Caracterización estructural de instancias

Mario Alberto Gutiérrez Carrales
1549273

30 de abril de 2019

Introducción

La presente práctica consiste en llevar a cabo cinco aplicaciones basadas en un caso de la vida real que sean consideradas como instancias del problema de máximo flujo de redes. Se analiza cada caso por separado realizando diferentes comparaciones de los elementos de la instancia para caracterizar a aquellos que la hacen hacen más efectiva.

Los principales objetos de estudio que se consideran son los tiempos de ejecución y valores de flujo máximo ya que son quienes determinan la eficiencia de la instancia, los siguientes objetos con mayor importancia son los elementos del grafo, por ejemplo como los nodos y aristas así como la capacidad en las aristas y la cantidad que fluye entre ellas.

El lenguaje de programación que se utiliza es `python` [1] y una de las principales librerías que es necesaria para esta práctica es `NetworkX` [2] con funciones especiales para grafos, en especial para crear, acomodar y resolver problemas de optimización en redes utilizando distintos algoritmos.

1. Ambiente computacional

Los experimentos realizados en esta práctica se realizaron en una PC con un sistema operativo de 64 bits, procesador AMD A9-9410 Radeon R5, 2 núcleos + 3G con 2.90 GHz de velocidad y 12 GB de memoria RAM.

Para llevar a cabo la experimentación en `python` y para poder utilizar las funciones requeridas se necesitan las siguientes librerías:

1. `NetworkX`. Como se mencionó con anterioridad, esta librería es de gran ayuda para la manipulación de grafos.
2. `Matplotlib` [3]. Esta librería contiene funciones que sirven para visualizar gráficas y guardar imágenes en distintos formatos, en particular el *eps*.
3. `Time` [4]. Contiene la función que determina el tiempo (en segundos) que uno o varios procesos tardan, dicho tiempo se calcula con una diferencia entre el tiempo inicial y el final.
4. `Pandas` [5]. Una de las funciones con las que cuenta esta librería es la lectura de archivos con extensión *csv* y la manipulación de *data frames* facilitando el uso de tablas.
5. `Seaborn` [6]. Se utiliza para la creación del diagrama de caja y bigotes.

Dichas librerías se importan al entorno de `python` tal como se muestra en el código 1.

Código 1: *Cargar librerías en* ***python***

```
1 import matplotlib.pyplot as plt
2 import networkx as nx
3 import seaborn as sns
4 import pandas as pd
5 import time
```

Para guardar imágenes en formato eps se utiliza el comando que proporciona el fragmento de código 2.

Código 2: *Guardar imágenes en formato eps*

```
1    plt.savefig("CA_1.eps")
```

Para evitar que los bordes de una imagen se recorten al momento de guardarla se le aplica el proceso que se muestra en el código 3.

Código 3: *Aumentar bordes para evitar cortes en la imagen*

```
1        plot_margin = 0.05
2        x0, x1, y0, y1 = plt.axis()
3        plt.axis((x0 - plot_margin,x1 + plot_margin,y0 -
    plot_margin,y1 + plot_margin))
```

2. Algoritmos para características de nodos

1. `degree_centrality`. La centralidad del grado para un nodo v es la fracción de nodos a los que está conectado. Recibe como único parámetro un grafo.

2. `closeness_centrality`. Este algoritmo calcula la centralidad de cercanía de los vértices, tomando el recíproco del promedio de las distancias más cortas y normalizando. Recibe como parámetro principal un grafo y un parámetro opcional que es la normalización de los valores.

3. `clustering`. Lo que mide este algoritmo es el nivel de agrupamiento que existe entorno a un vértice, que se calcula como el número total de aristas que conectan a los vecinos más cercanos entre el número máximo de aristas posibles entre todos los vecinos más cercanos. Este algoritmo calcula el coeficiente de agrupamiento de los vértices y recibe como parámetro un grafo.

4. `load_centrality`. Es la fracción de todas las rutas más cortas que pasan a través de ese nodo. Recibe como parámetro principal un grafo y un parámetro opcional que es la normalización de los valores.

5. `eccentricity`. La excentricidad de un nodo v es la distancia máxima de v a todos los demás nodos. Retorna la excentricidad de los nodos de un grafo y recibe como parámetros un grafo.

6. `pagerank`. El algoritmo pagerank calcula la probabilidad de que si uno navega las aristas del grafo aleatoriamente, vaya a dar a un nodo específico. Recibe un grafo como parámetro.

*Todos los valores retornados son guardados en una variable tipo diccionario en el que las claves son los nodos y los valores son los números que retorna el algoritmo. En el código 4 se muestra un ejemplo de como guardar dichos valores en un arreglo.

Código 4: *Guardar los valores de los algoritmos en un arreglo*

```
1 V1=[i for i in nx.degree_centrality(G).values()]
2 V2=[i for i in nx.closeness_centrality(G).values()]
3 V3=[i for i in nx.clustering(G).values()]
4 V4=[i for i in nx.load_centrality(G).values()]
5 V5=[i for i in nx.eccentricity(G).values()]
6 V6=[i for i in nx.pagerank(G).values()]
```

3. Metodología

Primero se elige una aplicación real, luego ciertos algoritmos para trabajar dicha aplicación y analizar diferentes variaciones. Para la construcción de cada instancia se toman en cuenta las siguientes características:

Aplicación. Un *sistema de tuberías* es un conducto que cumple la función de transportar agua u otros fluidos [7], en este caso supondremos que las aplicaciones son enfocadas en flujo de agua. Esta aplicación fue seleccionada debido a la gran importancia de los sistemas de tuberías en la industria y también por la eficiencia de un recurso vital como el agua, dependiendo de la situación dicho sistema debe cumplir con ciertas características físicas.

Algoritmo generador. *Barabasi_albert_graph* que toma dos parámetros, el primero es el número de nodos que tendrá el grafo y el segundo es el número de aristas que salen de cada nodo a otros. Se seleccionó este generador ya que gracias a su segundo parámetro se controla la cantidad de aristas presentes en el grafo y de esta manera aumentar su semejanza a una aplicación real en la que puede haber muchos nodos pero no tantas aristas.

Algoritmo de acomodo. *Kamada_kawai_layout* que toma como parámetro el grafo generado y retorna un diccionario con la la posición de cada nodo. Este algoritmo fue seleccionado gracias a que en la práctica dos de Gutiérrez [8] se realizó una comparación y debido a las características del grafo es el que mejor se adapta visualmente.

Implementación para el problema de máximo flujo. *Maximum_flow* que toma tres parámetros, el grafo que se resolvera, junto con el nodo inicial y el nodo final, esta función retorna dos objetos, un número que es el valor de máximo flujo de la instancia y también un diccionario que contiene los valores de flujo entre cada arco. A pesar de que este es un algoritmo que le toma mayor tiempo en resolver la instancia a comparación de otros el hecho de poder conocer el flujo de cada arista hace relevante esta implementación.

En el código 5 se muestra como se puede llevar a cabo el proceso de generación, acomodo y solución de instancia.

Código 5: *Proceso de solución de instancia*

```
#Generador
G=nx.barabasi_albert_graph(8,2)
#Algoritmo de acomodo
nx.draw_networkx(G,nx.kamada_kawai_layout(G),with_labels=False)
#Implementacion para el problema de maximo flujo
Max_Flujo, Flujo_Arcos = nx.maximum_flow(G, 0,3)
```

En la figura 1 se muestra el resultado del grafo generado y acomodado con los algoritmos indicados anteriormente.

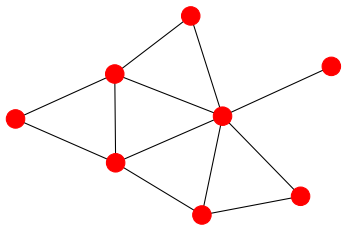


Figura 1: *Grafo con algoritmo de acomodo*

Posteriormente se lleva a cabo un análisis por instancia en el siguiente orden:

1. Visualización de instancia. Se muestran los nodos del grafo distinguiendo el inicio (rosa), final (rojo) y los intermedios (aqua). Además el grosor de la arista representa la capacidad en relación a las demás aristas.

2. Visualización de instancia con solución. Se muestra el grafo anterior añadiendo el flujo que corresponde a cada arista (celeste) preservando la idea de que el grosor del flujo indica una relación con los demás flujos, también se añade la variación de tamaño de los nodos de la instancia utilizando los algoritmos para características de nodos, esto con el fin de determinar cuales son los más relevantes (a mayor radio, mayor relevancia).

3. Visualización de instancia con solución y caracterización. Se toma el mismo gráfico anterior con la sutileza de que se le añade color a los nodos intermedios que fungen un papel importante, es decir, que si se

varía el nodo inicial o final para dicha instancia podrían ser un buen nodo inicial o final, gracias a que se mejora la función objetivo. Los nodos que son buen inicio pero no final (azul y verde limón), aquellos que son mejor nodo final pero no inicial (morado y verde limón) y por último los nodos que son tanto mejor inicio como mejor final (amarillo).

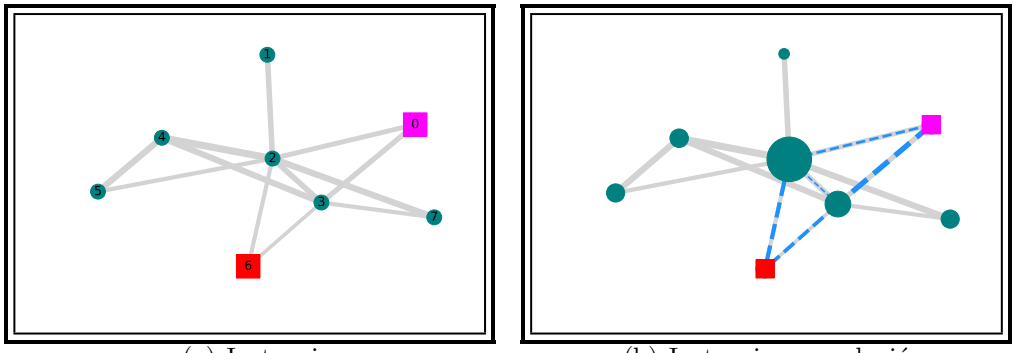
4. Discusión de las características de nodos y parejas que mejoran el flujo a la instancia.
5. Comparación del tiempo de ejecución (TE) variando el nodo inicial.
6. Comparación del tiempo de ejecución (TE) variando el nodo final.
7. Comparación tiempo de ejecución (TE) variando un nodo intermedio eliminado.
8. Discusión del mejor nodo posible, tiempos de ejecución y posibles nodos a eliminar de la instancia.

Para el caso de las visualizaciones de las instancias, solamente en la grafica 1 se muestra el número de nodo.

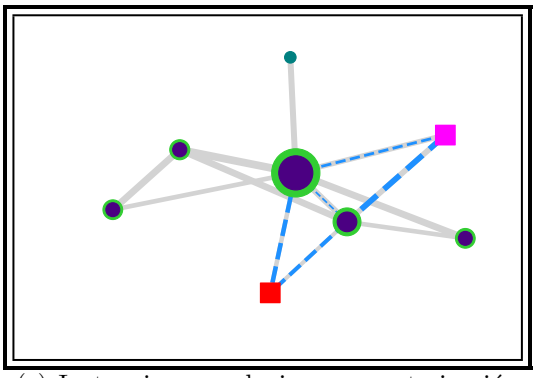
Para los casos 5, 6 y 7 se grafican los diagramas de caja y bigote identificandp los casos en los que la función objetivo mejora (rojo) y los que permanecen igual o disminuyen (azul).

Después de analizar cada instancia, se realiza una comparación global mediante un diagrama de caja para cada una de las características de los nodos correspondientes a las diferentes instancias.

Instancia 1



(a) Instancia (b) Instancia con solución



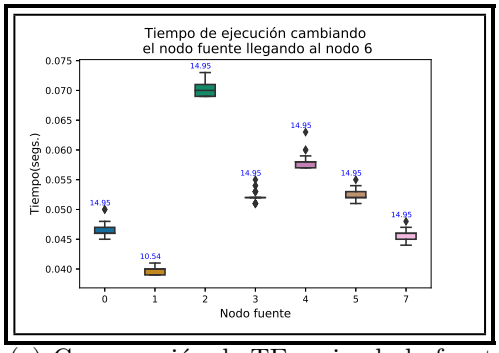
(c) Instancia con solucion y caracterización

Figura 2: *Visualización de la instancia 1.*

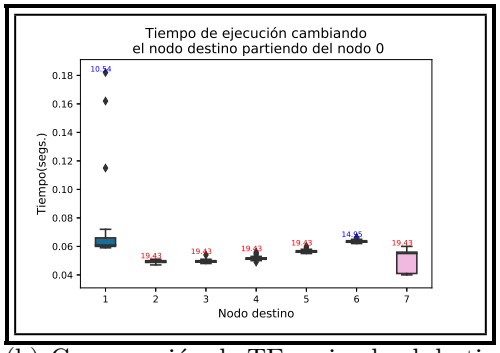
En la figura 2 se observa que del total de las aristas solo se usan 5, representando el 41.6 %.

También se muestra que los nodos 2 y 3 tiene en general mejores características que el resto y que para esta instancia los nodos 2, 3, 4, 5 y 7 son mejores nodos finales (destino) que el principal, obteniendo un aumento en la función objetivo.

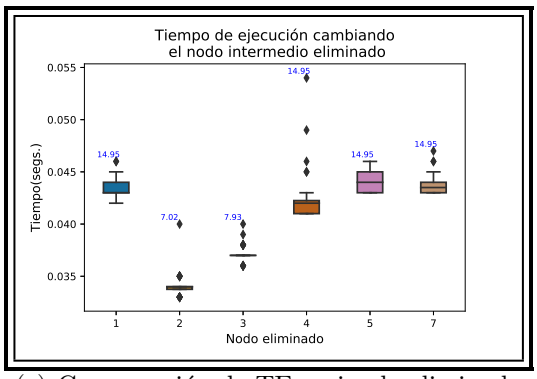
Por otro lado, si el problema se resuelve variando el nodo inicial no se obtiene mejora la función objetivo.



(a) Comparación de TE variando la fuente



(b) Comparación de TE variando el destino



(c) Comparación de TE variando eliminado

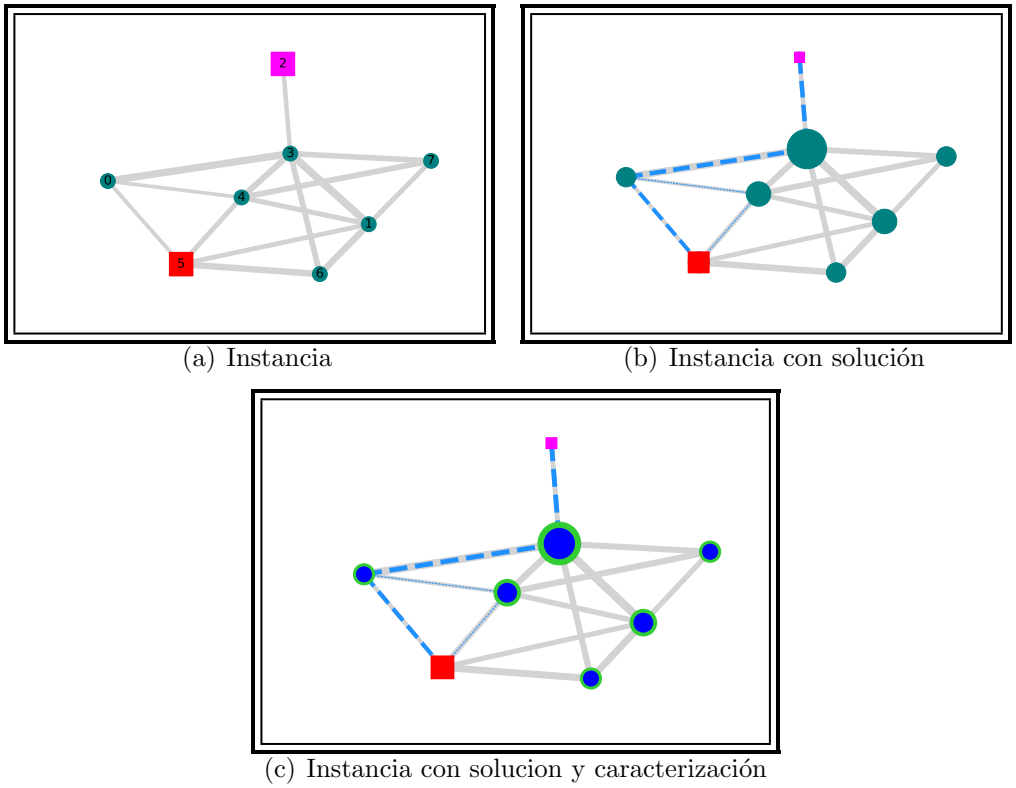
Figura 3: *Comparación del tiempo de ejecución para la instancia 1.*

En la figura 3 se observa que solamente empeora la función objetivo cuando se cambia el nodo inicial al 1, todos los demas son iguales al inicial, y los mejores en cuanto a tiempo de ejecución son el nodo 0 y el 7.

Como se mencionó en el apartado de visualización, cuando se varía el nodo final se obtienen varias mejoras, y de esas mejoras los que presentan mejores tiempos de ejecución son el nodo 2 y 7, sobretodo el 2.

Si se elimina un nodo intermedio, no se obtienen mejoras en la función objetivo, así que no conviene eliminar ningún nodo.

Instancia 2



(a) Instancia

(b) Instancia con solución

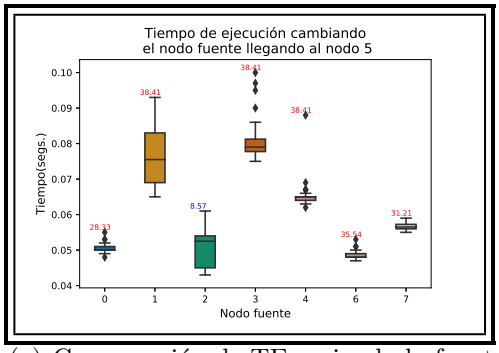
(c) Instancia con solucion y caracterización

Figura 4: *Visualización de la instancia 2.*

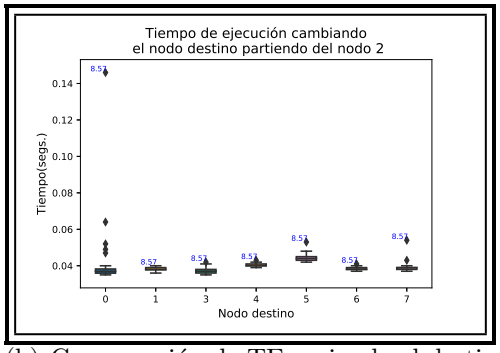
En la figura 4 se observa que del total de las aristas solo se usan 5, representando el 33.3 %.

También se muestra que los nodos 1, 3 y 4 tiene en general mejores características que el resto y que para esta instancia todos los nodos intermedios son mejores nodos iniciales (fuente) que el principal, obteniendo un aumento en la función objetivo.

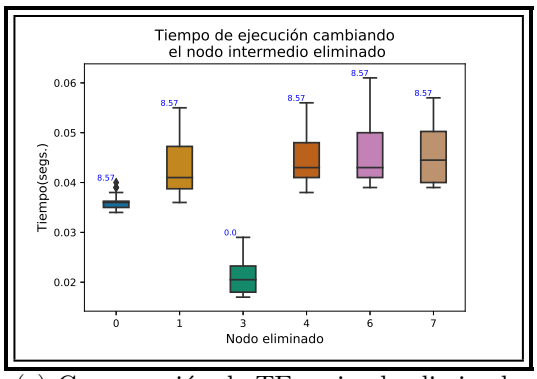
Por otro lado, si el problema se resuelve variando el nodo final no se obtiene mejora la función objetivo.



(a) Comparación de TE variando la fuente



(b) Comparación de TE variando el destino



(c) Comparación de TE variando eliminado

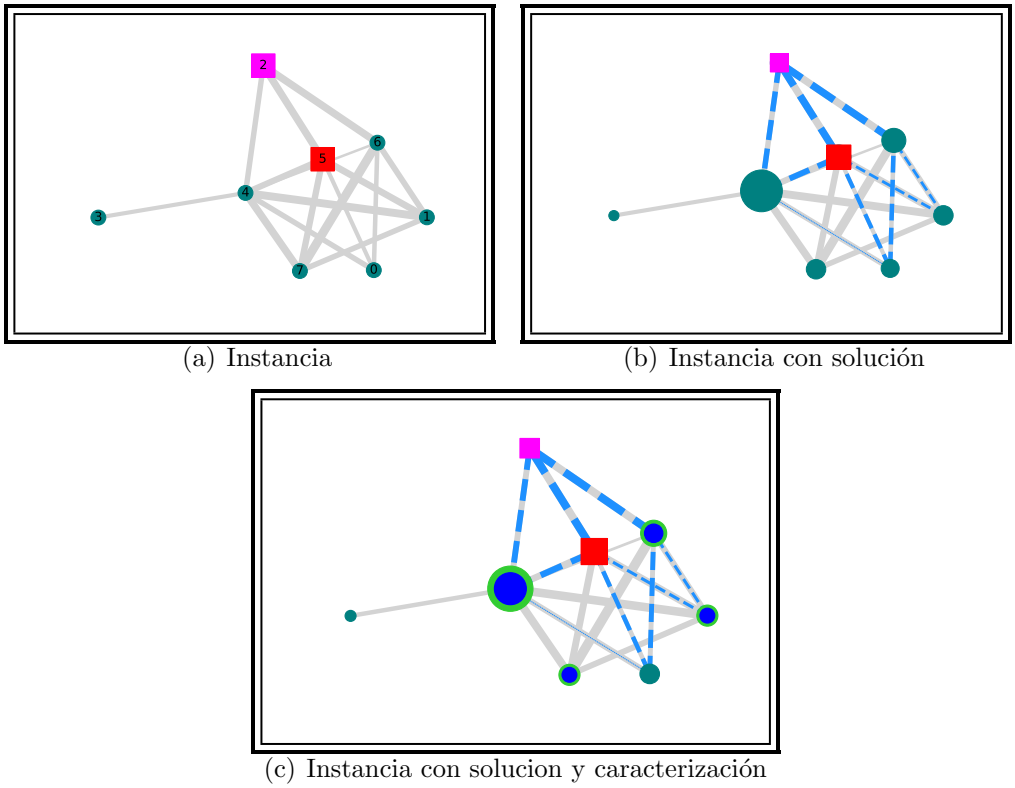
Figura 5: *Comparación del tiempo de ejecución para la instancia 2.*

En la figura 5 se observa que cuando se varía el nodo inicial se obtienen solamente mejoras, y de esas mejoras los nodos que presentan mejor función objetivo son el 1, 3 y 4, de los cuales el que presenta mejor tiempo de ejecución es el 3.

También se observa que fijando el nodo inicial y cambiando el nodo destino siempre se obtiene el mismo valor en la función objetivo, gracias a eso, y a la comparación de tiempos de ejecución, se podría considerar que los mejores nodos el 1 y el 7.

Si se elimina un nodo intermedio, no se obtienen mejoras en la función objetivo, así que no conviene eliminar ningún nodo, de hecho, si se elimina el nodo 3, el problema es infactible, por eso la importancia de este nodo.

Instancia 3



(a) Instancia

(b) Instancia con solución

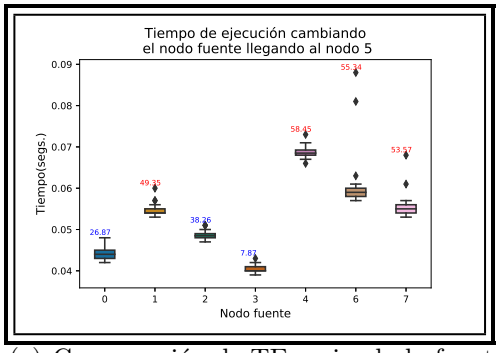
(c) Instancia con solucion y caracterización

Figura 6: *Visualización de la instancia 3.*

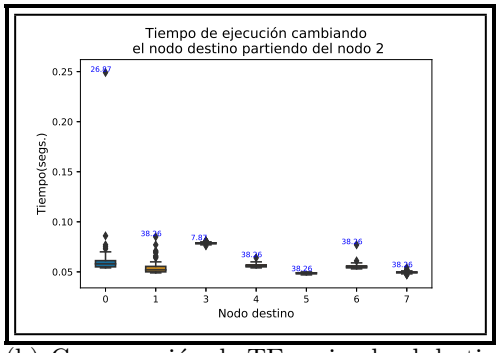
En la figura 6 se observa que del total de las aristas solo se usan 9, representando el 60 %.

También se muestra que los nodos 4 y 5 tiene en general mejores características que el resto y que para esta instancia los nodos 1, 4, 6 y 7 son mejores nodos iniciales (fuente) que el principal, obteniendo un aumento en la función objetivo.

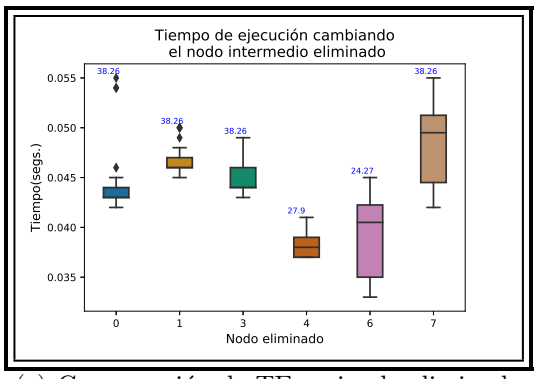
Por otro lado, si el problema se resuelve variando el nodo final no se obtiene mejora la función objetivo.



(a) Comparación de TE variando la fuente



(b) Comparación de TE variando el destino



(c) Comparación de TE variando eliminado

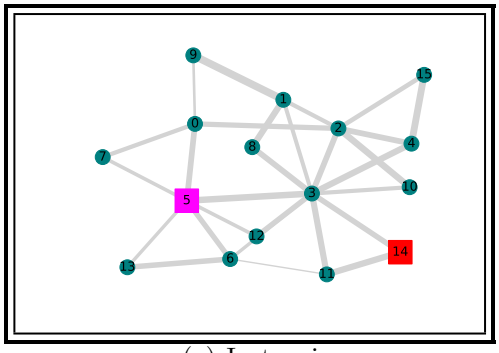
Figura 7: *Comparación del tiempo de ejecución para la instancia 3.*

En la figura 7 se observa que si se resuelve la instancia cambiando el nodo inicial, se obtienen diferentes mejoras, los nodos para los cuales se presenta eso son el 1, 4, 6 y 7 y de estos el que obtiene mayor función objetivo es el 4, pero también es el que más tiempo le toma en resolver la instancia.

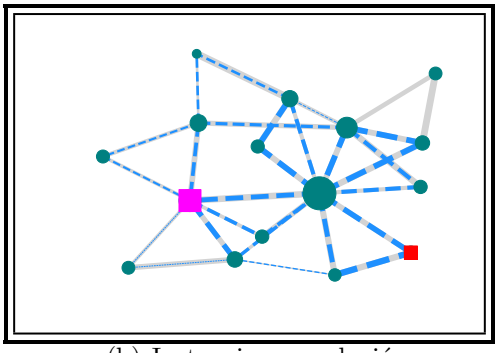
Como se mencionó en el apartado de visualización, cuando se varía el nodo final no se obtiene ninguna mejora, pero si se obtienen valores que empatan en función objetivo con la instancia original, los nodos para los que ocurre esto, son el 1, 4, 6 y 7, de los cuales los que se ejecutan en menor tiempo son el 7 y la instancia original con el nodo destino 5.

Si se elimina un nodo intermedio, no se obtienen mejoras en la función objetivo, pero si hay quienes se pueden eliminar y se conserva el valor del flujo óptimo, en este caso, se puede eliminar el nodo 3 por estar aislado.

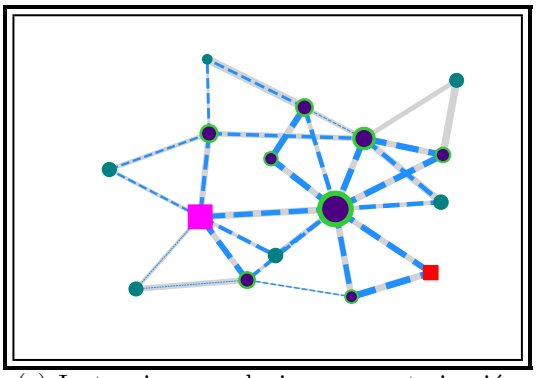
Instancia 4



(a) Instancia



(b) Instancia con solución



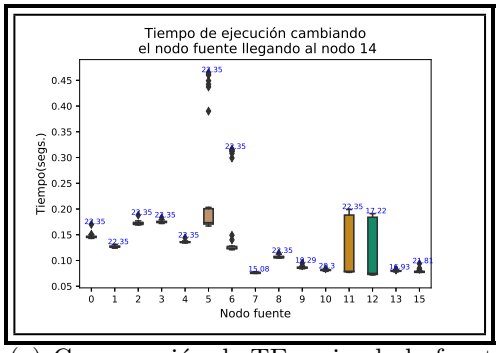
(c) Instancia con solucion y caracterización

Figura 8: *Visualización de la instancia 4.*

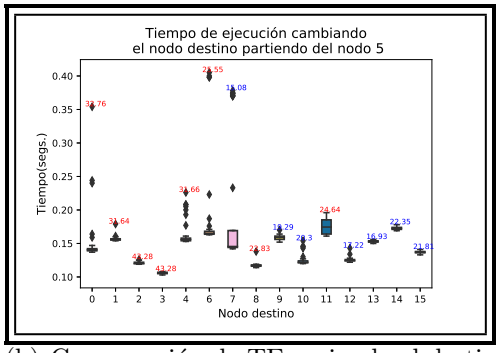
En la figura 8 se observa que del total de las aristas se usan 26, representando el 92.85 %.

También se muestra que los nodos 3 y 5 tienen en general mejores características que el resto y que para esta instancia los nodos 0, 1, 2, 3, 4, 6, 8 y 11 son mejores nodos finales (destino) que el principal, obteniendo un aumento en la función objetivo.

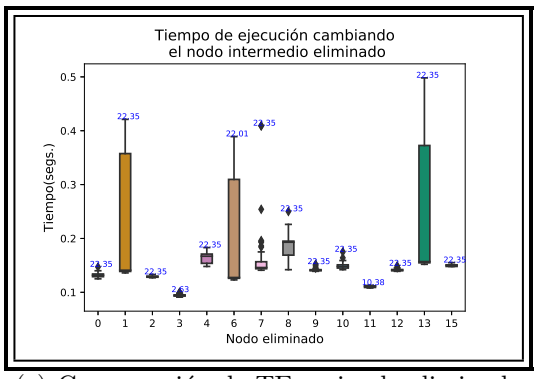
Por otro lado, si el problema se resuelve variando el nodo inicial no se obtiene mejora la función objetivo.



(a) Comparación de TE variando la fuente



(b) Comparación de TE variando el destino



(c) Comparación de TE variando eliminado

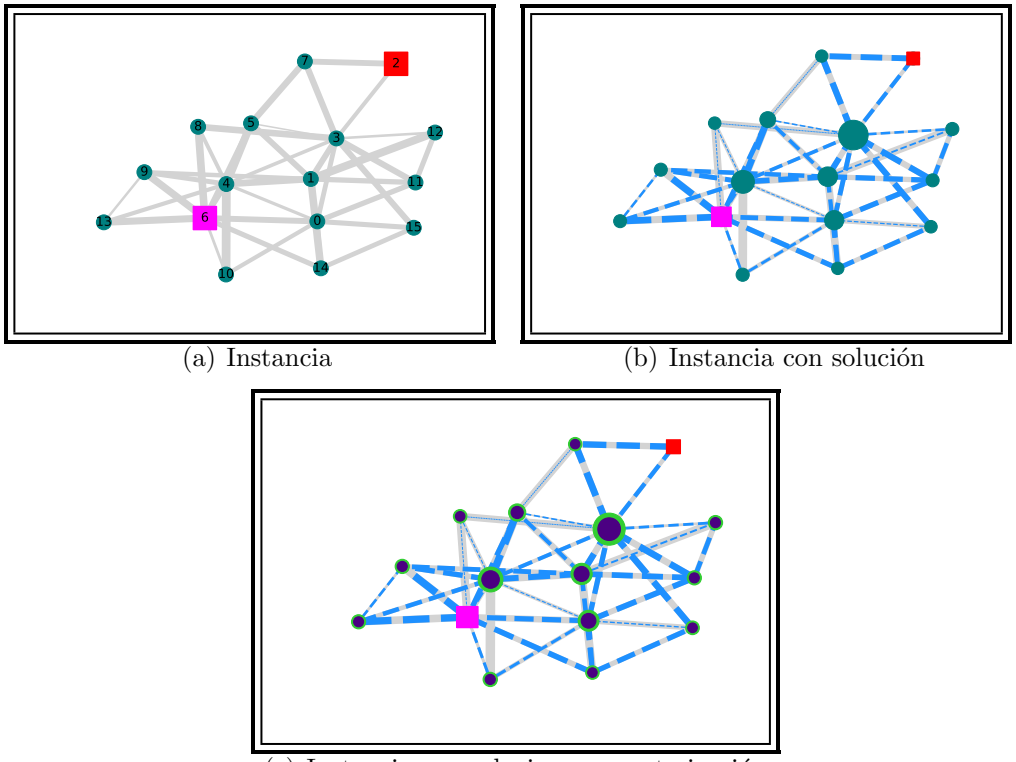
Figura 9: *Comparación del tiempo de ejecución para la instancia 4.*

En la figura 9 se observa que no hay mejoras en la función objetivo variando el nodo inicial y los que la empeoran son los nodos 7, 9, 10, 12, 13 y 15, el resto conserva el mismo flujo que el nodo inicial original, de los cuales el nodo 8 es el que presenta menor tiempo de ejecución al resolver la instancia.

Como se mencionó en el apartado de visualización, cuando se varía el nodo final se obtienen varias mejoras, y de esas mejoras los que presentan mayor función objetivo son los nodos 2 y 3, siendo el 2 el que presenta menor tiempo de ejecución.

Si se elimina un nodo intermedio, no se obtienen mejoras en la función objetivo, pero para algunos se conserva el flujo de la instancia original, así que se pudiera eliminar alguno de ellos, los mejores candidatos son 1, 9, 12 y 15 ya que son los que hacen que la distancia se resuelva más rápido.

Instancia 5



(a) Instancia

(b) Instancia con solución

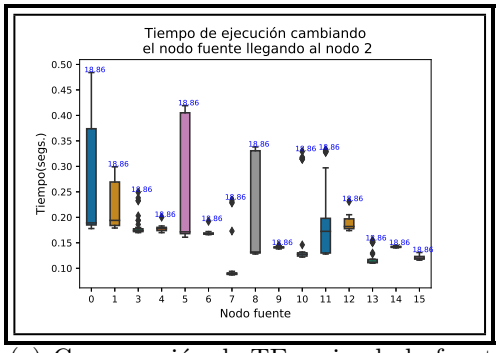
(c) Instancia con solucion y caracterización

Figura 10: *Visualización de la instancia 5.*

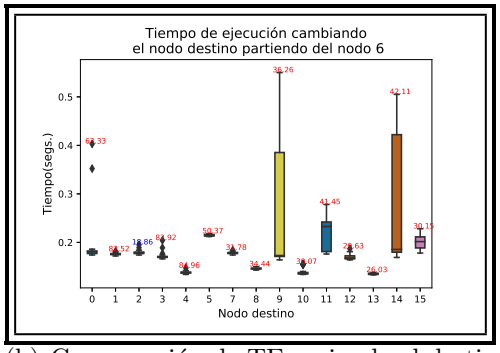
En la figura 10 se observa que del total de las aristas se usan 28, representando el 96.55 %.

También se muestra que los nodos 3 y 4 tiene en general mejores características que el resto y que para esta instancia todos los nodos intermedios son mejores nodos finales (destino) que el principal, obteniendo un aumento en la función objetivo.

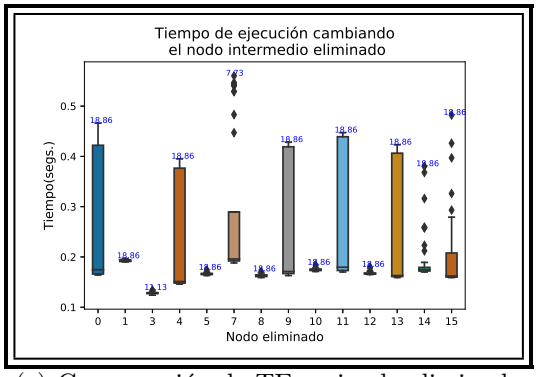
Por otro lado, si el problema se resuelve variando el nodo inicial no se obtiene mejora la función objetivo.



(a) Comparación de TE variando la fuente



(b) Comparación de TE variando el destino



(c) Comparación de TE variando eliminado

Figura 11: *Comparación del tiempo de ejecución para la instancia 5.*

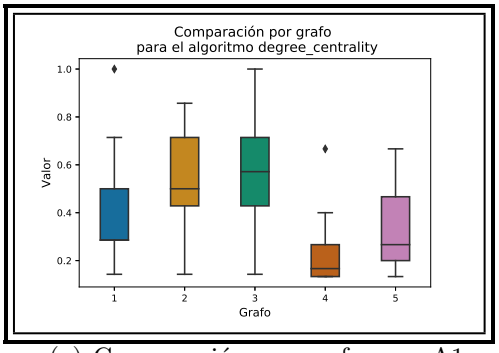
En la figura 11 se observa que no importa que nodo se tome como nodo fuente siempre se obtiene el mismo flujo y de estos el que resuelve la instancia más rápido es el nodo 7, presentando algunos casos atípicos con mayor tiempo.

Por otro lado, como se mencionó en el apartado de visualización cuando se varía el nodo final todos brindan mejor función objetivo, en especial los nodos 1, 3 y 4, siendo este ultimo el de mayor valor y el de menor tiempo de ejecución.

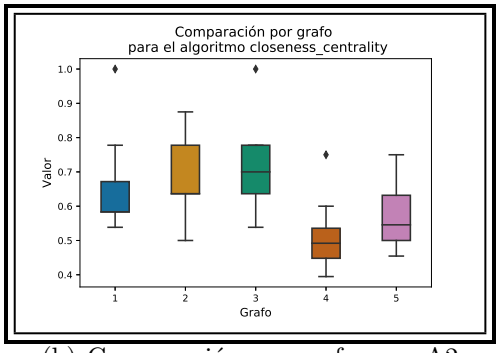
Si se elimina un nodo intermedio, no se obtienen mejoras en la función objetivo, pero para algunos se conserva el flujo de la instancia original, así que se pudiera eliminar alguno de ellos, los mejores candidatos son 5, 8, 10 y 12 ya que son los que hacen que la distancia se resuelva más rápido.

Comparación de instancias

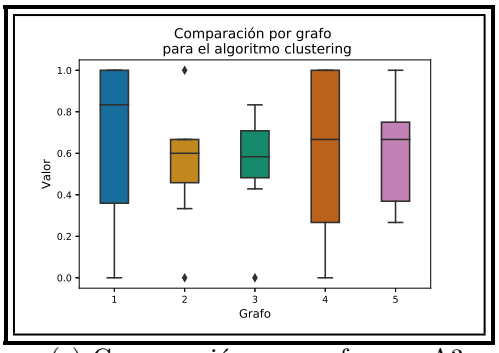
En este ultimo apartado, se comparan las características de los nodos para cada grafo, esperando poder determinar que característica pudiera parecer importante para un flujo más grande.



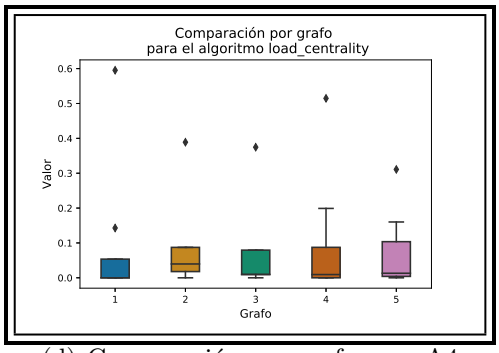
(a) Comparación por grafo para A1



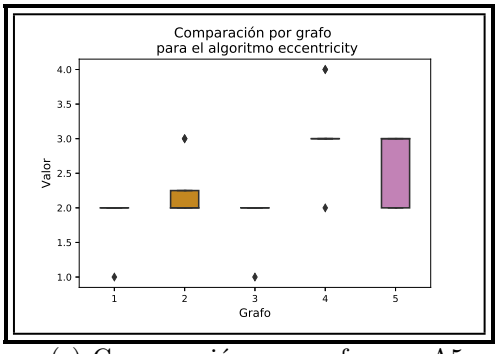
(b) Comparación por grafo para A2



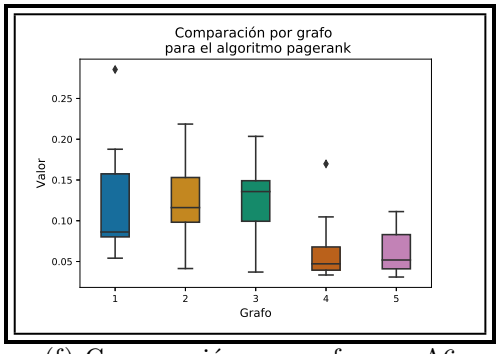
(c) Comparación por grafo para A3



(d) Comparación por grafo para A4



(e) Comparación por grafo para A5



(f) Comparación por grafo para A6

Figura 12: *Comparación de los valores por grafo para cada característica.*

Recordando los valores objetivos 14.95, 8.57, 38.26, 23.25 y 18.86, así como que el número de nodos de las primeras 3 instancias son de 8 nodos y el resto de 16, para las instancias 1, 2, 3, 4 y 5, respectivamente.

De acuerdo a los resultados de las comparaciones de las características se obtienen las siguientes conclusiones generales:

1. Dado que la instancia 3 fue la que mejor flujo obtuvo y la 4 fue la que le siguió pero obtuvieron buen y mal valor para el `degree_centrality` respectivamente, no pareciera que este algoritmo influencíe en la función objetivo.

2. Si se obtiene bajo valor de `closeness_centrality` obtiene mayor flujo, mientras que un alto valor, provoca menor flujo en el grafo.

3. En el algoritmo `clustering` hay mucha interacción entre los grafos y las funciones objetivos, dicho eso, no podemos decir que haya una relación lineal entre el valor de este algoritmo presente en el grafo y la función objetivo.

4. Las instancias 2 y 3 son la de menor y mayor objetivo respectivamente, pero para este algoritmo `load_centrality` son los que mayor valor tienen, así que, pareciera que no habría una relación clara en base a esta característica.

5. En este caso, la instancia 4 obtuvo muy buen valor de `eccentricity`, pero la instancia 3 muy malo, así que parece que habría constraste en los resultados de funciones objetivos, pareciendo que no hubiera relacion.

6. Para este algoritmo `pagerank` parece haber una tendencia entre que a mayor valor, menor flujo y para menor valor, mayor flujo.

Estas conclusiones se obtuvieron a partir de las instancias realizadas, si se desea mayor precisión, se puede aumentar el número de instancias y de variaciones en ellas.

Referencias

[1] Python Software Foundation Versión 3.7.2. https://www.python.org/.

[2] NetworkX developers Versión 2.2. https://networkx.github.io.

[3] The Matplotlib development team Versión 3.0.2. https://matplotlib.org/.

[4] Sulce A. PythonHow. https://pythonhow.com/measure-execution-time-python-code/.

[5] Augspurger T. and the pandas core team Versión 0.24.2. https://pandas.pydata.org/.

[6] Waskom M. Versión 0.9.0. Copyright 2012-2018. https://seaborn.pydata.org/.

[7] Francois J. Mecánica de fluidos. Escuela de Ingeniería Mecánica.

[8] Gutiérrez M. Repositorio optimización flujo en redes. https://github.com/MarioGtz14/Flujo_Redes_Mario.