Caracterización estructural de instancias.

4166

30 de abril de 2019

Introducción

En un grafo formado por aristas y nodos, donde cada arista tiene un valor que se interpreta como la capacidad, algo de interés es, dado dos nodos identificados como **fuente** y **sumidero**, es saber cual es el valor del flujo máximo que puede transportar desde el fuente al sumidero. Otros aspectos importantes es ver como las caracteristicas del nodo fuente y el nodo sumidero afectan al valor del flujo máximo en el mismo grafo.

En este documento se habla acerca de las caracteristicas que presentan los nodos en un grafo y de que manera afectan estos en el valor del flujo máximo mediante un algoritmo de `Networkx`.

Especificaciones técnicas

La computadora en la que se corrió los algoritmos es una Macbook Pro, cuyo procesador es un Intel Core i5 de 2.3 GHz. Tiene una memoria RAM de 8 GB 2133 MHz.

1. Generador

`watts_strogatz_graph()`

Este generador de grafos toma como parámetros la cantidad de nodos deseados, el numero de nodos vecinos con los que se puede conectar y la probabilidad de conexión desde ese nodo con los nodos que puede conectar.

Usualmente, este tipo de grafos, se usa para hacer las redes de mundo pequeño estas redes hacen referencia a que por mas alejado que esté un nodo del otro,

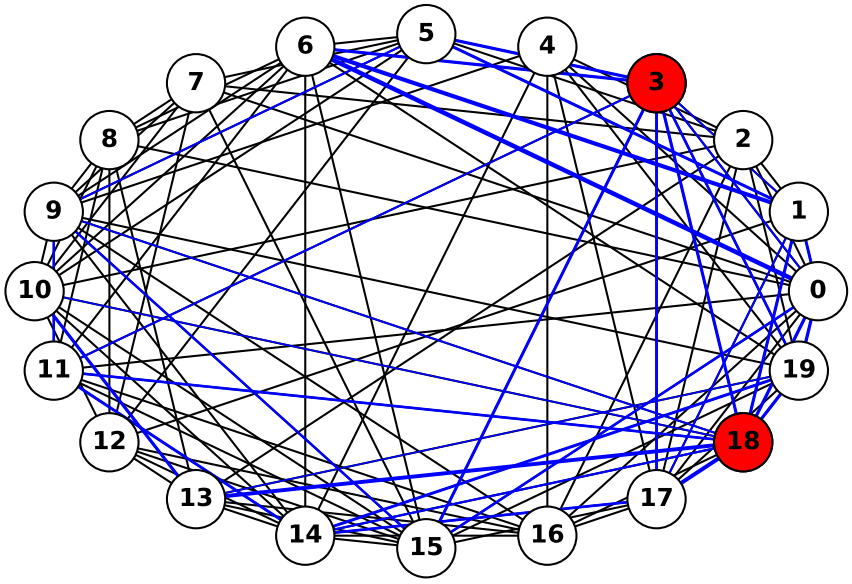

Figura 1: Grafo con rutas de flujo máimo.

todos los nodos son accesibles desde otro nodo.

En las figuras 1, 2, 3, 4, 5 se muestran grafos que son hechos con este generador.

Algoritmos

En esta sección se habla de los algoritmos que se usaron junto con una breve descripción.

Algoritmo de flujo máximo

El algoritmo implementado para encontrar el flujo máximo es `maximum_flow()` el cual toma como parámetros un grafo el cual debe tener como atributo la capacidad de cada arista, junto con el par de nodos fuente y sumidero. Al encontrar el flujo máximo devuelve el valor de este, asi como el flujo que pasa por cada arista para obtener ese valor del flujo máximo.

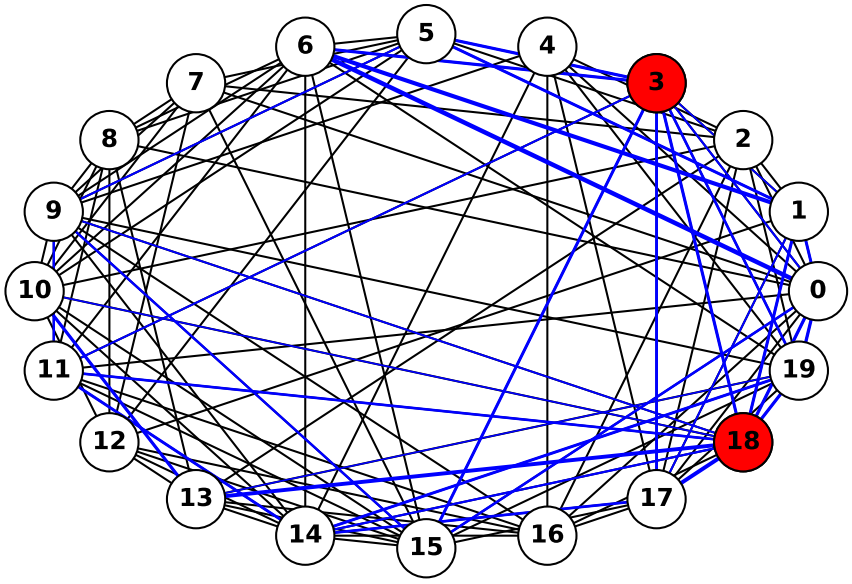

Figura 2: Grafo con rutas de flujo máimo.

`degree_centrality()`

`clustering()`

Este algoritmo que recibe como parámetro un grafo que puede ser con pesos en sus aristas o sin pesos; el caso que se tiene aristas con pesos, devuelve el promedio geométrico de los pesos del subgrafo.

`closeness_centrality()`

Mediante este algoritmo se obtiene una medida de centralidad de la red, es decir, para cada nodo se calcula que tan central es; entre más al centro esté el nodo está mas cerca de todos los demás nodos.

`load_centrality()`

Una vez que se obtienen todas las rutas mas cortas, el dato que ofrece este algoritmo es la fracción de todas las rutas mas cortas que pasan a través de ese nodo.

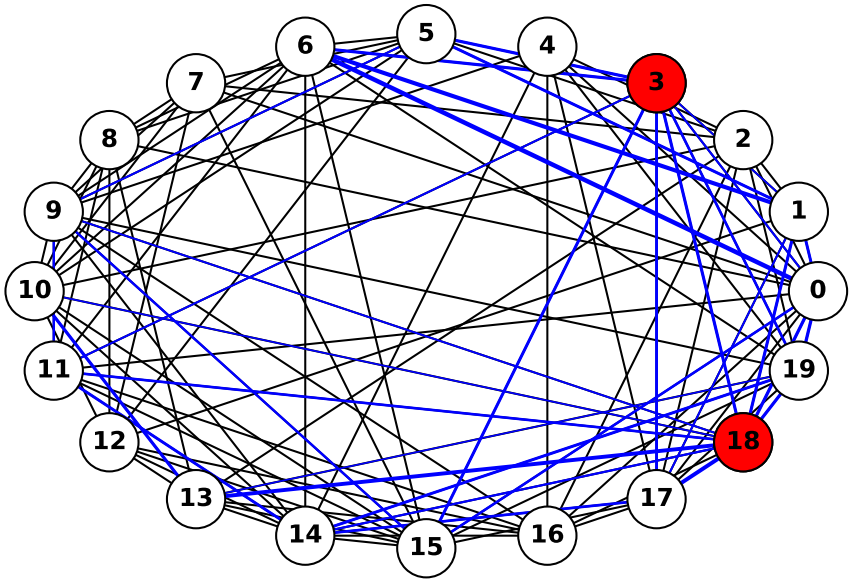

Figura 3: Grafo con rutas de flujo máimo.

`eccentricity()`

La información que aporta este algoritmo es para determinar que tan cerca o tan lejos está un nodo con respecto a los demás.

`pagerank()`

Este algoritmo calcula una clasificación de los nodos en el gráfico de acuerdo a las características de los nodos con quien esta conectado.

2. Metodología

El interés de este estudio es determinar qué características de los nodos influyen en un mayor valor de flujo máximo. Lo que se hace es generar un grafo de diferentes tamaños, desde veinte nodos hasta sesenta nodos con incrementos de diez nodos, una vez que el grafo ya tiene capacidad se eligen los pares de nodos fuente y sumidero, se calcula el valor del flujo máximo y se guarda las características que tienen éstos nodos.

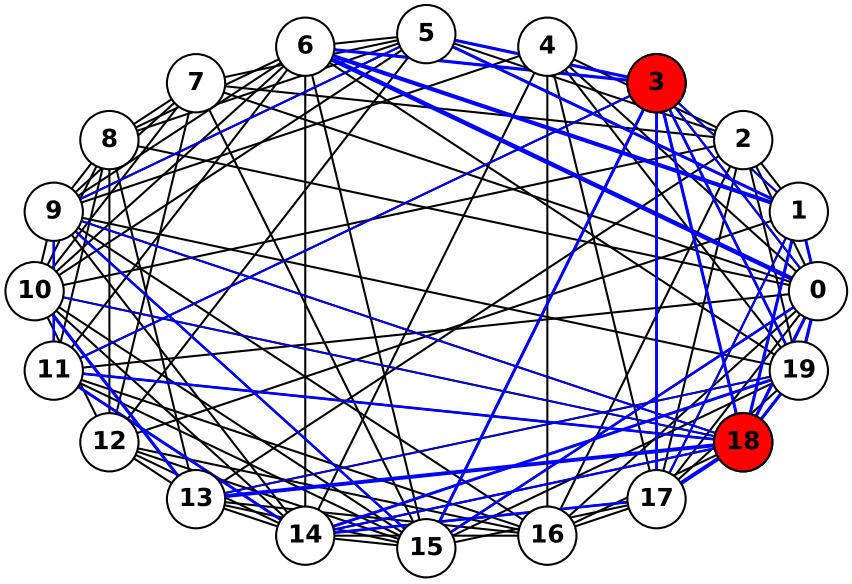

Figura 4: Grafo con rutas de flujo máimo.

La implementación en python se lleva a cabo con el código siguiente.

```
#-------------------Experimentacion-------------------------
datos=np.arange(sum(orden)*15, dtype=float).reshape(sum(orden),15)
fila=0
for i in range(len(orden)):
    H=nx.watts_strogatz_graph(orden[i], int(orden[i]/2) , 0.33 ,
    seed=None)
    initial=0
    final=0
    lista=[]
    lista[:]=H.edges
    width=np.arange(len(lista)*1,dtype=float).reshape(len(lista),1)
    for r in range(len(lista)):
        R=np.random.normal(loc=20, scale=5.0, size=None)
        width[r]=R
        H.add_edge(lista[r][0],lista[r][1],capacity=R)
    for w in range(orden[i]):
        initial=random.randint(0,round(len(H.nodes)/2))
        final=random.randint(initial,len(H.nodes)-2)
        while initial==final:
            initial=random.randint(0,round(len(H.nodes)/2))
            final=random.randint(initial,len(H.nodes)-2)
        tiempo_inicial=time()
        T=nx.maximum_flow(H, initial, final)
        tiempo_final=time()
        datos[fila,0]=T[0]
```

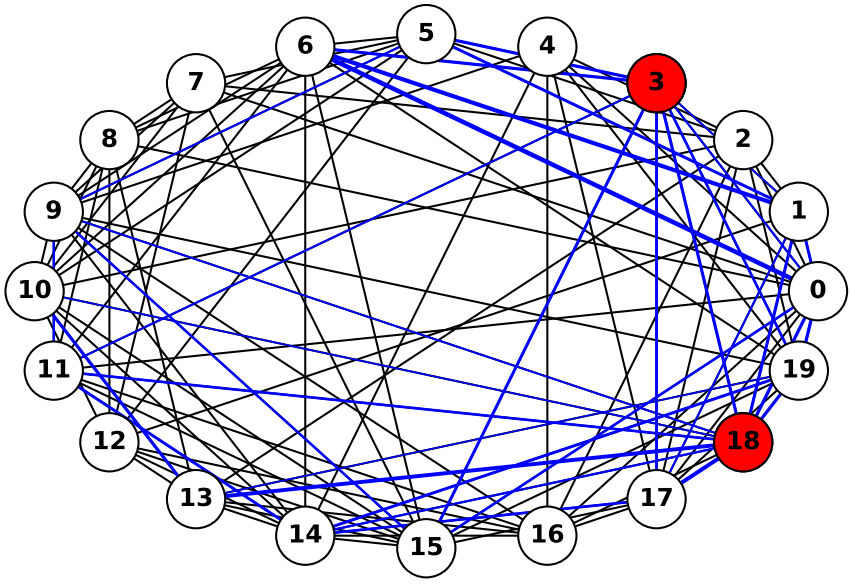

Figura 5: Grafo con rutas de flujo máimo.

```
25          datos[fila,1]=tiempo_final-tiempo_inicial
26 #--------------------Info Fuente-----------------------------------
27          datos[fila,2]=nx.degree_centrality(H)[initial]
28          datos[fila,3]=nx.clustering(H, nodes=initial)
29          datos[fila,4]=nx.closeness_centrality(H, u=initial)
30          datos[fila,5]=nx.load_centrality(H, v=initial)
31          datos[fila,6]=nx.eccentricity(H, v=initial)
32          datos[fila,7]=nx.pagerank(H, alpha=0.9, weight='weight')[
    initial]
33 #--------------------Info Sumidero---------------------------------
34          datos[fila,8]=nx.degree_centrality(H)[final]
35          datos[fila,9]=nx.clustering(H, nodes=final)
36          datos[fila,10]=nx.closeness_centrality(H, u=final)
37          datos[fila,11]=nx.load_centrality(H, v=final)
38          datos[fila,12]=nx.eccentricity(H, v=final)
39          datos[fila,13]=nx.pagerank(H, alpha=0.9, weight='weight')[
    final]
40          datos[fila,14]=orden[i]
41          fila+=1
```

caracterizacion.py

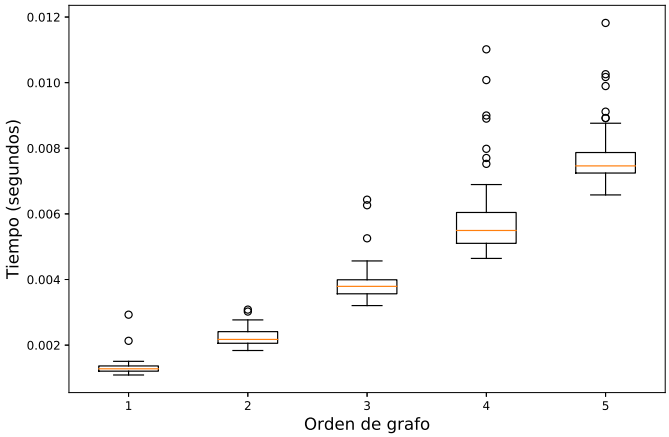

Figura 6: Diagrama de caja de tiempo.

3. Resultados

Se elaboró unos diagramas de caja para ver el comportamiento de las características de los nodos fuente y sumidero 6, 7, 8, 9, 10, 11, 12, 13, 14. Los valores del eje de las abscisas representan el orden de los grafos.

Con esa información se hace una matriz de correlación 15 donde el cero es el valor del flujo máximo, el uno es el valor del tiempo, la fila cinco es el valor de load centrality del nodo fuente, el siete representa el pagerank del nodo fuente, las filas once y trece es el valor de load centrality y pagerank del nodo sumidero respectivamente.

Las filas y columnas en negro, es debido a que representan la excentricidad y su valor permanece constante para todos los grafos.

Lo que nos muestra 15 es que el valor de load centrality y el pagerank, tanto del nodo fuente como del sumidero afectan inversamente al valor del flujo máximo.

Luego, se hizo un ANOVA para identificar si las medias de los conjuntos eran iguales o hay diferencias significativas en ellas. Se obtuvo los resultados que se muestra en la tabla llamada `ANOVA.txt` en el cual los valores de la tercer columna que son mayores a 0.1, sus medias son iguales y los valores menores 0.1 se rechaza la hipótesis de que las medias sean iguales.

`ANOVA.txt`

	sum_sq	df	F	PR>F
Deg_fuente	5846.313375	2.0	1.567752	2.112127e-01
Clstr_fuente	19842.361020	1.0	10.641886	1.311800e-03
Clsns_fuente	3926.661470	1.0	2.105953	1.483847e-01

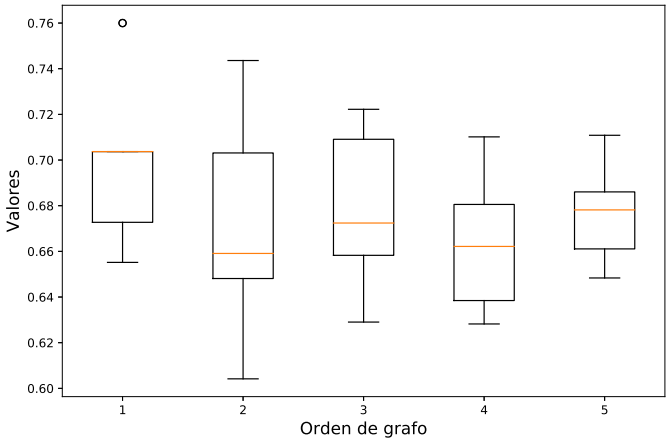

Figura 7: Diagrama de caja de centralidad de nodos fuentes.

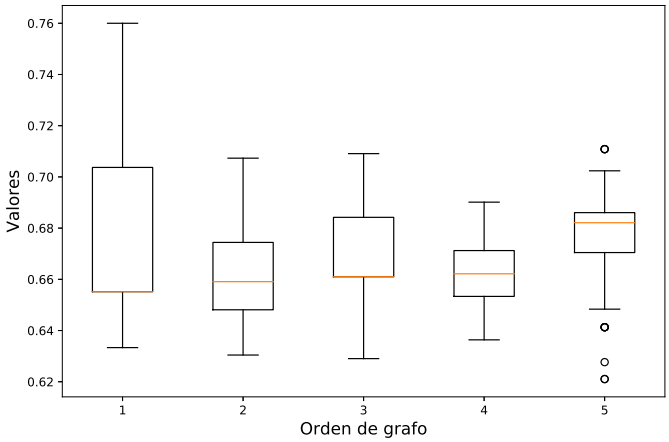

Figura 8: Diagrama de caja de centralidad de nodos sumidero.

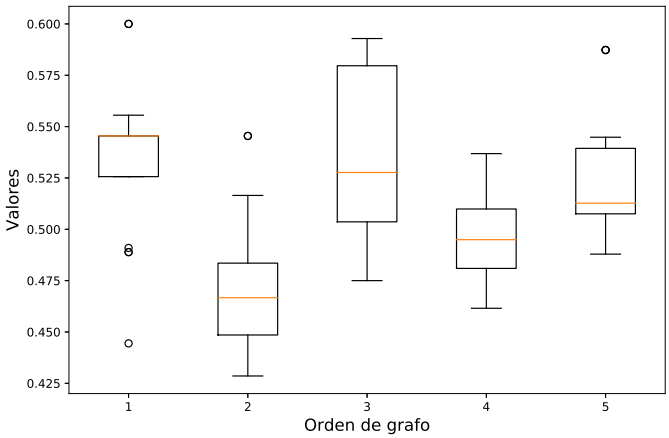

Figura 9: Diagrama de caja de clustering de nodos fuentes.

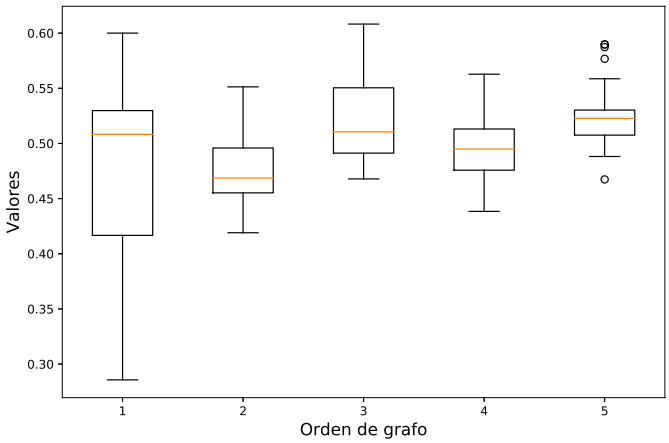

Figura 10: Diagrama de caja de clustering de nodos sumidero.

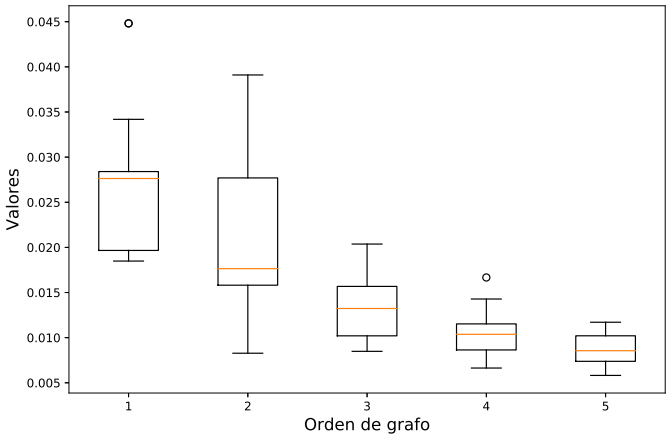

Figura 11: Diagrama de caja de load centrality de nodos fuentes.

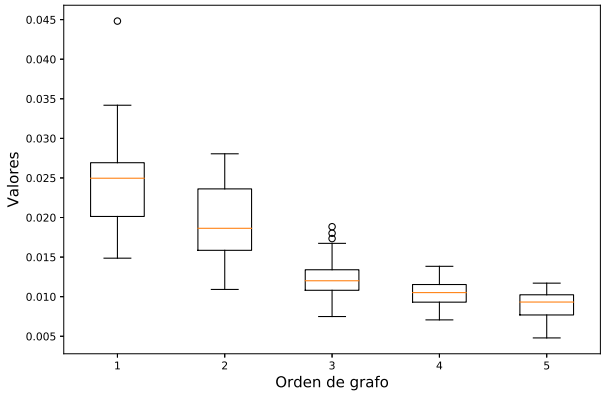

Figura 12: Diagrama de caja de load centrality de nodos sumidero.

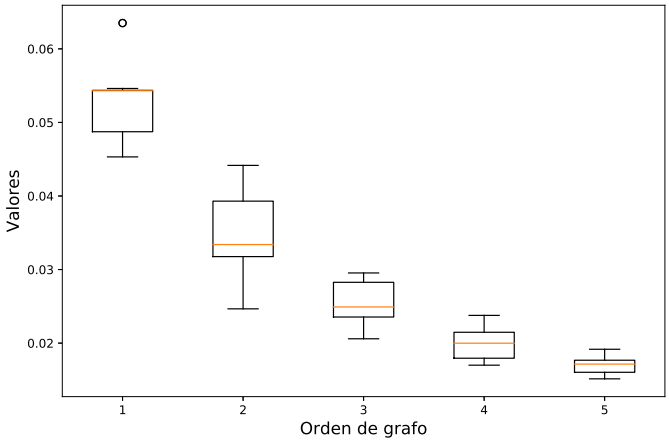

Figura 13: Diagrama de caja de pagerank de nodos fuentes.

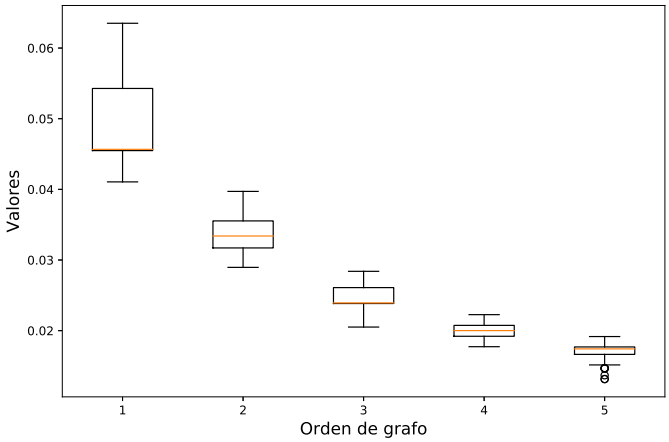

Figura 14: Diagrama de caja de load pagerank de nodos sumidero.

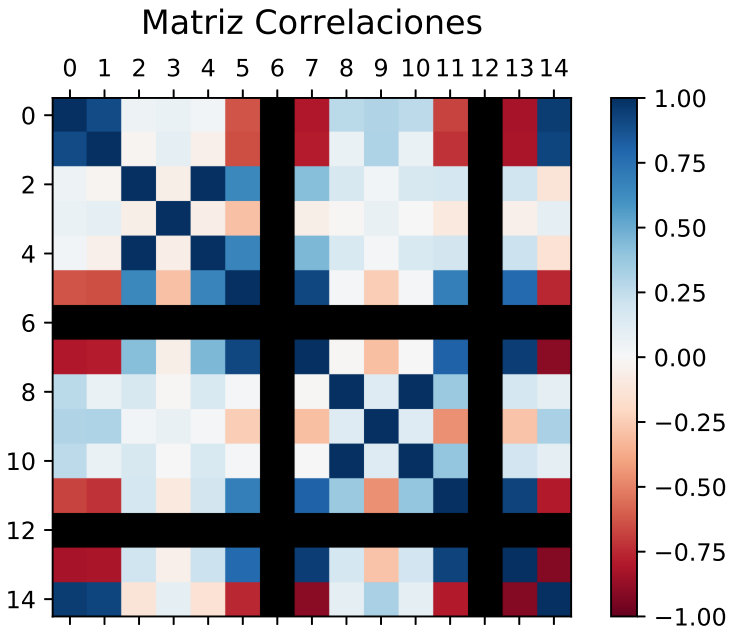

Figura 15: Correlaciones de las características de los nodos fuente y sumidero.

Load_fuente	33331.798014	1.0	17.876562	3.667508e-05
Ex_fuente	33.471134	1.0	0.017951	8.935590e-01
Prank_fuente	29028.620403	1.0	15.568675	1.120869e-04
Clstr_sumidero	1220.063324	1.0	0.654346	4.195802e-01
Clsns_sumidero	2692.447162	1.0	1.444017	2.309939e-01
Load_sumidero	3810.709229	1.0	2.043766	1.544818e-01
Ex_sumidero	33.471134	1.0	0.017951	8.935590e-01
Prank_sumidero	54301.746662	1.0	29.123197	2.018353e-07
Residual	352400.527109	189.0	NaN	NaN

Referencias

[1] NetworkX Developers, "Networkx documentation," 2012.

[2] J. L. Molina, "La ciencia de las redes," *Apuntes de Ciencia y Tecnología*, vol. 11, no. 1, pp. 36–42, 2004.

[3] J. M. Six and I. G. Tollis, "A framework for circular drawings of networks," in *International Symposium on Graph Drawing*. Springer, 1999, pp. 107–116.