Tarea 4

5272

1. Introducción

Para realizar el siguiente trabajo se utilizó la librería Networkx de Python y se aplicó el generador de grafo Dorogovtsev goltsev mendes para obtener un grafo de 14 vértices; los pesos de las aristas tienen una distribución normal con una media de 4 y una desviación estándar de 2.3.

La elección de este generador de grafo es que puede ser aplicado a una red de transporte capacitado, donde se desea enviar cierta cantidad de unidades de un producto desde un almacén hasta un cliente [1]. El algoritmo de acomodo que se utilizó para visualizar el grafo es diseño aleatorio, en este los vértices se diferencian por colores, el azul claro es la fuente mientras el azul oscuro es el sumidero y el color verde representa los restantes vértices; el ancho de las aristas representa la capacidad de estas y luego las que aparecen de rojo son por las que pasa el flujo y las negras son las que el flujo es 0.

2. Generador de Grafo

El generador de grafo utilizado se encuentra a continuación:

Generador Dorogovtsev-Mendes

Este generador siempre produce grafos planos, el mismo comienza creando tres vértices y aristas, formando un triángulo, para posteriormente ir incluyendo un vértice a la vez; cada vez que se agrega un vértice, se elige una arista al azar y el vértice se conecta a través de dos nuevas aristas a los dos extremos [2].

3. Algoritmo de Flujo Máximo

El algoritmo de flujo máximo utilizado se encuentra a continuación:

Flujo Máximo

Sea G un grafo, s y t nodos de G y *cap* una función de capacidad no negativa en las aristas de G. El flujo que pasa de s a t debe satisfacer las restricciones de capacidad, es decir el flujo sobre una arista no debe exceder su capacidad. En un flujo máximo, el flujo de salida de s es máximo entre todos los flujos desde s a t [1].

4. Resultados Computacionales

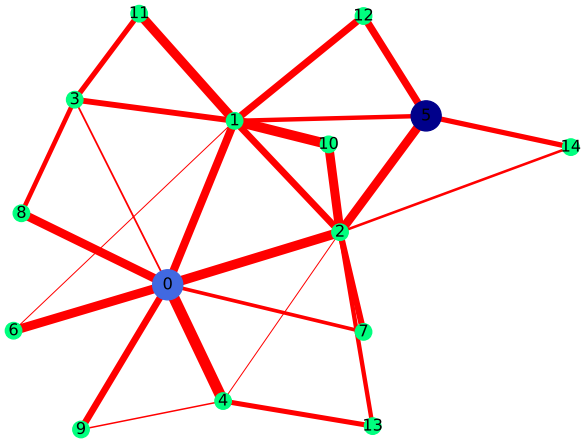
Los grafos que se muestran en la *Figura 1* representan cada una de las instancias de las 5 que se utilizaron, donde se pueden observar las fuentes y los sumideros, además se puede decir que hay instancias como la 1 y la 2 donde pasa flujo por todas sus aristas. La relación de fuentes y sumidero para cada instancia se muestra a continuación:

Instancia 1 $\Longrightarrow$ G: 0, 5
Instancia 2 $\Longrightarrow$ G: 1, 5
Instancia 3 $\Longrightarrow$ G: 0, 8
Instancia 4 $\Longrightarrow$ G: 3, 5
Instancia 5 $\Longrightarrow$ G: 5, 0

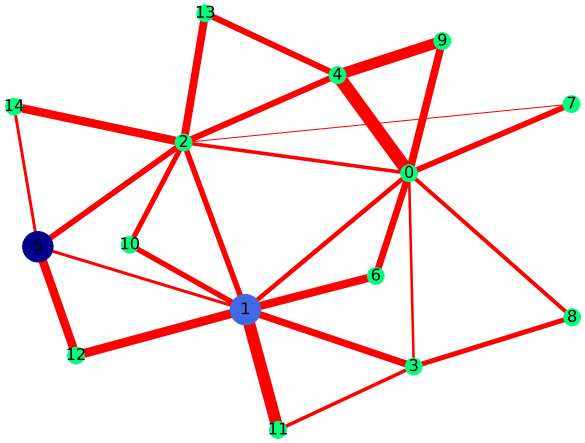
Se utilizaron las siguientes características estructurales para cada grafo visualizado:

- Distribución de grado.
- Coeficiente de agrupamiento.

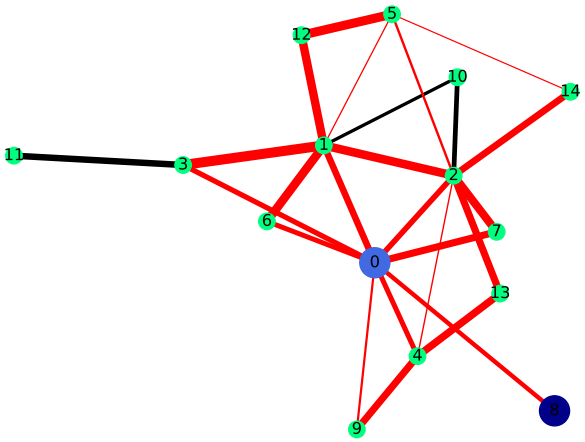
- Centralidad de cercanía.
- Centralidad de carga.
- Excentricidad.
- Rango de página.



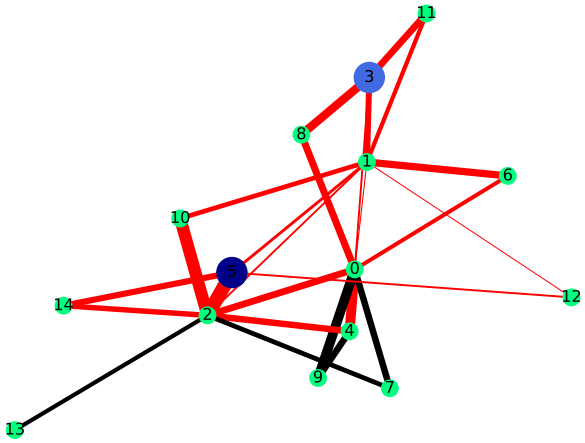
(a) Instancia 1



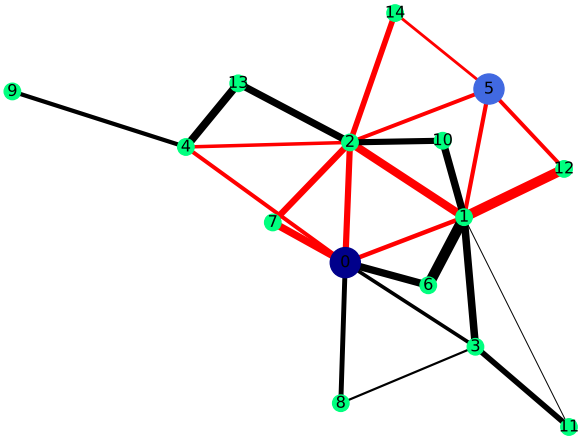
(b) Instancia 2



(c) Instancia 3



(d) Instancia 4



(e) Instancia 5

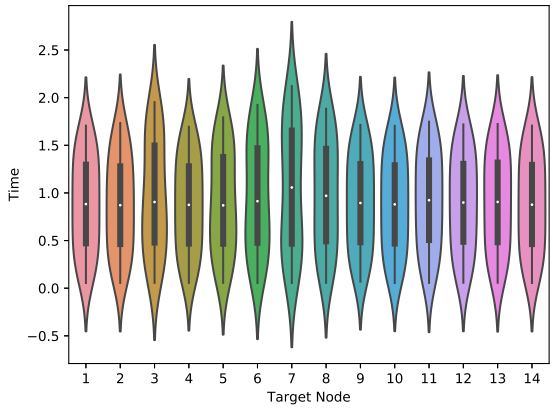
Figura 1: Instancias

Se analizó para cada instancia cuáles eran los vértices que constituyen buenas fuentes y los que son mejores sumideros *Cuadro 1*, los resultados se muestran a continuación:

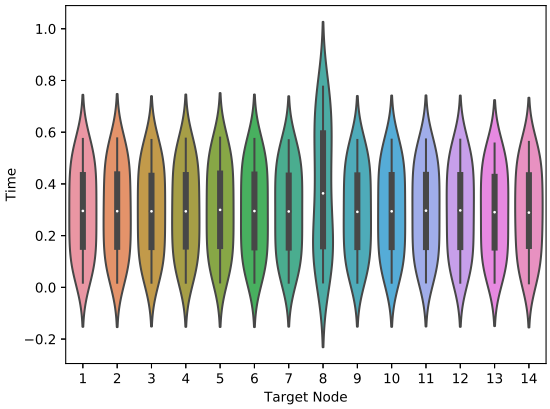
	Instancias				
	1	**2**	**3**	**4**	**5**
Coeficiente de Agrupamiento	4, 2	2, 1	2, 0	0, 12	6, 2
Centralidad de cercanía	1, 0	2, 4	2, 1	2, 1	8, 1
Distribución de grado	2, 1	2, 0	14, 1	1,3	2,5
Centralidad de carga	2, 0	2, 1	1, 0	13, 0	3, 7
Excentricidad	5, 2	0, 1	2, 5	5, 7	2, 1
Rango de página	2, 13	3, 6	3, 1	1, 0	0,9

Cuadro 1: Mejores fuente, sumideros

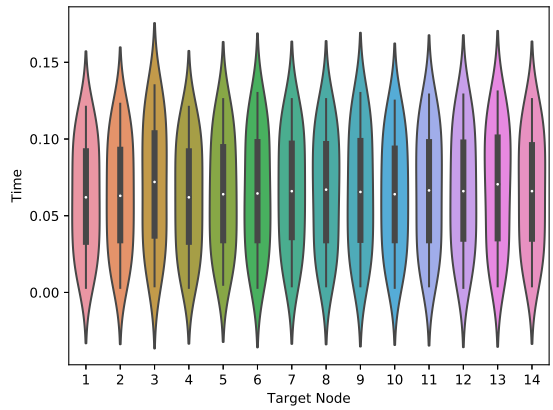
Se realiza una evaluación de como las características de los vértices que afectan a los tiempos de ejecución de los algoritmos seleccionados, en primer lugar se realizó el análisis para la instancia 1 *Figura 2* donde se puede decir que para las características estructurales Centralidad de cercanía y Coeficiente de agrupamiento los nodos tienes comportamientos similares, donde resalta el nodo 7 dado que tiene una duración de tiempo más grande, en el mismo los nodos 7 y 8 son los que poseen mayores funciones objetivos con una valor de 18.43 y 15.44 unidades de flujo para cada estructura respectivamente además que poseen las medias mas elevadas con un valor de 1.23 y 1.35. Para las demás estructuras los nodos tienen un comportamiento similar.



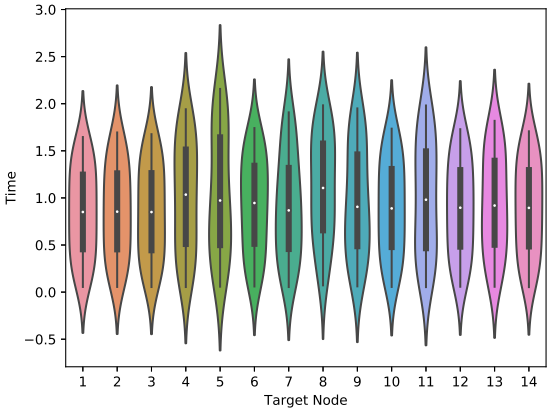
(a) Centralidad de cercanía



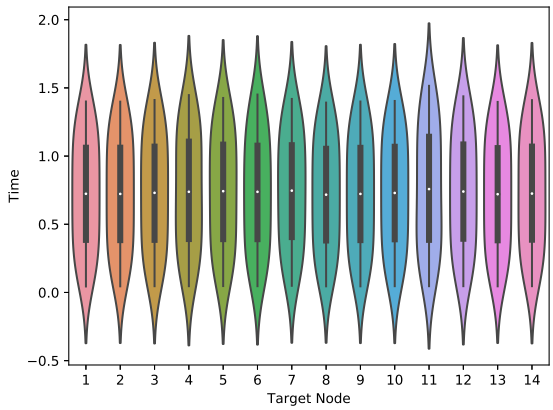
(b) Coeficiente de agrupamiento



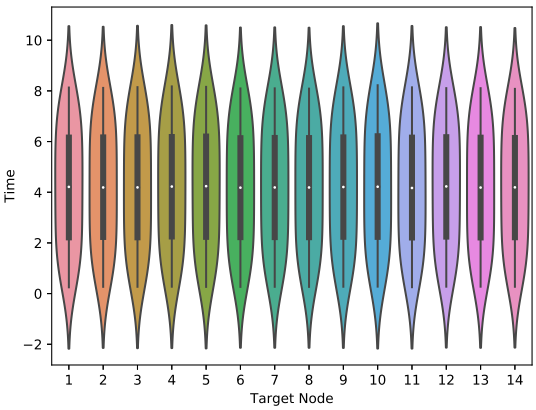
(c) Distribución de grado



(d) Excentricidad



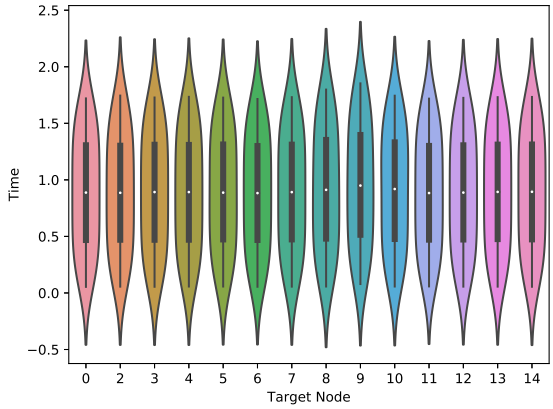
(e) Centralidad de carga



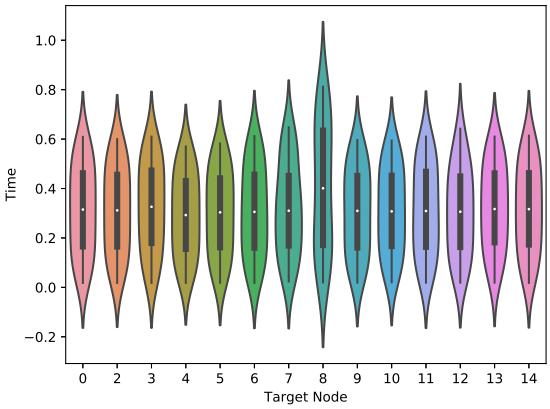
(f) Rango de página

Figura 2: Gráfico de violín: Instancia 1

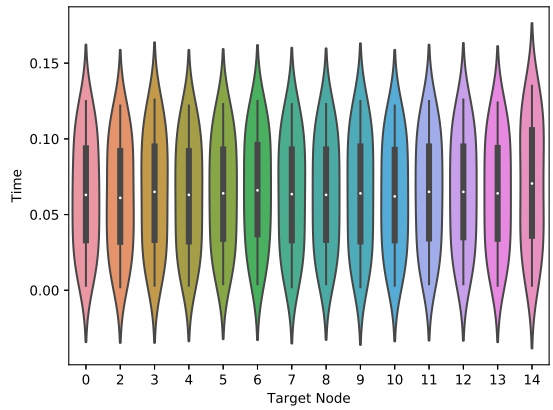
Para los nodos que pertenecen a la instancia 2 como se observa *Figura 2*, en la estructura de Rango de página es donde mayor se ve afectado el tiempo de ejecución, mientras que en la gráfica b el nodo 8 resalta sobre los demás con un mayor tiempo de ejecución el mismo presenta la mayor función objetivo con una valor de 21.28 unidades de flujo, las medias para estas estructuras son muy similares las cuales oscilan alrededor de 0.8.



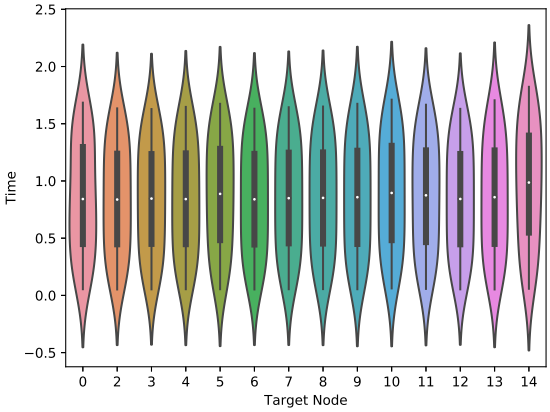
(a) Centralidad de cercanía



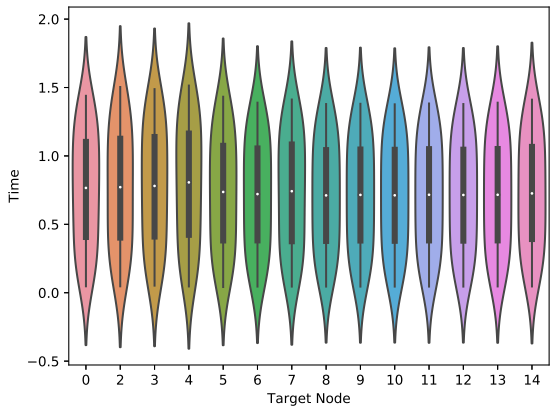
(b) Coeficiente de agrupamiento



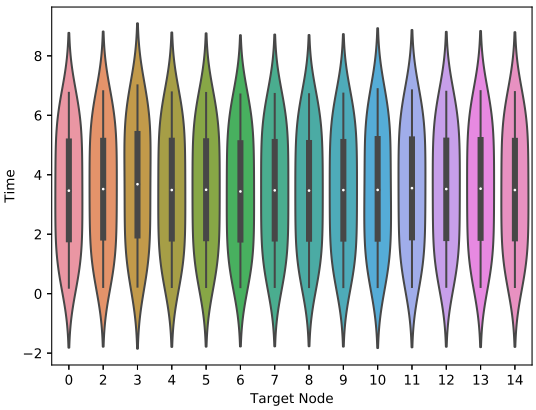
(c) Distribución de grado



(d) Excentricidad



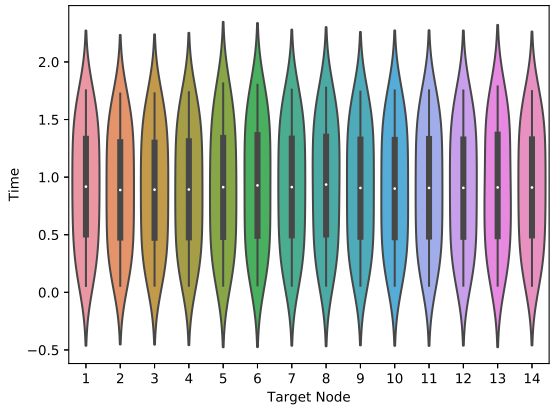
(e) Centralidad de carga



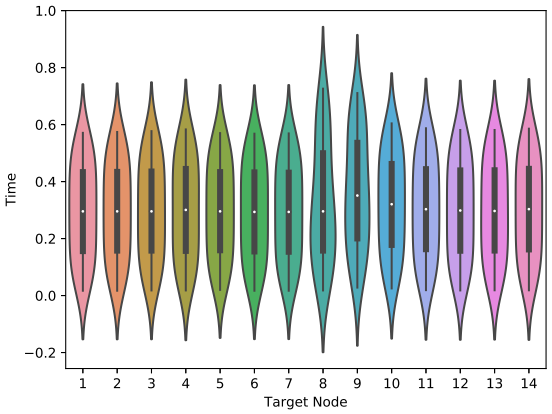
(f) Rango de página

Figura 3: Gráfico de violín: Instancia 2

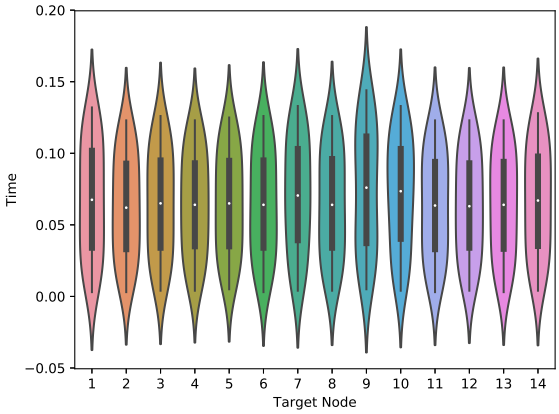
Al analizar la *Figura 3* se puede decir que al igual que la instancia anterior esta tiene un comportamiento similar, pero se puede destacar que la estructura que menos afecta el tiempo es Coeficiente de agrupamiento en esta la mayor función objetivo se encuentra en el nodo 8 con un valor de 21.28 unidades de flujo, las medias oscilan en el valor 0.3.



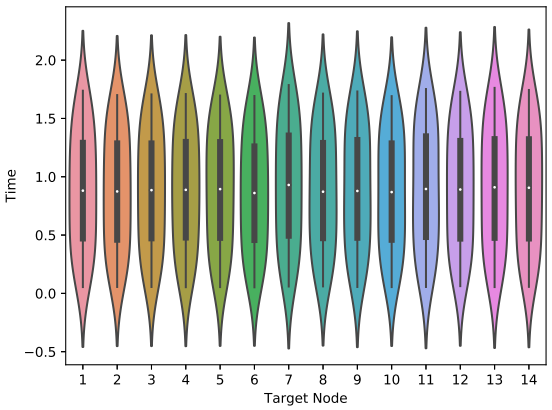
(a) Centralidad de cercanía



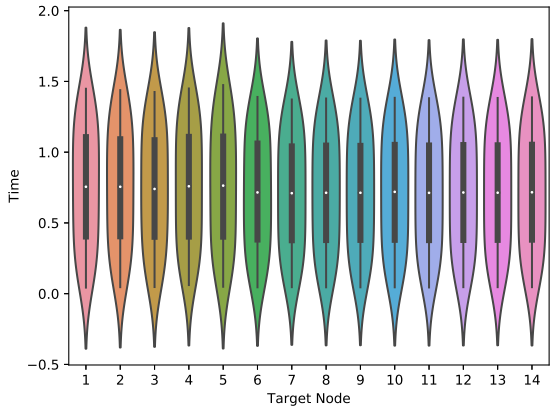
(b) Coeficiente de agrupamiento



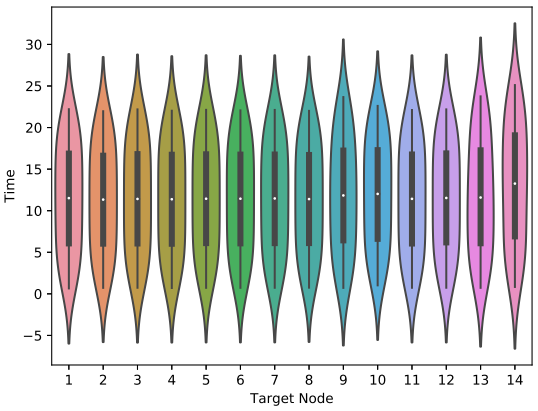
(c) Distribución de grado



(d) Excentricidad



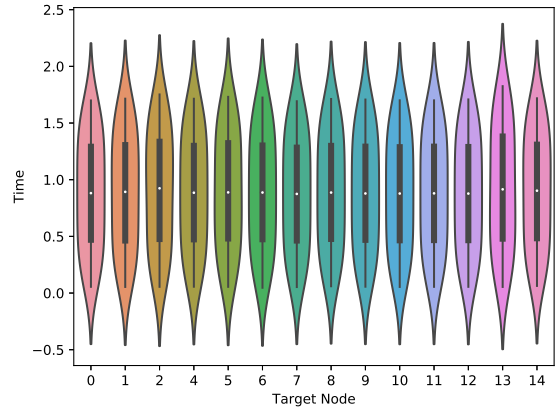
(e) Centralidad de carga



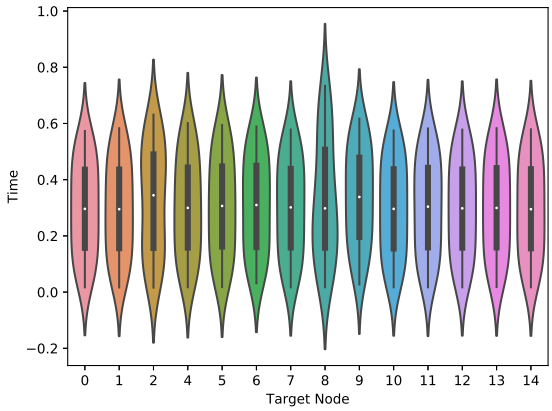
(f) Rango de página

Figura 4: Gráfico de violín: Instancia 3

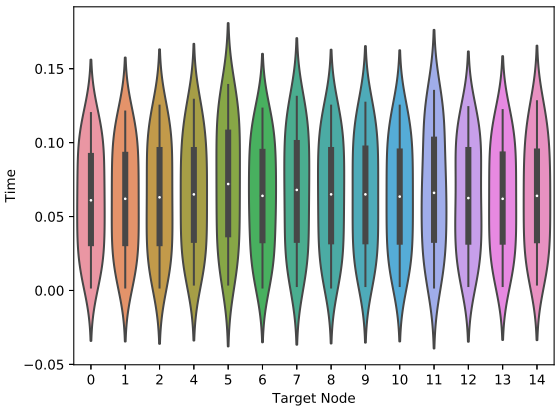
La cuarta instancia *Figura 4* el mayor valor de la función objetivo se encuentra en la característica estructural Centralidad de carga con un valor de 15.45 unidades de flujo y en este caso los mayores tiempos se encuentran en Centralidad de cercanía, las medias tienen un valor entre los 0.7 y los 0.9.



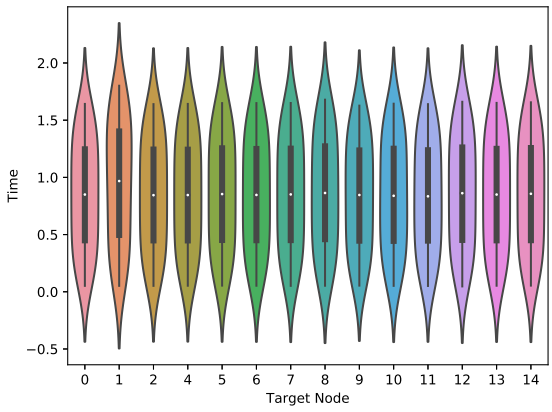
(a) Centralidad de cercanía



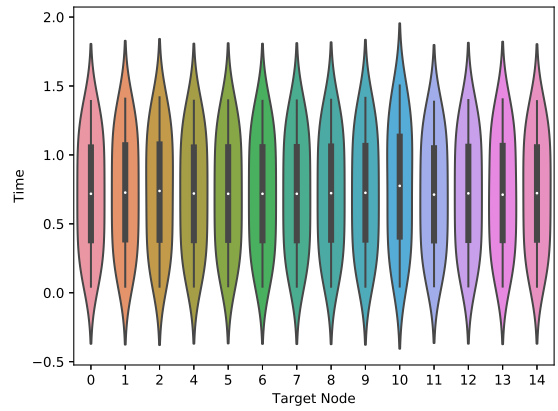
(b) Coeficiente de agrupamiento



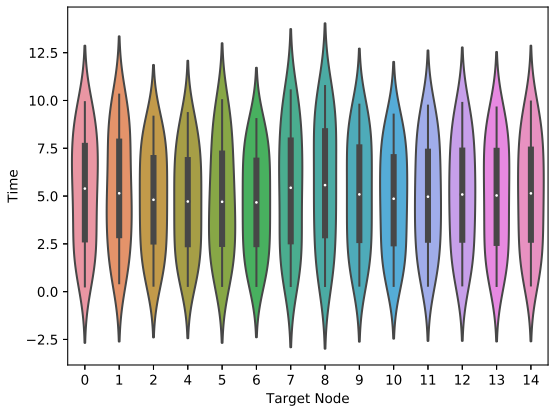
(c) Distribución de grado



(d) Excentricidad



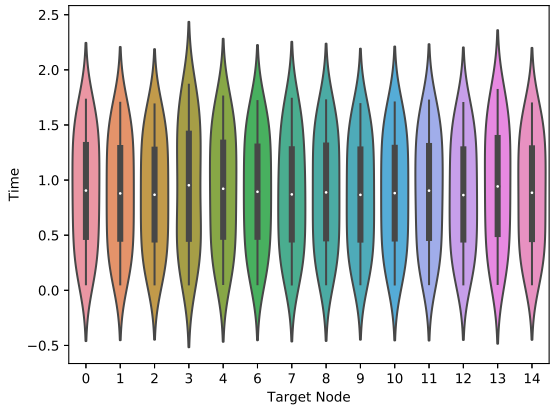
(e) Centralidad de carga



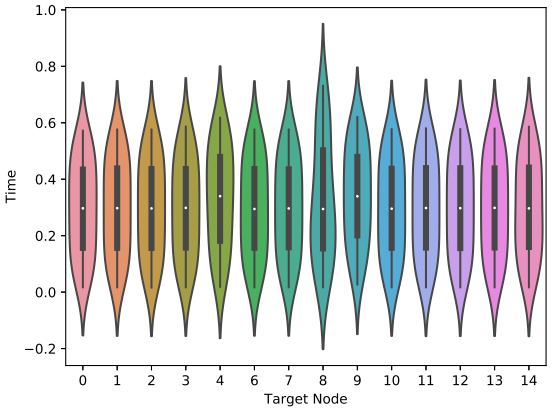
(f) Rango de página

Figura 5: Gráfico de violín: Instancia 4

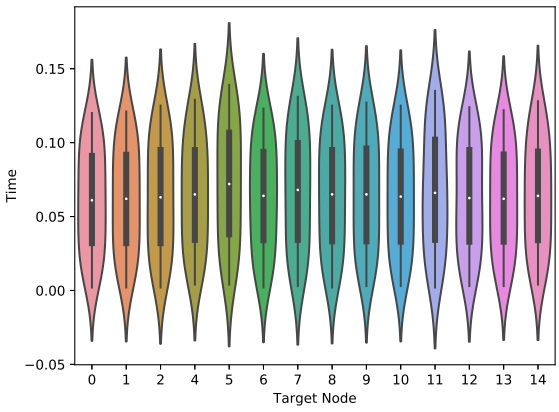
Por último se analizó la instancia 5 *Figura 5*, en donde los menores tiempos de ejecución se encuentran en Coeficiente de agrupamiento, el mayor valor de la función objetivo se encuentra en Rango de página con un valor de 18.17 unidades de flujo, mientras que las medias para cada una de las características vistas se encuentran alrededor del mismo valor.



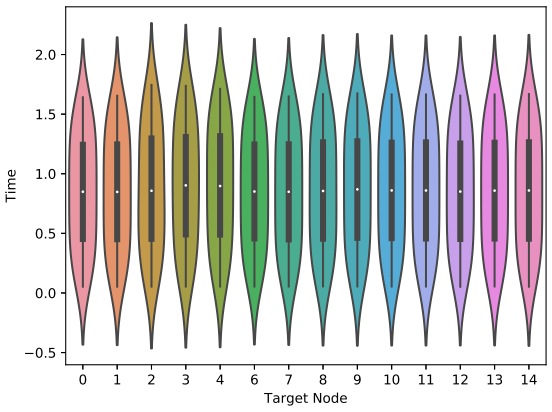
(a) Centralidad de cercanía



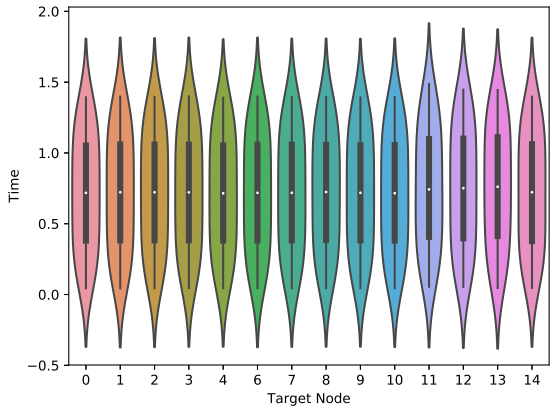
(b) Coeficiente de agrupamiento



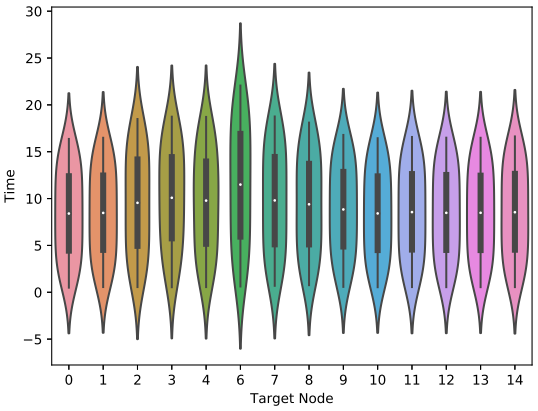
(c) Distribución de grado



(d) Excentricidad



(e) Centralidad de carga



(f) Rango de página

Figura 6: Gráfico de violín: Instancia 5

Se puede concluir que el algoritmo que más afecta al tiempo de ejecución es el Rango de página, mientras que el mayor valor para la función objetivo de todas las características lo presenta Coeficiente de agrupamiento con un valor de 21.28 unidades de flujo, mientras que la que más afecta es la característica estructural Excentricidad con el valor más bajo.

5. Fragmento de Código

```
weights = np.random.normal(4, 2.3, nx.number_of_edges(G))
m = 0
for t, s, p in G.edges(data=True):
    p['weight'] = weights[m]
    m += 1

st_list = []

for i in range(1):
    for j in range(1, 2):
        list_random_G = random.sample(range(len(G)), 2)
        for k in range(1, 2):
            value_dinitz = dinitz(G, list_random_G[0], list_random_G[1], capacity="weight")
            print("Fuente", list_random_G[0])
            print("Sumidero", list_random_G[1])
            nodos_fuente=[]
            nodos_fuente.append(list_random_G[0])
            nodos_sumidero = []
            nodos_sumidero.append(list_random_G[1])

            color_map = []
            node_T = []
            for node in G:
                if node == list_random_G[0]:
                    color_map.append('royalblue')
                    node_T.append(500)
                else:
                    if node == list_random_G[1]:
                        color_map.append('darkblue')
                        node_T.append(500)
                    else:
                        color_map.append('springgreen')
                        node_T.append(150)

    max_flujo, max_dic = nx.maximum_flow(G, list_random_G[0], list_random_G[1],
                                         capacity='weight')
    print("Diccionario", max_dic)
    print("Flujo Maximo", max_flujo)
    edge_colors = ['black' if max_dic[i][j] == 0 and max_dic[j][i] == 0 else 'red'
                   for i, j in G.edges]

    pos = nx.random_layout(G)

    nx.draw(G, node_color=color_map, edge_color=edge_colors, node_size=node_T,
            width=weights, with_labels=True) #Se dibuja el grafo
```

new.py

Referencias

[1] Ravindra K Ahuja and James B Orlin. A fast and simple algorithm for the maximum flow problem. *Operations Research*, 37(5):748–759, 1989.

[2] Abdelmalik Moujahid and Blanca Rosa Cases Gutiérrez. Analysis of spanish text-thesaurus as a complex network. 2014.