Tarea No.1: Rectificada

Dayli Machado de Armas (5275)

3 de junio de 2019

1. Gráfo simple no dirigido acíclico

Un grafo simple no dirigido acíclico, es aquel que no posee dirección en sus aristas, ni bucles, y por supuesto no es reflexivo [3].

Este tipo de grafos suele representarse por una secuencia de nodos unidos por aristas, donde de cada nodo generalmente solo sale, o llega una arista [1].

En la práctica este tipo de grafos puede emplearse para representar estructuras moleculares que sigan este comportamiento, por ejemplo, la representación de moléculas. Un ejemplo de estas moléculas sería la representación de alcanos de tipo lineal, donde los átomos serían los nodos y los enlaces entre estos las aristas. Este tipo de moléculas seleccionadas no poseen ciclos, ni bucles en su representación. Ver figura 1 en la página 2 donde se muestra la representación gráfica del mismo.

```
import matplotlib.pyplot as plt
import networkx as nx
H=nx.Graph()
H.add_path([0,1,2,3,4,])
nx.draw (H)

plt.savefig("imagenes/Fig01.eps")
```

Practica.py

2. Gráfo simple no dirigido cíclico

Un grafo simple no dirigido cíclico es aquel que no posee dirección en sus aristas, pero si se forma un ciclo que represente una figura cerrada que comience y termine en el mismo vértice, entonces es llamdo cíclico [2]. En los grafos no dirigidos, el flujo puede fluir en ambas direcciones.

Este tipo de grafos suele representarse por una secuencia de nodos unidos por aristas, pero estos pueden tener dentro un ciclo, o ser un ciclo en sí mismo. En la práctica, poseen múltiples aplicaciones, por ejemplo en la topología de redes circular, es un ejemplo de aplicación de este tipo de

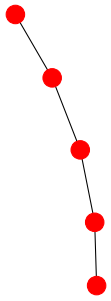

Figura 1: Representación de grafo simple no dirigido acíclico

grafo, otro ejemplo pudiera ser la representación de las autopistas de una región dada, o también la representación de otro tipo de moléculas de alcanos, por ejemplo en las de alcanos que se representan como ciclos, siendo el caso del ciclo pentano por ejemplo con cinco nodos unidos por aristas de manera circular.

El ejemplo a seguir son las llamadas que se realizan interdepartamentales en una muestra de tiempo de la jornada laboral en una empresa determinada, supongamos el departamento de producción, de marketing, comercial, calidad y planificación de la producción, donde cada nodo representa un departamento dentro de la empresa o área de esta, y las aristas las llamadas que se realizan, estas pueden tener un comportamiento que al graficarlo represente un grafo simple no dirigido cíclico. Su utilidad pudiera ser analizar el comportamiento del grafo para minimizar la cantidad de llamadas, o el tiempo de estas por ejemplo. Ver figura 2 en la página 3 donde se muestra la representación gráfica del mismo.

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

G = nx.Graph()

G.add_nodes_from([0,1,2,3,4])
pos = nx.spring_layout(G)

G.add_edge(0,1)
G.add_edge(0,3)
G.add_edge(1,4)
G.add_edge(1,2)
G.add_edge(3,1)
G.add_edge(0,4)

nx.draw_networkx_nodes(G, pos, node_size=2000, node_color='y', node_shape='o')
nx.draw_networkx_edges(G, pos, width=1, alpha=0.5, edge_color='r')


labels = {}
labels[0] = r'Markt'
labels[1] = r'Prod'
labels[2] = r'Cald'
labels[3] = r'Comerc'
labels[4] = r'Planif'
```

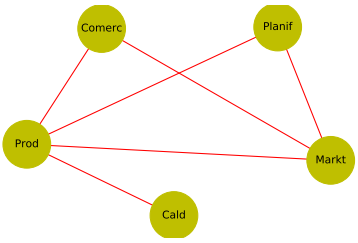

Figura 2: Representación de grafo simple no dirigido cíclico

```
26
27 plt.axis('off')
28
29 nx.draw_networkx_labels(G, pos, labels, font_size=10)
30 plt.savefig("imagenes/Fig02.eps")
31
32 plt.show()
```

grafo2.py

3. Gráfo simple no dirigido reflexivo

Un grafo simple en el cual no se permiten aristas multiples, además será no dirigido y reflexivo, si no poseen dirección en sus aristas y reflexivo si cuenta al menos con un bucle, lo que se refiere a una arista reflexiva en la que coinciden el vértice de origen y el vértice de destino [3].

Este grafo pudiera emplearse para representar el comportamiento del entrecruzamiento entre diferentes grupos raciales, donde cada grupo representa un nodo diferente, pudiendo existir cruzamiento entre ellos, y las relaciones entre los grupos son los hijos que tengan, es una relación no dirigida, a su vez dentro del grupo de la misma raza se pueden relacionar entre ellos mismos y estarían ocurriendo lazos reflexivos. Si se consideran los cuatro grupos representativos de las razas según la clasificación de especialistas serían: Blanco/caucásico, asiático/mongoloide, negroide/negro y australoide, esta clasificación posee dentro cada grupo una clasificación de hasta 30 subgrupos, pero solo se representarán los cuatro principales. Ver figura 3 en la página 4 donde se muestra la representación gráfica del mismo.

```
1 import matplotlib.pyplot as plt
2 import networkx as nx
3
4 G = nx.Graph()
5 G.add_nodes_from([1,2,3,4])
6
7 node_b = {1}
8 node_a = {2}
9 node_n = {3}
```

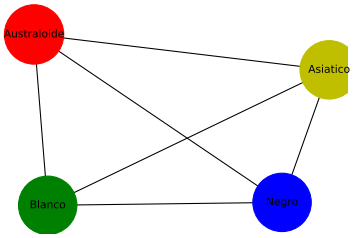

Figura 3: Representación de grafo simple no dirigido reflexivo

```
node_au = {4}

pos = nx.spring_layout(G)
#pos = {1:(5000,8020), 2:(7150,6620), 3:(6675, 5180), 4:(5475,8200)}

G.add_edges_from([(1,1), (1,2), (1,3), (1,4)])# Cada nodo es reflexivo pero no me sale la flechita
G.add_edges_from([(2,2), (2,1), (2,3), (2,4)])#igualmente no sale el reflexivo en si mismo
G.add_edges_from([(3,3), (3,1), (3,2), (3,4)])#no sale el reflexivo en si mismo
G.add_edges_from([(4,4), (4,1), (4,3), (4,2)])#es reflexivo en si mismo

nx.draw_networkx_nodes(G, pos, nodelist=node_b, node_size=3000, node_color='g', node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_a, node_size=3000, node_color='y', node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_n, node_size=3000, node_color='b', node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_au, node_size=3100, node_color='r', node_shape='o')

nx.draw_networkx_edges(G, pos, width=1, alpha=0.8, edge_color='black', )

labels = {}
labels[1] = r'Blanco'
labels[2] = r'Asiatico'
labels[3] = r'Negro'
labels[4] = r'Australoide'

plt.axis('off')

nx.draw_networkx_labels(G, pos, labels, font_size=10)

#plt.xlim (2000,16000)
#plt.autoscale (enable=true, axis='both')
plt.savefig("imagenes/Fig03.eps", bbox_inches='tight')
#nx.draw(G)
plt.show()
```

grafo3.py

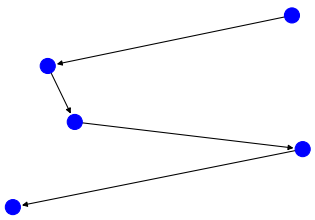

Figura 4: Representación de gráfo simple dirigido acíclico

4. Gráfo simple dirigido acíclico

Un grafo simple dirigido acíclico, es aquel que posee una dirección en sus aristas, y no posee bucles o reflexividad, ni ciclos[3].

En este caso, ejemplos de la vida real son aquellos que posean un origen del que puede salir una o varias aristas por diferentes caminos, sin que entre ellas existan ciclos y tengan un destino final diferente al del origen. Los árboles genealógicos, los organigramas en las empresas, el flujo de procesos industriales que no posean ciclos, que posean un inicio y un fin, pudieran resultar ser ejemplos de la vida real en los que se puede aplicar este tipo de grafos. Ver figura 4 en la página 5 donde se muestra la representación gráfica del mismo.

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

G = nx.DiGraph()

G.add_node(1)
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(5)

G.add_edge(1,2)
G.add_edge(2,3)
G.add_edge(3,4)
G.add_edge(4,5)
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size=200, node_color='b', node_shape='o')
nx.draw_networkx_edges(G, pos, width=1, alpha=0.8, edge_color='black')
plt.axis('off')

plt.savefig("imagenes/Fig04.eps")
plt.show()
```

grafo4.py

5. Gráfo simple dirigido cíclico

Un grafo simple dirigido cíclico es aquel que posee una dirección en sus aristas, y posee una figura cerrada que comienza y termina en el mismo vértice.

En este caso, ejemplos de la vida real son aquellos que posean un origen del que puede salir una o varias aristas por diferentes caminos, y que en un determinado nodo regresen a alguno anterior de modo que se forme uno o más ciclos. El destino final deberá ser diferente al del origen.

Como aplicación real de este tipo de grafo, puede ser el flujo de procesos industriales o artesanales que posean ciclos, por ejemplo: la elaboración artesanal de jugo de naranja, al representar este proceso en un flujograma mediante un diagrama OPERIN, en el que se reflejen sus operaciones. Una vez que se llegue al paso de exprimir las naranjas que se cae en un ciclo volviendo a la operación anterior de seleccionar otra y otra naranja hasta cubrir toda la capacidad del extractor de jugo. En este ejemplo las operaciones son los nodos y los enlaces entre los nodos la transformación que va teniendo la naranja durante el proceso. Pudiera representarse como se muestra en la figura 5 en la página 7 donde se muestra la representación gráfica del mismo.

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

G = nx.DiGraph()

G.add_node(1)
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(5)
G.add_node(6)

pos = {1:(100, 0), 2:(150,0), 3:(200, 0), 4:(250,0), 5:(300,0), 6:(350,0)}

G.add_edge(1,2)
G.add_edge(2,3)
G.add_edge(3,4)
G.add_edge(4,5)
G.add_edge(5,6)

G.add_cycle([5,4])
#G.add_cycle (5,4)

nx.draw_networkx_nodes(G, pos, node_size=600, node_color='r', node_shape='o')
nx.draw_networkx_edges(G, pos, width=1, alpha=0.8, edge_color='black')

labels = {}
labels[1] = r'$Op1$'
labels[2] = r'$Op2$'
labels[3] = r'$Op3$'
labels[4] = r'$Op4$'
labels[5] = r'$Op5$'
labels[6] = r'$Op6$'

nx.draw_networkx_labels(G, pos, labels, font_size=10)

```

Figura 5: Representación de gráfo simple dirigido cíclico

```
37 plt.axis('off')
38
39 plt.savefig("imagenes/Fig05.eps")
40
41 plt.show()
```

grafo5.py

6. Gráfo simple dirigido reflexivo

La diferencia con este grafo y el anterior reflexivo visto, es que en este caso las aristas si tienen sentido y el flujo que se analice debe ir en una dirección. En este caso deberá existir un nodo que se llame a sí mismo, o sea, que posea al menos un bucle.

Un ejemplo en el que pudiera emplearse la teoría de grafos, y que pudiera seguir un comportamiento dirigido reflexivo es al analizar entre personas de grupos sanguíneos diferentes, representar de quien pueden recibir sangre en función del tipo que posean. La reflexividad está dada en los casos en que cada grupo sanguíneo puede recibir de otros grupos y de sí mismo, excepto el grupo O- que solo puede recibir del mismo tipo.

Pudiera representarse como se muestra en la figura 6 en la página 9 donde se muestra la representación gráfica del mismo.

```
1 import matplotlib.pyplot as plt
2 import networkx as nx
3
4 G = nx.DiGraph()
5
6 G.add_node(1)
```

```
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(5)
G.add_node(6)
G.add_node(7)
G.add_node(8)

node_a = {1,2}
node_b = {3,4}
node_ab = {5,6}
node_o = {7,8}

pos = nx.spring_layout(G)

G.add_edges_from([(1,1), (8,1), (7,1), (2,1)])# Cada nodo es reflexivo pero no me sale la flechita
G.add_edges_from([(2,2), (8,2)])#igualmente no sale el reflexivo en si mismo
G.add_edges_from([(3,3), (8,3), (7,3), (4,3)])#no sale el reflexivo en si mismo
G.add_edges_from([(4,4), (8,4)])#es reflexivo en si mismo
G.add_edges_from([(5,5), (1,5), (2,5), (3,5), (4,5), (6,5), (7,5), (8,5)])# es reflexivo en si mismo
G.add_edges_from([(6,6), (4,6), (8,6), (2,6)])
G.add_edges_from([(7,7), (8,7)])
G.add_edges_from([(8,8)])

nx.draw_networkx_nodes(G, pos, nodelist=node_a, node_size=800, node_color='g', node_shape='o', alpha=0.3)
nx.draw_networkx_nodes(G, pos, nodelist=node_b, node_size=800, node_color='y', node_shape='o', alpha=0.3)
nx.draw_networkx_nodes(G, pos, nodelist=node_ab, node_size=800, node_color='b', node_shape='o', alpha=0.3)
nx.draw_networkx_nodes(G, pos, nodelist=node_o, node_size=800, node_color='r', node_shape='o', alpha=0.8)
#nx.draw_networkx_nodes(G, pos, nodelist=node_o, node_size=800, node_color='r', node_shape='on', alpha=0.1)
nx.draw_networkx_edges(G, pos, width=1, alpha=0.8, edge_color='black', )

labels = {}
labels[1] = r'A+'
labels[2] = r'A-'
labels[3] = r'B+'
labels[4] = r'B-'
labels[5] = r'AB+'
labels[6] = r'AB-'
labels[7] = r'O+'
labels[8] = r'O-'

plt.axis('off')
nx.draw_networkx_labels(G, pos, labels, font_size=10)
plt.savefig("imagenes/Fig06.eps")
plt.show()
```

grafo6.py

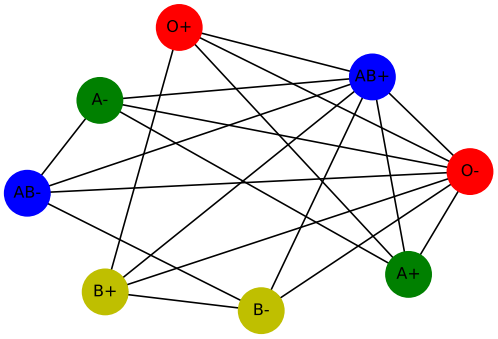

Figura 6: Representación de gráfo simple dirigido reflexivo

7. Multigrafo no dirigido acíclico

Cuando haya más de una arista entre un par de vértices, el grafo se llama multígrafo. Si no se permiten aristas múltiples, el grafo es simple[3]. Es no dirigido cuando las aristas no poseen sentido restringido, sino que el flujo fluye en ambas direcciones, y en este caso no presenta ciclos.

Un ejemplo práctico de multígrafo no dirigido acíclico pudiera decirse que es la representación de determinados enlaces moleculares, como ejemplo pudiera citarse el tipo de elemento que posea un enlace doble éster. Para la representación de los elementos con este tipo de enlace puede emplearse el grafo en cuestión. Específicamente pudiera decirse que la representación de éster sulfúrico pudiera representarse con este comportamiento, como se muestra en la figura 7 en la página 11 donde se muestra la representación gráfica del mismo.

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

G=nx.MultiGraph()

G.add_node(1)
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(5)
G.add_node(6)
G.add_node(7)

node_o = {4,5,6,7}
node_r = {1,2,3}

G.add_edge(4,3, weight=3)
G.add_edge(4,3, weight=5)
G.add_edge(3,7, weight=3)
```

```
G.add_edge(3,7, weight=5)
G.add_edge(1,5)
G.add_edge(5,3)
G.add_edge(3,6)
G.add_edge(6,2)

blue=[(4,3),(3,7),(1,5),(5,3),(3,6),(6,2)]
red=[(4,3),(3,7)]
pos = {1:(0,220), 2:(150,220), 3:(75, 180), 4:(75,200), 5:(25,180),6:(125,180),
    7:(75,160)}
nx.draw_networkx_nodes(G, pos, nodelist=node_o, node_size=600, node_color='b',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_r, node_size=600, node_color='y',
    node_shape='o')
nx.draw_networkx_edges(G, pos, edgelist=blue,width=6, alpha=0.5,
edge_color='b', style='dashed')
nx.draw_networkx_edges(G, pos, edgelist=red,width=6, alpha=0.5,
edge_color='r')

labels = {}
labels[1] = r'R1'
labels[2] = r'R2'
labels[3] = r'S'
labels[4] = r'O'
labels[5] = r'O'
labels[6] = r'O'
labels[7] = r'O'

plt.axis('off')
nx.draw_networkx_labels(G, pos, labels, font_size=10)
pos = nx.spring_layout(G)
plt.savefig("imagenes/Fig07.eps")
plt.show()
```

grafo7.py

8. Multigrafo no dirigido cíclico

Cuando haya más de una arista entre un par de vértices, el grafo se llama multígrafo. En este caso será no dirigido cuando las aristas no poseen sentido restringido, sino que el flujo fluye en ambas direcciones, y deberán existir zonas cerradas, o sea, que regresen al nodo inicial.

Este tipo de grafo es de gran utilidad para representar el comportamiento de redes sociales y neuronales por ejemplo. El ejemplo práctico que se me ocurre mostrar, es cuando tenemos necesidad de comunicarnos con alguien y las vías de comunicación que tenemos en la actualidad, donde cada vía de comunicación seleccionada tendrá un costo (peso), y existen muchas vías para comunicarnos con otras personas, y a su vez para que esa persona se comunique con nosotros, por lo que no es dirigido, pero sí cíclico, debido a los lazos que se pueden formar entre las personas que necesitan comunicarse. El ejemplo se restringe a solo dos vías de comunicación, ya sea por whatsap, o por llamadas telefónicas, aunque en la práctica existen muchas más. Este grafo se muestra en la figura 8 en la página 12 donde se muestra la representación gráfica del mismo.

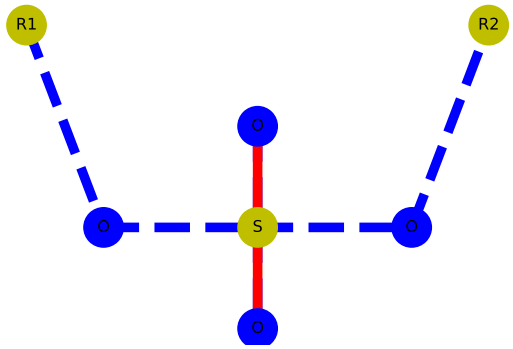

Figura 7: Representación de multigrafo no dirigido acíclico

```
1
2
3  import matplotlib.pyplot as plt
4  import networkx as nx
5
6  G = nx.MultiGraph ()
7
8  G.add_node(1)
9  G.add_node(2)
10 G.add_node(3)
11 G.add_node(4)
12 G.add_node(5)
13 E=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
14 for e in E:
15     (u,v)=e
16     G.add_edge(u,v,3)
17     G.add_edge(u,v,4)
18 green=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
19 yelow=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
20 pos = nx.spring_layout(G)
21 plt.axis('off')
22 nx.draw_networkx_nodes(G, pos ,node_size=400, node_color='black', node_shape='o')
23 nx.draw_networkx_edges(G, pos, edgelist=green, width=3, alpha=0.5,
24 edge_color='g', style='dashed')
25 nx.draw_networkx_edges(G, pos, edgelist=yelow,width=4, alpha=0.5,
26 edge_color='y')
27 plt.savefig("imagenes/Fig08.eps")
28 plt.show()
```

grafo8.py

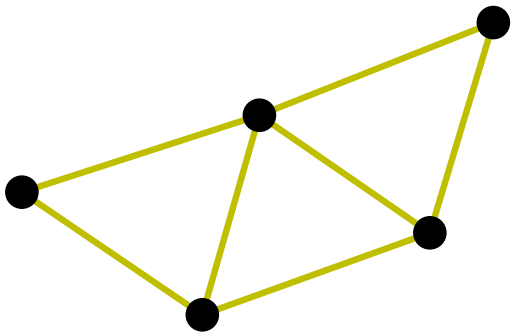

Figura 8: Representación de Multigrafo no dirigido cíclico

9. Multigrafo no dirigido reflexivo

Este caso se refiere a un multigrafo en el existe más de una arista entre un par de vértices, sin dirección entre estas, pero de un nodo deben salir más de una arista que regresen al mismo nodo, sin pasar por otro.

Una aplicación pudiera ser la relación que existe entre cinco signos zodiacales, donde cada nodo representa a las personas de un signo determinado y de cada uno de ellos puede salir una arista que represente las relaciones positivas entre los diferentes signos y otra las relaciones negativas, y a su vez las personas del mismo signo se relacionarán con sus iguales de manera positiva o negativa, lo que representa la reflexibilidad. Este grafo se muestra, para una muestra de cinco signos cualesquiera, en la figura 9 en la página 13 donde se muestra la representación gráfica del mismo.

```
1
2  import matplotlib.pyplot as plot
3  import networkx as nx
4
5  G = nx . MultiGraph ()
6
7  G.add_node(1)
8  G.add_node(2)
9  G.add_node(3)
10 G.add_node(4)
11 G.add_node(5)
12
13 E=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
14 for e in E:
15     (u,v)=e
16     G.add_edge(u,v,3)
17     G.add_edge(u,v,4)
18 node_A = {1}
19 node_S = {2}
```

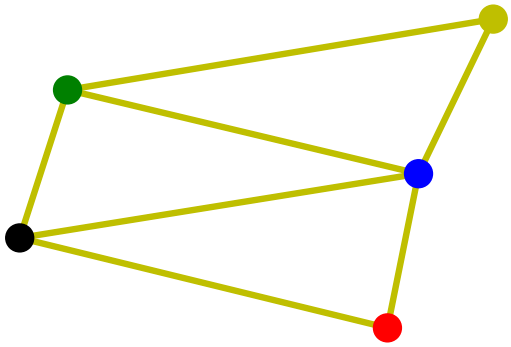

Figura 9: Representación de Multigrafo no dirigido reflexivo

```
node_E = {3}
node_V = {4}
node_C = {5}
green=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
yelow=[(1,2),(2,3),(3,4),(4,5),(5,1),(1,3),(5,3)]
pos = nx.spring_layout(G)
plt.axis('off')
nx.draw_networkx_nodes(G, pos, nodelist=node_A, node_size=300, node_color='g',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_S, node_size=300, node_color='y',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_E, node_size=300, node_color='b',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_V, node_size=300, node_color='r',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=node_C, node_size=300, node_color='black',
    node_shape='o')

nx.draw_networkx_edges(G, pos, edgelist=green, width=3, alpha=0.5,
edge_color='g', style='dashed')
nx.draw_networkx_edges(G, pos, edgelist=yelow,width=4, alpha=0.5,
edge_color='y')

plt.savefig("imagenes/Fig09.eps")

plt.show()
```

grafo9.py

10. Multigrafo dirigido acíclico

Para este tipo de grafo deben existir más de una arista entre un par de nodos, con dirección y no debe formarse ninguna figura cerrada dentro del grafo.

Este ejemplo aplicado a la práctica pudiera ser la representación de un viajero que desea ir de una ciudad a otra para visitarlas y tiene que elegir entre varias rutas posibles para llegar, o pudiera elegir también entre diferentes medios de transporte para trasladarse entre las ciudades, o una combinación de ambos. En este caso las ciudades serían los nodos, y las rutas o medios de transporte posibles a elegir entre un nodo u otro serían las aristas. Una ruta pudiera ser ir de la ciudad de Matanzas en Cuba, a la Habana, de esta a Monterrey, de Monterrey a Torreón y de ahí a Sinaloa. Este grafo se muestra en la figura 10 en la página 15 donde se muestra la representación gráfica del mismo.

```
1
2
3  import matplotlib.pyplot as plt
4  import networkx as nx
5
6  G = nx.MultiDiGraph()
7
8  G.add_node(1)
9  G.add_node(2)
10 G.add_node(3)
11 G.add_node(4)
12 G.add_node(5)
13
14 G.add_edge(1,2, weight=3)
15 G.add_edge(1,2, weight=4)
16 G.add_edge(2,3, weight=3)
17 G.add_edge(2,3, weight=4)
18 G.add_edge(3,4, weight=3)
19 G.add_edge(3,4, weight=4)
20 G.add_edge(4,5, weight=3)
21 G.add_edge(4,5, weight=4)
22
23 medio=[(1,2),(2,3),(3,4),(4,5)]
24 ruta=[(1,2),(2,3),(3,4),(4,5)]
25
26 pos = nx.spring_layout(G)
27
28 nx.draw_networkx_nodes(G, pos, node_size=500, node_color='b', node_shape='o')
29
30 nx.draw_networkx_edges(G, pos, edgelist=medio, width=6, alpha=0.8,
31 edge_color='black', style='dashed')
32
33 nx.draw_networkx_edges(G, pos, edgelist=ruta,width=4, alpha=0.5,
34 edge_color='r')
35
36 labels = {}
37 labels[1] = r'Mat'
38 labels[2] = r'Hab'
39 labels[3] = r'Monty'
40 labels[4] = r'Torr'
41 labels[5] = r'Sing'
42
```

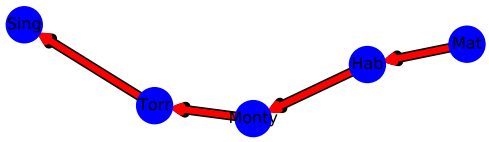

Figura 10: Representación de multigrafo dirigido acíclico

```
43 plt.axis('off')
44
45 nx.draw_networkx_labels(G, pos, labels, font_size=10)
46
47 plt.savefig("imagenes/Fig10.eps")
48 plt.show()
```

grafo10.py

11. Multigrafo dirigido cíclico

Para ser multigrafo dirigido cíclico debe existir más de una arista entre un par de nodos, con dirección y además formarse al menos una figura cerrada dentro del grafo.

Un ejemplo que representa este tipo de grafo sería la representación del flujo que siguen los pacientes una vez que asisten a la consulta de cuerpo de guardia del ISSSTE, en la que las operaciones del proceso serían los nodos y la clasificación del tipo de paciente y su recorrido dentro de la instalación las aristas. En este caso el cliente al llegar pasa por la consulta de clasificación y ahí le asignan un color en función de la gravedad de su dolencia, se asumen tres colores, rojo, amarillo y verde, que van de mayor gravedad a menor respectivamente y de esta dependerá el tiempo de espera para pasar a la siguiente consulta donde se encuentra el doctor, existe tres consultas con doctores disponibles y de ahí pueden pasar al laboratorio, u a otras de las salas. Luego, del laboratorio pueden pasar nuevamente a la consulta del doctor para revisar los resultados y ocurriría un ciclo, o del doctor directo a la sala, luego al laboratorio y retornar nuevamente al doctor, y ocurriría otro ciclo. Este grafo se muestra en la figura 11 en la página 17.

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

```
import networkx as nx

G = nx.MultiDiGraph()

G.add_node(1)
G.add_node(2)
G.add_node(3)
G.add_node(4)
G.add_node(5)

G.add_edge(1,2, weight=3)
G.add_edge(1,2, weight=4)
G.add_edge(1,2, weight=5)
G.add_edge(2,3, weight=3)
G.add_edge(2,3, weight=4)
G.add_edge(2,4, weight=3)

G.add_edge(2,5, weight=3)
G.add_edge(3,2, weight=4)
G.add_edge(3,2, weight=3)


y=[(1,2),(2,3),(3,2)]
g=[(1,2),(2,3),(3,2)]
r=[(1,2),(2,4),(2,5)]

pos = nx.spring_layout(G)

nx.draw_networkx_nodes(G, pos, node_size=400, node_color='b', node_shape='o')

nx.draw_networkx_edges(G, pos, edgelist=y, width=10, alpha=0.2,
edge_color='y', style='dashed')

nx.draw_networkx_edges(G, pos, edgelist=g, width=6, alpha=0.5,
edge_color='g')

nx.draw_networkx_edges(G, pos, edgelist=r, width=2, alpha=0.8,
edge_color='r')

labels = {}
labels[1] = r'1'
labels[2] = r'2'
labels[3] = r'Lab'
labels[4] = r'3'
labels[5] = r'4'


plt.axis('off')

nx.draw_networkx_labels(G, pos, labels, font_size=10)

plt.savefig("imagenes/Fig11.eps")
plt.show()
```

grafo11.py

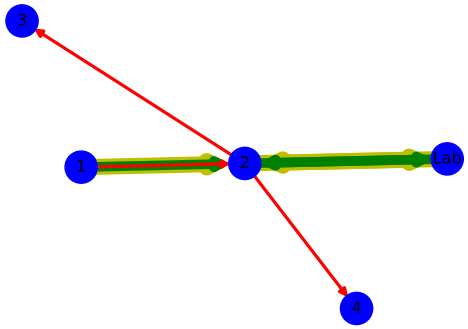

Figura 11: Representación de multigrafo dirigido cíclico

12. Multigrafo dirigido reflexivo

Para ser un multigrafo dirigido reflexivo debe existir más de una arista entre un par de nodos, con dirección y además cada nodo debe llamarse a sí mismo.

Un ejemplo en el que el empleo de la teoría de grafos como los que se analizan en esta sección puede emplearse es en el estudio de la influencia de la religión en la sociedad, como un subejemplo de la aplicación en redes sociales [2]. Por ejemplo, si cada nodo es una persona religiosa creyente, ese nodo (persona) constantemente se realiza un autoexamen de conciencia a sí mismo, evaluando cómo ha sido su actitud con respecto a las variables a analizar en este autoexamen. Estas variables pueden ser operacionalizadas como cada una de las doctrinas de cada religión, las cuales varían de una religión a otra, pero son más de una y pueden agruparse por categorías. Estas variables representan las aristas, cada una puede tener un peso determinado según sea el interés del estudio, a su vez estas mismas doctrinas (convertidas en variable) serán las que cada persona creyente se encargará de transmitir a otras personas creyentes o no, pudiera decirse que se transmite el tipo de actitud que se debe tener ante cada una de estas variables de una persona a otra.

En este ejemplo se puede evaluar el impacto en un grupo poblacional que puede tener un tipo de creencia u otro, en la medida en que aumente la red. Una simplificación de este tipo de grafo se representa en la figura 12 en la página 19 donde se muestra la representación gráfica del mismo.

```
1
2 import matplotlib.pyplot as plt
3 import networkx as nx
4
5 G = nx.MultiDiGraph()
6
7 G.add_node(1)
8 G.add_node(2)
9 G.add_node(3)
10 G.add_node(4)
```

```
G.add_node(5)

a= {1}
b= {2}
c= {3}
d= {4}
e= {5}

G.add_edge(1,2, weight=3)
G.add_edge(1,2, weight=4)
G.add_edge(1,2, weight=5)
G.add_edge(2,3, weight=3)
G.add_edge(2,3, weight=4)
G.add_edge(2,3, weight=4)
G.add_edge(3,4, weight=3)
G.add_edge(3,4, weight=3)
G.add_edge(3,4, weight=3)
G.add_edge(4,5, weight=3)
G.add_edge(4,5, weight=3)
G.add_edge(4,5, weight=3)


y=[(1,2),(2,3),(3,4),(4,5)]
g=[(1,2),(2,3),(3,4),(4,5)]
r=[(1,2),(2,3),(3,4),(4,5)]

pos = nx.spring_layout(G)

nx.draw_networkx_nodes(G, pos, nodelist=a, node_size=300, node_color='black',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=b, node_size=300, node_color='y', node_shape
    ='o')
nx.draw_networkx_nodes(G, pos, nodelist=c, node_size=300, node_color='b', node_shape
    ='o')
nx.draw_networkx_nodes(G, pos, nodelist=d, node_size=300, node_color='r', node_shape
    ='o')
nx.draw_networkx_nodes(G, pos, nodelist=e, node_size=300, node_color='g', node_shape
    ='o')

nx.draw_networkx_edges(G, pos, edgelist=y, width=8, alpha=1,
edge_color='y', style='dashed')

nx.draw_networkx_edges(G, pos, edgelist=g, width=6, alpha=0.5,
edge_color='g')

nx.draw_networkx_edges(G, pos, edgelist=r, width=2, alpha=0.8,
edge_color='r')

plt.axis('off')

plt.savefig("imagenes/Fig12.eps")
plt.show()
```

grafo12.py

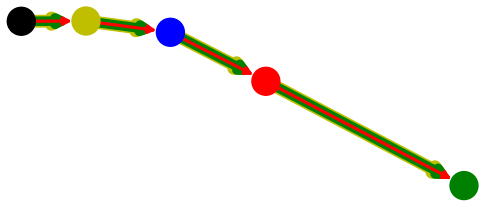

Figura 12: Representación de multigrafo dirigido reflexivo

Referencias

[1] Pieter Swart Aric Hagberg, Dan Schult. *NetworkX Reference, Release 2.3rc1.dev20190113142952*, 2019.

[2] Anwesha Chakraborty, Trina Dutta, Sushmita Mondal, and Asoke Nath. Application of graph theory in social media. *International Journal of Computer Sciences and Engineering*, 6, 10 2018. doi: 10.26438/ijcse/v6i10.722729.

[3] Elisa Schaeffer. *Complejidad computacional de problemas y el análisis y diseño de algoritmos*. 2017.