Tarea 2

5280

1 de junio de 2019

Introducción

La teoría de grafos tiene aplicación en diversas áreas, debido a que las redes aparecen prácticamente en todas las actividades de nuestra vida diaria: sistemas de comunicaciones, sistemas hidráulicos, circuitos eléctricos y electrónicos, sistemas mecánicos, redes sociales, entre otros.

Como constata Ravindra [2], las redes físicas son quizás las más comunes y mejor identificables. Los sistemas de transporte, cualquiera que sea el medio suelen modelarse en sistemas de distribución y decisiones logísticas complejos, siendo el problema de transporte un ejemplo típico de investigación de operaciones.

Otro ejemplo representativo de uso de grafos son los algoritmos de búsqueda web de Google. Estos se basan en el gráfico WWW, que contiene todas las páginas web como vértices y los hipervínculos como bordes [3].

La teoría de los grafos también es usada en química, debido a la posibilidad de representar los modelos estructurales mediante diagramas. En un grafo molecular, los vértices representan a los átomos y los lados a los enlaces químicos que conectan ciertas parejas de átomos [6].

Para acomodar los grafos, de manera que su forma refleje lo que se desea, se han desarrollo varios *layouts* : algoritmos que devuelven una lista de posiciones para los nodos, según parámetros definidos.

Entre los *layouts* más conocidos se destacan:

- *Bipartite layout*
- *Circular layout*
- *Kamada kawai layout*
- *Random layout*
- *Rescale layout*
- *Shell layout*
- *Spring layout*
- *Spectral layout*

Figura 1: Representación con un grafo de ocho estaciones de la Línea 2 del Metro de Monterrey.

1. Grafo simple no dirigido acíclico

La mayoría de los esquemas de líneas de autobuses, tranvías o trenes del transporte público pueden ser representados por grafos simples no dirigidos acíclicos.

En la figura 1 (página 2) se muestra una sección de ochos estaciones de la Línea 2 del Metro de Monterrey, donde las estaciones son los nodos y el tramo de línea entre ellos, los vértices.

Para posicionar los nodos de este grafo se empleó el *spring layout*. Se obtuvieron resultados igualmente satisfactorios empleando otros *layouts*.

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

G = nx.Graph()
G.add_edge('Sendero','Santiago Tapia')
G.add_edge('Santiago Tapia','San Nicolas')
G.add_edge('San Nicolas','Anahuac')
G.add_edge('Anahuac','Universidad')
G.add_edge('Universidad','Ninos Heroes')
G.add_edge('Ninos Heroes','Regina')
G.add_edge('Regina','General Anaya')

positions = nx.spring_layout(G, scale=.15)

nx.draw_networkx_nodes(G, positions, node_size=500, node_color='y')
nx.draw_networkx_edges(G, positions, width=1,alpha=1, edge_color='b')
nx.draw_networkx_labels(G, positions, font_size=12,font_family='sans-serif')
plot.axis('off')
plot.savefig("fig1.eps")
plot.show()
```

Ej1.py

2. Grafo simple no dirigido cíclico

Este tipo de grafos es adecuado para representar relaciones comerciales entre empresas al igual que relaciones de parentesco, amistad o romance entre personas. En la figura 2 (página 3) se muestra

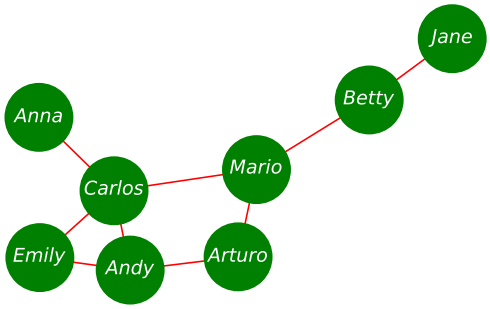

Figura 2: Relaciones de amistad en un grupo de personas.

un grafo que representa las relaciones de amistad en un grupo de diez personas. Los vértices son las personas y las aristas, las personas que tienen una relación de amistad.

Para posicionar este grafo se usó el *Fruchterman-Reingold layout*. Este fue diseñado para dibujar grafos no dirigidos, con el objetivo de distribuir los vértices uniformemente en el marco, haciendo que las longitudes de los bordes sean uniformes y reflejando la simetría [4].

```
1
2  import matplotlib.pyplot as plot
3  import networkx as nx
4
5  G = nx . Graph ()
6  G.add_edge(1,2)
7  G.add_edge(1,3)
8  G.add_edge(1,4)
9  G.add_edge(4,5)
10 G.add_edge(2,6)
11 G.add_edge(3,8)
12 G.add_edge(7,8)
13 G.add_edge(4,8)
14 G.add_edge(4,7)
15 pos=nx.fruchterman_reingold_layout(G,scale=.45)
16
17 nx.draw_networkx_nodes(G, pos, node_size=1800, node_color='g', node_shape='o')
18 nx.draw_networkx_edges(G, pos, width=1, alpha=0.5, edge_color='red')
19
20 labels = {}
21 labels[1] = r'$ Mario $'
22 labels[2] = r'$Betty$'
23 labels[3] = r'$Arturo$'
24 labels[4] = r'$Carlos$'
25 labels[5] = r'$Anna$'
26 labels[6] = r'$Jane$'
27 labels[7] = r'$Emily$'
28 labels[8] = r'$Andy$'
29
30 nx.draw_networkx_labels(G, pos, labels, font_size=12, font_color='w')
31
32 plot.axis('off')
33 plot.savefig("fig2.eps")
34
35 plot.show()
```

Ej2.py

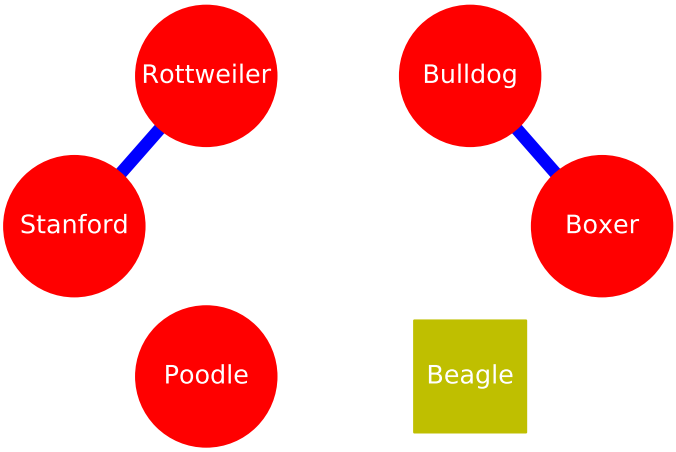

Figura 3: Estado de cría de razas de perro

3. Grafo simple no dirigido reflexivo

Un criador de perros posee seis razas diferentes de estos animales. Con un grafo de este tipo puede representar de cuales de estos animales ha obtenido crías, siendo los vértices las razas de perro y las aristas la unión entre razas que han tenido crías.

En la figura 3 se puede observar que todas las razas excepto la beagle, que es recién adquirida y aún no se ha reproducidos, han tenido crías con su misma raza (vértices en rojo). También se observa que han habido cruzamientos entre boxer y bulldog y entre pitbull y rottweiler.

Para posicionar este grafo se usó el *Kamada-Kawai layout*, diseñado para dibujar grafos ponderados no dirigidos [1]. Los *layouts circular* y *shell* mostraron también resultados similares.

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

G = nx.Graph()

G.add_edge(1,2)
G.add_edge(3,4)
G.add_node(5)
G.add_node(6)
apareados={1,2,3,4,5}
noApareado={6}
pos = nx.kamada_kawai_layout(G, scale=.45)

nx.draw_networkx_nodes(G, pos, nodelist=apareados, node_size=4400, node_color='r',
    node_shape='o')
nx.draw_networkx_nodes(G, pos, nodelist=noApareado, node_size=2800, node_color='y',
    node_shape='s')
nx.draw_networkx_edges(G, pos, width=6, alpha=0.6, edge_color='b')

labels = {}
labels[1] = r'Boxer'
labels[2] = r'Bulldog'
labels[3] = r'Rottweiler'
labels[4] = r'Stanford'
```

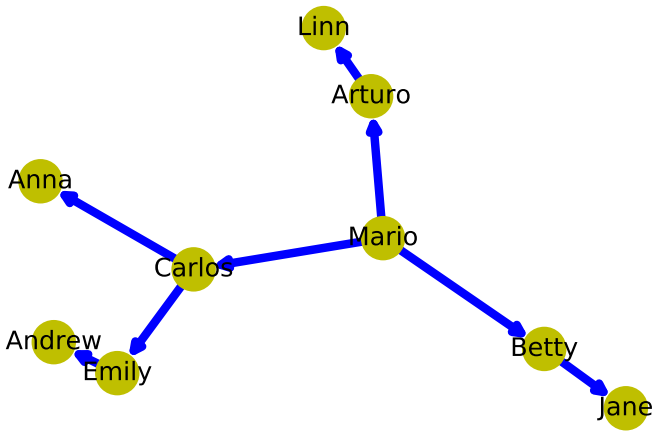

Figura 4: Representación de la transmisión de una ITS en un grupo de personas.

```
labels[5] = r'Poodle'
labels[6] = r'Beagle'

nx.draw_networkx_labels(G, pos, labels, font_size=12,font_color='w')

plot.axis('off')
plot.savefig("fig3.eps")
plot.show()
```

Ej3.py

4. Grafo simple dirigido acíclico

La cadena de propagación de las ITS (Infecciones de Transmisión Sexual) para las cuales no se conoce cura o las que solo afectan al individuo una vez en la vida, pueden representarse mediante grafos simples dirigidos acíclicos. En este caso cada sujeto es un vértice y la dirección de transmisión de la enfermedad es representada en la arista. La enfermedad se propaga desde la persona portadora hacia el sujeto sano, que posteriormente se convierte en portador y contagia a otros sujetos sanos.

De los *layouts* empleados para posicionar los elementos de la figura 4 el Fruchterman-Reingold fue el más adecuado.

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

G = nx.DiGraph()
G.add_edge(1,2)
G.add_edge(1,3)
G.add_edge(1,4)
G.add_edge(4,5)
G.add_edge(2,6)
G.add_edge(7,8)
G.add_edge(4,7)
G.add_edge(3,9)
pos = nx.fruchterman_reingold_layout(G,scale=.55,iterations=30)
nx.draw_networkx_nodes(G, pos, node_size=400, node_color='y', node_shape='o')
nx.draw_networkx_edges(G, pos, width=4, edge_color='b')

labels = {}
labels[1] = r'Mario'
labels[2] = r'Betty'
labels[3] = r'Arturo'
labels[4] = r'Carlos'
labels[5] = r'Anna'
labels[6] = r'Jane'
labels[7] = r'Emily'
labels[8] = r'Andrew'
labels[9] = r'Linn'

nx.draw_networkx_labels(G, pos, labels, font_size=12)
plot.axis('off')
plot.savefig("fig4.eps")
plot.show()
```

Ej4.py

5. Grafo simple dirigido cíclico

En ciudades con carreteras estrechas como Santiago de Cuba, en diferentes sectores, las carreteras son de un solo sentido y se representan con grafos de este tipo. Las aristas representan las calles y los vértices son las intersecciones entre al menos dos aristas.

Luego de probar varios *layouts*, el *spectral* es el que arroja el resultado deseado para dibujar la figura 5.

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

G = nx.DiGraph()

G.add_edge(1,2,calle='c')
G.add_edge(2,3,calle='c')
G.add_edge(5,4,calle='b')
G.add_edge(6,5,calle='b')
G.add_edge(8,9,calle='a')
G.add_edge(7,8,calle='a')
G.add_edge(1,4,calle='19')
G.add_edge(4,7,calle='19')
G.add_edge(8,5,calle='21')
G.add_edge(5,2,calle ='21')
G.add_edge(3,6,calle ='23')
G.add_edge(6,9,calle ='23')
```

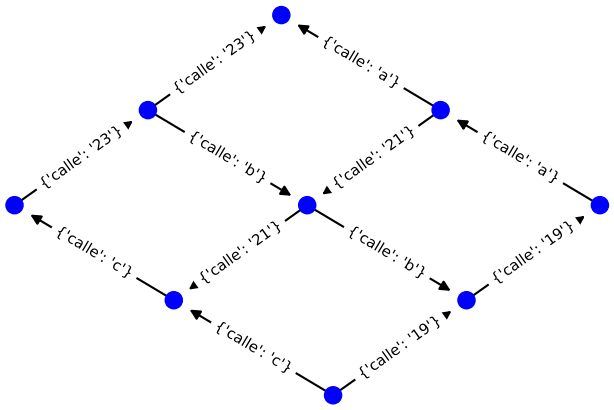

Figura 5: Carreteras de sentido único en Santiago de Cuba.

```
18
19 pos = nx.spectral_layout(G)
20 nx.draw_networkx_nodes(G, pos, node_size=60, node_color='b', node_shape='o')
21 nx.draw_networkx_edges(G, pos, width=1, edge_color='black')
22 edge_labels=nx.draw_networkx_edge_labels(G,pos,font_size=7,label_pos=.5)
23
24
25 plot.axis('off')
26 plot.savefig("fig5.eps")
27 plot.show()
```

Ej5.py

6. Grafo simple dirigido reflexivo

Un grafo en el que cada vértice es una empresa de servicios y las aristas, la unión con cada una de las otras empresas a las que le presta servicios, puede representarse mediante un grafo dirigido reflexivo, pues algunas de estas empresas se brindan servicio a ellas mismas.

Un sitio web pequeño, en el cual se accede a todas sus páginas a través de un menú estático también se puede representar con este tipo de grafo.

Otro ejemplo está dado por la representación de un grupo de personas conectadas a una red social y los perfiles de otras personas que tengan abiertos en el navegador en ese momento. Cada una de esas persona también pudiesen estar mirando su propio perfil. En el grafo de la figura 6 los vértices son las personas y las aristas indican el perfil de qué personas tiene cada uno abierto en el navegador. Puede apreciarse en color rojo que Mayra y Andrew tienen abiertos sus propios perfiles.

Para obtener la forma deseada del grafo ofrecen resultados similares el *circular layout* y el *shell*.

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

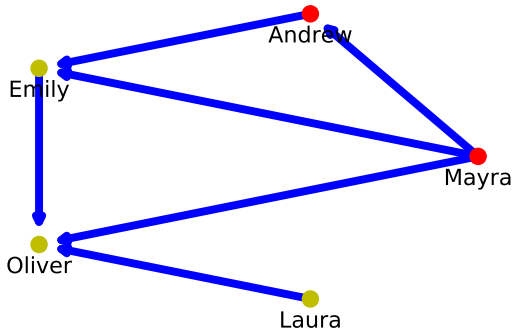

Figura 6: Representación de personas mirando perfiles en redes sociales.

```
6
7  G.add_edge(1,2)
8  G.add_edge(1,3)
9  G.add_edge(1,5)
10 G.add_edge(2,3)
11 G.add_edge(3,5)
12 G.add_edge(4,5)
13 node1 = {1,2}
14 node2 = {3,4,5}
15
16 pos = nx.circular_layout(G)
17
18 nx.draw_networkx_nodes(G, pos, nodelist=node1,node_size=100, node_color='r',
       node_shape='o',alpha=0.8)
19 nx.draw_networkx_nodes(G, pos, nodelist=node2,node_size=100, node_color='y',
       node_shape='o')
20 nx.draw_networkx_edges(G, pos,width=5, alpha=0.4, edge_color='b')
21
22 labels = {}
23 labels[1] = r'Mayra'
24 labels[2] = r'Andrew'
25 labels[3] = r'Emily'
26 labels[4] = r'Laura'
27 labels[5] = r'Oliver'
28
29 pos1=pos
30 for i in pos:
31     pos1[i][1] = pos1[i][1] -0.15
32 nx.draw_networkx_labels(G, pos1, labels, font_size=14)
33
34 plot.axis('off')
35 plot.savefig("fig6.eps")
36 plot.show()
```

Ej6.py

7. Multigrafo no dirigido acíclico

La ruta entre Santiago de Cuba y Holguín puede representarse como un multigrafo no dirigido cíclico. En un pueblo llamado Caballería la carretera principal se bifurca, pudiendo continuar por la ruta principal hasta Banes, o por el camino alternativo de Caballería, que, aunque es más largo,

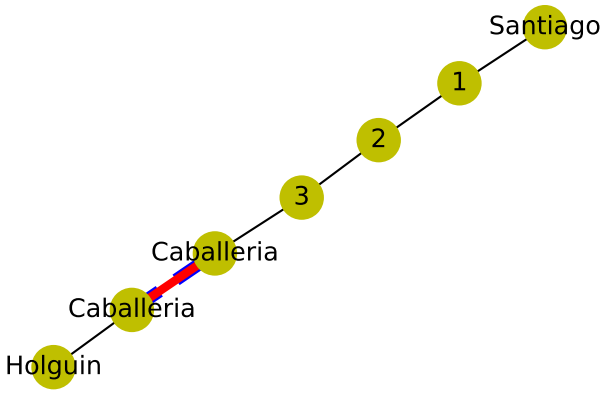

Figura 7: Posible ruta entre Santiago de Cuba y Holguín

puede recorrerse en menor tiempo a causa del poco tránsito.

En la figura 7 los vértices representan las cabeceras municipales, que están simplemente enumeradas por no ser de interés para el ejemplo mencionado. Para la visualización se empleó el *layout Kamada-Kawai*.

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

G = nx . MultiGraph ()


G.add_edge(1,2, weight=1)
G.add_edge(2,3 ,weight=1)
G.add_edge(3,4, weight=1)
G.add_edge(4,5, weight=1)
G.add_edge(5,6, weight=1)
G.add_edge(5,6,weight=2)
G.add_edge(6,7, weight=1)
black=[(1,2),(2,3),(3,4),(4,5),(6,7)]
blue=[(5,6)]
red=[(5,6)]
pos = nx.kamada_kawai_layout(G, scale=.2)

nx.draw_networkx_nodes(G, pos ,node_size=400, node_color='y', node_shape='o')
nx.draw_networkx_edges(G, pos,edgelist=black,width=1,edge_color='black', alpha=0.8)
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=4, alpha=0.5,
edge_color='r')
labels = {}
labels[1] = r'Santiago'
labels[2] = r'1'
labels[3] = r'2'
labels[4] = r'3'
labels[5] = r'Caballeria'
labels[6] = r'Caballeria'
labels[7] = r'Holguin'

nx.draw_networkx_labels(G, pos, labels, font_size=12)

```

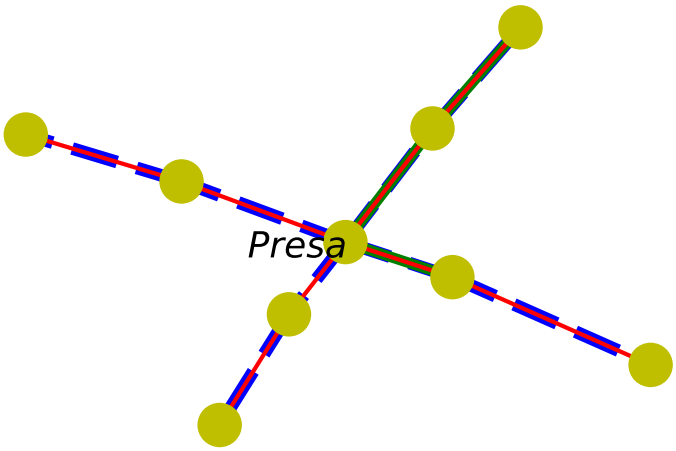

Figura 8: Canales para irrigación del suelo creados en la presa El cacao.

```
36 plot.axis('off')
37 plot.savefig("fig7.eps")
38 plot.show()
```

Ej7.py

8. Multigrafo no dirigido cíclico

En la presa El cacao, ubicada en el Municipio Cotorro, Ciudad de La Habana, se realizó un proyecto para favorecer la agricultura. Al rededor de la presa se construyeron una serie de canales para mantener el suelo húmedo durante todo el año. Este sistema puede ser representado mediante el grafo de la figura 8, el cual los vértices representan la unión entre uno o más canales y las aristas, cada uno de los canales. El *layout Kamada-Kawai* fue el que mejores resultados de visualización arrojó.

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

G = nx . MultiGraph ()

G.add_edge(1,2, weight=3)
G.add_edge(1,2, weight=5)
G.add_edge(1,2, weight=6)
G.add_edge(2,3, weight=4)
G.add_edge(2,3, weight=3)
G.add_edge(2,3, weight=5)
G.add_edge(1,4, weight=4)
G.add_edge(1,4, weight=3)
G.add_edge(4,5, weight=5)
G.add_edge(4,5, weight=3)
G.add_edge(1,6, weight=3)
G.add_edge(1,6,weight=2)
G.add_edge(6,7, weight=3)
G.add_edge(6,7, weight=4)
G.add_edge(1,8, weight=5)
G.add_edge(1,8, weight=4)
G.add_edge(1,8, weight=2)
G.add_edge(8,9, weight=5)
G.add_edge(8,9, weight=4)

blue=[(1,2),(2,3),(1,4),(4,5),(1,6),(6,7),(1,8),(8,9)]
red=[(1,2),(2,3),(1,4),(4,5),(1,6),(6,7),(1,8),(8,9)]
green=[(1,2),(2,3),(1,8)]

pos = nx.kamada_kawai_layout(G)

nx.draw_networkx_nodes(G, pos ,node_size=400, 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=green,width=5, alpha=0.5,
edge_color='g')
nx.draw_networkx_edges(G, pos, edgelist=red,width=2, alpha=0.5,
edge_color='r')
labels = {}
labels[1] = r'$Presa$'
for i in pos:
    pos[i][0] = pos[i][0] -0.15
    pos[i][1] = pos[i][1] -0.03
nx.draw_networkx_labels(G, pos, labels, font_size=17)

plot.axis('off')
plot.savefig("fig8.eps")
plot.show()
```

Ej8.py

9. Multigrafo no dirigido reflexivo

En la Universidad de Oriente se quiere realizar un concurso de habilidades entre las carreras de Ingeniería en Electrónica, Ingeniería en Automática, Licenciatura en Física, Ingeniería en Informática y Licenciatura en Matemática-Cibernética. Las habilidades a evaluar serán matemática y programación.

La figura 9 muestra como está organizado el concurso. Cada vértice representa a los estudiantes que estudian una de las carreras y las aristas, con qué estudiantes pueden participar en cada habilidad.

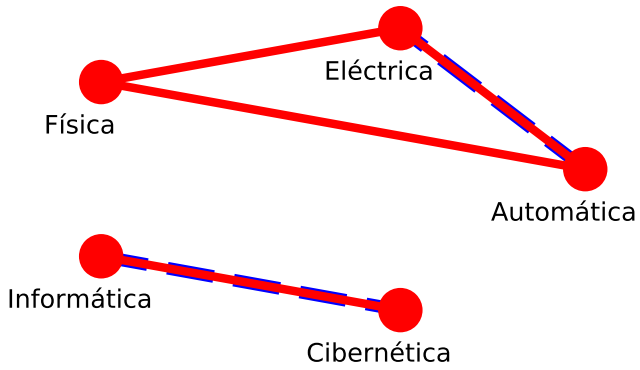

Figura 9: Concurso de habilidades.

Todos los vértices son de color rojo porque todos los estudiantes pueden competir con estudiantes de su misma carrera.

Los vértices rojos unen a los que pueden concursar en entre sí en matemáticas y los azules, los que pueden concursar en programación.

Para la visualización se emplea el *layout circular*.

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

G = nx . MultiGraph ()

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

blue=[(1,2),(4,5)]
red=[(1,2),(2,3),(1,3),(4,5)]

pos = nx.circular_layout(G, scale=0.1)

nx.draw_networkx_nodes(G, pos ,node_size=400, node_color='r', 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=4, alpha=0.5,
edge_color='r')
labels = {}
labels[1] = r'Automatica'
labels[2] = r'Electrica'
labels[3] = r'Fisica'
labels[4] = r'Informatica'
labels[5] = r'Cibernetica'

for i in pos:
    pos[i][1] = pos[i][1] -0.03
    pos[i][0] = pos[i][0] -0.008
```

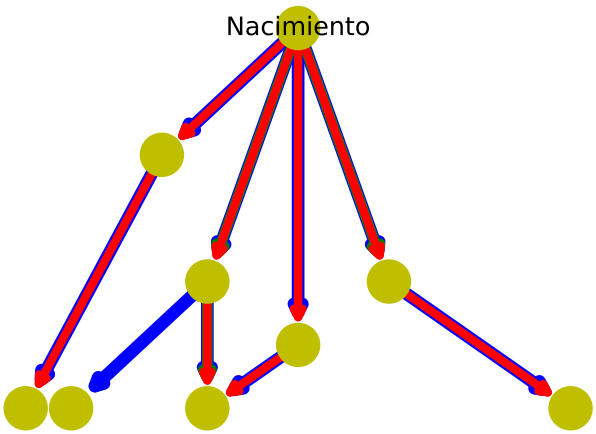

Figura 10: Representación del delta de un río en su transcurso hacia el mar.

```
nx.draw_networkx_labels(G, pos, labels, font_size=12)

plot.axis('off')
plot.savefig("fig9.eps")
plot.show()
```

Ej9.py

10. Multigrafo dirigido acíclico

Estos grafos pueden utilizarse para representar la cuenca hidrográfica de un río en su flujo hacia el mar. En la figura 10 los vértices representan los puntos en los que al menos dos ramificaciones del delta confluyen o se separan. Para que el grafo tuviese exactamente la forma deseada, se insertaron las coordenadas de los vértices de manera manual.

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

G = nx.MultiDiGraph()

G.add_edge(1,2, weight=3)
G.add_edge(1,2, weight=5)
G.add_edge(1,2, weight=6)
G.add_edge(2,3, weight=4)
G.add_edge(2,5, weight=3)
G.add_edge(2,5, weight=5)
G.add_edge(1,4, weight=4)
G.add_edge(1,4, weight=3)
G.add_edge(4,5, weight=5)
G.add_edge(4,5, weight=3)
G.add_edge(1,6, weight=3)
G.add_edge(1,6,weight=2)
G.add_edge(6,7, weight=1)
G.add_edge(6,7, weight=4)
G.add_edge(1,8, weight=5)
G.add_edge(1,8, weight=4)
G.add_edge(1,8, weight=2)
```

```
G.add_edge(8,9, weight=5)
G.add_edge(8,9, weight=4)

blue=[(1,2),(2,3),(2,5),(1,4),(4,5),(1,6),(6,7),(1,8),(8,9)]
red=[(1,2),(2,5),(1,4),(4,5),(1,6),(6,7),(1,8),(8,9)]
green=[(1,2),(2,5),(1,8)]

pos = {1:(400, 700), 2:(300,300), 3:(150, 100), 4:(400,200), 5:(300,100),6:(250,500)
    , 7:(100
       ,100), 8:(500, 300), 9:(700,100)}
nx.draw_networkx_nodes(G, pos, node_size=400, 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=green, width=5, alpha=0.5,
edge_color='g')
nx.draw_networkx_edges(G, pos, edgelist=red, width=4, alpha=0.5,
edge_color='r')
labels = {}
labels[1] = r'Nacimiento'

nx.draw_networkx_labels(G, pos, labels, font_size=12)

plot.xlim(20,800)
plot.axis('off')
plot.savefig("fig10.eps")
plot.show()
```

Ej10.py

11. Multigrafo dirigido cíclico

Este tipo de grafo es especialmente útil para representar viajes por lugares a los que se puede llegar mediante diferentes vías, regresando luego al punto de origen. Un ejemplo concreto de esto sería un recorrido vacacional por varias ciudades. Partiendo de la ciudad A, hay tres posibles vías para llegar a la ciudad B, en bus, en auto de alquiler o en tren, cada uno de estos medios de transporte representaría una arista diferente para unir los vértices. Así sucesivamente, se tienen en cuenta las posibles vías de transporte hacia las distintas ciudades hasta completar el recorrido y regresar a casa.

Para posicionar los vértices en la figura 11 se emplea el *layout shell*, aunque el circular también ofrece buenos resultados en este tipo de grafos.

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

G = nx.MultiDiGraph()
G.add_edge(1,2, weight=3)
G.add_edge(1,2, weight=5)
G.add_edge(1,2, weight=6)
G.add_edge(2,3, weight=3)
G.add_edge(2,3, weight=5)
G.add_edge(2,3, weight=6)
G.add_edge(3,4, weight=3)
G.add_edge(4,5, weight=3)
G.add_edge(4,5, weight=5)
G.add_edge(5,6, weight=3)
G.add_edge(6,7, weight=6)
G.add_edge(7,1, weight=6)

blue=[(1,2),(2,3),(3,4),(4,5),(5,6)]
```

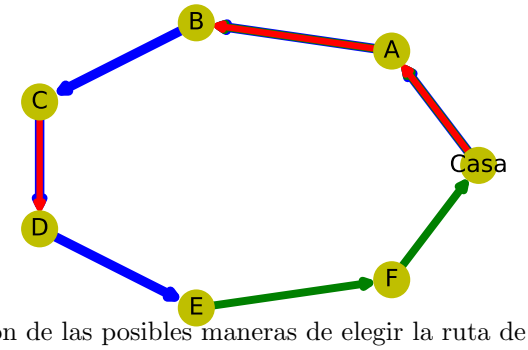

Figura 11: Representación de las posibles maneras de elegir la ruta de un circuito vacacional.

```
red=[(1,2),(2,3),(4,5)]
green=[(1,2),(2,3),(6,7),(7,1)]

pos = nx.shell_layout(G)

nx.draw_networkx_nodes(G, pos ,node_size=500, 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=green,width=5, alpha=.7,
edge_color='g')
nx.draw_networkx_edges(G, pos, edgelist=red,width=4, alpha=0.5,
edge_color='r')

labels = {}
labels[1] = r'Casa'
labels[2] = r'A'
labels[3] = r'B'
labels[4] = r'C'
labels[5] = r'D'
labels[6] = r'E'
labels[7] = r'F'

nx.draw_networkx_labels(G, pos, labels, font_size=15, )
plot.axis('off')
plot.savefig("fig11.eps")
plot.show()
```

Ej11.py

12. Multigrafo dirigido reflexivo

González-Cervantes [5] emplea teoría de grafos para representar el potencial eléctrico en el corazón, demostrando que se pueden incorporar las leyes fisiológicas involucradas. Cada uno de los vértices representa uno de los puntos principales que generan los impulsos eléctricos y que llevan la electricidad a cada parte del corazón, las aristas describen el valor máximo de voltaje y su duración en tiempo que descarga cada vértice. Además, puede proporcionar información con respecto al potencial eléctrico por zonas para una mejor localización. En la figura 12 se muestra la representación con grafos del intervalo PR. Debido a que se siguió la forma del corazón para dibujar el grafo, las coordenadas de los vértices fueron ingresadas manualmente.

```
import matplotlib.pyplot as plot
```

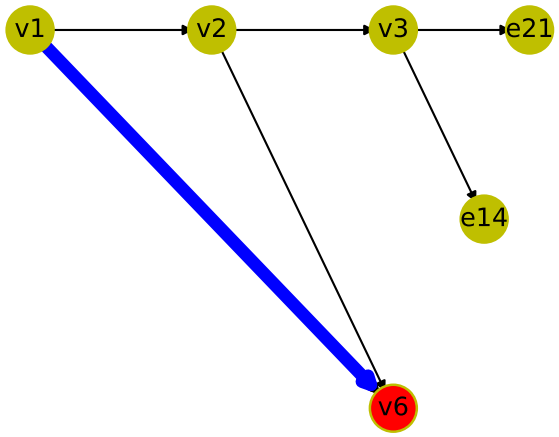

Figura 12: Representación con grafos del intervalo PR.

```
import networkx as nx

G = nx.MultiDiGraph()

G.add_edge(1,2)
G.add_edge(2,3)
G.add_edge(3,4)
G.add_edge(3,5)
G.add_edge(1,6,weight=3)
G.add_edge(1,6,weight=1)
blue=[(1,6)]
G.add_edge(2,6)
node1 = {6}

pos = {1:(50, 350), 2:(250,350), 3:(450, 350), 4:(600,350), 5:(550,340), 6:(450,330)
    }
nx.draw_networkx_nodes(G, pos, node_size=500, node_color='y', node_shape='o')
nx.draw_networkx_edges(G, pos, width=1, alpha=0.8, edge_color='black')
nx.draw_networkx_nodes(G, pos, nodelist=node1,node_size=400, node_color='r',
    node_shape='o')
nx.draw_networkx_edges(G, pos, edgelist=blue,width=6, alpha=0.5,
edge_color='b', style='dashed')

labels = {}
labels[1] = r'v1'
labels[2] = r'v2'
labels[3] = r'v3'
labels[4] = r'e21'
labels[5] = r'e14'
labels[6] = r'v6'

nx.draw_networkx_labels(G, pos, labels, font_size=12)
plot.xlim(20,800)
plot.axis('off')
plot.savefig("fig12.eps")
plot.show()
```

Ej12.py

Referencias

[1] An algorithm for drawing general y otros. *Information processing letters*, 31(1):7–15, 1989.

[2] Ravindra K Ahuja. *Network flows: theory, algorithms, and applications*. Pearson Education, 2017.

[3] Fan Chung. Graph theory in the information age. *Notices of the AMS*, 57(6):726–732, 2010.

[4] Edward M Fruchterman, Thomas MJ y Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[5] Aurora y Salido-Ruiz Ricardo González-Cervantes, Natalia y Espinoza-Valdez. Potencial eléctrico en el corazón: Representación mediante un grafo. *ReCIBE*, 5(3), 2016.

[6] Amador Menéndez Velázquez. Una breve introducción a la teoría de grafos. *Suma*, 28:11–26, 1998.