Tarea 2 Optimización de Flujo de Redes

1985274

26 de febrero de 2019

En este trabajo se mostrarán varios tipos de grafos con ejemplos empleando distintos algorítmos de acomodamiento. Para esto se ha utilizado el lenguaje **Python** en su versión 3.7 [11], el editor de código Spyder en su versión 3.3.1 y el editor Texmaker 5.0.2 para redactar el documento. Además, se han empleado las librerías **networkX** 1.5 [10] y **Matplotlib** [7], para dibujar los grafos mostrados.

1. Grafo simple no dirigido acíclico

Este grafo se asemeja a la situación del recorrido de la Línea dos del metro de Monterrey, México que conecta varias estaciones de la ciudad. El metro comienza su recorrido en la estación de Sendero y finaliza en la estación de General Zaragoza. En ejemplo de la figura 1 cada estación es considerada como un nodo y están enlazadas por aristas que no tienen dirección y todas las estaciones están conectadas. Basado en el plano del libro Plan Sectorial de Transporte y Vialidad del Área Metropolitana de Monterrey 2008–2030 de la página 100 [4].

En el ejemplo de la figura 1 se utilizó el algoritmo de acomodamiento spring layout para representar los nodos y arcos del ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G, pos, dim, k, iterations y el parámetro fixed. Los nodos verdes indican el nodo inicial y final del recorrido del metro [8] .

1.1. Código

```
1
2  G.add_edge('a', 'b',  weight=0.2)
3  G.add_edge('b',   'c',  weight=0.7)
4  G.add_edge('c',   'd',weight=0.7)
5  G.add_edge('d', 'e',weight=0.7)
6  G.add_edge('e', 'f',weight=0.9)
7  G.add_edge('f', 'g', weight=0.7)
8  G.add_edge('g',  'h',   weight=0.7)
9  G.add_edge('h',   'i',  weight=0.7)
10
```

```
11 initialpos = {'a':(0,0), 'b':(0,3), 'c':(0,-1), 'd':(2,5), 'e':(5,5), 'f'
       :(4,4), 'g':(4,9), 'h':(6,1),'i':(7,2),  }
12 pos = nx.spring_layout(G, k=10, pos = initialpos, fixed=None,  iterations
       =100,dim=2, weight='weight',scale=0.5)
13
14 #pos = nx.spring_layout(G)
15 size = [700, 700, 700, 700, 700, 700,700,700,700]
16 color=[ "#2da05f" , "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"
       , "#A0CBE2", "#A0CBE2", "#2da05f" ]
17
18 nx.draw_networkx_nodes(G, pos, node_color=color,node_size=size,)#
       node_size=700,s
19 nx.draw_networkx_edges(G, pos, width=6)
20 nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
```

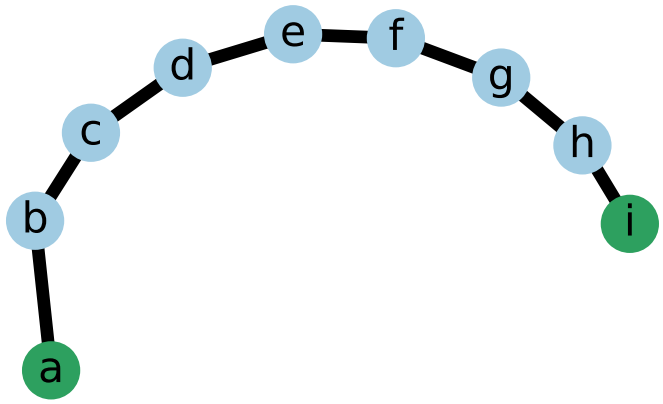

Figura 1: Grafo simple no dirigido acíclico

2. Grafo simple no dirigido cíclico

Este grafo se asemeja a la estructura del grafito. El grafito es un tipo de material que presenta una estructura laminar conocida como grafeno. Éste está compuesto por átomos de carbono formando una red hexagonal plana, la cual está formada por seis átomos enlazados formando un anillo. En este tipo de estructura cada átomo está representado en la figura 2 como un nodo del grafo y las aristas representan los enlaces entre los átomos. Basado en la figura uno

de la tesis Modificación superficial de materiales de carbono: grafito y grafeno de la página 6 [5].

En el ejemplo de la figura 2 se utilizó el algoritmo de acomodamiento circular layout para representar el ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G, scale, center y dim. Este algoritmo devuelve una lista de las posiciones por cada nodo[9].

2.1. Código

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3  G = nx.Graph()
4  G.add_edge('A', 'B', weight=0.6)
5  G.add_edge('B', 'C', weight=0.7)
6  G.add_edge('C', 'D', weight=0.7)
7  G.add_edge('D', 'E', weight=0.7)
8  G.add_edge('E', 'F', weight=0.9)
9  G.add_edge('F', 'A', weight=0.7)
10 size = [700, 700, 700, 700, 700, 700]
11 color=[ "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2" , "#A0CBE2" ,"#A0CBE2"
       ]
12 initialpos = {'a':(0,0), 'b':(0,3), 'c':(0,-1), 'd':(5,2), 'e':(3,1), 'f'
      :(4,4)}
13 pos = nx.circular_layout(G, scale=0.5, dim=2)
14 nx.draw_networkx_nodes(G, pos, node_size=size, node_color=color)
15 nx.draw_networkx_edges(G, pos, width=6)
16 nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
17 plt.axis('off')
18 plt.savefig("GraSNoDiCiclicoTipoLayout.eps")
19 plt.show()
```

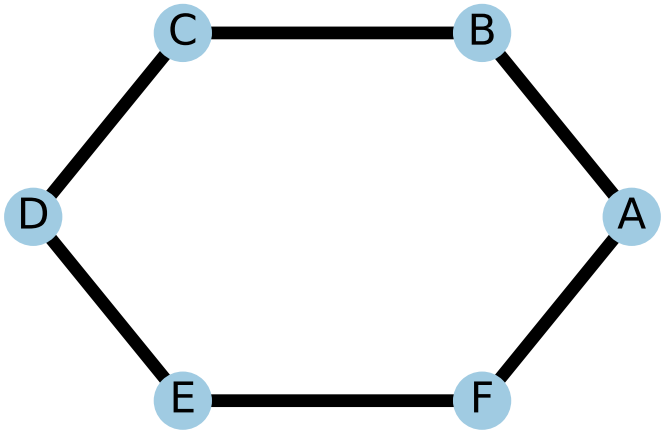

Figura 2: Grafo simple no dirigido cíclico

3. Grafo simple no dirigido reflexivo

Este grafo se asemeja a la situación en la que se tiene un conjunto de inspectores de rutas de transporte interurbanas, donde los nodos que son las paradas, son los puntos de embarques de las personas. Las aristas que unen a los nodos son las autopistas y carreteras que unen las paradas. Este grafo es no dirigido porque por esas rutas (las autopistas, carreteras), los camiones pueden pasar en dos direcciones. Los inspectores tienen que inspeccionar determinadas rutas críticas. Las aristas tienen un peso o ponderación de cero si no es crítica y uno si es crítica.

Los inspectores comen en determinada hora y esta actividad se puede representar en el grafo una arista al mismo nodo y esa arista indica que ese es un nodo de comida.

En el ejemplo de la figura 3 el nodo reflexivo está marcado de color rojo. En esa figura una parada sería el nodo **a**, otra parada sería el nodo **b**, que como es reflexivo sería un nodo de comida, es decir una parada donde los inspectores pueden comer.

Un bucle es una arista reflexiva, donde coinciden el vértice de origen y el vértice destino, si un grafo cuenta con al menos un nodo con esta característica se dice que es un grafo reflexivo [12, pág 6].

Para visualizar este grafo se utilizó el algoritmo de acomodamiento spring layout representado en la figura 3.

3.1. Código

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.Graph()
G.add_node( "a", size=10 ,pos=(1,60))
G.add_node( "b" , size=10 , pos=(40,80))
G.add_node( "c", size=10 , pos=(80,20))
G.add_node( "d", size=10, pos=(80,-50))
G.add_node( "e", size=10 ,pos=(10,20))
G.add_node( "f", size=10 ,pos=(40,1))
G.add_node( "g", size=10 ,pos=(30,-50))
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('e', 'a',  color='red',weight=8)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)
G.add_edge('g', 'e',  color='black',weight=3)
pos=nx.spring_layout(G, k=10, iterations=100, dim=2,weight='weight',scale
    =0.5 )
labels = {('a','e'):'15', ('b','f'):'70',('d','c'):'10'}
size = [700, 700, 700, 700, 700, 700, 700]
color=["#A0CBE2", "#FF0000", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2",
    "#A0CBE2"]
nx.draw(G,pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=3, node_shape='s')
nx.draw_networkx_nodes(G, pos, node_size=size, node_color=color)
nx.draw_networkx_edges(G, pos, width=6, edge_color='blue')
nx.draw_networkx_edge_labels(G,pos, labels, font_size=10, font_color='red
    ',font_family='sans-serif')
plt.axis('off')
plt.savefig("GNDirigidoReflexivo.eps")
plt.show()
```

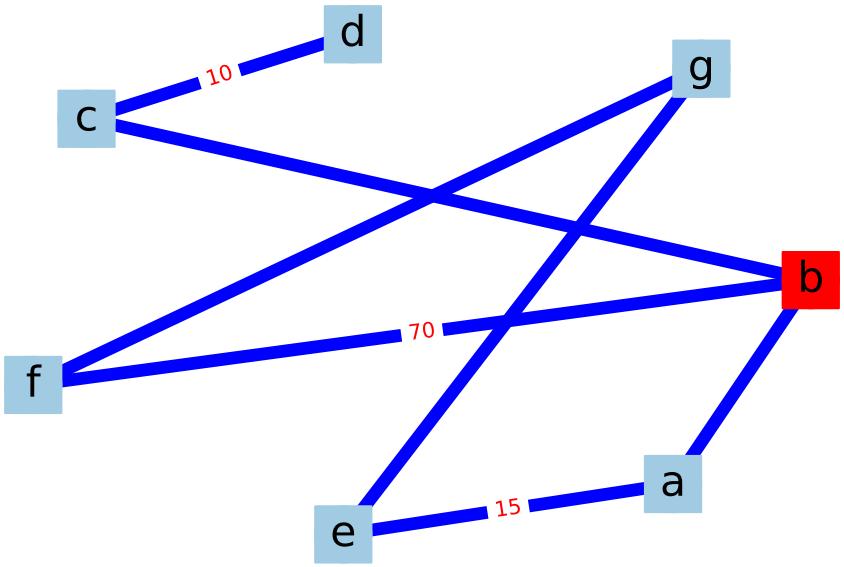

Figura 3: Grafo simple no dirigido reflexivo

4. Grafo simple dirigido acíclico

Un ejemplo de este grafo es el flujograma del proceso de fabricación del asfalto, que incluye la dosificación, mezcla y carga en una planta de asfalto. En este grafo cada actividad es un nodo. El flujograma tiene una actividad donde comienza el proceso y donde finaliza el proceso y cada actividad tiene un orden específico, lo cual determina que se pueda representar como un grafo simple y dirigido. Las aristas representan el cambio de una actividad a la otra.

En el grafo que se ilustra en la figura 4 el nodo **Inicio**, representa el inicio del proceso con la fabricación de cemento, el nodo dos representa el **proceso 1** que se refiere a la elevación de los materiales pétreos a la criba, el nodo tres representa el **proceso 2** que se refiere a la Separación y almacenaje en tolvas de los elementos en caliente, el nodo cuatro representa el **proceso 3** que se refiere al Pesaje del asfalto y materiales pétreos, el nodo cinco representa el **proceso 4** que se refiere a la mezcla, el nodo seis representa el **proceso 5** que se refiere al Transporte de mezcla y almacenaje en tolvas de carga, el nodo siete representa el fin del Proceso de fabricación [2, pág 78].

En el ejemplo de la figura 4 se utilizó el algoritmo de acomodamiento kamada kawai layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G, que es el grafo, dist, pos weight, scale, center y dim.

4.1. Código

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()
G.add_edge('I', 'a', weight=0.6)
G.add_edge('a', 'b', weight=0.6)
G.add_edge('b', 'c', weight=0.6)
G.add_edge('c', 'd', weight=0.6)
G.add_edge('d', 'e', weight=0.7)
G.add_edge('e', 'F', weight=0.7)
pos = nx.kamada_kawai_layout(G, weight='weight', scale=1, center=None,dim
    =2)
labels = {('I', 'a' ):'20', ('b','c'):'70',('c','d'):'10'}
size = [700, 700, 700, 700, 700, 700,700]
color=[ "#2da05f" , "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"
    , "#A0CBE2" ]
nx.draw(G, pos=pos, with_labels = True, scale=1, align='vertical',center=
    None,edge_color='black',node_color=color,font_size=20, width=2)
nx.draw_networkx_edge_labels(G,pos, labels, font_size=10, font_color='red
    ')
nx.draw_networkx_nodes( G, pos, node_size=700, node_color=color)
nx.draw_networkx_edges( G, pos,width=6, edge_color='blue')
nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
plt.axis('off')
plt.savefig("GrafoSimpleDAcicNew.eps")
plt.show()
```

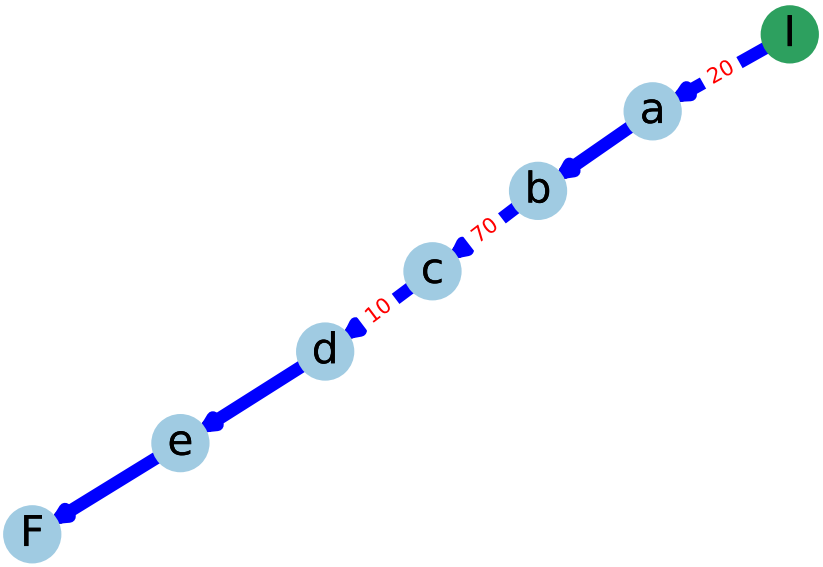

Figura 4: Grafo simple dirigido acíclico

5. Grafo simple dirigido cíclico

Un ejemplo de este grafo es el flujo de actividades por las que transita el instrumental quirúrgico a esterilizar que es utilizado en un hospital. En la figura 5 cada actividad puede ser representada por un nodo y al estar todas conectadas se forma un grafo, que siempre sigue un orden, por esto es un grafo dirigido. El instrumental puede ser esterilizado muchas veces, por lo que se considera un grafo cíclico. El flujo se inicia cuando el instrumental es recolectado del salón de operación y esta actividad está representada por el nodo a**nodo a**. A continuación, se pasa a la Descontaminación, que es el nodo b **nodo b**, después se pasa a la Desinfección, que corresponde al nodo c **nodo c**. Le sigue la actividad de Limpieza (nodo d)**nodo d**, luego se pasa al Enjuague (nodo e)**nodo e**, luego se pasa al Secado (nodo f)**nodo f**, luego el material pasa a la actividad de Inspección y empaquetamiento del material(nodo g)**nodo g** y seguidamente se efectúa el Control de calidad(nodo h)**nodo h**. El siguiente nodo representa el material quirúrgico in situ empleado por el cirujano. Basado en el libro Manual de esterilización para centros de salud [6].

En el ejemplo de la figura 5 se utilizó el algoritmo de acomodamiento rescale layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, se encuentran: el parámetro pos y scale.

5.1. Código

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()

G.add_node( "a", size=10 )
G.add_node( "b" , size=10 )
G.add_node( "c", size=10 )
G.add_node( "d", size=10 )
G.add_node( "e", size=10 )
G.add_node( "f", size=10 )
G.add_node( "g", size=10 )
G.add_node( "h", size=10 )
G.add_node( "i", size=10 )
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('d', 'e',  color='black',weight=3)
G.add_edge('e', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)
G.add_edge('g', 'h',  color='black',weight=3)
G.add_edge('h', 'i',  color='black',weight=3)
G.add_edge('i', 'a',  color='black',weight=3)
pos = nx.shell_layout(pos)
labels = {('a', 'b' ):'20', ('b','c'):'70',('c','d'):'10'}
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2",
    "#A0CBE2", "#A0CBE2", "#A0CBE2"]
nx.draw(G, pos=pos, with_labels = True, scale=1, align='horizontal',
    center=None,edge_color='black',node_color=color,font_size=20, width
    =2)
nx.draw_networkx_edge_labels(G,pos, labels, font_size=10, font_color='red
    ')
nx.draw_networkx_nodes( G, pos, node_size=400, node_color=color)
nx.draw_networkx_edges( G, pos,width=3, edge_color='blue')
nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
```

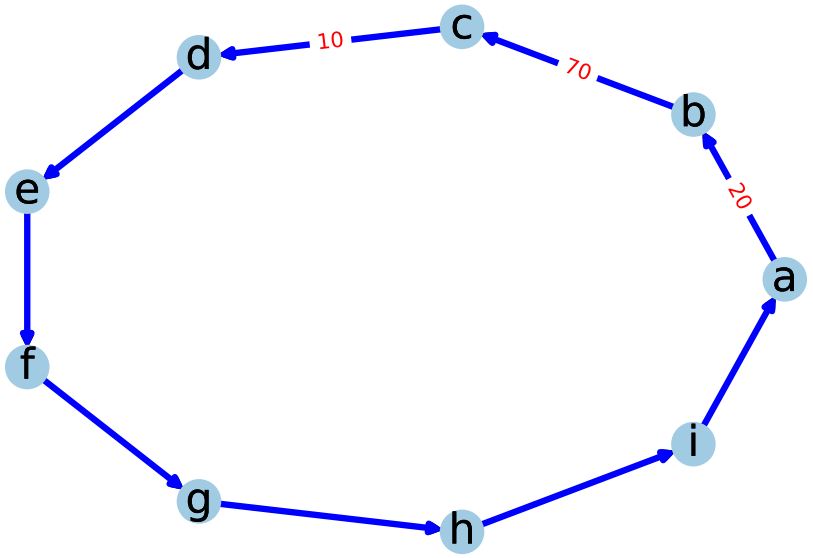

Figura 5: Grafo simple dirigido cíclico

6. Grafo simple dirigido reflexivo

Un ejemplo de este tipo de grafo está representado en el diagrama de Entidad-Relación en la construcción de una base de datos. En la figura 6, cada nodo representa una tabla de la base de datos, y las aristas simbolizan la relación con otras tablas. En este tipo de diagramas se pueden establecer relaciones binarias, reflexivas y ternarias entre las tablas de la base de datos [3].

En el ejemplo de la figura 6 se utilizó el algoritmo de acomodamiento bipartite layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentra: el parámetro G (grafo a visualizar), el parámetro nodes, align (para alinear los nodos), center y aspect ratio.

6.1. Código

```
1 import matplotlib.pyplot as plt
2
3 G = nx.DiGraph()
4 G.add_nodes_from([1,2,3,4], bipartite=0)
5 G.add_nodes_from(['a','b', 'c', 'd' ], bipartite=1)
6 G.add_edges_from([(1,'a'), (1,'b'), (2,'b'), (2,'c'), (3,'c'),  (4,'a')
      ,(4,'d') ])
7 l, r = nx.bipartite.sets(G)
```

```
pos = {}
pos.update((node, (1, index)) for index, node in enumerate(l))
pos.update((node, (2, index)) for index, node in enumerate(r))
labels = {(1,'a'):'2', (2,'b'):'7'}
color=["#2da05f", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2",
    "#A0CBE2", "#A0CBE2"]
nx.draw(G, pos=pos, with_labels = True, scale=1, align='vertical',center=
    None,edge_color='black',node_color=color,font_size=20, width=2)
nx.draw_networkx_edge_labels(G,pos, labels, font_size=10, font_color='red
    ')
nx.draw_networkx_nodes(G, pos, node_size=700,  node_color=color,
    with_labels=True)
nx.draw_networkx_edges( G, pos,width=6, edge_color='blue')
nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
```

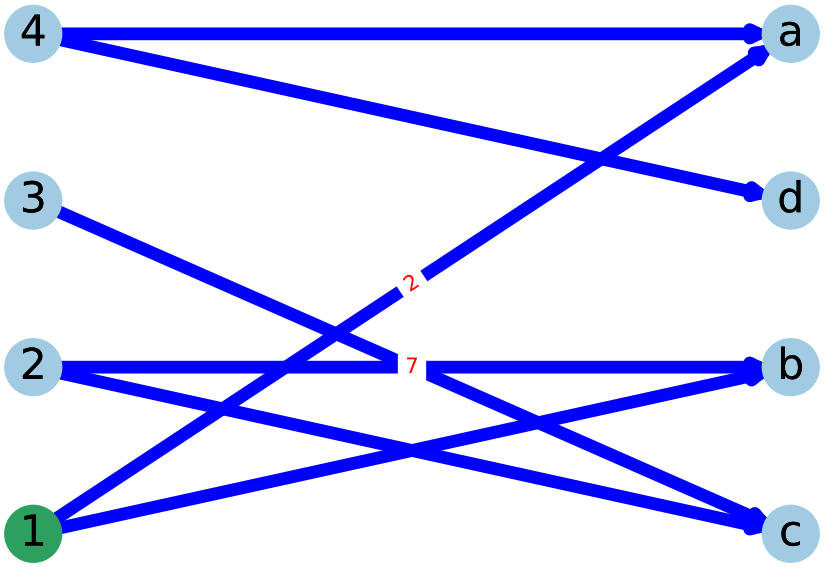

Figura 6: Grafo simple dirigido reflexivo

7. Multigrafo no dirigido acíclico

Un ejemplo aplicado de este tipo de multigrafo está basado en la representación del diagrama de rutas entre ciudades donde cada ciudad está representada como un nodo y cada arista es una vía para llegar a esa ciudad con una ponderación, por ejemplo por vía aérea, por vía terrestre. La ponderación es el tiempo que se demora en llegar de una ciudad a otra, o de un nodo al otro. En la figura 7 cada nodo representa a una ciudad, que puede ser el origen del viaje hacia

otro nodo pasando por varias otras ciudades o nodos. Los arcos son las vías que existen para llegar, representados en la figura de color verde y negro.

En el ejemplo de la figura 7 se utilizó el algoritmo de acomodamiento Spectral layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G(es el grafo a visualizar) y el parámetro weight(con este parámetro se controla el ancho de los arcos).

7.1. Código

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.MultiGraph()
G.add_node( "a", size=60 ,  pos=(1,60))
G.add_node( "b" , size=60 , pos=(40,80))
G.add_node( "c", size=60, pos=(80,20))
G.add_node( "d", size=60,  pos=(80,-50))
G.add_node( "e", size=60,  pos=(10,20))
G.add_node( "f", size=60, pos=(40,1))
G.add_node( "g", size=60, pos=(30,-50))
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('a', 'b',  color='black',weight=3)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('b', 'c',  color='black',weight=3)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('c', 'd',  color='black',weight=3)
G.add_edge('e', 'a',  color='red',weight=8)
G.add_edge('e', 'a',  color='black',weight=3)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)

pos=nx.spectral_layout(G)
size = [700, 700, 700, 700, 700, 700]
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"]
nx.draw(G,pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=8)
```

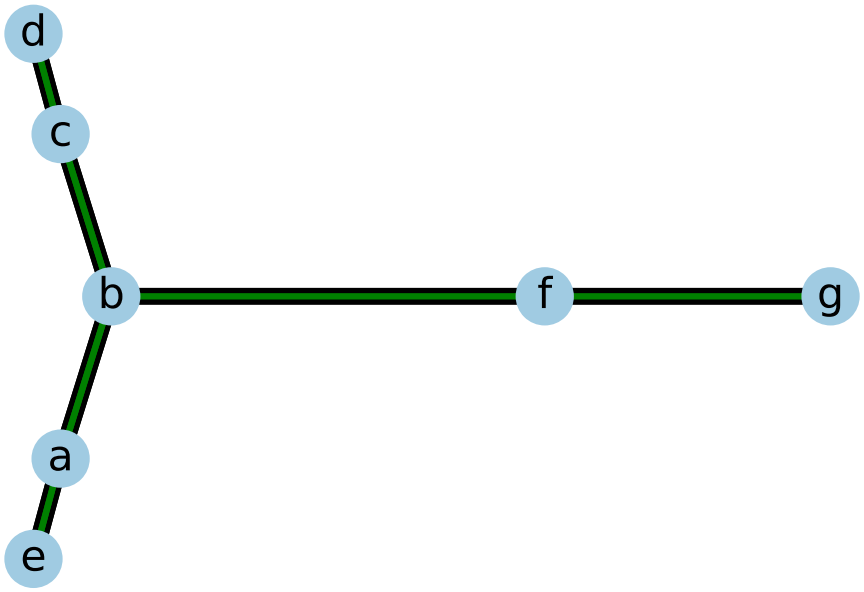

Figura 7: Multigrafo no dirigido acíclico

8. Multigrafo no dirigido cíclico

Un ejemplo aplicado de este tipo de multigrafo se asemeja a la situación del diagrama de relación de usuarios de redes sociales como Facebook. En la figura 8 se muestra un ejemplo donde los individuos están representadas por los nodos y los arcos representan las relaciones entre los individuos. En este tipo de multigrafos los nodos pueden ser además de los individuos, organizaciones, grupos sociales, usuarios de un servicio de salud, miembros de una comunidad. Los arcos representan las relaciones sociales como links o bonds. Estas relaciones entre individuos se refieren relaciones de amistad, de pareja, y trabajo entre otras. Basado en la figura 4.0 del libro Redes sociales y análisis de redes de la página 105 [13].

En el ejemplo de la figura 8 se utilizó el algoritmo de acomodamiento fruchterman reingold layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G(es el grafo a visualizar), el parámetro dim, k(distancia óptima entre nodos),pos, fixed, iterations y weight(con este parámetro se controla el ancho de los arcos).

8.1. Código

```
1 import networkx as nx
```

```
import matplotlib.pyplot as plt
G = nx.MultiGraph()

G.add_node( "a", size=60 ,  pos=(5,40))
G.add_node( "b" , size=60 , pos=(20,50))
G.add_node( "c", size=60, pos=(40,40))
G.add_node( "d", size=60,  pos=(50,40))
G.add_node( "e", size=60,  pos=(10,30))
G.add_node( "f", size=60, pos=(30,40))
G.add_node( "g", size=60, pos=(30,30))
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('a', 'b',  color='black',weight=3)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('b', 'c',  color='black',weight=3)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('c', 'd',  color='black',weight=3)
G.add_edge('e', 'a',  color='red',weight=8)
G.add_edge('e', 'a',  color='black',weight=3)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)
G.add_edge('g', 'e',  color='black',weight=3)
pos=nx.fruchterman_reingold_layout(G)
size = [700, 700, 700, 700, 700, 700]
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"]
nx.draw(G,pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=8)
nx.draw_networkx_edges(G, pos,  edge_color='green', label="S", arrowstyle
    ='-|>', arrows=True, width=3)
plt.axis('off')
```

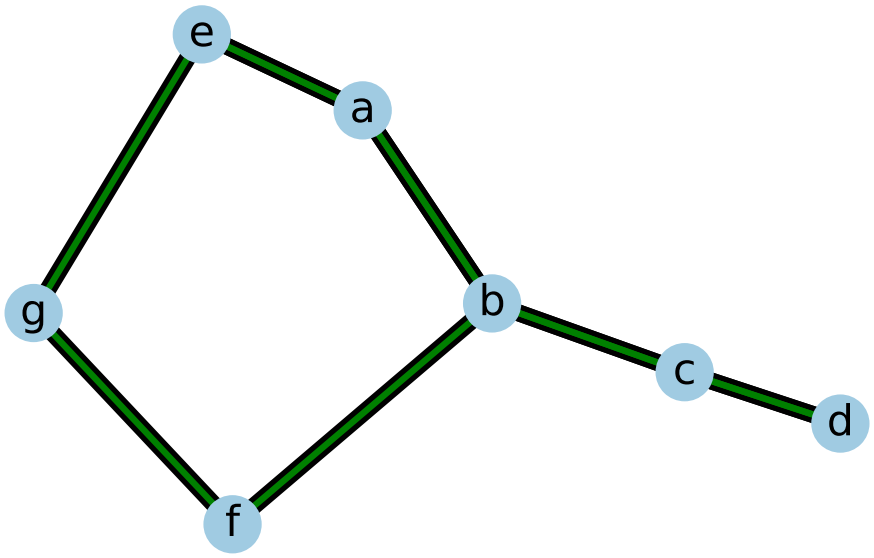

Figura 8: Multigrafo no dirigido cíclico

9. Multigrafo no dirigido reflexivo

Un ejemplo aplicado de este tipo de multigrafo se asemeja al organigrama de competencias de un torneo de ajedrez, donde cada nodo representa a una escuela (nodos de color rojo). Primero se efectúan las competencias locales entre los concursantes de una misma escuela (representado en el grafo como un nodo reflexivo), luego se realizan competencias entre escuelas. En este ejemplo de la figura 9 cada nodo representa a una escuela y los arcos reflejan las competencias efectuadas. Los nodos azules son las escuelas que pasan a la segunda etapa de competencias. El nodo representa a las competencias entre las escuelas ganadoras [1].

En el ejemplo de la figura 9 se utilizó el algoritmo de acomodamiento circular layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G (grafo a visualizar), scale, center y dim.

9.1. Código

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 G = nx.MultiGraph()
4
5 G.add_node( "a", size=60 ,  pos=(1,60))
```

```
G.add_node( "b" , size=60 , pos=(10,80))
G.add_node( "c", size=60, pos=(80,20))
G.add_node( "d", size=60,  pos=(80,-50))
G.add_node( "e", size=60,  pos=(10,20))
G.add_node( "f", size=60, pos=(40,1))
G.add_node( "g", size=60, pos=(30,-50))
G.add_edge('a', 'e',  color='red',weight=8)
G.add_edge('b', 'e',  color='red',weight=8)
G.add_edge('c', 'f',  color='red',weight=8)
G.add_edge('d', 'a',  color='red',weight=8)
G.add_edge('e', 'g',  color='red',weight=8)
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('g', 'd',  color='red',weight=8)
G.add_edge('g', 'c',  color='red',weight=8)
G.add_edge('b', 'f',  color='red',weight=8)
pos = nx.circular_layout(G, scale=0.5, dim=2)
size = [1000, 1000, 1000, 1000, 1000, 1000, 1000]
color=["#FF0000", "#A0CBE2", "#A0CBE2", "#FF0000", "#FF0000", "#A0CBE2",
    "#FF0000"]
nx.draw(G, pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=8)
nx.draw_networkx_edges(G, pos,  edge_color='green', label="S", arrowstyle
    ='-|>', arrows=True, width=3)
nx.draw_networkx_nodes( G, pos, node_size=400, node_color=color)
nx.draw_networkx_labels(G, pos, font_size=20, font_family='sans-serif')
plt.axis('off')
```

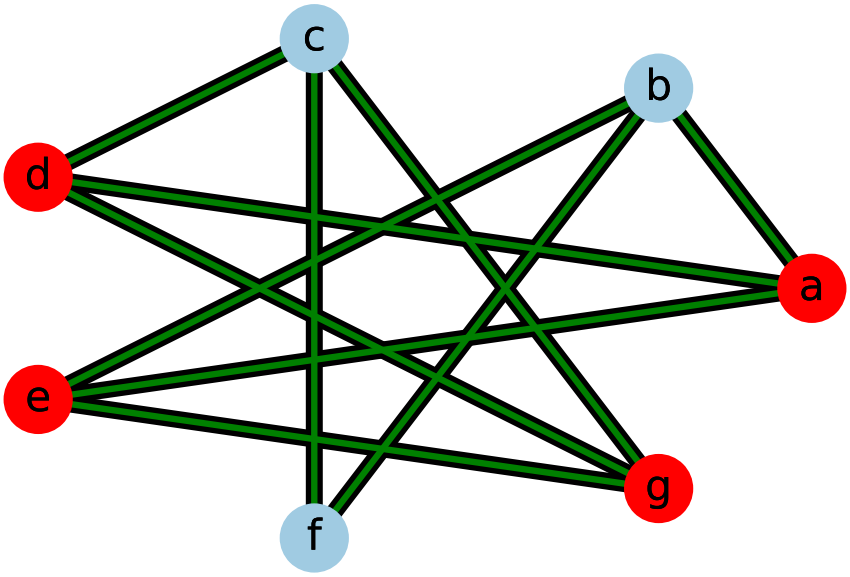

Figura 9: Multigrafo no dirigido reflexivo

10. Multigrafo dirigido acíclico

Un ejemplo aplicado de este tipo de multigrafo es el diagrama de opciones de aerolíneas que tiene un viajero para ir de una ciudad a la otra, por ejemplo, para ir de la Ciudad de la Habana a la ciudad de Miami, el viajero tiene varias opciones como la aerolínea American Airlines, Interjet, Aeroméxico y Copa Airlines entre otras. En la figura diez cada ciudad se representa como un nodo, por lo que existe una ciudad origen y otra ciudad destino y los arcos o aristas son las distintas opciones de aerolíneas que el viajero tiene para seleccionar y de manera similar para cada destino, estos se encuentran en la figura 10 de color verde y negro.

En el ejemplo de la figura 10 se utilizó el algoritmo de acomodamiento random layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G (el grafo a visualizar), dim (dimensión del layout) y seed (semilla para generar el layout de los nodos).

10.1. Código

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 G = nx.MultiDiGraph()
4 G.add_node( "a", size=10 ,  pos=(1,60))
```

```
G.add_node( "b" , size=10 , pos=(40,80))
G.add_node( "c", size=10, pos=(80,20))
G.add_node( "d", size=10,  pos=(80,-50))
G.add_node( "e", size=10, pos=(30,-50)    )# pos=(10,20) )
G.add_node( "f", size=10, pos=(40,1))
G.add_node( "g", size=10, pos=(10,20)  )   #pos=(30,-50) )
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('a', 'b',  color='black',weight=3)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('b', 'c',  color='black',weight=3)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('c', 'd',  color='black',weight=3)
G.add_edge('e', 'a',  color='red',weight=8)
G.add_edge('e', 'a',  color='black',weight=3)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)
#G.add_edge('g', 'e',  color='black',weight=3)
pos=nx.get_node_attributes(G,'pos')

pos = nx.random_layout(G) # scale=0.5, dim=2)
size = [1000, 1000, 1000, 1000, 1000, 1000, 1000]
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"]

nx.draw(G,pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=6)
```

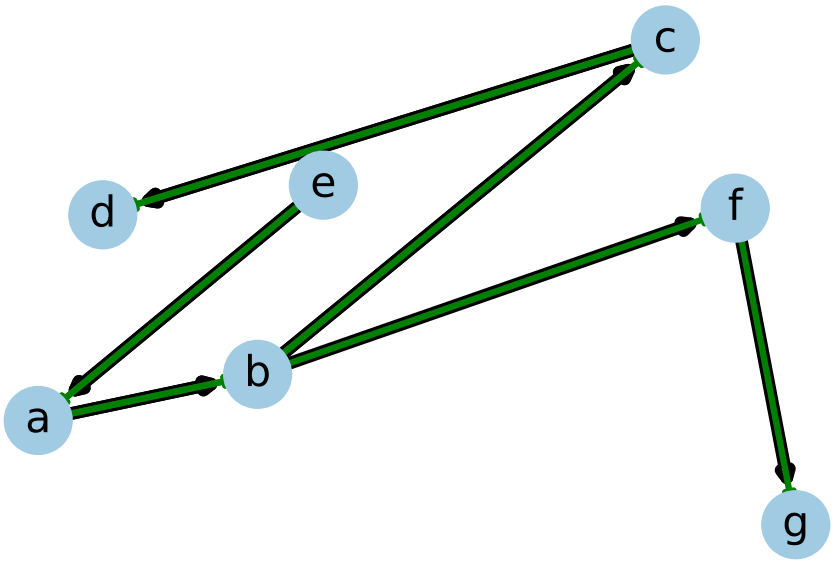

Figura 10: Multigrafo dirigido acíclico

11. Multigrafo dirigido cíclico

Un ejemplo aplicado de este tipo de multigrafo es la representación de las opciones que tiene un viajero que se dispone a realizar un viaje en redondo, partiendo de la ciudad donde vive y regresando al final del viaje a la ciudad de donde partió, la ciudad origen. Cada país o ciudad visitada está representada como un nodo y las aristas representan la transición de un país o ciudad a la otra ciudad o país. El viajero tiene varias opciones para realizar su viaje en redondo, a partir de las distintas opciones que le brindan las agencias de viaje. En la figura 11 cada nodo es una ciudad o país, y las aristas varias opciones para llegar de un nodo al otro.

En el ejemplo de la figura 11 se utilizó el algoritmo de acomodamiento spring layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G (representa el grafo a visualizar), parámetro k (distancia optimal entre los nodos), pos(posición inicial para los nodos) y el parámetro iterations (número máximo de iteraciones).

11.1. Código

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
```

```
G = nx.MultiDiGraph()

G.add_node( "a", size=10 ,  pos=(1,60))
G.add_node( "b" , size=10 , pos=(40,80))
G.add_node( "c", size=10, pos=(80,20))
G.add_node( "d", size=10,  pos=(80,-50))
G.add_node( "e", size=10,  pos=(10,20))
G.add_node( "f", size=10, pos=(40,1))
G.add_node( "g", size=10, pos=(30,-50))
G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('a', 'b',  color='black',weight=3)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('b', 'c',  color='black',weight=3)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('c', 'd',  color='black',weight=3)
G.add_edge('e', 'a',  color='red',weight=8)
G.add_edge('e', 'a',  color='black',weight=3)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=3)
G.add_edge('g', 'e',  color='black',weight=3)
pos = nx.spring_layout(G, scale=0.5, dim=2)
size = [1000, 1000, 1000, 1000, 1000, 1000, 1000]
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2", "#A0CBE2"]
nx.draw(G,pos, with_labels=True, edge_color='black', node_size=size,
    node_color=color, font_size=20, width=7)

nx.draw_networkx_edges(G, pos,  edge_color='green', label="S", arrowstyle
    ='-|>', arrows=True, width=3)

plt.axis('off')
```

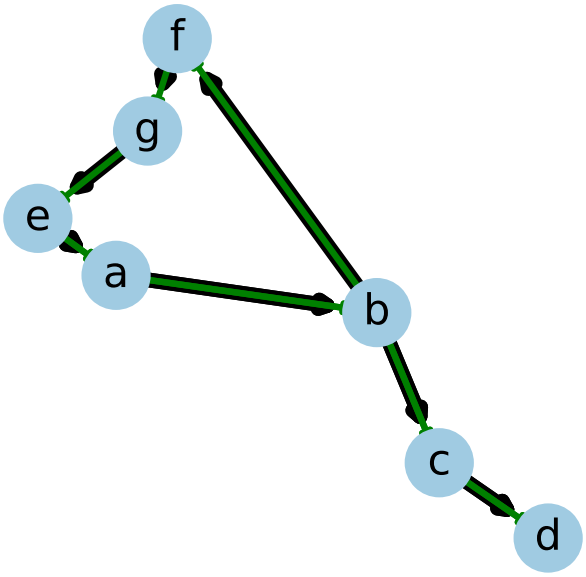

Figura 11: Multigrafo dirigido cíclico

12. Multigrafo dirigido reflexivo

Un ejemplo aplicado de este tipo de multigrafo es el diseño de la red de transporte urbano por la que van a transitar luego los camiones o vehículos de distribución. Los nodos son las ciudades, los puntos más importantes por los que pasan los vehículos de distribución y en ocasiones existe más de un camino para llegar al lugar y cada camino es una arista, que tiene varios atributos por ejemplo el tiempo de recorrido. Pueden existir varias aristas entre los nodos, lo que da lugar a un multigrafo.

Lo mismo ocurre con el diseño de rutas en el transporte urbano. En el ejemplo de la figura 12 los nodos son puntos de carga y descarga de pasajeros y se puede llegar por diferentes vías de la ciudad, hay calles principales y hay que determinar cuál es el más adecuado. Este tipo de multigrafo es reflexivo, porque en este tipo de diseño se toma un tiempo ajustar el itinerario de los vehículos y estos tienen que hacer un tiempo en las paradas, por lo que ese tiempo es un lazo al mismo nodo, convirtiendo al multigrafo, en multigrafo reflexivo.

En el ejemplo de la figura 12 se utilizó el algoritmo de acomodamiento circular layout para representar los nodos y arcos acorde al ejemplo aplicado. Este algoritmo tiene varios parámetros que lo caracterizan, entre los que se encuentran: el parámetro G (representa el grafo a visualizar), parámetro scale, center y dim (determina la dimensión del layout).

12.1. Código

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.MultiDiGraph()

G.add_node( "a", size=10 ,  pos=(1,60))
G.add_node( "b" , size=10 , pos=(40,80))
G.add_node( "c", size=10, pos=(80,20))
G.add_node( "d", size=10,  pos=(80,-50))
G.add_node( "e", size=10,  pos=(10,20))
G.add_node( "f", size=10, pos=(40,1))
G.add_node( "g", size=10, pos=(30,-50))
G.add_node( "h", size=10, pos=(50,30))
G.add_node( "i", size=10, pos=(30,30))
G.add_node( "i", size=10, pos=(30,30))

G.add_edge('a', 'b',  color='red',weight=8)
G.add_edge('a', 'b',  color='black',weight=3)
G.add_edge('b', 'c',  color='red',weight=8)
G.add_edge('b', 'c',  color='black',weight=3)
G.add_edge('c', 'd',  color='red',weight=8)
G.add_edge('c', 'd',  color='black',weight=3)
G.add_edge('b', 'f',  color='black',weight=3)
G.add_edge('f', 'g',  color='black',weight=4)
G.add_edge('g', 'e',  color='black',weight=3)
G.add_edge('f', 'h',  color='black',weight=5)
G.add_edge('f', 'i',  color='black',weight=3)
pos=nx.circular_layout(G)
size = [1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000]
color=["#A0CBE2", "#A0CBE2", "#A0CBE2", "#FF0000", "#A0CBE2", "#A0CBE2",
    "#A0CBE2", "#A0CBE2", "#A0CBE2"]
```

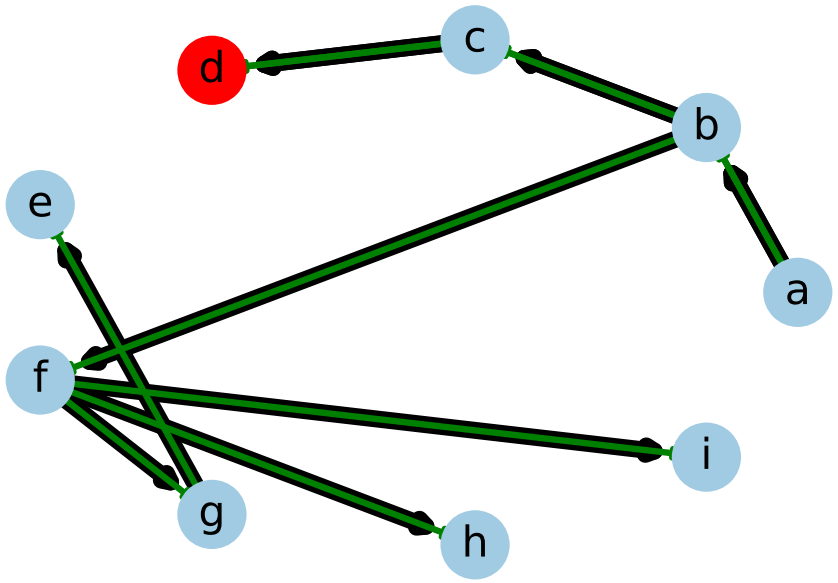

Figura 12: Multigrafo dirigido reflexivo

Referencias

[1] R. K. Ahuja. *Network Flows: Theory, Algorithms, and Applications*. Pearson Education, primera edition, 2017.

[2] Alfonso López Céspedes. *Diseño de una metodología de automatización y control para los procesos de dosificación, mezcla y carga en una planta de asfalto*. 2007.

[3] Departamento de Lenguajes y Sistemas Informáticos. *Base dea Datos Tema 4 Modelo Entidad/Interrelación*. 2005.

[4] Agencia Estatal de Transporte. *Plan Sectorial de Transporte y Vialidad del Área Metropolitana de Monterrey 2008-2030*. 1998.

[5] Pablo Solis Fernández. *Modificación superficial de materiales de carbono: grafito y grafeno*. 2011.

[6] Silvia Acosta Gnass. *Manual de esterilización para centros de salud*. 2008.

[7] Matplotlib developers. `https://matplotlib.org`.

[8] Matplotlib developers. `https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.drawing.layout.spring_layout.html`.

[9] Matplotlib developers. https://networkx.github.io/documentation/stable/reference/generated/networkx.drawing.layout.circular_layout.html#networkx.drawing.layout.circular_layout.

[10] Networkx developers con última actualización el 19 de Septiembre 2018. https://networkx.github.io/documentation/stable.

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

[12] E. Schaeffer. *Análisis de Algoritmos. Teoría de grafos.* 2014.

[13] José Hernando Ávila Toscano. *Redes sociales y análisis de redes.* 2005.