrame(position)
df.to_csv("position1" + ".csv", index=None, header=None)
plt.axis("off")
plt.savefig("Grafo1a" + ".png", bbox_inches='tight')
plt.savefig("Grafo1a" + ".eps", bbox_inches='tight')
plt.show(G)

d = shortest_augmenting_path(G, 4, 8, capacity="weight")
flowma = []
widths1 = []
widths0 = []
widths2 = []
widths3 = []
maxi = 0
nodos = []
nodosF = [4]
nodosS = [8]
nodos = listnodos(d, 4, 8)
for edges in d.edges():
    widths.append(d.edges[edges]["flow"])
    if d.edges[edges]["flow"] > 0:
        widths1.append(d.edges[edges]["capacity"])
        widths0.append(edges)
        flowma.append(d.edges[edges]['flow'])

    elif d.edges[edges]["flow"] == 0:
        widths2.append(edges)
        widths3.append(d.edges[edges]["capacity"])
print(widths1)
print(widths3)
flowma[:] = [x + 20 for x in flowma]
for i in flowma:
     if i > maxi:
        maxi = i
widths[:] = [x/10*x/6 for x in widths]
widths1[:] = [(x - 7)/2 for x in widths1]
widths3[:] = [(x - 7)/2 for x in widths3]
plt.figure(figsize=(15, 15))
position = pd.read_csv('position1.csv', header=None)
for u, v, data in d.edges(data=True):
    if data['flow'] >= 0:
     labels[(u, v)] = data['flow']
nx.draw_networkx_nodes(d, position,nodelist=nodos, node_size=800,
    node_color="#de8919", cnode_shape="8")
nx.draw_networkx_nodes(d, position, nodelist=nodosF,node_size=800,
    node_color="red", cnode_shape="8")
nx.draw_networkx_nodes(d, position,nodelist=nodosS, node_size=800,
    node_color="green", cnode_shape="8")

nx.draw_networkx_edges(d, position, edgelist=widths2, edge_color='
    black', width=widths3, arrows=False)
nx.draw_networkx_edges(d, position, edgelist=widths0, edge_cmap=plt
    .cm.Purples, width=widths1, edge_color=flowma,
```

```
                        edge_vmin=0, edge_vmax=maxi)

nx.draw_networkx_edge_labels(d, position, edge_labels=labels,
    font_size=10, label_pos=.5)
nx.draw_networkx_labels(d, position, font_size=15)

plt.axis("off")


plt.savefig("Grafo1cf" + ".png", bbox_inches='tight')
plt.savefig("Grafo1cf" + ".eps", bbox_inches='tight')

plt.show(G)
list=["Grafo1","Grafo2","Grafo3","Grafo4","Grafo5"]

for k in list:
    print(k)
    l = Leer_grafos(k)
    Tiempos(l,k)
```

Leer_grafo.py

La aplicación del algoritmo de flujo máximo *Shortest augmenting path* a todas las posibles combinaciones de fuentes y sumidero de cada uno de los cinco grafos nos dio como resultado los valores de flujo máximo para cada una de estas combinaciones y nos muestra cómo va variando el óptimo desde la peor combinación fuente sumidero a la mejor.

Para el Grafo1 esto se muestra en la figura 2 de la página 9.

Para el Grafo2 esto se muestra en la figura 3 de la página 10.

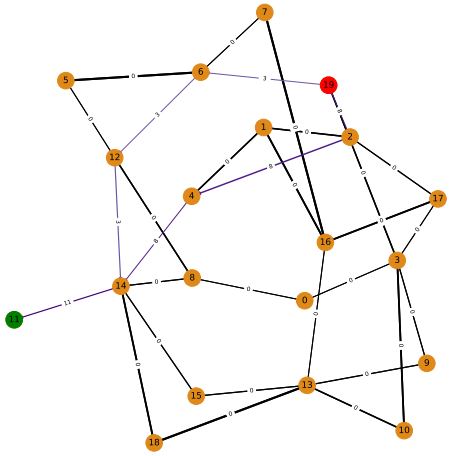
Para el Grafo3 esto se muestra en la figura 4 de la página 11.

Para el Grafo4 esto se muestra en la figura 5 de la página 12.

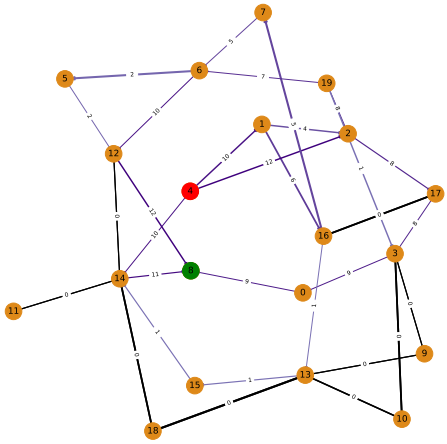
Para el Grafo5 esto se muestra en la figura 6 de la página 13.

4. Resultados del análisis de los datos

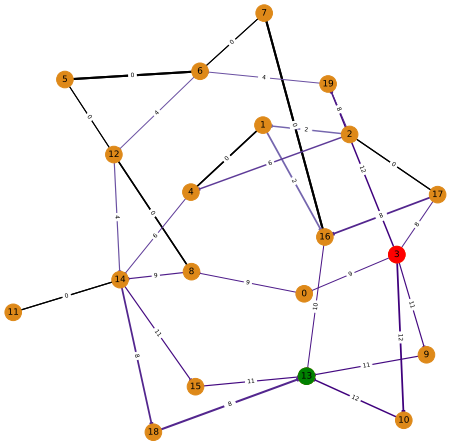
Con los datos recopilado de las ejecuciones del algoritmo de flujo máximo en los cinco grafos y todas las posibles combinaciones de fuentes y sumidero, se realizó histogramas para cada una de las seis características en los cinco grafos para determinar si los valores de la media del tiempo de ejecución tienen una distribución normal. Estos histogramas nos indican que la distribución no es normal como muestran la figuras 7, 8, 9, 10, 11, 12 de las páginas 14, 15, 16, 17, 18, 19 respectivamente.



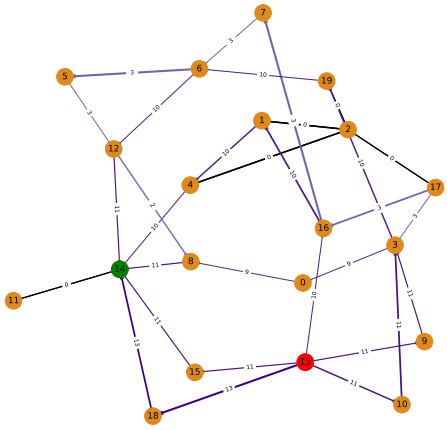
(a) *Grafo1*, flujo máximo 11



(b) *Grafo1*, flujo máximo 32

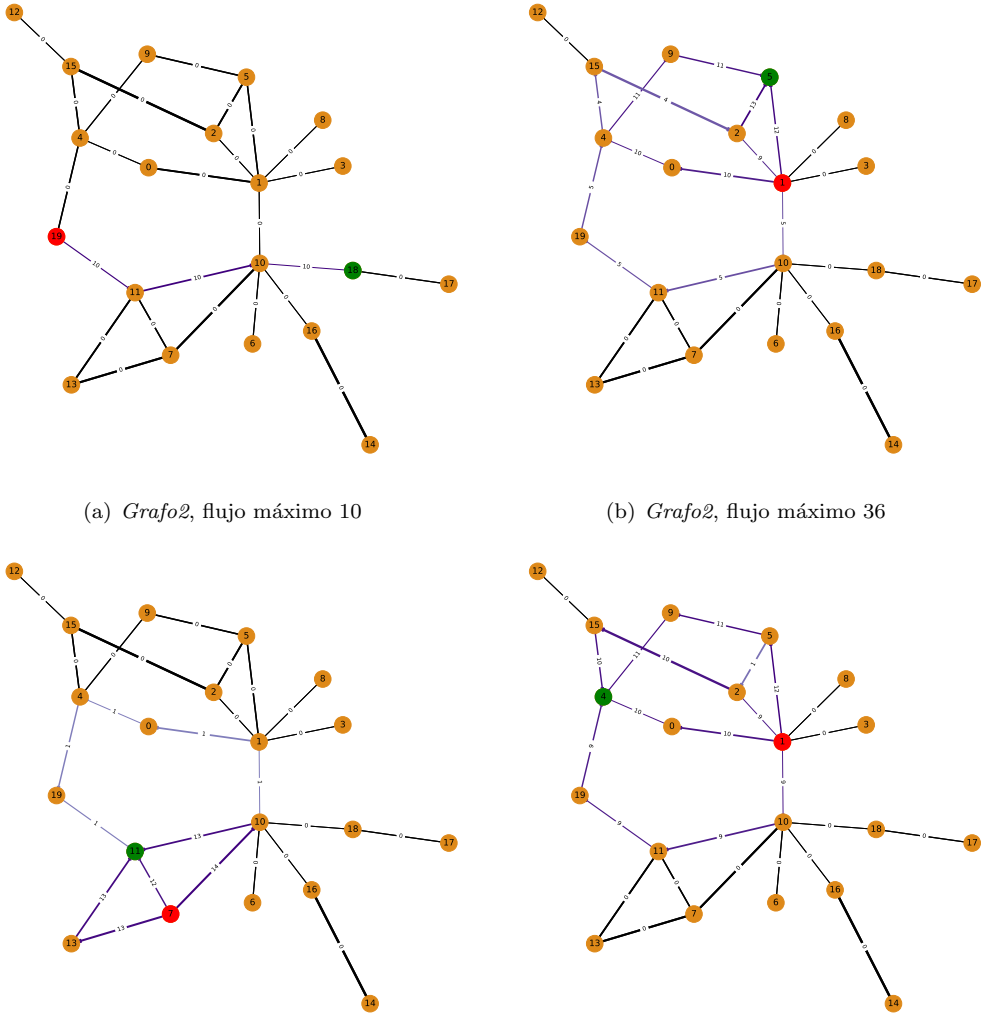


(c) *Grafo1*, flujo máximo 52



(d) *Grafo1*, flujo máximo 56

Figura 2: Grafos resultantes de la aplicación del algoritmo de flujo máximo, donde el grosor de las aristas representa la capacidad de las mismas, el color negro la ausencia de flujo, el color violeta la presencia de flujo y la intensidad del color violeta la cantidad de flujo que pasa por la arista. El color rojo del vértice representa la fuente y el vértice verde el sumidero.



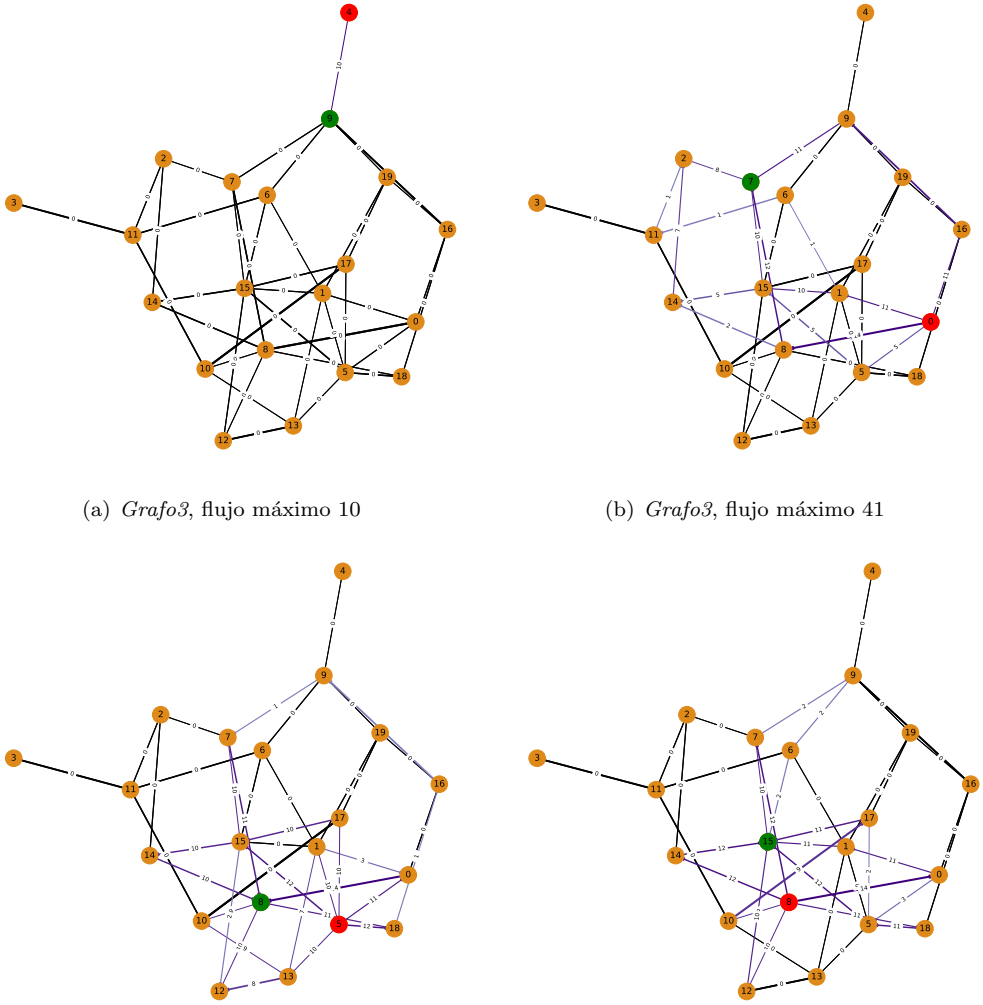
(a) *Grafo2*, flujo máximo 10

(b) *Grafo2*, flujo máximo 36

(c) *Grafo2*, flujo máximo 39

(d) *Grafo2*, flujo máximo 40

Figura 3: Grafos resultantes de la aplicación del algoritmo de flujo máximo, donde el grosor de las aristas representa la capacidad de las mismas, el color negro la ausencia de flujo, el color violeta la presencia de flujo y la intensidad del color violeta la cantidad de flujo que pasa por la arista. El color rojo del vértice representa la fuente y el vértice verde el sumidero.



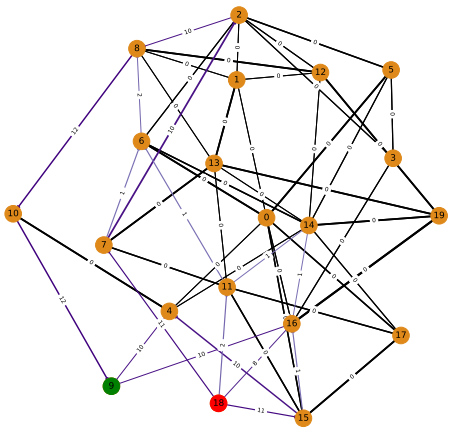
(a) *Grafo3*, flujo máximo 10

(b) *Grafo3*, flujo máximo 41

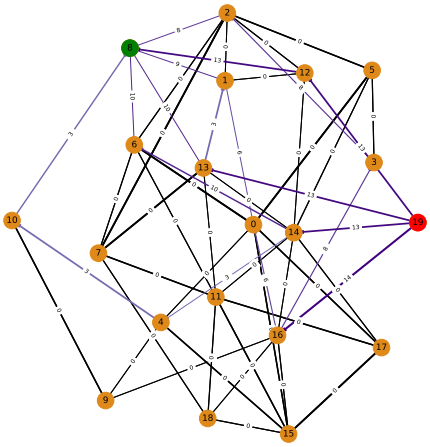
(c) *Grafo3*, flujo máximo 65

(d) *Grafo3*, flujo máximo 68

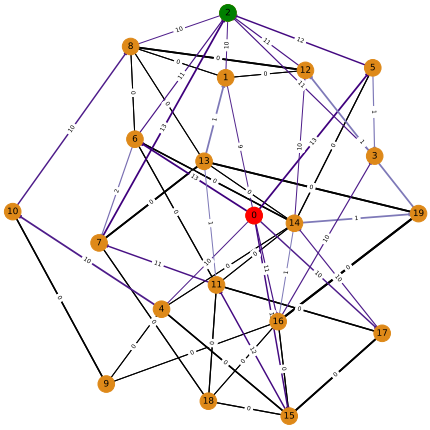
Figura 4: Grafos resultantes de la aplicación del algoritmo de flujo máximo, donde el grosor de las aristas representa la capacidad de las mismas, el color negro la ausencia de flujo, el color violeta la presencia de flujo y la intensidad del color violeta la cantidad de flujo que pasa por la arista. El color rojo del vértice representa la fuente y el vértice verde el sumidero.



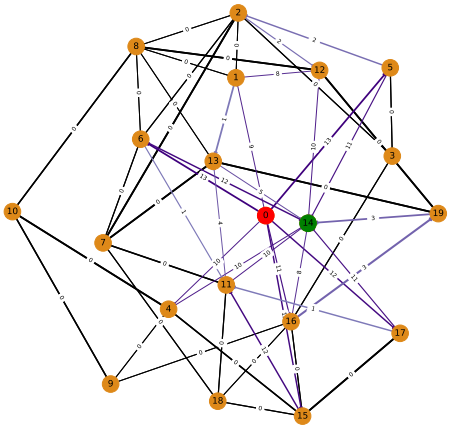
(a) *Grafo4*, flujo máximo 32



(b) *Grafo4*, flujo máximo 53

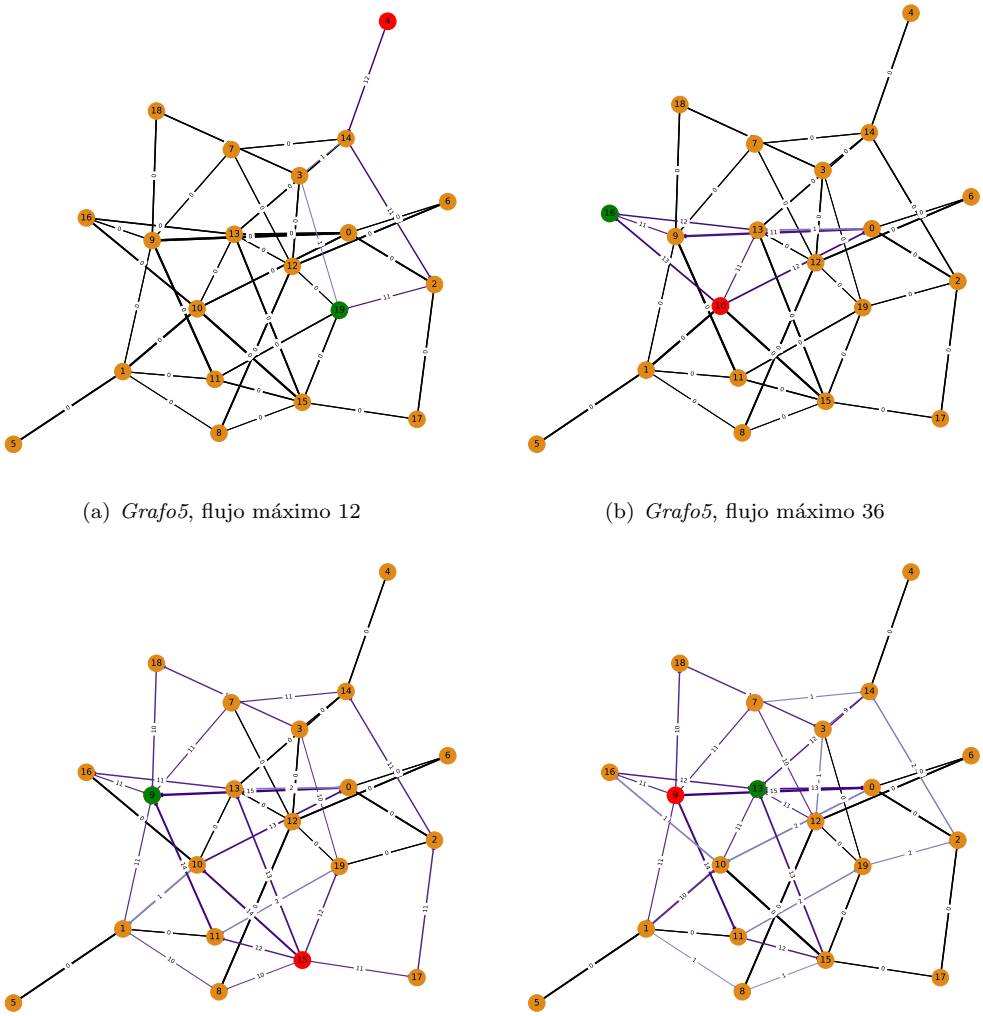


(c) *Grafo4*, flujo máximo 78



(d) *Grafo4*, flujo máximo 80

Figura 5: Grafos resultantes de la aplicación del algoritmo de flujo máximo, donde el grosor de las aristas representa la capacidad de las mismas, el color negro la ausencia de flujo, el color violeta la presencia de flujo y la intensidad del color violeta la cantidad de flujo que pasa por la arista. El color rojo del vértice representa la fuente y el vértice verde el sumidero.



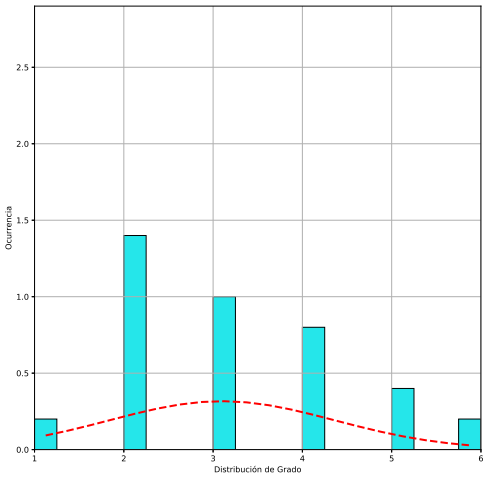
(a) *Grafo5*, flujo máximo 12

(b) *Grafo5*, flujo máximo 36

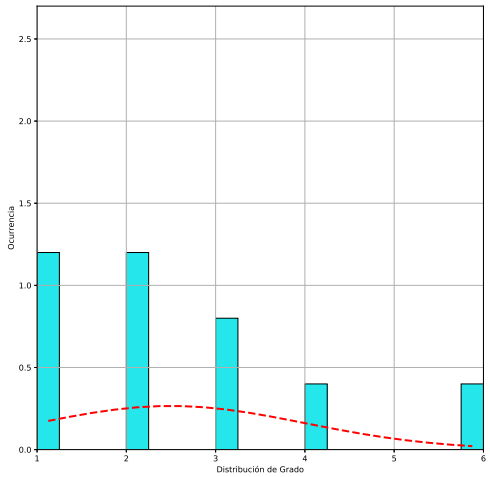
(c) *Grafo5*, flujo máximo 72

(d) *Grafo5*, flujo máximo 72

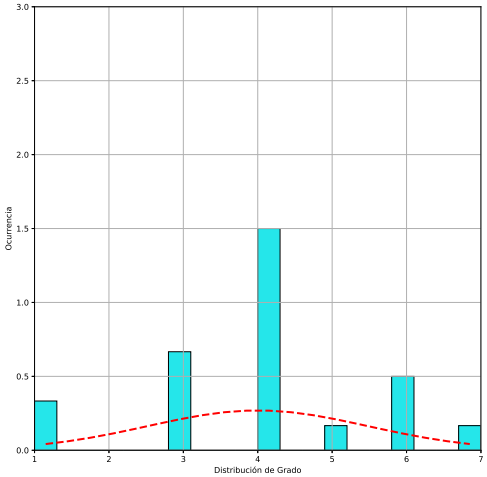
Figura 6: Grafos resultantes de la aplicación del algoritmo de flujo máximo, donde el grosor de las aristas representa la capacidad de las mismas, el color negro la ausencia de flujo, el color violeta la presencia de flujo y la intensidad del color violeta la cantidad de flujo que pasa por la arista. El color rojo del vértice representa la fuente y el vértice verde el sumidero.



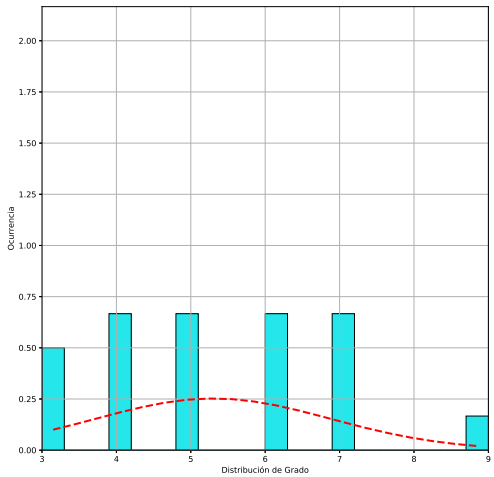
(a) *Grafo1*



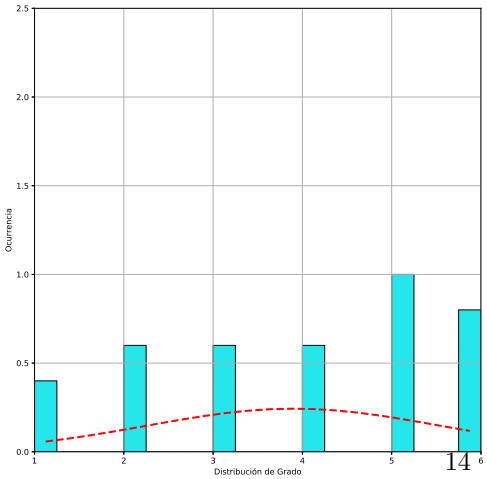
(b) *Grafo2*



(c) *Grafo3*

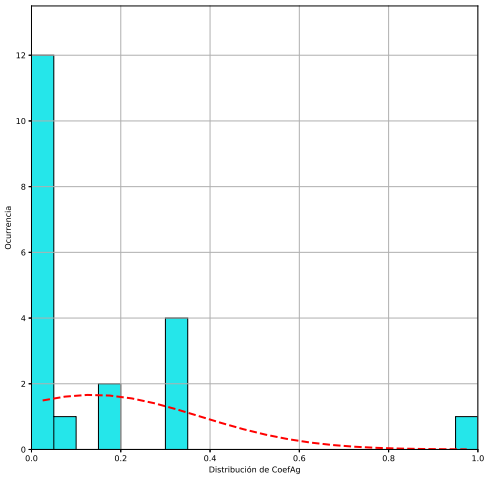


(d) *Grafo4*

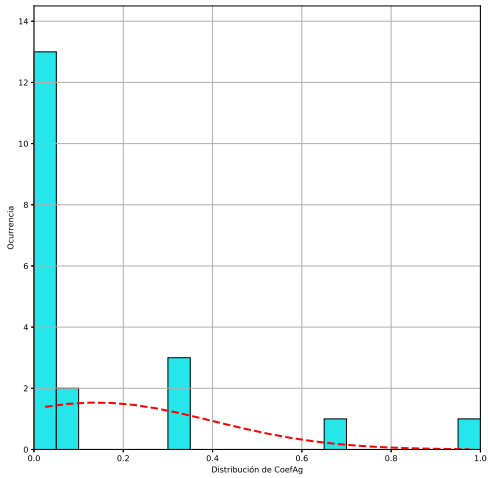


(e) *Grafo5*

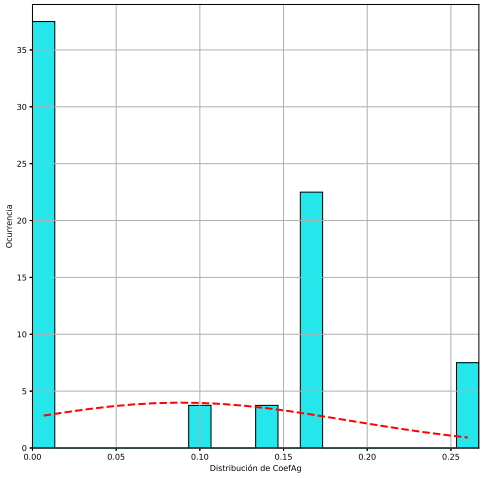
Figura 7: Histogramas que muestran la distribución de los valores de la característica distribución de grado.



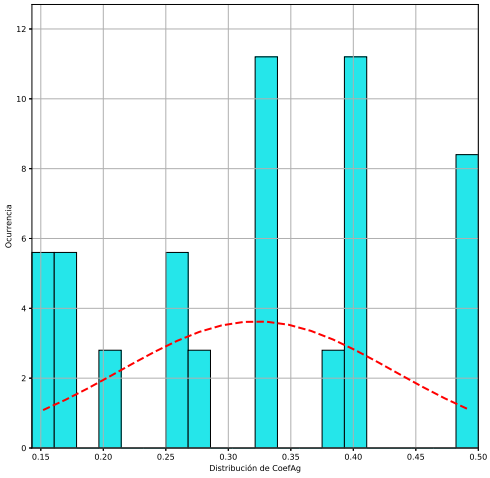
(a) *Grafo1*



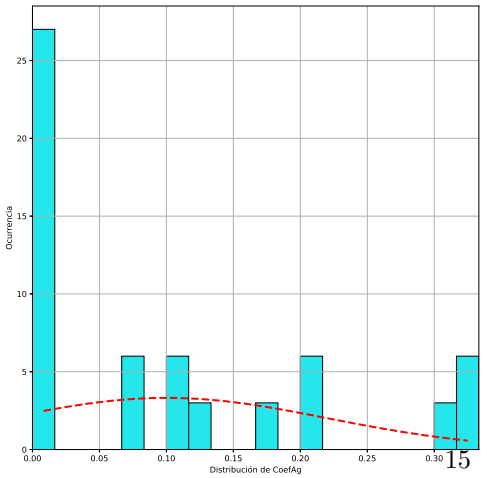
(b) *Grafo2*



(c) *Grafo3*

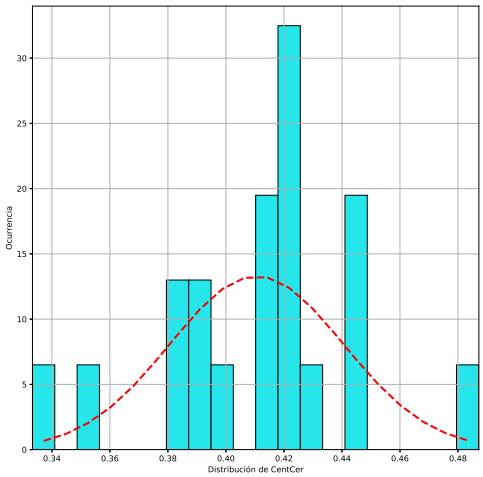


(d) *Grafo4*

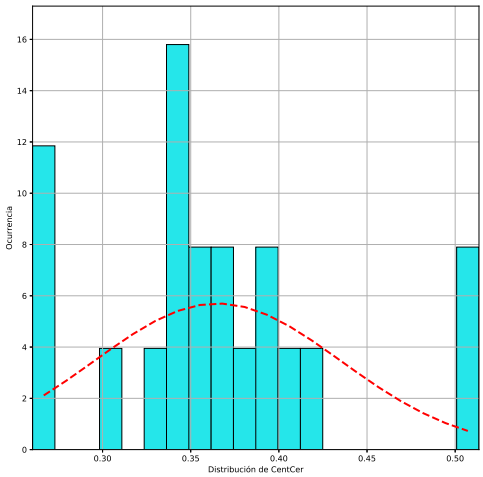


(e) *Grafo5*

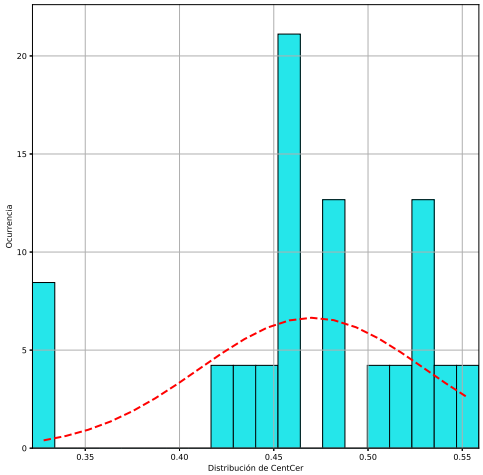
Figura 8: Histogramas que muestran la distribución de los valores de la característica coeficiente de agrupamiento.



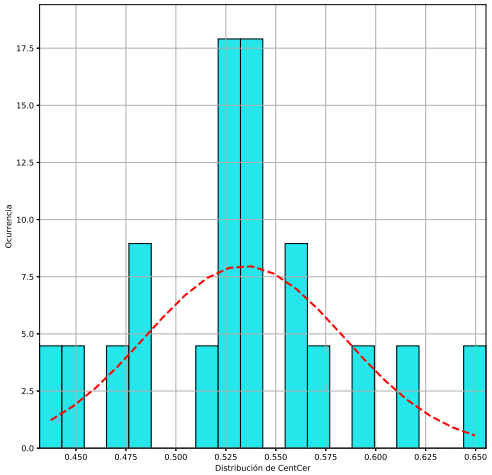
(a) *Grafo1*



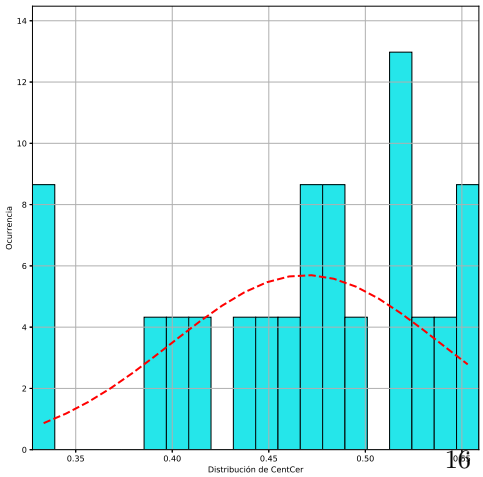
(b) *Grafo2*



(c) *Grafo3*

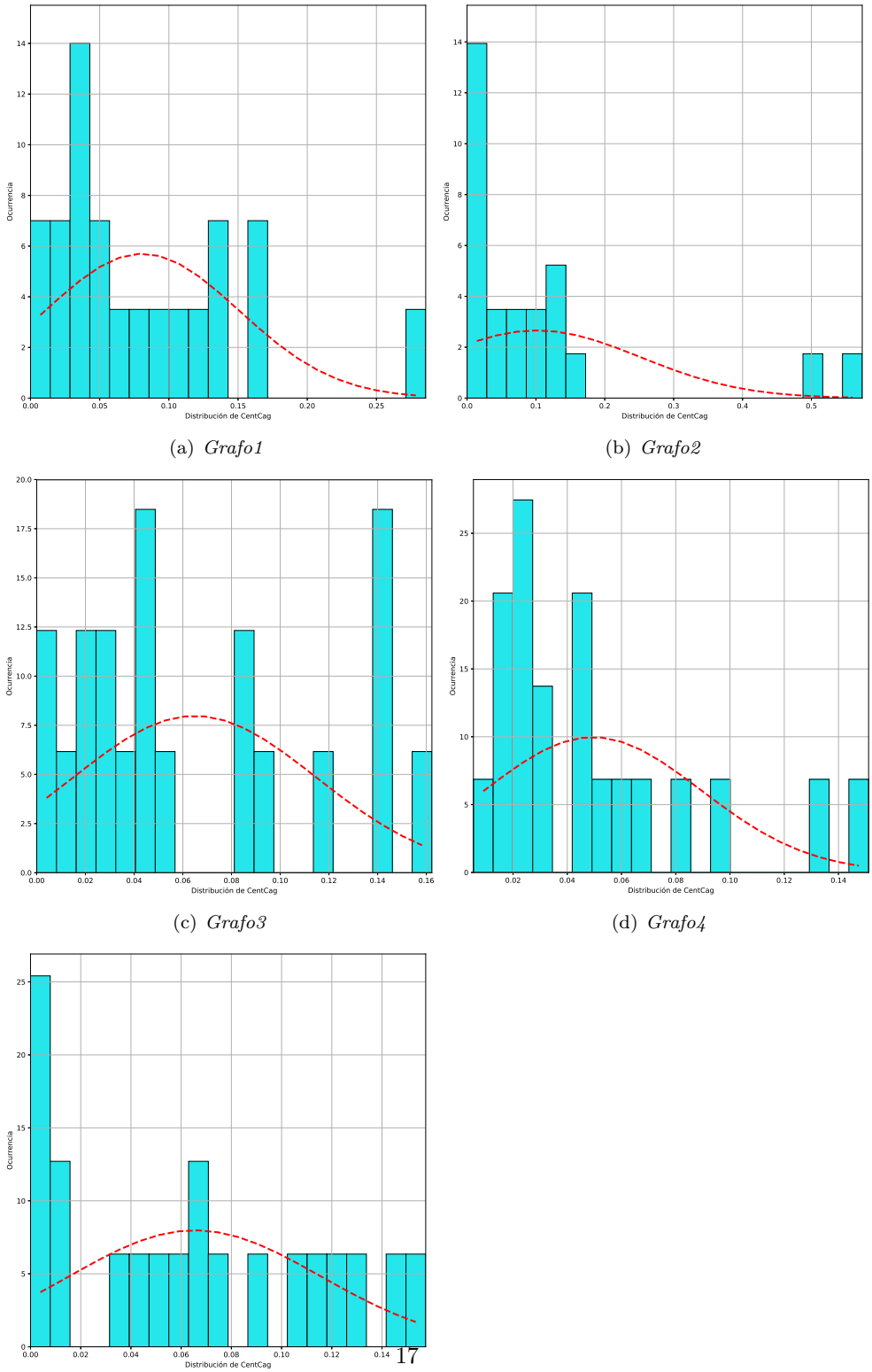


(d) *Grafo4*



(e) *Grafo5*

Figura 9: Histogramas que muestran la distribución de los valores de la característica centralidad de cercanía.



(a) *Grafo1*

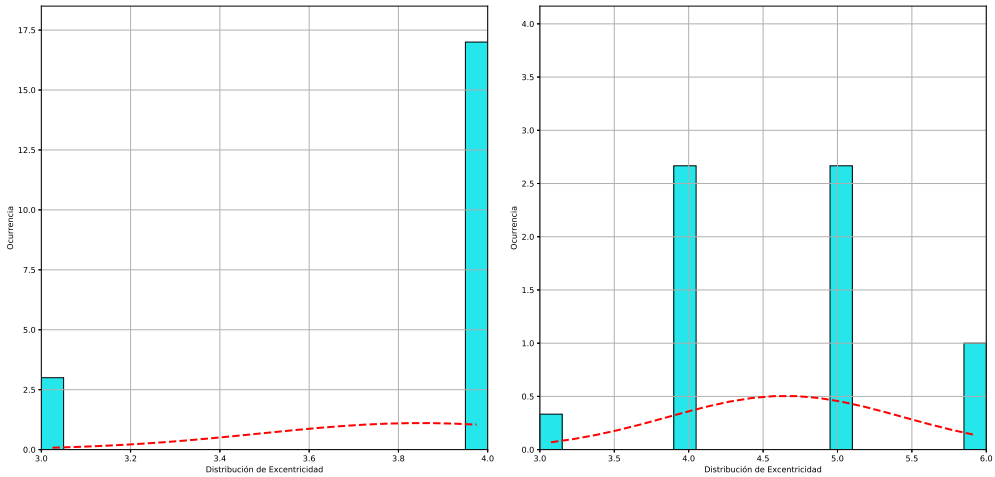
(b) *Grafo2*

(c) *Grafo3*

(d) *Grafo4*

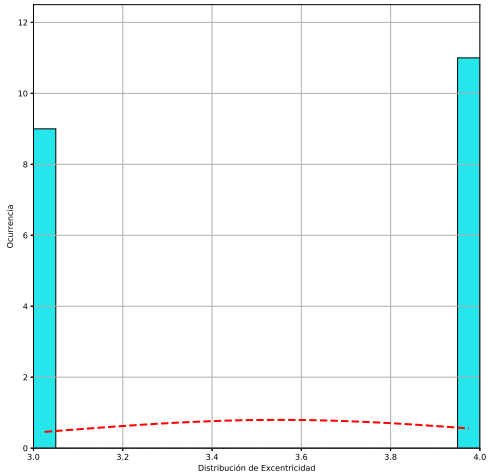
(e) *Grafo5*

Figura 10: Histogramas que muestran la distribución de los valores de la característica centralidad de carga.

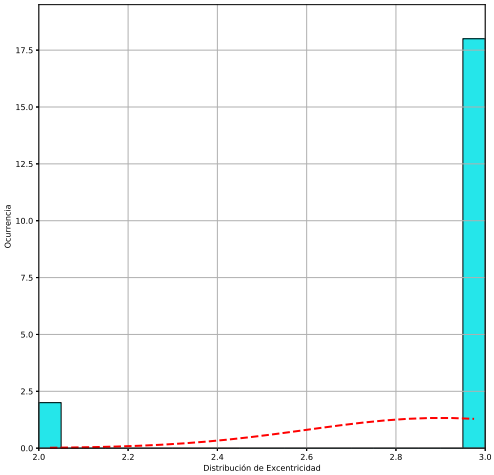


(a) *Grafo1*

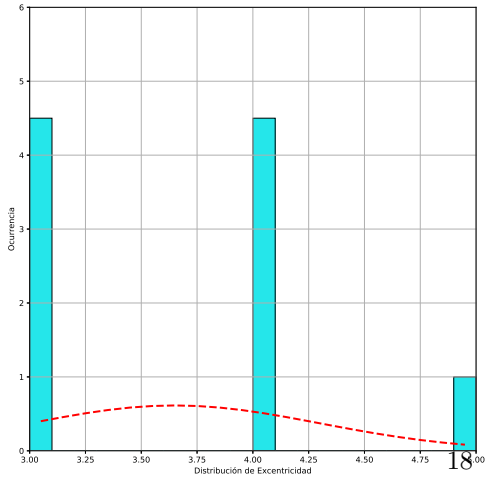
(b) *Grafo2*



(c) *Grafo3*

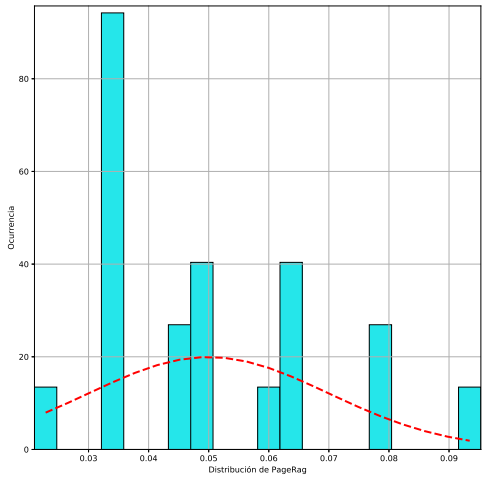


(d) *Grafo4*

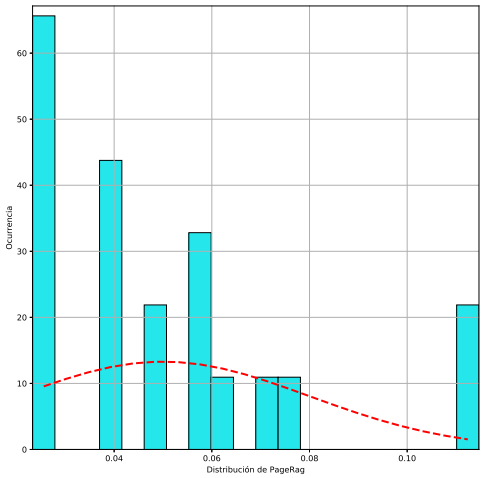


(e) *Grafo5*

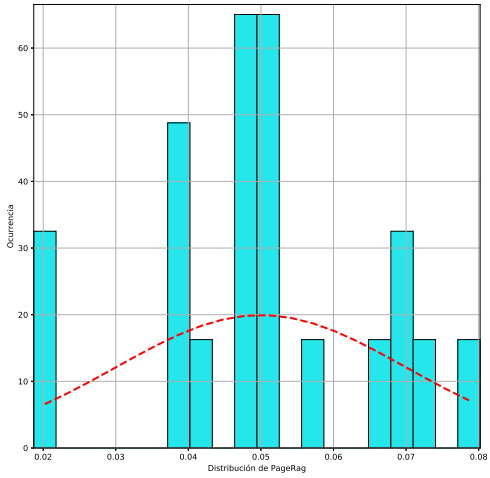
Figura 11: Histogramas que muestran la distribución de los valores de la característica excentricidad.



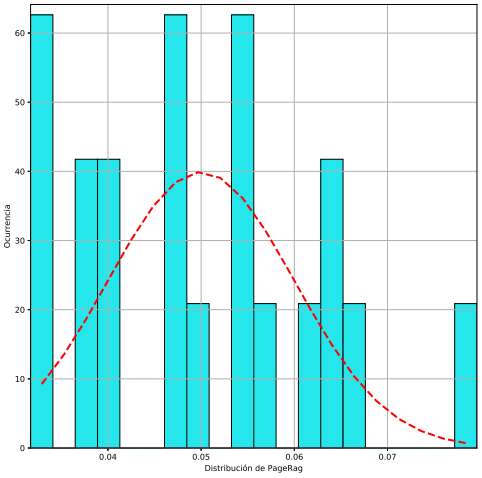
(a) *Grafo1*



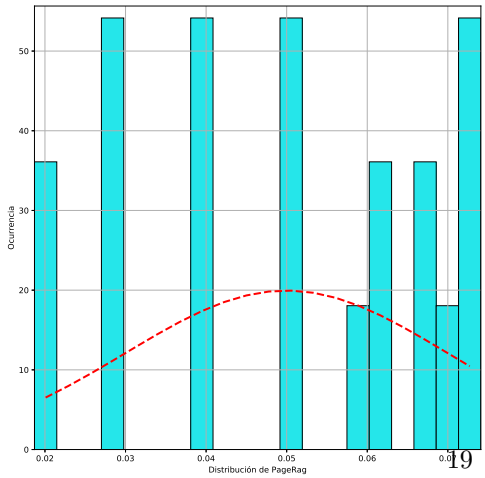
(b) *Grafo2*



(c) *Grafo3*



(d) *Grafo4*



(e) *Grafo5*

Figura 12: Histogramas que muestran la distribución de los valores de la característica *PageRank*.

Dado que la distribución de los valores de las características no es normal, se hizo una categorización por rangos para hacer cómoda la visualización de la relación con las variables dependientes, estos rangos se obtuvieron a través de los contenedores que devolvió histogramas que se le aplicaron a cada propiedad.

Con esta categorización de las características se realizaron las pruebas estadísticas para analizar la in fluencia de las características estructurales de los grafos en el tiempo de ejecución del algoritmo de flujo máximo y el valor de flujo máximo.

El siguiente fragmento de código nos muestra la realización de las pruebas antes mencionadas.

```
logX = np.log1p(df['Mediana'])
df = df.assign(mediana_log=logX.values)
df.drop(['Mediana'], axis= 1, inplace= True)

factores=["Grado","CoefAg","CentCer","CentCag","Excentricidad","PageRag"]
plt.figure(figsize=(8, 6))
for i in factores:
    print(rp.summary_cont(df['FlujoMax'].groupby(df[i])))

    anova = pg.anova (dv='FlujoMax', between=i, data=df, detailed=True , )
    pg._export_table (anova,("ANOVAsFlujoMax"+i+".csv"))

    ax=sns.boxplot(x=df["FlujoMax"], y=df[i], data=df, palette="cubehelix")

    plt.savefig("boxplot_FlujoMax" + i + ".eps", bbox_inches='tight')
    tukey = pairwise_tukeyhsd(endog = df["FlujoMax"], groups= df[i], alpha=0.05)

    tukey.plot_simultaneous(xlabel='Flujo Maximo', ylabel=i)

    plt.savefig("simultaneous_tukey" + i + ".eps", bbox_inches='tight')

    print(tukey.summary())
    t_csv = open("TukeyFlujoMax"+i+".csv", 'w')
    with t_csv:
        writer = csv.writer(t_csv)
        writer.writerows(tukey.summary())
    plt.show()

modelo = ols('FlujoMax ~ Grado + CoefAg + CentCer + CentCag + Excentricidad + PageRag +'
             ' Grado*CoefAg + Grado*CentCer+ Grado*CentCag + Grado*Excentricidad + Grado*PageRag'
             '+CoefAg*CentCer + CoefAg*CentCag + CoefAg*Excentricidad + CoefAg*PageRag + '
             'CentCer*CentCag + CentCer*Excentricidad + CentCer*PageRag + CentCag*Excentricidad'
```

```
33                  ' + CentCag*PageRag + Excentricidad*PageRag' ,data=df)
        .fit()
34  print(modelo.summary())
35  modelo_csv = open("Anova_MultFlujoMax.csv", 'w')
36  aov_table = sm.stats.anova_lm(modelo, typ=2)
37  df1=pd.DataFrame(aov_table)
38  df1.to_csv("modeloFlujoMax.csv")
```

Analisis_datos1.py

4.1. Análisis de varianza (ANOVA)

El análisis de varianza (ANOVA) es la técnica central en el análisis de datos experimentales. La idea general de esta técnica es separar la variación total en las partes con las que contribuye cada fuente de variación en el experimento. En el caso de los diseños completamente al azar se separan la variabilidad debida a los tratamientos y la debida al error. Cuando la primera predomina sobre la segunda, es cuando se concluye que las medias son diferentes. Cuando los tratamientos no dominan contribuyen igual o menos que el error, por lo que se concluye que las medias son iguales [2]. Para analizar si los diferentes factores (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad, *pagerank*) influyen en lsa variable dependiente *tiempo de ejecución* y valores de flujo máximo se realizó un ANOVA de un factor para cada caso.

4.1.1. Influencia de la distribución de grado en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

Cuadro 3: Influencia de la distribución de grado en el tiempo de ejecución

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
Grado	209597.452	2.000	104798.726	662.186	0.000	0.411
Within	300222.666	1897.000	158.262	-	-	-

En el cuadro **??** se muestra que existen diferencia entre las medianas de los grupos de factores ya que el$p - unc$es menor que 0,05 por lo que se rechaza la hipótesis de que la distribución de grado no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 13 de la página 22.

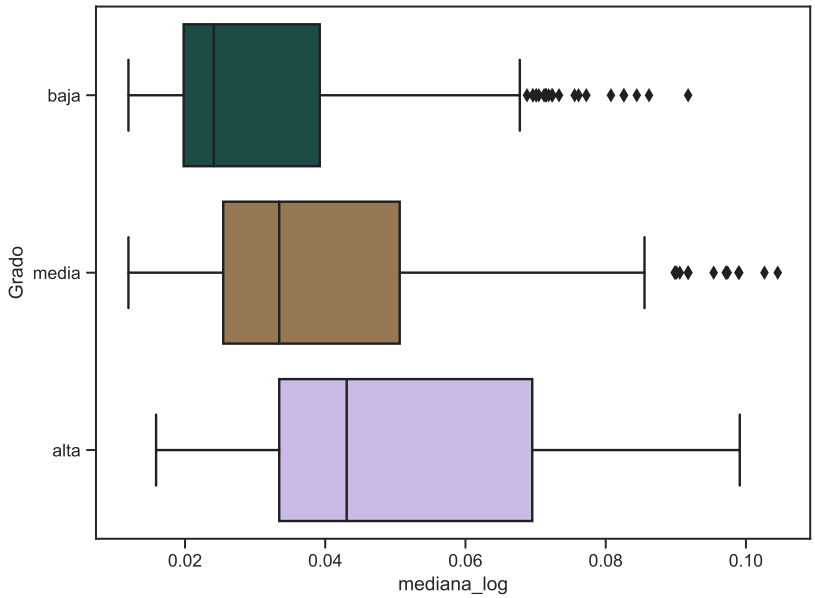


Figura 13: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con la distribución de grado.

En el cuadro 3 se muestra que existen grandes diferencia entre las medianas de los grupos de factores ya que el $p - unc$ es menor que 0,05 por lo que se se rechaza la hipótesis de que la distribución de grado no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 4 de la página 11. Por tal motivo se realiza la prueva de Tukey mostrara las diferencias entre las medianas de los factores, lo que se evidencia en el cuadro 4 de la página 22.

Cuadro 4: Influencia de la distribución de grado en el tiempo de ejecución *(Tukey)*

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	-0.020	-0.024	-0.016	True
alta	media	-0.011	-0.015	-0.007	True
baja	media	0.009	0.007	0.011	True

4.1.2. Influencia del coeficiente de agrupamiento en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

Cuadro 5: Influencia del coeficiente de agrupamiento en el tiempo de ejecución

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
CoefAg	0.013	2.000	0.006	19.962	0.000	0.021
Within	0.610	1897.000	0.000			

En el cuadro 5 se muestra que existen diferencia entre las medianas de los grupos de factores ya que el $p-unc$ es menor que 0,05 por lo que se rechaza la hipótesis de que el coeficiente de agrupamiento no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 14 de la página 23.

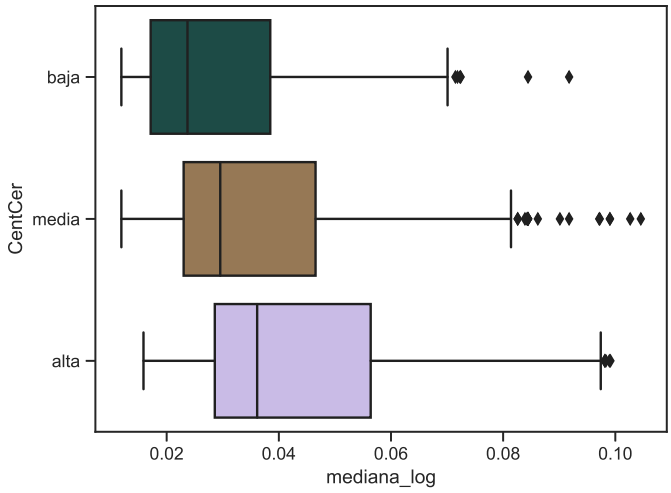


Figura 14: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con el coeficiente de agrupamiento.

En el cuadro 5 se muestra que existen marcadas diferencia entre las medianas de dos de los grupos de factores $p-unc$ es menor que 0,05 por lo que se se rechaza la hipótesis de que el coeficiente de agrupamiento no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 14 de la página 23. Por tal motivo se realiza la prueva de Tukey mostrara las diferencias entre las medianas de los factores, lo que se evidencia en el cuadro 6 de la página 24.

Cuadro 6: Influencia del coeficiente de agrupamiento en el tiempo de ejecución

group1	*group2*	*meandiff*	*lower*	*upper*	*reject*
alta	baja	0.005	-0.001	0.010	False
alta	media	0.014	0.007	0.020	True
baja	media	0.009	0.005	0.013	True

4.1.3. Influencia de la centralidad de cercanía en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

Cuadro 7: Influencia de la centralidad de cercanía en el tiempo de ejecución

Source	*SS*	*DF*	*MS*	*F*	*p-unc*	*np2*
CentCer	0.054	2.000	0.027	89.249	0.000	0.086
Within	0.570	1897.000	0.000			

En el cuadro 7 se muestra que existen diferencia entre las medianas de los grupos de factores ya que el $p-unc$ es menor que 0,05 por lo que se rechaza la hipótesis de que la centralidad de cercanía no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 15 de la página 24.

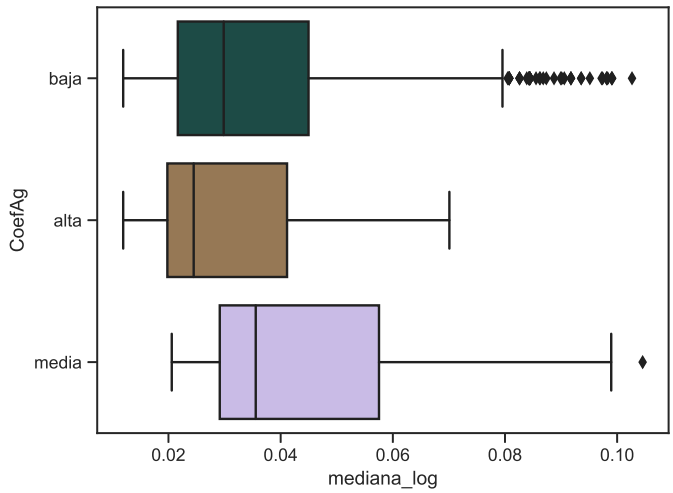


Figura 15: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con la centralidad de cercanía.

En el cuadro 7 se muestra que existen marcadas diferencia entre las medianas de dos de los grupos de factores $p - unc$ es menor que 0,05 por lo que se se rechaza la hipótesis de que la centralidad de cercanía no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 15 de la página 24. Por tal motivo se realiza la prueva de Tukey mostrara las diferencias entre las medianas de los factores, lo que se evidencia en el cuadro 8 de la página 25.

Cuadro 8: Influencia de la centralidad de cercanía en el tiempo de ejecución

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	-0.016	-0.018	-0.013	*True*
alta	media	-0.008	-0.011	-0.006	*True*
baja	media	0.007	0.005	0.010	*True*

4.1.4. Influencia de la centralidad de carga en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

Cuadro 9: Influencia de la centralidad de carga en el tiempo de ejecución(ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
CentCag	0.000	2.000	0.000	0.476	0.621	0.001
Within	0.623	1897.000	0.000			

En el cuadro 9 se muestra que no existen diferencia entre las medianas de los grupos de factores ya que el $p - unc$ es meyor que 0,05 por lo que se acepta la hipótesis de que la centralidad de carga no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 16 de la página 26.

4.1.5. Influencia de la excentricidad en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

En el cuadro 10 se muestra que existen diferencia entre las medianas de los grupos de factores ya que el $p - unc$ es menor que 0,05 por lo que se rechaza la hipótesis de que la excentricidad no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 17 de la página 26.

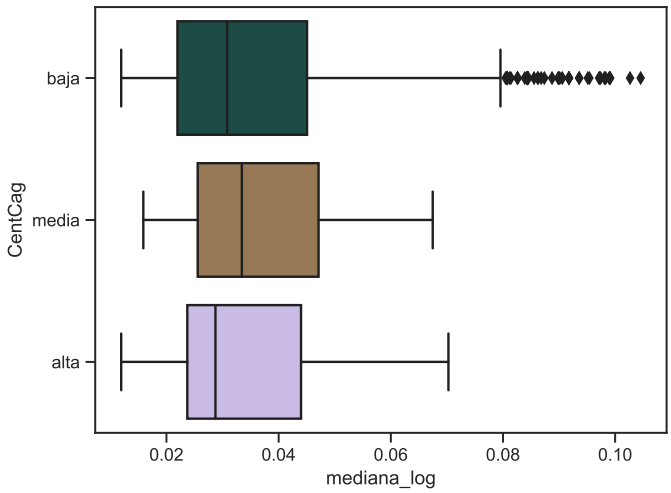


Figura 16: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con la centralidad de carga.

Cuadro 10: Influencia de la excentricidad en el tiempo de ejecución (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
Excentricidad	0.043	2.000	0.021	70.117	0.000	0.069
Within	0.580	1897.000	0.000			

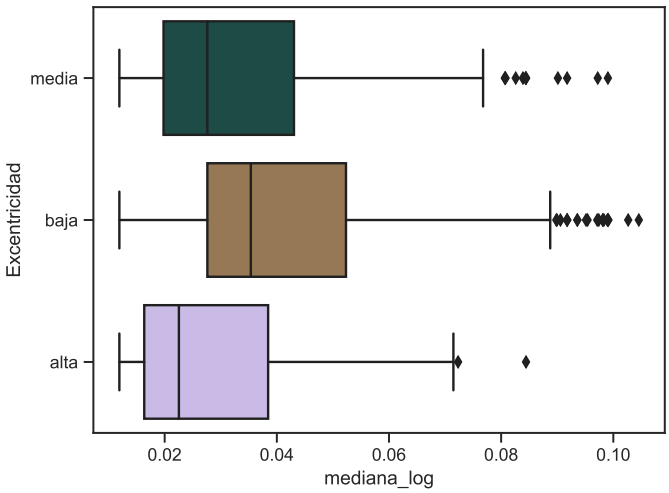


Figura 17: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con la excentricidad.

En el cuadro 10 se muestra que existen diferencia entre las medianas de dos de los grupos de factores $p - unc$ es menor que 0,05 por lo que se se rechaza la hipótesis de que la excentricidad no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 17 de la página 26. Por tal motivo se realiza la prueva de Tukey mostrara las diferencias entre las medianas de los factores, lo que se evidencia en el cuadro 11 de la página 27.

Cuadro 11: Influencia de la excentricidad en el tiempo de ejecución (Tukey)

group1	**group2**	***meandiff***	***lower***	***upper***	***reject***
alta	baja	0.013	0.010	0.016	*True*
alta	media	0.004	0.001	0.007	*True*
baja	media	-0.008	-0.010	-0.006	*True*

4.1.6. Influencia del *PageRank* en el tiempo de ejecución

El siguiente cuadro muestra el resultado de la aplicación del ANOVA.

Cuadro 12: Influencia del *PageRank* en el tiempo de ejecución (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
PageRag	0.016	2.000	0.008	25.358	0.000	0.026
Within	0.607	1897.000	0.000			

En el cuadro 12 se muestra que existen diferencia entre las medianas de los grupos de factores ya que el $p - unc$ es menor que 0,05 por lo que se rechaza la hipótesis de que el *PageRank* no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 18 de la página 28

En el cuadro 12 se muestra que existen diferencia entre las medianas de dos de los grupos de factores $p - unc$ es menor que 0,05 por lo que se se rechaza la hipótesis de que el *PageRank* no influye en el *tiempo de ejecución*. Esto se puede observar en la figura 17 de la página 26. Por tal motivo se realiza la prueva de Tukey mostrara las diferencias entre las medianas de los factores, lo que se evidencia en el cuadro 13 de la página **??**.

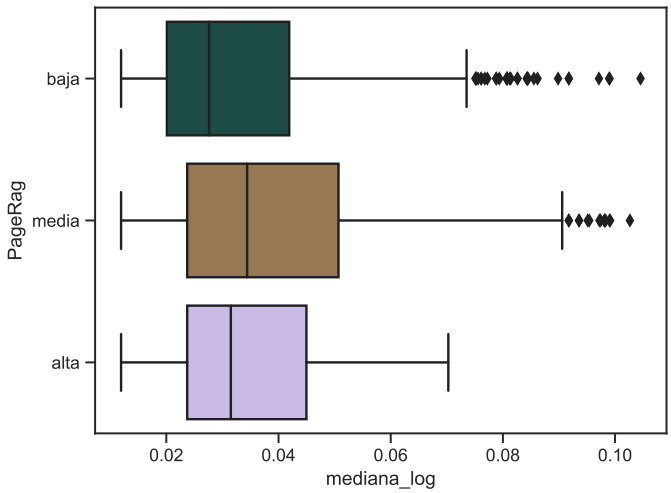


Figura 18: Diagrama de caja y bigotes que relaciona los tiempos de ejecución con el *PageRank*.

Cuadro 13: Influencia del *PageRank* en el tiempo de ejecuciónAdd caption

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	-0.001	-0.006	0.005	*False*
alta	media	0.005	-0.001	0.011	*False*
baja	media	0.006	0.004	0.008	*True*

4.1.7. Influencia de los seis factores (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad, *pagerank*) en el tiempo de ejecución

Para analizar este caso se realizo ANOVA multifactorial dando como resultado el cuadro 14 de la página 29.

Cuadro 14: Influencia de los seis factores (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad, *pagerank*) en el tiempo de ejecución (MANOVA)

	sum_sq	***df***	***F***	*PR(>F)*
Grado	0.000	2.000	0.000	1.000
CoefAg	0.000	2.000	0.000	1.000
CentCer	0.000	2.000	-0.409	1.000
CentCag	0.000	2.000	0.000	1.000
Excentricidad	0.000	2.000	0.000	1.000
PageRag	0.000	2.000	0.000	1.000
Grado:CoefAg	0.001	4.000	0.594	0.619
Grado:CentCer	0.001	4.000	1.048	0.381
Grado:CentCag	0.001	4.000	0.854	0.491
Grado:Excentricidad	0.001	4.000	1.064	0.373
Grado:PageRag	0.001	4.000	0.858	0.489
CoefAg:CentCer	0.001	4.000	0.560	0.571
CoefAg:CentCag	0.001	4.000	0.525	0.665
CoefAg:Excentricidad	0.000	4.000	0.203	0.816
CoefAg:PageRag	0.001	4.000	0.430	0.731
CentCer:CentCag	0.001	4.000	0.758	0.517
CentCer:Excentricidad	0.001	4.000	0.543	0.653
CentCer:PageRag	0.001	4.000	0.755	0.519
CentCag:Excentricidad	0.001	4.000	0.584	0.626
CentCag:PageRag	0.001	4.000	0.640	0.527
Excentricidad:PageRag	0.001	4.000	0.677	0.608
Residual	0.549	1876.000		

En el cuadro 7 se puede observar que los factores que más influyen en el tiempo de ejecución son el número de vértices y la densidad de los grafos, que además existe una relación entre el número de nodos y la densidad de los grafos.

4.1.8. Influencia de las seis propiedades en el flujo máximo

Los cuadro 15, 16, 17, 18, 19, 19 de las páginas 30, 30, 30, 30,30, 30 respectivamente, muestran los resultados de la aplicación del ANOVA en las seis propiedades con respecto los valores de flujo máximo.

Cuadro 15: Influencia de la distribución de grado en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
Grado	209597.452	2.000	104798.726	662.186	0.000	0.411
Within	300222.666	1897.000	158.262			

Cuadro 16: Influencia de la distribución de coeficiente de agrupamiento en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
CoefAg	59963.814	2.000	29981.907	126.431	0.000	0.118
Within	449856.304	1897.000	237.141			

Cuadro 17: Influencia de la distribución de centralidad de cercanía en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
CentCer	256496.045	2.000	128248.023	960.377	0.000	0.503
Within	253324.073	1897.000	133.539			

Cuadro 18: Influencia de la centralidad de carga en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
CentCag	6469.019	2.000	3234.510	12.190	0.000	0.013
Within	503351.098	1897.000	265.341			

Cuadro 19: Influencia de la excentricidad en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
Excentricidad	200853.698	2.000	100426.849	616.603	0.000	0.394
Within	308966.420	1897.000	162.871	-	-	-

Cuadro 20: Influencia de el *pagerank* en el flujo máximo (ANOVA)

Source	***SS***	***DF***	***MS***	***F***	***p-unc***	***np2***
PageRag	67070.251	2.000	33535.125	143.684	0.000	0.132
Within	442749.867	1897.000	233.395			

En los cuadro anteriores se muestra que existen diferencia entre las medianas de los grupos de factores ya que el$p - unc$es menor que 0,05 por lo que se rechaza la hipótesis de que las propiedades no influye en el *flujo máximo*. Esto se puede observar claramente en las figuras 19, 20, 21, 22, 23, 23 de las páginas 31, 32, 32, 33, 33, 34 respectivamente.

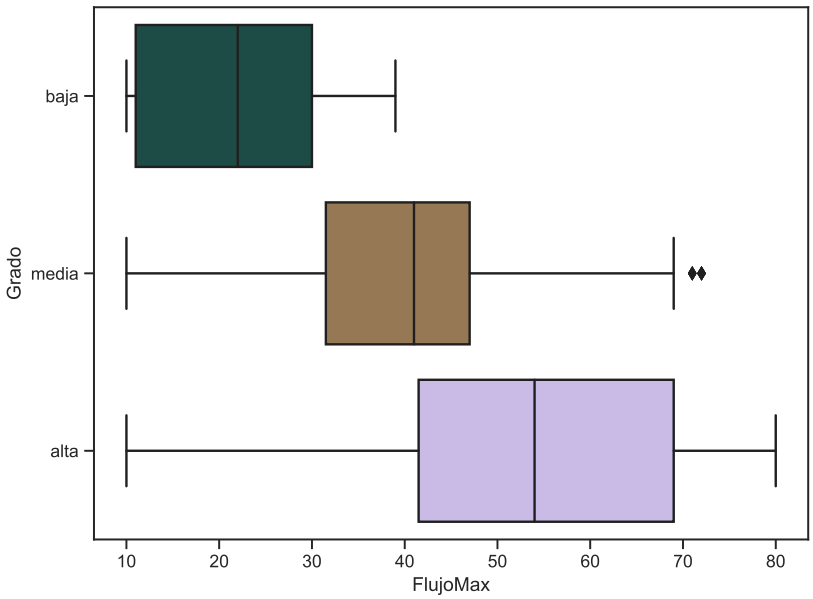


Figura 19: Diagrama de caja y bigotes que relaciona los flujos máximos con la distribución de grado.

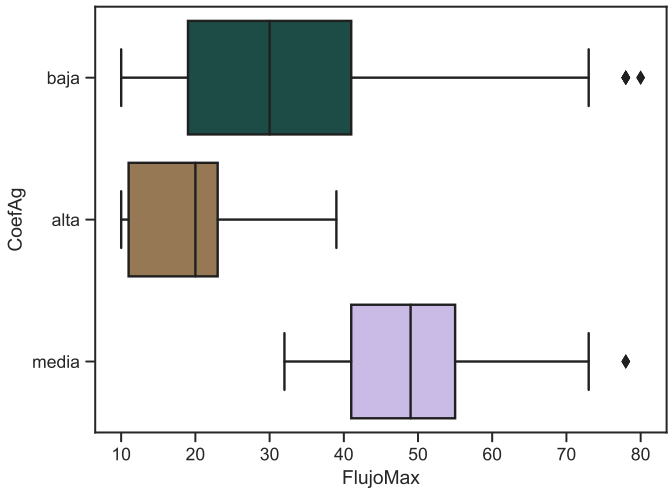


Figura 20: Diagrama de caja y bigotes que relaciona los flujos máximos con el coeficiente de agrupamiento.

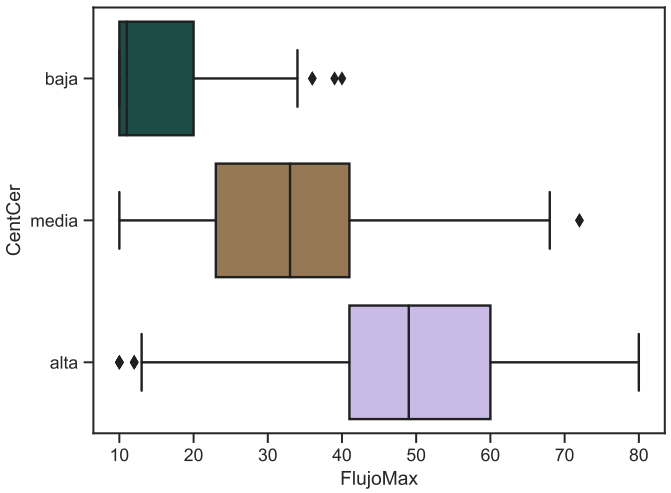


Figura 21: Diagrama de caja y bigotes que relaciona los flujos máximos con la centralidad de cercanía.

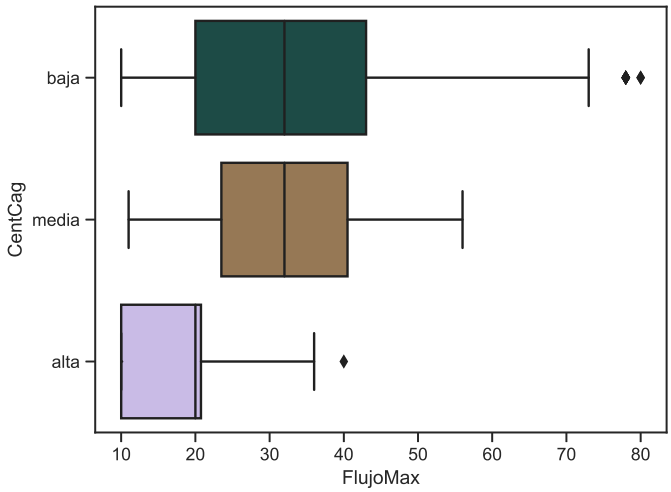


Figura 22: Diagrama de caja y bigotes que relaciona los flujos máximos con la centralidad de carga.

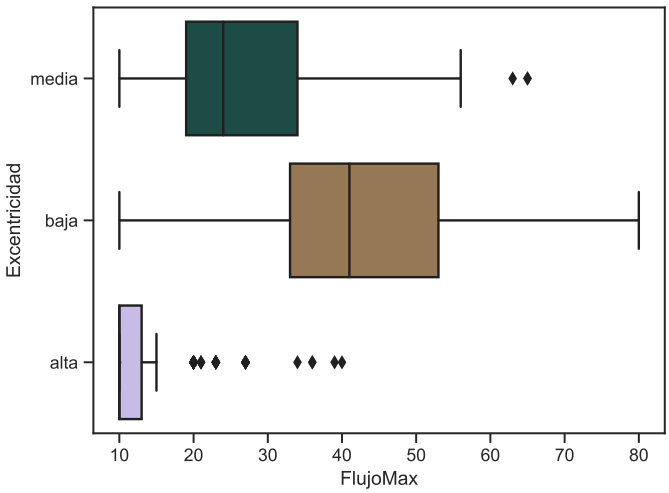


Figura 23: Diagrama de caja y bigotes que relaciona los flujos máximos con la excentricidad.

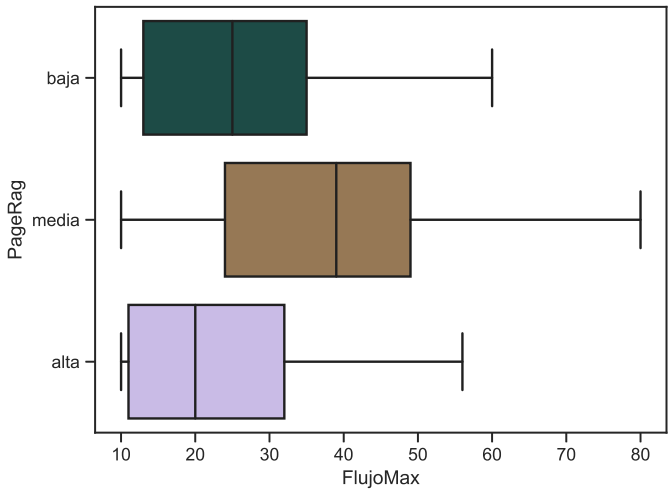


Figura 24: Diagrama de caja y bigotes que relaciona los flujos máximos con el *pagerank*.

Cuadro 21: Influencia de la distribución de grado en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	-33.721	-36.659	-30.782	*True*
alta	media	-15.682	-18.613	-12.751	*True*
baja	media	18.039	16.642	19.436	*True*

Cuadro 22: Influencia de el coeficiente de agrupamiento en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	11.253	6.389	16.118	*True*
alta	media	30.351	24.741	35.961	*True*
baja	media	19.098	16.039	22.156	*True*

Cuadro 23: Influencia de la centralidad de cercanía en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	-33.901	-35.715	-32.086	*True*
alta	media	-17.016	-18.568	-15.463	*True*
baja	media	16.885	15.355	18.415	*True*

Cuadro 24: Influencia de la centralidad de carga en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	13.165	6.904	19.427	*True*
alta	media	13.816	3.080	24.551	*True*
baja	media	0.651	-8.160	9.461	*False*

Cuadro 25: Influencia de la excentricidad en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	29.093	26.913	31.272	*True*
alta	media	12.855	10.692	15.017	*True*
baja	media	-16.238	-17.711	-14.764	*True*

Debido a las diferencias obtenidas en todas las ANOVAS realizadas, procedemos a relizar la prueba de Tukey cuyus resultados se muestran el los cuadros

En los cuadros 21, 22, 23, 24, 25, 26 de las páginas 34, 34, 34, 35, 35, 35 se pueden observar claramente que las seis características influye en el valor máximo de flujo.

4.1.9. Influencia de los seis factores (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad, *pagerank*) en el flujo máximo

Para analizar este caso se realizo ANOVA multifactorial dando como resultado el cuadro 27 de la página 36.

En el cuadro 27 de la página 36 se observa claramente que todos las características influye en los valores del flujo máximo corroborando lo arrojado en las pruebas anteriores.

Cuadro 26: Influencia del *pagerank* en el flujo máximo (Tukey)

group1	***group2***	***meandiff***	***lower***	***upper***	***reject***
alta	baja	3.867	-0.998	8.733	*False*
alta	media	15.846	10.916	20.776	*True*
baja	media	11.979	10.269	13.689	*True*

Cuadro 27: Influencia de los seis factores (distribución de grado, coeficiente de agrupamiento, centralidad de cercanía, centralidad de carga, excentricidad, *pagerank*) en el flujo máximo (MANOVA)

	sum_sq	***df***	***F***	***PR(¿F)***
Grado	0.001	2.000	0.000	1.000
CoefAg	-45.503	2.000	-0.210	1.000
CentCer	2523.680	2.000	11.667	0.000
CentCag	0.000	2.000	0.000	1.000
Excentricidad	0.000	2.000	0.000	1.000
PageRag	0.000	2.000	0.000	1.000
Grado:CoefAg	34.118	4.000	0.079	0.971
Grado:CentCer	37.572	4.000	0.087	0.987
Grado:CentCag	54.632	4.000	0.126	0.973
Grado:Excentricidad	35.792	4.000	0.083	0.988
Grado:PageRag	59.651	4.000	0.138	0.968
CoefAg:CentCer	55.193	4.000	0.128	0.880
CoefAg:CentCag	38.260	4.000	0.088	0.966
CoefAg:Excentricidad	34.182	4.000	0.079	0.924
CoefAg:PageRag	45.273	4.000	0.105	0.957
CentCer:CentCag	46.187	4.000	0.107	0.956
CentCer:Excentricidad	41.053	4.000	0.095	0.963
CentCer:PageRag	74.952	4.000	0.173	0.915
CentCag:Excentricidad	90.135	4.000	0.208	0.891
CentCag:PageRag	79.931	4.000	0.185	0.831
Excentricidad:PageRag	51.771	4.000	0.120	0.976
Residual	202895.688	1876.000		

Referencias

[1] Bartłomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych-Pawlewicz. Shortest augmenting paths for online matchings on trees. *Theory of Computing Systems*, 62(2):337–348, Feb 2018.

[2] Humberto Gutiérrez and Román de la Vara. *Análisis y diseño de experimentos*. The McGraw-Hill Companies, Inc., segunda edición edition, 2008. 60–74.

[3] Desarrolladores NetworkX. `https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html#networkx.generators.random_graphs.erdos_renyi_graph`. Accessed: 01-04-2019.

[4] Desarrolladores NetworkX. `https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.random_graphs.fast_gnp_random_graph.html#networkx.generators.random_graphs.fast_gnp_random_graph`. Accessed: 01-04-2019.

[5] Desarrolladores NetworkX. `https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.random_graphs.binomial_graph.html#networkx.generators.random_graphs.binomial_graph`. Accessed: 01-04-2019.

[6] Desarrolladores NetworkX. `https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.flow.maximum_flow.html#networkx.algorithms.flow.maximum_flow`. Accessed: 01-04-2019.

[7] Desarrolladores NetworkX. `https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.coloring.greedy_color.html`. Accessed: 18-03-2019.

[8] Sadegh Nobari, Xuesong Lu, Panagiotis Karras, and Stéphane Bressan. Fast random graph generation. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 331–342, New York, NY, USA, 2011. ACM.