Representación de redes a través de la teoría de grafos

Mario Alberto Gutiérrez Carrales

12 de febrero de 2019

En este reporte se desarrolla una investigación teórica acerca de las redes debido a que son de vital importancia ya que en el mundo se presenta una gran diversidad de problemas que pueden ser modelados a través de una red, de hecho, la existencia de las redes es tan relevante que la teoría de grafos [1] se encarga de estudiar sus propiedades.

Utilizando ésta teoría se pretende estructurar las propiedades de algunos grafos y además con la ayuda del lenguaje de programación interpretado `python` [2] se desean construir ejemplos que ayuden a visualizar de forma sencilla cada grafo.

En el código se utilizaron dos módulos, uno de ellos es NetworkX [3], que contiene estructura de datos que almacena grafos y el otro Matplotlib [4] que permite visualizar y guardar imágenes en formato eps.

El uso de dichos módulos va declarado al inicio del código, tal como se muestra a continuación:

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

Antes de comenzar con la descripción de varios tipos de grafos se presentan algunas definiciones preliminares que son necesarias.

Definiciones preliminares

Un grafo G es una tripleta G=(V,E,δ), donde V representa el conjunto de puntos llamados vértices (o nodos), E es el conjunto de arcos llamados aristas (o lados) y δ: E $\rightarrow$ V $\times$ V una función que asigna a cada arista e $\in$ E un par de vértices u,v $\in$ V, en muchas ocasiones la función δ viene implícita en el dibujo del grafo.

Un grafo simple es aquel que entre cualquier pareja de vértices existe a lo más una arista que los une, en caso contrario se dice que es un multigrafo.

Se dice que un grafo es no dirigido si no se toma en cuenta el orden en que los aristas se asignan a los pares de vértices es decir (v_i,v_j)=(v_j,v_i) $\forall$ v_j,v_i $\in$ V , en caso contrario, es decir, si no se tiene la igualdad en alguna pareja de vértices se dice que es un grafo dirigido.

Si en un grafo se puede encontrar una lista de vértices conectados de tal manera que el primer vértice y el último sean el mismo entonces se dice que es un grafo cíclico, si un grafo no es cíclico entonces se dice que es acíclico.

Un grafo reflexivo es aquel en el que al menos un vertice tiene un arista que lo conecta consigo mismo.

Es fácil determinar el tipo de grafo que se tiene haciendo uso de la matriz de multiplicidad para multigrafos y adyacencia para grafos simples, por un lado si se desea saber si un grafo es dirigido basta con verificar que se cumpla que la matriz correspondiente sea simetrica, por otra parte para confirmar si un grafo es reflexivo es suficiente observar la diagonal principal de la matriz y verificar si hay un valor distinto de 0.

Es posible combinar varias propiedades, obteniendo asi diferentes tipos de grafos como se muestran en las siguientes secciones.

NOTA: En los siguientes ejemplos se utilizan aristas azules para hacer referencia a que hay múltiples aristas entre 2 vértices. Se utilizan nodos verdes para indicar que un nodo es reflexivo y tiene un lazo, en cambio un nodo rojo no hace referencia a algo en especial.

1. Grafo simple no dirigido acíclico

Un modelo aplicado que se puede llevar a cabo con este tipo de grafos son las semifinales de un torneo de fútbol, tal como se muestra en la figura 1.

```
G=nx.Graph()
pos={0:(1,0),1:(2,0),2:(3,0),3:(4,0),4:(1.5,.8),5:(3.5,.8)
    ,6:(2.5,1.6)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5',5:'6',6:'7'}
Conexion=[(0,4),(1,4),(2,5),(3,5),(4,6),(5,6)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(7),
    node_color='r')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)
plt.axis('off')
plt.savefig("Ejemplo1.eps")
plt.show()
```

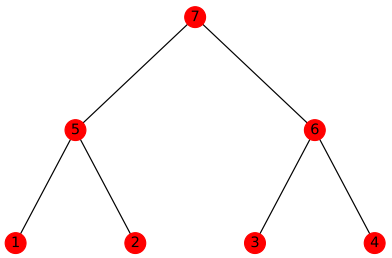

Figura 1: *Seminifinales de torneo de fútbol*

Cuando en este grafo hay una manera de llegar entre cada pareja de vértices se le conoce como árbol, por la forma parecida a un árbol real en como se van desprendiendo las aristas.

Otras aplicaciones de este tipo de grafo son los organigramas en general, distribución de bienes, árboles genealógicos, isomeros de cadena, etc.

2. Grafo simple no dirigido cíclico

Una de las aplicaciones más utilizadas hoy en dia que se basan en este tipo de grafo es la representación de usuarios de una red social como Facebook [5], Twitter [6], etc. así como se observa en la figura 2.

```
G=nx.Graph()
pos={0:(2,1),1:(2.5,1),2:(1,2),3:(2.25,2),4:(3,2),5:(2,3)
    ,6:(2.5,3)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5',5:'6',6:'7'}
Conexion=[(0,2),(1,3),(2,3),(3,4),(2,5),(3,6),(4,6)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(7),
    node_color='r')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)
plt.axis('off')
plt.savefig("Ejemplo2.eps")
plt.show()
```

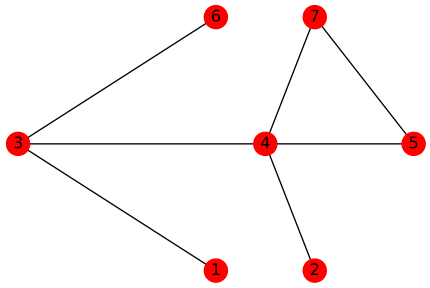

Figura 2: *Usuarios en una red social*

Otra aplicación importante es la modelación de territorios y de ahí surge uno de los problemas más importantes dentro de la optimización matemática que es el problema de la ruta más corta. (Considerando que todas las calles, avenidas ó carreteras son de doble sentido).

3. Grafo simple no dirigido reflexivo

Este grafo se encuentra presente en la forma en como están construidos ciertos cruces de avenidas en diferentes ciudades, de tal manera que las avenidas son perpendiculares entre sí y en medio se encuentra un retorno, que representa un lazo en el grafo, y su objetivo es cambiar la orientación en la que se dirige el automovilista. Observar figura 3.

```
G=nx.Graph()
pos={0:(0,1),1:(1,0),2:(1,1),3:(2,1),4:(1,2)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5'}
Conexion=[(0,2),(1,2),(3,2),(4,2)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[2],
    node_color='g')
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[0,1,3,4],
    node_color='r')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)
plt.axis('off')
plt.savefig("Ejemplo3.eps")
plt.show()
```

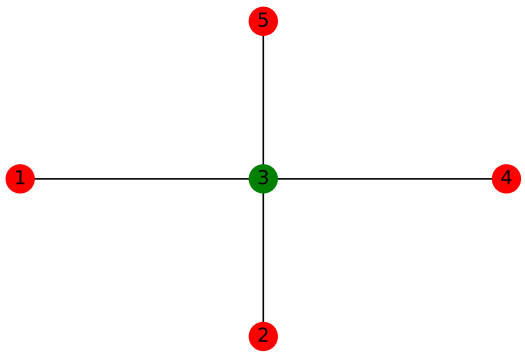

Figura 3: *Conexión entre avenidas y un retorno*

4. Grafo simple dirigido acíclico

Algunas de las principales aplicaciones de los grafos dirigidos es la modelación de redes de transporte, cadenas de suministro, organigramas con dirección, etc.

Supongamos que el dueño de una empresa quiere llevar a cabo un proyecto, el reparto de actividades se distribuye jerárquicamente como se muestra en la figura 4.

```
G=nx.DiGraph()

pos={0:(1.75,3),1:(.5,2),2:(3,2),3:(0,1),4:(1,1),5:(2.5,1)
    ,6:(3.5,1)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5',5:'6',6:'7'}
Conexion=[(0,1),(0,2),(1,3),(1,4),(2,5),(2,6)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(7),
    node_color='r')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)
plt.axis('off')

plt.savefig("Ejemplo4.eps")
plt.show()
```

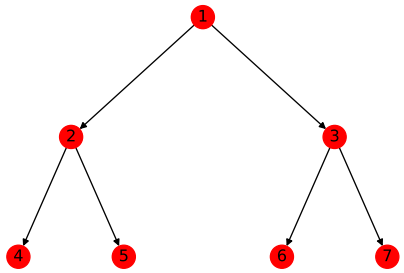

Figura 4: *Jerarquía en planeación de un proyecto*

5. Grafo simple dirigido cíclico

En muchas ocasiones el tráfico vehicular se ve afectado por diversos factores haciendo que este sea lento, debido a ello en muchas calles solamente se permite el acceso en un solo sentido. La figura 5 muestra un ejemplo del tránsito vehicular en una cierta localidad.

```
G=nx.DiGraph()

pos={0:(0,1),1:(1,0),2:(2,1),3:(1.6,2),4:(.5,2)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5'}
Conexion=[(0,1),(1,2),(2,3),(2,4),(3,4),(4,0)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(5),
    node_color='r')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)
plt.axis('off')

plt.savefig("Ejemplo5.eps")
plt.show()
```

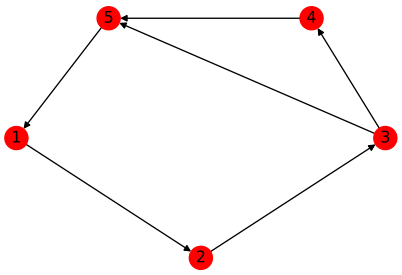

Figura 5: *Calles con sentido restringido*

6. Grafo simple dirigido reflexivo

En un modelo matemático se desea contemplar tantas características reales mientras sea posible para lograr entender un fenómeno, en este caso es posible mezclar la condición de un grafo dirigido que contenga un lazo tal como se observó en la sección 3, y un ejemplo de esto es el como están hechas ciertas colonias en algunos países en el que suele haber una calle en la esquina de una localidad y es acompañada de un retorno o parque para facilitar la salida vehicular. En la figura 6 se muestra un ejemplo de esta situación.

```
G=nx.DiGraph()

pos={0:(0,1),1:(1,0),2:(2,0),3:(2,1),4:(1,1)}
labels={0:'$1$',1:'2',2:'3',3:'4',4:'5'}
Conexion=[(0,1),(1,0),(1,2),(1,4),(2,3),(3,4),(4,3),(4,0)]

nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[0,1,2,4],
    node_color='r')
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[3],
    node_color='g')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=Conexion,width=1)
nx.draw_networkx_labels(G,pos,labels,font_size=12)

plt.axis('off')
plt.savefig("Ejemplo6.eps")
plt.show()
```

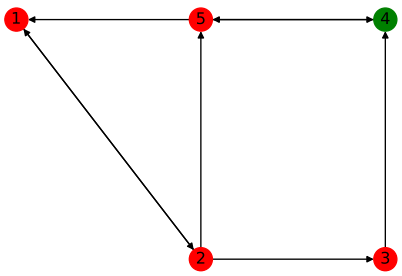

Figura 6: *Colonia*

7. Multigrafo no dirigido acíclico

En redes, un multigrafo es aquel que conecta algún par de vértices de múltiples maneras, esto hace que nuestro objeto de estudio sea más completo ya que en la mayoria de los casos esta herramienta es muy útil en la modelación. Un ejemplo de este grafo es una fábrica que produce productos y los suministra a diferentes almacénes pero los vehículos pueden transitar no solamente por una carretera, sino por múltiples, en este caso 2. Ver figura 7.

```
G=nx.MultiGraph()

pos={0:(1,.4),1:(2,.4),2:(2,0),3:(2,.8),4:(3.5,.8)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5'}

nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(1,2),(3,4)],
    width=1,edge_color='b')
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1),(1,3)],
    width=1)
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(5),
    node_color='r')
nx.draw_networkx_labels(G,pos,labels,font_size=12)

plt.axis('off')
plt.savefig("Ejemplo7.eps")
plt.show()
```

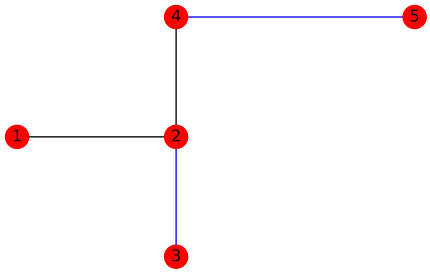

Figura 7: *Ruta de un almacén*

8. Multigrafo no dirigido cíclico

Este tipo de grafo se ve relacionado en general en la modelación donde se vean involucrados puentes. Una característica resaltante de este grafo es que se puede llegar a cualquier vértice por la conexión de las aristas. De ésto surge el prolema particular de los 7 puentes de Konigsberg. En la figura 8 se muestra un ejemplo.

```
1  G=nx.MultiGraph()
2
3  pos={0:(0,0),1:(2,0),2:(2,2),3:(0,2),4:(0,1)}
4  labels={0:'1',1:'2',2:'3',3:'4',4:'5'}
5
6  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1),(1,2),(2,3)
       ],width=1)
7  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,4),(3,4)],
       width=1,edge_color='b')
8  nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=range(5),
       node_color='r')
9  nx.draw_networkx_labels(G,pos,labels,font_size=12)
10
11 plt.axis('off')
12 plt.savefig("Ejemplo8.eps")
13 plt.show()
```

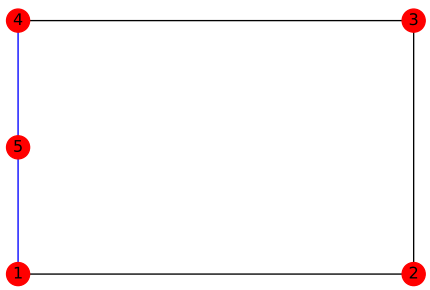

Figura 8: *Puentes de una ciudad*

9. Multigrafo no dirigido reflexivo

Sitúandonos en la sección 6, se puede extender el modelo de una colonia en la que la salida a una avenida puede estar dada por múltiples formas y no solo a la avenida, es decir, no solo un arista puede ser múltiple, sino varios. y además, puede considerarse múltiples lazos en la calle que funge el papel del nodo reflexivo, esto puede verse como un parque, callejón, retorno, etc. En la figura 9 se muestra un ejemplo.

```
G=nx.MultiGraph()

pos={0:(1,1),1:(0,0),2:(2,0),3:(2,2),4:(0,2)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5'}

nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1),(0,2),(0,3)
    ,(0,4),(1,4),(2,3),(3,4)],width=1)
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(1,2)],width=1,
    edge_color='b')
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[0,1,2,3],
    node_color='r')
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[4],
    node_color='g')
nx.draw_networkx_labels(G,pos,labels,font_size=12)

plt.axis('off')
plt.savefig("Ejemplo9.eps")
plt.show()
```

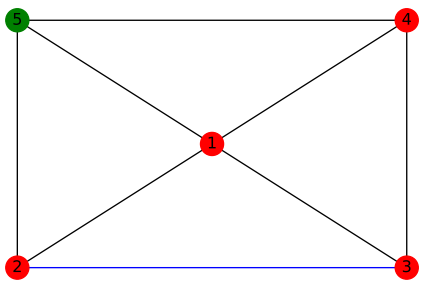

Figura 9: *Colonia con múltiples conexiones*

10. Multigrafo dirigido acíclico

Supongamos que se quiere llegar de una localidad a otra y se conocen los posibles caminos por los cuales se puede transitar y se desea encontrar la ruta que contenga menor costo para llevar a cabo el viaje. Dicho problema es el de la ruta más corta y este grafo puede modelarlo adecuadamente, así como se muestra en la figura 10.

```
1  G=nx.MultiDiGraph()
2
3  pos={0:(0,1),1:(1,0),2:(1,2),3:(2,0),4:(2,2)}
4  labels={0:'1',1:'2',2:'3',3:'4',4:'5'}
5
6  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1),(1,3),(1,2)
       ,(2,3),(2,4),(3,4)],width=1)
7  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,2)],width=1,
       edge_color='b')
8  nx.draw_networkx_nodes(G,pos,node_size=300,nodelist=[0,1,2,3,4],
       node_color='r')
9  nx.draw_networkx_labels(G,pos,labels,font_size=12)
10
11 plt.axis('off')
12 plt.savefig("Ejemplo10.eps")
13 plt.show()
```

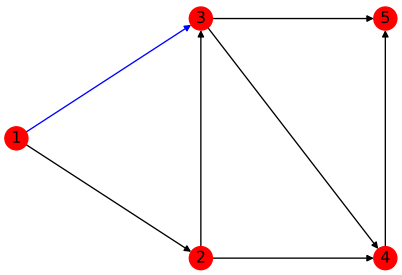

Figura 10: *Mapa transitorio*

11. Multigrafo dirigido cíclico

Este tipo de grafo es el que mejor modela los problemas de ruteo de vehículos, pues cuando estamos en alguna localidad siempre podemos llegar de nuevo a ella misma utilizando otras calles que, en ocasiones, se tiene la opción de elegir una entre varias calles para llegar a una localidad y también existe la posibilidad de que cualquier calle sea de un solo sentido. En la figura 11 se encuentra un ejemplo de este planteamiento.

```
G=nx.MultiDiGraph()

pos={0:(0,1),1:(1,0),2:(1,2),3:(2,0),4:(2,2),5:(3,1)}
labels={0:'1',1:'2',2:'3',3:'4',4:'5',5:'6'}

nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,2),(1,0),(1,3)
    ,(1,2),(2,3),(2,4),(3,1),(3,4),(3,5),(4,2),(4,5)],width=1)
nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1)],width=1,
    edge_color='b')
nx.draw_networkx_nodes(G,pos,node_size=300,nodelist
    =[0,1,2,3,4,5],node_color='r')

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

plt.axis('off')
plt.savefig("Ejemplo11.eps")
plt.show()
```

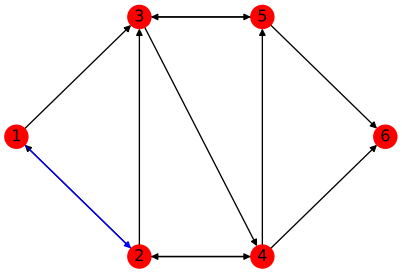

Figura 11: *Calles principales de una ciudad*

12. Multigrafo dirigido reflexivo

Un ejemplo aplicado de este tipo de grafo es la conectividad de cables y circuitos en una computadora, tanto los cables entrantes como por ejemplo los de encender y de internet, los circuitos mismos en el cpu, y los cables de salida como teclado, mouse, etc. Dicha representación de la red computacional se encuentra en la figura 12.

```
1  G=nx.MultiDiGraph()
2
3  pos={0:(0,2),1:(1,2),2:(1,1),3:(2,1),4:(2,2)}
4  labels={0:'1',1:'2',2:'3',3:'4',4:'5'}
5
6  nx.draw_networkx_nodes(G,pos,node_size=400,nodelist=[1],
       node_color='g')
7  nx.draw_networkx_nodes(G,pos,node_size=400,nodelist=[0,2,3,4],
       node_color='r')
8  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(0,1)],width=1,
       edge_color='b')
9  nx.draw_networkx_edges(G,pos,alpha=1,edgelist=[(1,2),(1,3),(1,4)
       ],width=1,edge_color='k')
10 nx.draw_networkx_labels(G,pos,labels,font_size=12)
11
12 plt.axis('off')
13 plt.savefig("Ejemplo12.eps")
14 plt.show()
```

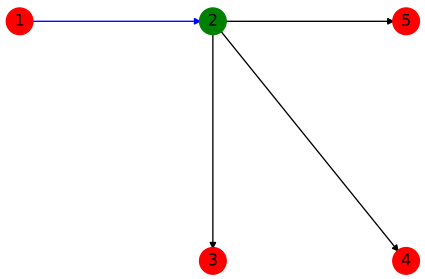

Figura 12: *Red de conexión computacional*

Referencias

[1] John Michael Harris, Jeffry L Hirst, and Michael J Mossinghoff. *Combinatorics and graph theory*, volume 2. Springer, 2008.

[2] https://www.python.org/.

[3] https://networkx.github.io/documentation/networkx 1.10/reference/-classes.html.

[4] https://matplotlib.org/.

[5] https://www.facebook.com/.

[6] https://twitter.com.