Tarea 2

5271

26 de febrero de 2019

1. Introducción

Cuando una información se manifiesta mediante objetos y sus relaciones, visualizarla puede ser el primer paso para resolver problemas de diversa índole. La modelación y teoría de grafos constituyen la forma más común de modelar información relacional, y su visualización una forma de representar la información [1].

Sin embargo el modelado y acomodo de un grafo que refleje situaciones de la vida real resulta complejo, es por ello que se han desarrollado algoritmos dedicados a perfeccionar el trazado de los mismos según el tipo problema y grafo que representen, llamados algoritmos de diseño o *layout*.

Los algoritmos de diseño son los que devuelven una lista de posiciones para los nodos según diversos parámetros definidos para cada algoritmo, buscando cumplir con ciertos criterios estéticos como son: minimizar cruces entre aristas, maximizar el ángulo entre aristas adyacentes, maximizar el ángulo entre aristas que se cruzan, mostrar simetría, distribución uniforme de los vértices y longitud uniforme de aristas [1].

Entre los algoritmos de diseño destacan los siguientes:

- *Bipartite layout* (posiciona los nodos en dos líneas rectas, es decir que divide el conjunto de nodos en dos subconjuntos X e Y donde no existe adyacencia entre los elementos de un mismo subconjunto) [4].
- *Circular layout* (ubica los nodos en forma circular) [4].
- *Kamada kawai layout* (posiciona los nodos utilizando la función de costo de longitud de camino Kamada-Kawai) [4].
- *Random layout* (posiciona los nodos de forma aleatoria) [4].
- *Rescale layout* (reescala dado una matriz de posiciones) [4].
- *Shell layout* (posiciona los nodos en círculos concéntricos) [4].

- *Spring layout* (posiciona los nodos utilizando el algoritmo dirigido por fuerza de Fruchterman-Reingold) [4].
- *Spectral layout* (posiciona nodos utilizando los vectores propios del gráfico laplaciano) [4].

2. Grafo simple no dirigido acíclico

Se pueden encontrar varias aplicaciones a la modelación de un grafo simple no dirigido acíclico, una de las más claras son los árboles, "el árbol (árbol libre) que es un grafo no dirigido, conexo y acíclico. Un árbol también puede definirse como un grafo no dirigido en elque hay exactamente un camino entre todo par de vértices"[2].

Un ejemplo de usos de árboles es en topología de red la de árbol, en esta topología los nodos de la red están ubicados en forma de árbol. esta conexión es similar a muchas redes en estrella interconectadas con la diferencia de no poseer un nodo central, en cambio posee un nodo troncal desde el cual se ramifican el resto de los nodos como se muestra en la figura 1 que es una pequeña representación con solo ocho nodos de la red que posee la UEB Rolando Peréz Gollanes entidad dedicada al sacrificio y procesamiento de aves.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 1 el algoritmo *spring layout*, dado que fue el que cumplió con la mayor cantidad de los criterios estéticos para este ejemplo.

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

A=nx.Graph()
A.add_edges_from([('R1','Sw1'),('Sw1','Sw2'),
                  ('Sw1','Sw3'),('Sw2','Pc2'),('Sw2','Pc3'),
                  ('Sw3','Pc4'),('Sw3','Pc5')])


sw=['R1','Sw1','Sw2','Sw3']
pc=['Pc2','Pc3','Pc4','Pc5']
swc=[('R1','Sw1'),('Sw1','Sw2'),('Sw1','Sw3')]
pcc=[('Sw2','Pc2'),('Sw2','Pc3'),('Sw3','Pc4'),('Sw3','Pc5')]
lista=['Sw2','Pc2','Pc3','Sw3','Pc4','Pc5','R1'],['Sw1']
#posicion=nx.shell_layout(A, nlist=lista, scale=0.8, center=None,
    dim=2)
#
#posicion=nx.random_layout(A, center=None, dim=2)
#
#posicion=nx.circular_layout(A, scale=0.8, center=None, dim=2)
#
```

```
#posicion=nx.spectral_layout(A, weight='weight', scale=1, center=
    None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.5, center=None, dim=2)

posicion=nx.spring_layout(A, k=1, iterations=300, threshold=0.0001,
     weight='distans', scale=0.5)

nx.draw_networkx_nodes(A,posicion, node_size=800, node_shape='s',
    nodelist=sw, node_color='#c9b323')
nx.draw_networkx_nodes(A,posicion, node_size=800, node_shape='o',
    nodelist=pc, node_color='#c68282')
nx.draw_networkx_edges(A,posicion, width=4, edgelist=swc,style='
    dashed',edge_vmax=1, edge_vmin=1)
nx.draw_networkx_edges(A,posicion, width=2, edgelist=pcc)
nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial
    ')


plot.axis('off')

plot.savefig("Graf1_spring_layout.eps")
plot.show(A)
```

Graf1.py

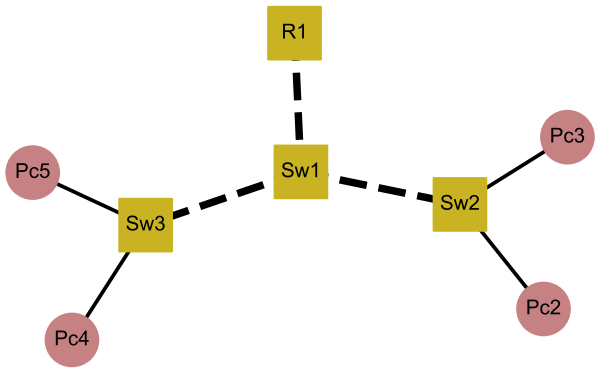

Figura 1: Grafo simple no dirigido acíclico, donde las lineas discontinuas representan los cables de fibra optica y los lineas continuas los cables UTP.

3. Grafo simple no dirigido cíclico

Un grafo simple no dirigido cíclico puede usarse áreas como geografía. Si se considera un mapa, digamos de Europa: que cada país sea un vértice y conecte dos vértices con una arista si esos países comparten una frontera. Un problema famoso que quedó sin resolver durante más de cien años fue el problema de los cuatro colores. Aproximadamente esto indica que cualquier mapa puede ser coloreado con a lo sumo cuatro colores de tal manera que los países adyacentes no tengan el mismo color. Este problema motivó muchos desarrollos en la teoría de grafos y finalmente se demostró con la ayuda de una computadora en 1976 [6].

Otra aplicación es en la representación de redes sociales, una red social se conceptualiza como un grafo, es decir, un conjunto de vértices (o nodos, unidades, puntos) que representan entidades u objetos sociales y un conjunto de líneas que representan una o más relaciones sociales entre ellos [3].
Por ejemplo, si consideramos un grupo de nueve doctores del núcleo académico de PISIS (Posgrado en Ingeniería de Sistemas) y construimos una red, tomando como vértices a los doctores y la colaboración de ellos en artículos publicados como aristas dará lugar a un grafo simple no dirigido cíclico como se muestra en la figura 2.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 2 el algoritmo *circular layout*, ya que el mismo permite una mejor compresión del grafo del ejemplo en cuestión.

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


A=nx.Graph()
A.add_edges_from([('Dr.FL','Dra.AA'),('Dr.FL','Dra.YR'),
                  ('Dr.FL','Dr.RS'),('Dr.FL','Dra.ES'),
                  ('Dra.YR','Dr.VB'),('Dra.YR','Dr.RR'),
                  ('Dra.YR','Dr.RS'),('Dra.AA','Dra.IM'),
                  ('Dra.AA','Dr.RR'),('Dra.IM','Dr.VB'),
                  ('Dra.IM','Dra.AS'),('Dra.IM','Dr.RS'),
                  ('Dra.IM','Dr.VB'),('Dr.VB','Dr.RS'),
                  ('Dra.AS','Dr.VB'),('Dra.AS','Dr.RR'),
                  ('Dra.AS','Dr.RS'),('Dra.ES','Dr.RR'),
                  ('Dra.AA','Dra.ES'),('Dra.IM','Dra.ES')])
lista=['Dr.FL','Dra.AA','Dra.YR','Dr.RR','Dra.IM','Dra.AS','Dr.VB',
    'Dra.ES'],['Dr.RS']
#
#posicion=nx.shell_layout(A, nlist=lista, scale=0.8, center=None,
    dim=2)
#
#posicion=nx.random_layout(A, center=None, dim=2)
```

```
#
posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)
#
#posicion=nx.spectral_layout(A, weight='weight', scale=1, center=
    None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.5, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=300, threshold
    =0.0001, weight='distans', scale=0.5)

nx.draw_networkx_nodes(A,posicion, node_size=2000, node_color=range
    (9),cmap=plot.cm.Blues)
nx.draw_networkx_edges(A,posicion, width=2,edge_color='#46267d')
nx.draw_networkx_labels(A,posicion, font_size=13,font_family='arial
    ',
                        font_color='#846c16', font_weight='bold')


plot.axis('off')
plot.savefig("Graf2_circular_layout.eps")
plot.show(A)
```

Graf2.py

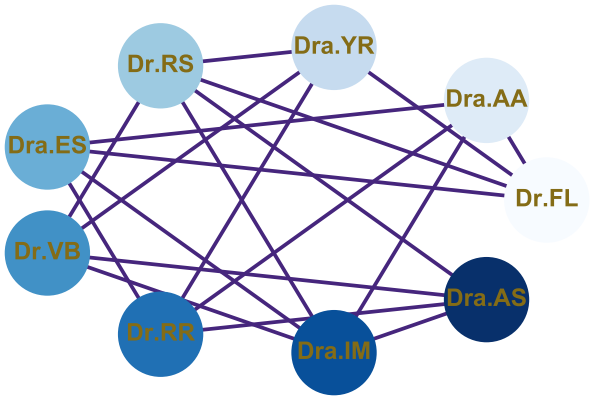

Figura 2: Grafo simple no dirigido cíclico, donde los vértices representa los diferentes doctores y las aristas la colaboración entre ellos.

4. Grafo simple no dirigido reflexivo

Una de las aplicaciones de este tipo de grafo es en la modelación de comportamientos de elementos sociales en la vida real.

Por ejemplo, en un estudio sobre el comportamiento sexual de un grupo de adolescentes con edades comprendida entre 15 y 18 años, se representar las relaciones sexuales consentidas (aristas) que existen entre los individuos (vértices) del grupo, así como la satisfacción, para así saber tendencias por edades, posibles esquemas de propagación de enfermedades y promiscuidad entre otros factores del interés de los sexólogos. Este ejemplo se muestra en el grafo de la figura 3.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 3 el algoritmo *kamada kawai layout*, dado que este algoritmo es uno de los más usados para la representación de grafos conexos y no dirigidos, por los resultados de acomodo tan buenos que arroja en estos tipos de grafos.

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


A=nx.Graph()
A.add_edge('Luis(15)','Maria(15)')
A.add_edge('Luis(15)','Luis(15)')
A.add_edge('Maria(15)','Yanet(17)')
A.add_edge('Yanet(17)','Yanet(17)')
A.add_edge('Maria(15)','Ferndo(16)')
A.add_edge('Ferndo(16)','Desisy(18)')
A.add_edge('Desisy(18)','Pedro(17)')
A.add_edge('Pedro(17)','Pedro(17)')
A.add_edge('Pedro(17)','Julia(18)')
A.add_edge('Pedro(17)','Rosi(16)')
A.add_edge('Ferndo(16)','Claudia(16)')
nodes_reflex = {'Luis(15)','Ferndo(16)','Julia(18)'}
nodes_no_reflex = {'Maria(15)','Yanet(17)','Rosi(16)','Desisy(18)',
                   'Pedro(17)','Claudia(16)'}

lista =['Luis(15)','Julia(18)','Yanet(17)','Rosi(16)','Desisy(18)',
    'Claudia(16)'], ['Maria(15)','Ferndo(16)','Pedro(17)']

#posicion=nx.shell_layout(A, nlist=lista, scale=0.5, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.8, center=None, dim=2)

#posicion=nx.spectral_layout(A, weight='distans', scale=1, center=
    None, dim=2)
```

```
30
31 posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
       weight', scale=0.5, center=None, dim=2)
32
33 #posicion=nx.spring_layout(A, k=1, iterations=200, threshold
       =0.0001, weight='weight', scale=0.50)
34
35 #posicion=nx.rescale_layout( pos , escala = 1 )
36
37 #posicion=nx.bipartite_layout(A, nodes, align='vertical', scale=1,
       center=None, aspect_ratio=1.3333333333333333)
38
39
40
41 nx.draw_networkx_nodes(A,posicion,nodelist=nodes_reflex,
42                        node_size=1500, node_color= '#ea6767')
43 nx.draw_networkx_nodes(A,posicion,nodelist=nodes_no_reflex,
44                        node_size=1500, node_color= 'y')
45 nx.draw_networkx_edges(A,posicion, width=4,edge_color='#85d9ce')
46
47 nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial
       ',
48                         font_color='Black', font_weight='bold')
49
50
51 plot.axis('off')
52 plot.savefig("Graf3_kamada_kawai_layout.eps")
53 plot.show(A)
```

Graf3.py

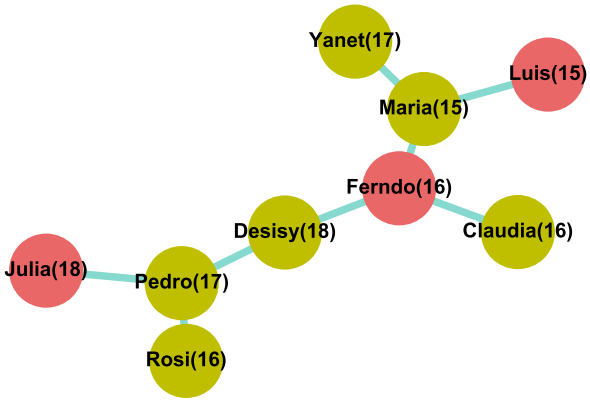

Figura 3: Grafo simple no dirigido reflexivo,donde los vértices rojos representan los nodos donde están las aristas reflexivas.

5. Grafo simple dirigido acíclico

Encontraos entre las aplicaciones de los grafos dirigidos o dígrafos acíclico las siguientes:

- Una red bayesiana queda especificada formalmente por una dupla $B = (G, O)$, donde G es un grafo dirigido acíclico (GDA) y O es el conjunto de distribuciones de probabilidad. Definimos un grafo como un par $G = (V, E)$, donde V es un conjunto finito de vértices nodos o variables y E es un subconjunto del producto cartesiano $V \times V$ de pares ordenados de nodos que llamamos enlaces o aristas [7].
- Los arboles dirigidos son un ejemplo clásico de grafos dirigidos acíclico, estos tienen múltiples en la cotidianidad como pueden ser los arbole genealógicos, los organigramas de una empresa (referido a las jerarquías entre los empleados) y el árbol de directorios de Windows.

En la figura 4 se muestra un ejemplo de red bayesiana (topología de red para el cáncer de pulmón), donde C (cáncer), Con (Contaminación), F (Fumador), D (disnea), Rx (rayos-x) son los vértices y sus relaciones son las aristas.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 4 el algoritmo *shell layout*,

dado que este algoritmo ubica los nodos en círculos concéntricos a un nodo dado, es la mejor opción para representar este ejemplo.

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

A=nx.DiGraph()
A.add_edge('Con','C',weight=0.8)
A.add_edge('F','C'  ,weight=0.8)
A.add_edge('C','D'  ,weight=3)
A.add_edge('C','Rx' ,weight=3)
lista=['Con','F','D','Rx'],['C']
nodes=['Con','F','D','Rx']
posicion=nx.shell_layout(A, nlist=lista, scale=0.8, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.8, center=None, dim=2)

#posicion=nx.spectral_layout(A, weight='distans', scale=1, center=
    None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.5, center=None, dim=2)

#posicion=nx.spring_layout(A, k=3, iterations=200, threshold
    =0.0001, weight='weight', scale=1)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, nodes, align='vertical', scale=1,
    center=None, aspect_ratio=1.3333333333333333)

nx.draw_networkx_nodes(A,posicion, node_size=700, nodelist=['C'],
    node_color= '#bbf1f0',alpha=1)
nx.draw_networkx_nodes(A,posicion, nodelist= ['Con','F','D','Rx'],
    node_size=600, node_color= '#6ed3c5',alpha=1)
nx.draw_networkx_edges(A,posicion, width=4,edge_color='#35665f')

nx.draw_networkx_labels(A,posicion, font_size=13,font_family='arial
    ',
                        font_color='#147645', font_weight='bold')


plot.axis('off')
plot.savefig("Graf4_shell_layout.eps")
plot.show(A)
```

Graf4.py

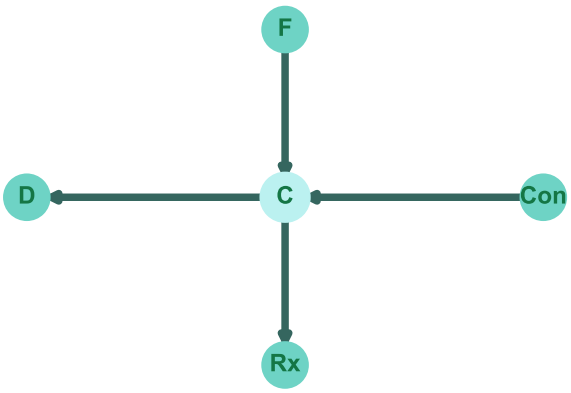

Figura 4: Grafo simple dirigido acíclico.

6. Grafo simple dirigido cíclico

Aplicaciones de los grafos dirigidos cíclicos:

- En ingeniería eléctrica se utilizan grafos dirigidos cíclicos en el análisis de circuito desde Kirchoff en los años 1850.
- En redes sociales, otro enfoque de redes sociales en su análisis puede arrojar un grafo dirigido cíclico si te tomamos como vértices a personas y como aristas el sentimiento de amistad de una persona hacia otra, este ejemplo se refleja en el grafo de la figura 5.

  Algorito de diseño.
  Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 5 el algoritmo *kamada kawai layout*, a pesar de no ser este ejemplo precisamente el más adecuado para este algoritmo fue el de mejor resultado estético.

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


A=nx.DiGraph()
A.add_edge('Ana','Felix')
A.add_edge('Ana','Alberto')
A.add_edge('Alberto','Yanet')
```

```
A.add_edge('Alberto','Fernando')
A.add_edge('Yanet','Roger')
A.add_edge('Teresa','Roger')
A.add_edge('Felix','Marta')
A.add_edge('Marta','Fernando')
A.add_edge('Yanet','Ana')
A.add_edge('Marta','Ana')
lista=['Ana','Felix','Alberto','Yanet','Fernando'],['Roger','Teresa','Marta',]

#posicion=nx.shell_layout(A, nlist=lista, scale=0.50, center=None, dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

#posicion=nx.spectral_layout(A, weight='distans', scale=0.50, center=None, dim=2)

posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='weight', scale=0.5, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=200, threshold=0.0001, weight='weight', scale=1)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, nodes, align='vertical', scale=1, center=None, aspect_ratio=1.3333333333333333)

nx.draw_networkx_nodes(A,posicion,
                       node_size=500, node_color= 'y')
nx.draw_networkx_edges(A,posicion, width=2,edge_color='b')
for p in posicion:
    posicion[p][1] += 0.05
nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial',
                        font_color='Black', font_weight='bold')


plot.axis('off')
plot.savefig("Graf5_kamada_kawai_layout.eps")
plot.show(A)
```

Graf5.py

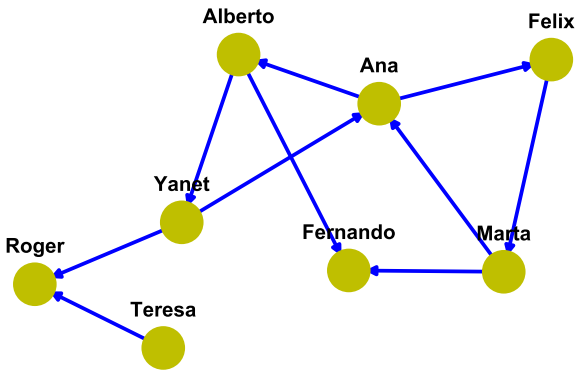

Figura 5: Grafo simple dirigido cíclico, donde los vértices representan las personas y las aristas la relación de amistad entre ellas.

7. Grafo simple dirigido reflexivo

Las aplicaciones de estos grafos van desde la representación del funcionamiento de las páginas web, el modelado de una empresa de servicio que brinda el mismo tanto a sus clientes como a ella misma, hasta representar un grafo que modele la comprobación de una red de computadoras.

Por ejemplo, suponiendo que se está comprobando la conectividad entre los nodos (equipos) de una red de computadoras, es decir que un nodo puede dar ping a cualquier otro nodo, significa que dicho nodo está conectado con el resto, además se debe asegurar que el mismo nodo pueda recibir un auto ping, si esto no ocurre existe un problema de conectividad. La figura 6 muestra dicho ejemplo.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 6 el algoritmo *random layout*, en este ejemplo se pudo usar este algoritmo por su simpleza ya que con grafos más complejos no es aconsejable el uso del mismo por sus malos resultados estéticamente hablando.

```
1 import networkx as nx
2 import matplotlib.pyplot as plot
3
```

```
4
5  A=nx.DiGraph()
6  A.add_edge('Pc1','Pc1')
7  A.add_edge('Pc1','Pc2')
8  A.add_edge('Pc1','Pc3')
9  A.add_edge('Pc1','Pc4')
10 A.add_edge('Pc1','Pc5')
11 A.add_edge('Pc1','Pc6')
12 A.add_edge('Pc1','Pc7')
13 nodes_reflex = {'Pc1'}
14 nodes_no_reflex = {'Pc2','Pc3','Pc4','Pc5','Pc6','Pc6','Pc7'}
15 lista=['Pc2','Pc3','Pc4','Pc5','Pc6','Pc6','Pc7'], ['Pc1']
16
17 #posicion=nx.shell_layout(A, nlist=lista , scale=0.50, center=None,
       dim=2)
18
19 posicion=nx.random_layout(A, center=None, dim=2)
20
21 #posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)
22
23 #posicion=nx.spectral_layout(A, weight='distans', scale=0.50,
       center=None, dim=2)
24
25 #posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
       weight', scale=0.5, center=None, dim=2)
26
27 #posicion=nx.spring_layout(A, k=1, iterations=200, threshold
       =0.0001, weight='weight', scale=1)
28
29 #posicion=nx.rescale_layout( pos , escala = 1 )
30
31 #posicion=nx.bipartite_layout(A, {'Pc1'}, align='vertical', scale
       =1, center=None, aspect_ratio=1.3333333333333333)
32 nx.draw_networkx_nodes(A,posicion ,nodelist=nodes_reflex ,
33                        node_size=500, node_color= 'r')
34 nx.draw_networkx_nodes(A,posicion ,nodelist=nodes_no_reflex ,
35                        node_size=500, node_color= 'y')
36 nx.draw_networkx_edges(A,posicion , width=2)
37 nx.draw_networkx_labels(A,posicion , font_size=11,font_family='arial
       ')
38
39
40 plot.axis('off')
41 plot.savefig("Graf6_random_layout.eps")
42 plot.show(A)
```

Graf6.py

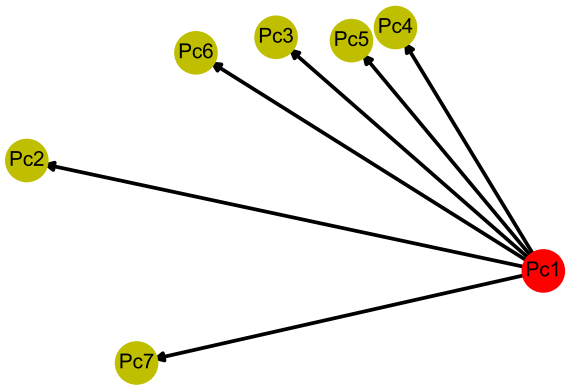

Figura 6: Grafo simple dirigido reflexivo, donde el vértice rojo representa la arista reflexiva.

8. Multigrafo no dirigido acíclico

Una de las aplicaciones de este tipo de grafo es en el tasado de rutas. Por ejemplo, considerando que se quiere tranzar los diferentes caminos (sin importar el sentido de estos) que comunican a varios Municipios de La Habana en un orden específico (Lisa, Marianao, Playa, Vedado, Habana Vieja, Habana del Este). Tomando como vértices los municipios y como aristas las carreteras que los unen, esto da lugar al grafo que muestra la figura 7.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 7 el algoritmo *spring layout*, este acomoda los nodos con el algoritmo dirigido por fuerza de Fruchterman-Reingold que minimiza la distancia entre los vértices conectados y de mayor peso.

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


A=nx.MultiGraph()
A.add_edge('Lisa','Marianao', weight=2)
A.add_edge('Marianao','Playa', weight=2)
A.add_edge('Marianao','Playa', weight=4)
A.add_edge('Playa','Vedado', weight=2)
A.add_edge('Vedado','Habana Vieja', weight=2)
```

```
A.add_edge('Vedado','Habana Vieja', weight=4)
A.add_edge('Vedado','Habana Vieja', weight=5)
A.add_edge('Habana Vieja','Habana del Este', weight=2)
A.add_edge('Habana Vieja','Habana del Este', weight=4)

black=[('Lisa','Marianao' ),('Playa','Vedado'),('Vedado','Habana
    Vieja'),
   ('Habana Vieja','Habana del Este'),('Marianao','Playa')]
red=[('Marianao','Playa'),('Vedado','Habana Vieja'),
   ('Habana Vieja','Habana del Este')]
bl=[('Vedado','Habana Vieja')]

lista=['Lisa','Marianao','Playa','Vedado','Habana Vieja'],['Habana
    del Este']

#posicion=nx.shell_layout(A, nlist=lista, scale=0.50, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

#posicion=nx.spectral_layout(A, weight='distans', scale=0.40,
    center=None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.30, center=None, dim=2)

posicion=nx.spring_layout(A, k=1, iterations=200, threshold=0.0001,
     weight='weight', scale=0.5)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, {'Marianao','Vedado','Habana Vieja
    '}, align='vertical', scale=1, center=None, aspect_ratio
    =1.3333333333333333)

nx.draw_networkx_nodes(A,posicion,
                       node_size=800, node_color= 'y')
nx.draw_networkx_edges(A, posicion, edgelist=bl, width=10, alpha
    =0.5,
edge_color='b')
nx.draw_networkx_edges(A, posicion, edgelist=red, width=5, alpha
    =0.5,
edge_color='r')
nx.draw_networkx_edges(A, posicion, edgelist=black, width=2,
edge_color='Black')
for p in posicion:
    posicion[p][1] -= 0.01
nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial
    ',
                        font_color='Black', font_weight='bold')


plot.axis('off')
plot.savefig("Graf7_spring_layout.eps")
plot.show(A)
```

Graf7.py

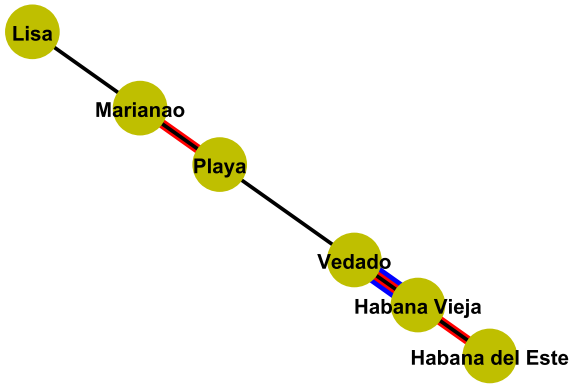

Figura 7: Multigrafo no dirigido acíclico, donde la diferencia de colores de los arcos representan los diferentes caminos.

9. Multigrafo no dirigido cíclico

Un ejemplo donde podemos utilizar este tipo de grafos es: se quiere representar cuantas llamadas (aristas)se realizaron entre un grupo de personas (vértices), donde nos importa saber la duración de cada llamada durante un día determinado. Este grafo se ve representado en la figura 8.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 8 el algoritmo *spectral layout*, dando buenos resultados estetico.

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


A=nx.MultiGraph()
A.add_edge('Tel_1','Tel_2', weight=2)
A.add_edge('Tel_1','Tel_2', weight=3)
```

```
A.add_edge('Tel_1','Tel_2', weight=4)
A.add_edge('Tel_1','Tel_3', weight=2)
A.add_edge('Tel_1','Tel_4', weight=4)
A.add_edge('Tel_2','Tel_4', weight=2)
A.add_edge('Tel_2','Tel_4', weight=3)
A.add_edge('Tel_2','Tel_3', weight=2)
A.add_edge('Tel_3','Tel_5', weight=2)
A.add_edge('Tel_3','Tel_5', weight=3)
A.add_edge('Tel_3','Tel_5', weight=4)
A.add_edge('Tel_5','Tel_6', weight=2)
A.add_edge('Tel_5','Tel_6', weight=3)
A.add_edge('Tel_1','Tel_6', weight=2)
A.add_edge('Tel_1','Tel_6', weight=3)
black=[('Tel_1','Tel_2' ),('Tel_1','Tel_3'),('Tel_2','Tel_4'),
    ('Tel_2','Tel_3'),('Tel_3','Tel_5'),('Tel_3','Tel_5'),
    ('Tel_5','Tel_6'),('Tel_1','Tel_6')]

red=[('Tel_1','Tel_2'),('Tel_2','Tel_4'),
    ('Tel_1','Tel_2'),('Tel_1','Tel_6'),('Tel_5','Tel_6')]
bl=[('Tel_1','Tel_2'),('Tel_3','Tel_5')]

lista=['Tel_1','Tel_2','Tel_3','Tel_4','Tel_5'],['Tel_6']
#posicion=nx.shell_layout(A, nlist=lista , scale=0.50, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

posicion=nx.spectral_layout(A, weight='weight', scale=0.50, center=
    None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.30, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=200, threshold
    =0.0001, weight='weight', scale=0.5)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, {'Marianao','Vedado','Habana Vieja
    '}, align='vertical', scale=1, center=None, aspect_ratio
    =1.3333333333333333)

nx.draw_networkx_nodes(A,posicion ,
                       node_size=1000, node_color= 'y')
nx.draw_networkx_edges(A, posicion , edgelist=bl, width=10, alpha
    =0.5,
edge_color='b')
nx.draw_networkx_edges(A, posicion , edgelist=red, width=5, alpha
    =0.5,
edge_color='r')
nx.draw_networkx_edges(A, posicion , edgelist=black, width=2,
edge_color='Black')

nx.draw_networkx_labels(A,posicion , font_size=11,font_family='arial
    ',
```

```
                        font_color='Black', font_weight='bold')


plot.axis('off')
plot.savefig("Graf8_spectral_layout.eps")
plot.show(A)
```

Graf8.py

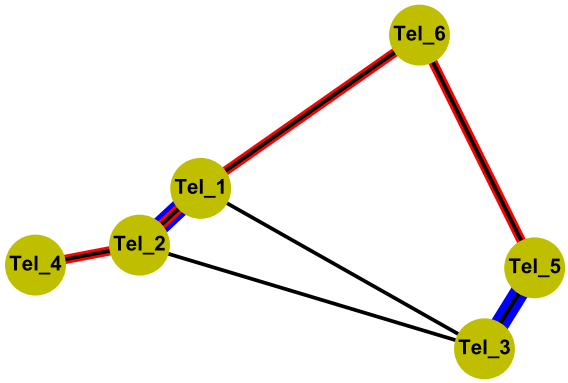

Figura 8: Multigrafo no dirigido cíclico, donde las aristas de diferentes colores representan la existencia de las llamadas con distinta duración.

10. Multigrafo no dirigido reflexivo

Partiendo del ejemplo de la sección 3 se pretende construir un grafo más informativo a la hora de su análisis, por lo que se le agrega como arista (el hecho que las relaciones sexuales mantenidas fueran dentro de una relación formal). Este grafo se muestra en la figura 9.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 9 el algoritmo *circular layout*, ya que el mismo permite una mejor compresión de este ejemplo mediante la ubicación circular de los nodos.

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


A=nx.MultiDiGraph()


A.add_edge('Luis(15)','Luis(15)'     ,weight=2)
A.add_edge('Maria(15)','Yanet(17)'   ,weight=2)
A.add_edge('Yanet(17)','Yanet(17)'   ,weight=2)


A.add_edge('Maria(15)','Fernando(16)' ,weight=2)
A.add_edge('Fernando(16)','Desisy(18)',weight=3)
A.add_edge('Fernando(16)','Desisy(18)',weight=2)
A.add_edge('Pedro(17)','Pedro(17)'   ,weight=2)
A.add_edge('Pedro(17)','Julia(18)'   ,weight=2)
A.add_edge('Pedro(17)','Rosi(16)'    ,weight=3)
A.add_edge('Pedro(17)','Rosi(16)'    ,weight=2)
A.add_edge('Fernando(16)','Claudia(16)',weight=2)

nodes_reflex = {'Luis(15)','Fernando(16)','Julia(18)'}
nodes_no_reflex = {'Maria(15)','Yanet(17)','Rosi(16)','Desisy(18)',
                   'Pedro(17)','Claudia(16)'}

black=[('Maria(15)','Yanet(17)'),
  ('Maria(15)','Fernando(16)'),('Fernando(16)','Desisy(18)'),
  ('Yanet(17)','Julia(18)'),('Desisy(18)','Pedro(17)'),
  ('Pedro(17)','Julia(18)'),  ('Pedro(17)','Rosi(16)'),
  ('Fernando(16)','Claudia(16)')]

bl=[('Fernando(16)','Desisy(18)'),('Pedro(17)','Rosi(16)')]

lista=['Luis(15)','Julia(18)','Yanet(17)','Rosi(16)','Desisy(18)','
    Claudia(16)'], ['Maria(15)','Fernando(16)','Pedro(17)']

#posicion=nx.shell_layout(A, nlist=lista , scale=0.50, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

#posicion=nx.spectral_layout(A, weight='distans', scale=0.50,
    center=None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.30, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=200, threshold
    =0.0001, weight='weight', scale=0.5)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, {'Marianao','Vedado','Habana Vieja
    '}, align='vertical', scale=1, center=None, aspect_ratio
    =1.3333333333333333)
```

```
nx.draw_networkx_nodes(A,posicion,nodelist=nodes_reflex,
                       node_size=1800, node_color='#ea6d6d')
nx.draw_networkx_nodes(A,posicion,nodelist=nodes_no_reflex,
                       node_size=1500, node_color='#cde95b')
nx.draw_networkx_edges(A, posicion, edgelist=bl, width=5, alpha
    =0.5,
edge_color='b')
nx.draw_networkx_edges(A, posicion, edgelist=black, width=2,
edge_color='Black')
nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial
    ',
                        font_color='#4f5148', font_weight='bold')


plot.axis('off')
plot.savefig("Graf9_circular_layout.eps")
plot.show(A)
```

Graf9.py

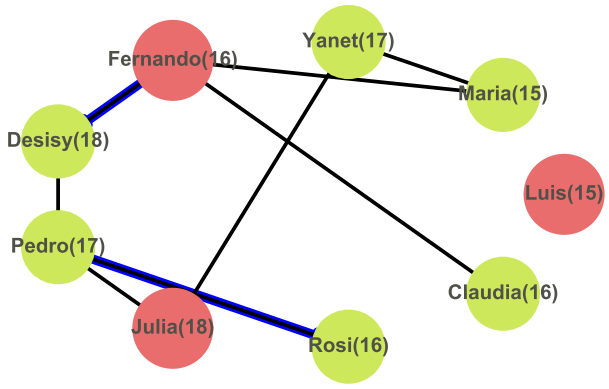

Figura 9: Multigrafo no dirigido reflexivo, donde las aristas de diferentes colores representan la existencia de una relación formal entre los individuos y los vértices de color rojo representa los vértices con aristas reflexivas.

11. Multigrafo dirigido acíclico

Considerando que se quiere trazar las diferentes rutas (teniendo en cuenta el sentido de los mimas) que llevan de una provincia a otra en Cuba (Pinar del

Río, Artemisa, La Habana, Mayabeque, Matanza, Cienfuegos, Villa Clara). Tomando como vértices las provincias y como aristas las carreteras que las unen, como muestra la figura 10.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 10 el algoritmo *spectral layout*, ya que el mismo logro un buen resultado en la ubicación de los nodos facilitando la compresión del grafo.

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


A=nx.MultiDiGraph()
A.add_edge('PR','A',      weight=2)
A.add_edge('A','LH',      weight=2)
A.add_edge('A','LH',      weight=4)
A.add_edge('LH','May',    weight=2)
A.add_edge('May','Mat',   weight=2)
A.add_edge('May','Mat',   weight=4)
A.add_edge('LH','May',    weight=4)
A.add_edge('LH','May',    weight=5)
A.add_edge('May','Mat',   weight=5)
A.add_edge('Mat','C',     weight=2)
A.add_edge('Mat','C',     weight=4)

black=[('PR','A' ),('LH','May'),('May','Mat'),
    ('Mat','C'),('A','LH')]
red=[('A','LH'),('May','Mat'),
    ('Mat','C'),('LH','May')]
bl=[('May','Mat'),('LH','May')]
lista=['PR','A','Mat','May','LH'],['C']
#posicion=nx.shell_layout(A, nlist=lista, scale=0.50, center=None,
    dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

posicion=nx.spectral_layout(A, weight='distans', scale=0.50, center
    =None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
    weight', scale=0.30, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=200, threshold
    =0.0001, weight='weight', scale=0.5)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, {'A','May','C'}, align='vertical',
     scale=0.5, center=None, aspect_ratio=1.3333333333333333)

nx.draw_networkx_nodes(A,posicion,
```

```
                        node_size=500, node_color= 'y')
nx.draw_networkx_edges(A, posicion, edgelist=bl, width=10, alpha
    =0.5,
edge_color='b')
nx.draw_networkx_edges(A, posicion, edgelist=red, width=5, alpha
    =0.5,
edge_color='r')
nx.draw_networkx_edges(A, posicion, edgelist=black, width=2,
edge_color='Black')
nx.draw_networkx_labels(A,posicion, font_size=11,font_family='arial
    ',
                        font_color='Black', font_weight='bold')


plot.axis('off')
plot.savefig("Graf10_spectral_layout.eps")
plot.show(A)
```

Graf10.py

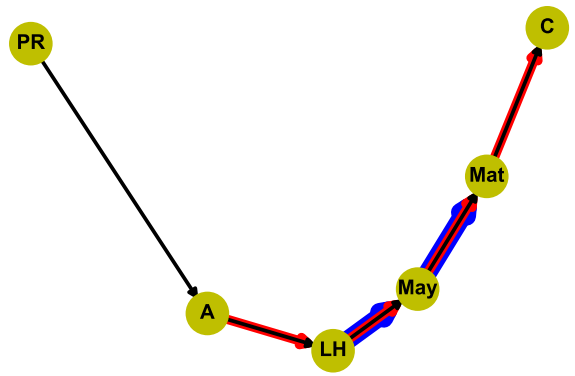

Figura 10: Multigrafo dirigido acíclico, donde las aristas de diferentes colores representan las existencia de mas de una ruta entre las diferente provincias.

12. Mltigrafo dirigido cíclico

"En el caso particular de que las redes reflejen una realidad social, los nodos pueden representar personas o entidades relacionadas con sus contextos, y las conexiones representarán relaciones sociales existentes entre ellos (amistad, parentesco, membresía, afinidad, etc.). A pesar de que intuitivamente las redes sociales se asemejan a los grafos matemáticos, es más habitual que en ellas se trabaje con distintos tipos de relaciones"[5] por lo que es necesario la utilización de multígrafos, que es la herramienta que contempla más de una relación entre dos nodos, con esto ganamos mayor riqueza en los datos a analizar. Por ejemplo, tenemos un grupo de personas que laboran en un departamento de Informática y se hace una encuesta donde se les pide que marque cuales de tres sentimientos (respeto, afinidad, rechazo) sienten por sus compañeros de trabajo, como muestra la figura 11.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 11 el algoritmo *kamada kawai layout*, ya que el mismo logro un buen resultado estético.

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


A=nx.MultiDiGraph()
A.add_edge('Persona1','Persona2', weight=2)
A.add_edge('Persona1','Persona2', weight=3)
A.add_edge('Persona1','Persona2', weight=4)
A.add_edge('Persona1','Persona3', weight=2)
A.add_edge('Persona1','Persona4', weight=4)
A.add_edge('Persona2','Persona4', weight=2)
A.add_edge('Persona2','Persona4', weight=3)
A.add_edge('Persona4','Persona3', weight=3)
A.add_edge('Persona3','Persona2', weight=2)
A.add_edge('Persona3','Persona5', weight=2)
A.add_edge('Persona3','Persona5', weight=3)
A.add_edge('Persona3','Persona5', weight=4)
A.add_edge('Persona5','Persona6', weight=2)
A.add_edge('Persona5','Persona6', weight=3)
A.add_edge('Persona1','Persona6', weight=2)
A.add_edge('Persona1','Persona6', weight=3)
black=[('Persona1','Persona2' ),('Persona1','Persona3'),('Persona2'
    ,'Persona4'),
   ('Persona3','Persona2'),('Persona3','Persona5'),('Persona3','
    Persona5'),
   ('Persona5','Persona6'),('Persona1','Persona6'),('Persona4','
    Persona3')]

red=[('Persona1','Persona2'),('Persona2','Persona4'),
   ('Persona1','Persona2'),('Persona1','Persona6'),('Persona5','
    Persona6')]
bl=[('Persona1','Persona2'),('Persona3','Persona5')]
```

```
29
30 lista=['Persona1','Persona2','Persona3','Persona4','Persona5'],['
       Persona6']
31 #posicion=nx.shell_layout(A, nlist=lista , scale=0.50, center=None,
       dim=2)
32
33 #posicion=nx.random_layout(A, center=None, dim=2)
34
35 #posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)
36
37 #posicion=nx.spectral_layout(A, weight='distans', scale=0.45,
       center=None, dim=2)
38
39 posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
       weight', scale=0.30, center=None, dim=2)
40
41 #posicion=nx.spring_layout(A, k=1, iterations=200, threshold
       =0.0001, weight='weight', scale=0.5)
42
43 #posicion=nx.rescale_layout( pos , escala = 1 )
44
45 #posicion=nx.bipartite_layout(A, {'Marianao','Vedado','Habana Vieja
       '}, align='vertical', scale=1, center=None, aspect_ratio
       =1.3333333333333333)
46
47 nx.draw_networkx_nodes(A,posicion ,
48                        node_size=500
49                        , node_color= 'y')
50 nx.draw_networkx_edges(A, posicion , edgelist=bl, width=10, alpha
       =0.5,
51 edge_color='b')
52 nx.draw_networkx_edges(A, posicion , edgelist=red, width=5, alpha
       =0.5,
53 edge_color='r')
54 nx.draw_networkx_edges(A, posicion , edgelist=black, width=2,
55 edge_color='#297033')
56 for p in posicion:
57     posicion[p][1]+= 0.07
58 nx.draw_networkx_labels(A,posicion , font_size=11,font_family='arial
       ',
59                         font_color='Black', font_weight='bold')
60
61
62 plot.axis('off')
63 plot.savefig("Graf11_kamada_kawai_layout.eps")
64 plot.show(A)
```

Graf11.py

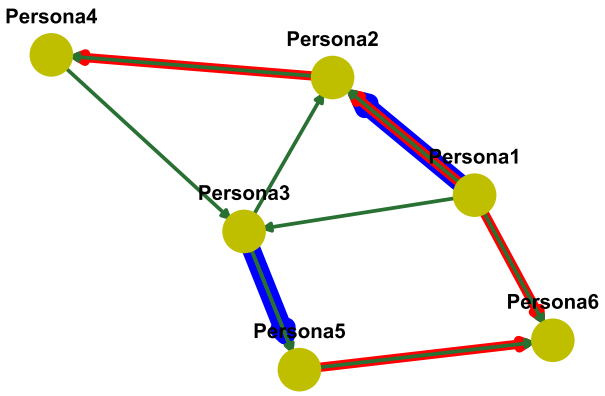

Figura 11: Multigrafo dirigido cíclico, donde las aristas de diferentes colores representan las existencia de más de una opinión sobre sus compañeros.

13. Multigrafo dirigido reflexivo

Una de las aplicaciones de estos grafos es en el modelado del funcionamiento de una máquina de estados. Por ejemplo, se desea modelar un grafo que represente el funcionamiento de una máquina muy sencilla de golosinas, que responde a las siguientes reglas:

1 Cada golosina vale $ 0.25.

2 La máquina acepta sólo monedas de $ 0.10 y de $ 0.05.

3 La máquina NO da vuelto.

Para modelar el grafo, debemos considerar distintos "estados de dinero ingresado", y como se va pasando de uno a otro. Por ejemplo, si en un momento tenemos ingresados 5 centavos, y agregamos 5 centavos más, pasamos a otro estado, que es el mismo que si hubiésemos ingresado 10 centavos al principio. Llamemos A, B, C, D, E y F a los estados que representan 0, 5, 10, 15, 20, 25 centavos ingresados respectivamente. Las transiciones de un estado a otro se harán por ingreso de 5 o 10 centavos, o al presionar el botón para obtener los caramelos (G). Como se muestra en la figura 12.

Algorito de diseño.
Después de probar con varios de los algoritmos de diseño existentes, se decide usar para la representación del grafo de la figura 12 el algoritmo *spectral*

layout, ya que este algoritmo mostro el mejor de los resultados a la hora de representar este ejemplo.

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


A=nx.MultiDiGraph()
A.add_edge('A','A',     weight=2)
A.add_edge('A','B',     weight=2)
A.add_edge('A','C',     weight=2)
A.add_edge('B','B',     weight=2)
A.add_edge('B','C',     weight=2)
A.add_edge('B','D',     weight=2)
A.add_edge('C','C',     weight=2)
A.add_edge('C','D',     weight=2)
A.add_edge('C','E',     weight=2)
A.add_edge('D','D',     weight=2)
A.add_edge('D','E',     weight=2)
A.add_edge('D','F',     weight=2)
A.add_edge('E','E',     weight=2)
A.add_edge('E','F',     weight=2)
A.add_edge('E','F',     weight=4)
A.add_edge('F','F',     weight=2)
A.add_edge('F','A',     weight=2)


black=[('A','A' ),('A','B'),('A','C'),
    ('B','B'),('B','C'),('B','D'),('C','C'),('C','D'),('C','E'),
    ('D','D'),('D','E'),('D','F'),('E','E'),('E','F'),('E','F'),('F'
     ,'A')]

bl=[('E','F')]
lista=['A','B','C','D','E'],['F']
#posicion=nx.shell_layout(A, nlist=lista , scale=0.50, center=None,
     dim=2)

#posicion=nx.random_layout(A, center=None, dim=2)

#posicion=nx.circular_layout(A, scale=0.5, center=None, dim=2)

posicion=nx.spectral_layout(A, weight='distans', scale=0.50, center
    =None, dim=2)

#posicion=nx.kamada_kawai_layout(A, dist=None, pos=None, weight='
     weight', scale=0.30, center=None, dim=2)

#posicion=nx.spring_layout(A, k=1, iterations=200, threshold
     =0.0001, weight='weight', scale=0.5)

#posicion=nx.rescale_layout( pos , escala = 1 )

#posicion=nx.bipartite_layout(A, {'Marianao','Vedado','Habana Vieja
     '}, align='vertical', scale=1, center=None, aspect_ratio
     =1.3333333333333333)

nx.draw_networkx_nodes(A,posicion ,
```

```
                        node_size=600, node_color= '#48b457')
nx.draw_networkx_edges(A, posicion, edgelist=bl, width=10, alpha
    =0.5,
edge_color='b',arrowsize=20)

nx.draw_networkx_edges(A, posicion, edgelist=black, width=2,
edge_color='Black',arrowsize=20)
nx.draw_networkx_labels(A,posicion, font_size=14,font_family='arial
    ',
                        font_color='Black', font_weight='bold')


plot.axis('off')
plot.savefig("Graf12_spectral_layout.eps")
plot.show(A)
```

Graf12.py

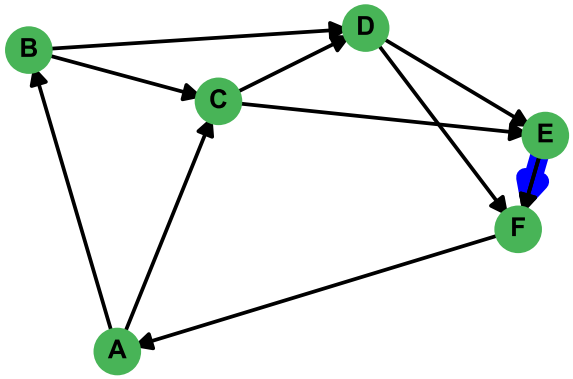

Figura 12: Multigrafo dirigido reflexivo, donde las aristas de diferentes colores representan las existencia de mas de una forma de pasar de un estado a otro.

Referencias

[1] Andrés Aiello and Rodrigo Ignacio Silveira. Trazado de grafos mediante métodos dirigidos por fuerzas: revisión del estado del arte y presentación de algoritmos para grafos donde los vértices son regiones geográficas. Technical report, Universidad de Buenos Aires, https://dccg.upc.edu/people/rodrigo/pubs/MScThesis.pdf, Diciembre 2004.

[2] Amalia Duch Brown. Grafos. `https://www.cs.upc.edu/~duch/home/duch/grafos.pdf`, Octubre 2007.

[3] Chakraborty, Anwesha, Trina Dutta, Mondal, Sushmita, Nath, and Asoke. Application of graph theory in social media. *International Journal of Computer Sciences and Engineering*, 6:722–729, 10 2018.

[4] NetworkX Developers Copyright 2004-2019. Overview of networkx. https://networkx.github.io/documentation/latest/index.html, Febrero 2019.

[5] R. A. Hanneman and M. Riddle. Introduction to social network methods. `http://faculty.ucr.edu/~hanneman/.Consultado`, September 2013.

[6] Kimball Martin. *Graph Theory and Social Networks*. Spring, Abril 2014.

[7] Mauricio Beltrán Pascuala, Azahara Muñoz Martínez, and Ángel Muñoz Alamillos. Redesbayesianas aplicadas a problemas de credit scoring.una aplicación práctica. `http://www.elsevier.es/en-revista-cuadernos-economia-329-articulo-redes-bayesianas/-aplicadas-problemas-credit`, Octubre 2013.