Tarea 2

Jesus Angel Patlán Castillo (5621)

26 de febrero de 2019

En esta tarea se analizarán distintos tipos de acomodo visual grafos generados a partir de la librería NetworkX [4] de Python [1]. Se estudiará particularmente los grafos utilizados para la tarea 1, la librería Matplotlib [5] para generar el grafo en los distintos algoritmos estudiados y para guardar el grafo en el formato ".eps". El código empleado se obtuvo consultando la documentación oficial de la librería NetworkX [3] y guías suplementarias [7, 11]. Las imágenes y el código se encuentran disponibles directamente en mi repositorio [10]. Las distintas formas de trazar los grafos son:

```
#Distintos algoritmos para trazar los grafos
nx.draw(G, pos=nx.spectral_layout(G), node_color='r', edge_color='b', with_labels=True)
nx.draw(G, pos=nx.circular_layout(G), node_color='r', edge_color='b', with_labels=True)
nx.draw(G, pos=nx.random_layout(G), node_color='r', edge_color='b', with_labels=True)
nx.draw(G, pos=nx.shell_layout(G), node_color='r', edge_color='b', with_labels=True)
nx.draw(G, pos=nx.spring_layout(G), node_color='r', edge_color='b', with_labels=True)
```

1. Algoritmo Circular

Este algoritmo cuenta con las siguientes propiedades [12]:

- Los nodos del grafo se encuentran sobre una misma circunferencia.
- Las aristas se representa con lineas rectas.
- Requieren a lo más $O(n)$ tiempo para su trazado, siendo n el número de aristas.
- Se utilizan particularmente para grafos en las que se desea mostrar claramente la biconectividad entre nodos.

1.1. Grafo simple no dirigido acíclico

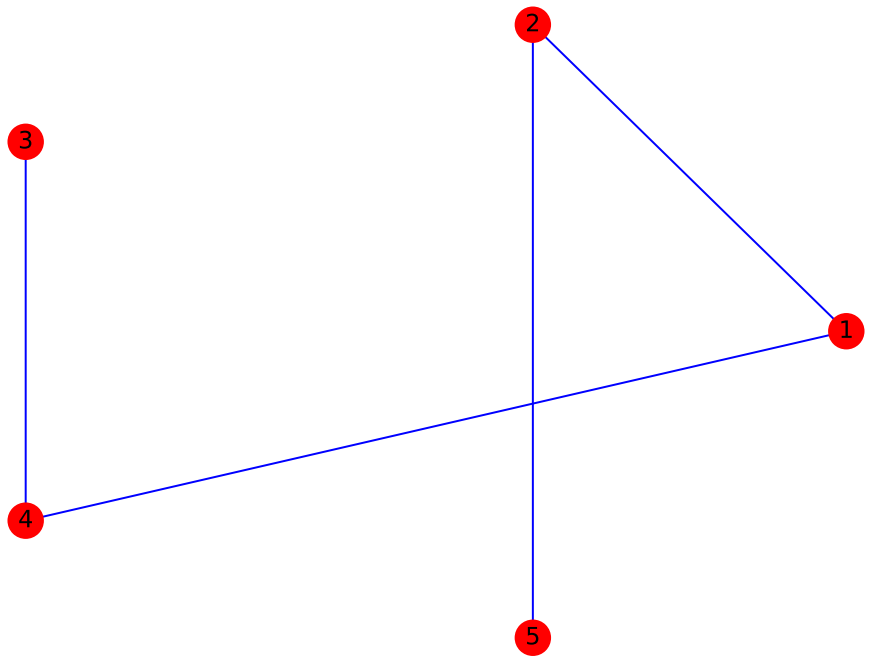

Figura 1: Grafo simple no dirigido acíclico.

1.2. Grafo simple no dirigido cíclico

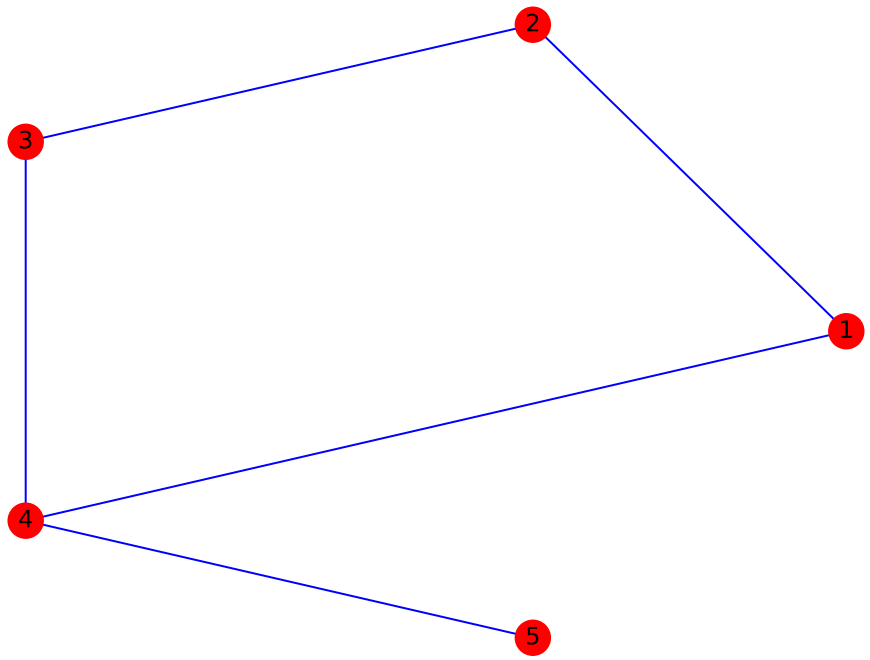

Figura 2: Grafo simple no dirigido cíclico.

1.3. Grafo simple dirigido cíclico

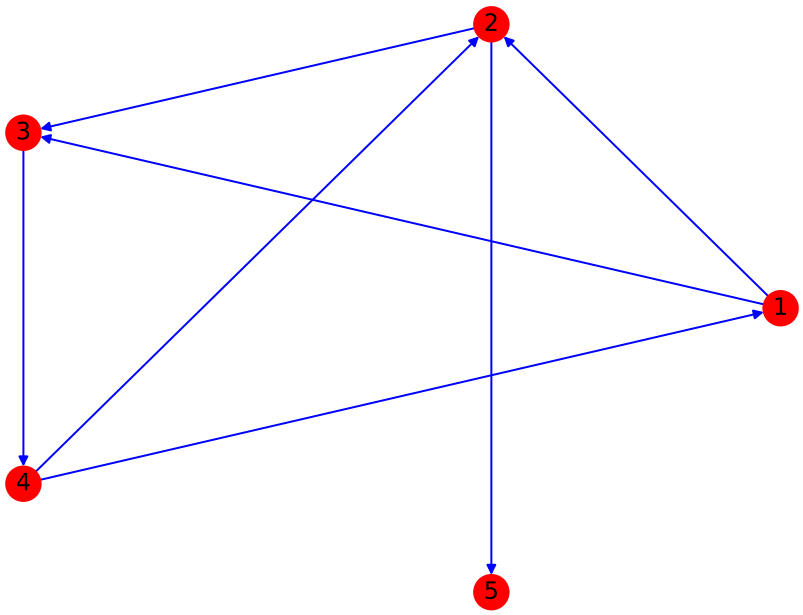

Figura 3: Grafo simple dirigido cíclico.

1.4. Multigrafo dirigido acíclico

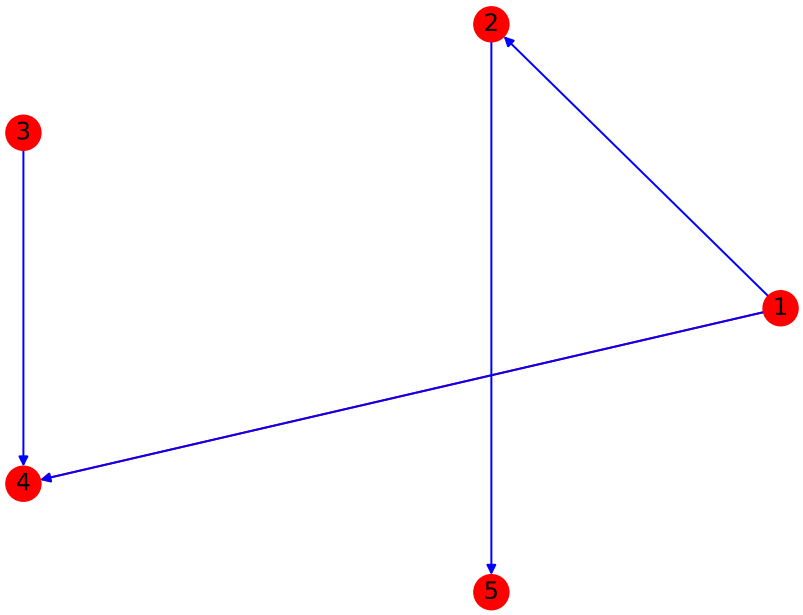

Figura 4: Multigrafo dirigido acíclico (los nodos 1 y 4 tienen múltiples aristas).

2. Algoritmo Random

Realiza el posicionamiento de los nodos de manera aleatoria, teniendo únicamente de restricción la ubicación de los nodos para no acomodar nodos encima de otros. Dada esta única restricción, su complejidad es nula [9].

2.1. Multigrafo no dirigido cíclico

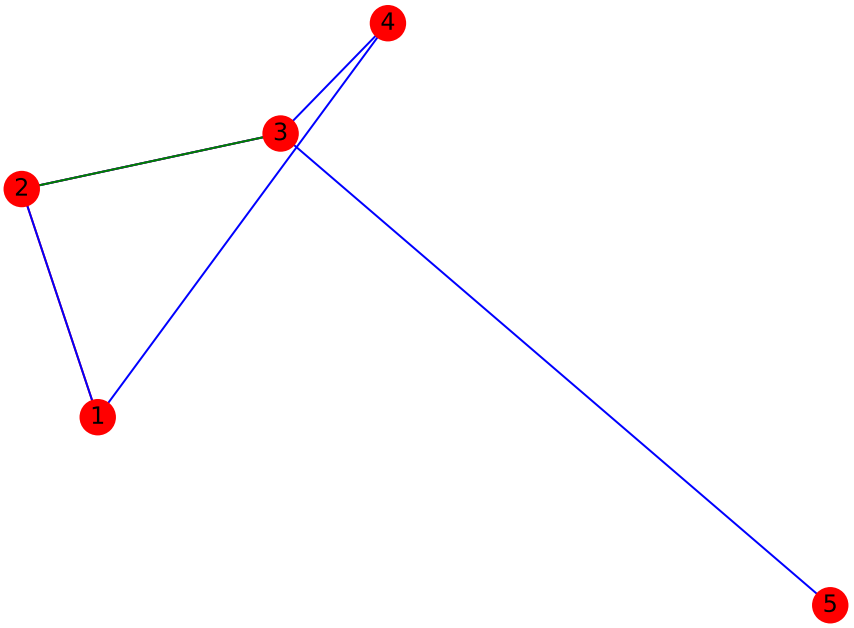

Figura 5: Multigrafo no dirigido cíclico (los nodos 1, 2 y 3 tienen múltiples aristas).

3. Algoritmo *Spectral*

El algoritmo *spectral* utiliza eigenvectores de la matriz del grafo (también conocido como el laplaciano). Dado un grafo ponderado, el algoritmo tiene como fin dar una relación entre los pesos de las aristas y la longitud de las aristas entre los nodos, esto es, entre mayor sea el peso de las aristas, más corta será la distancia entre los nodos [6].

3.1. Grafo simple dirigido reflexivo

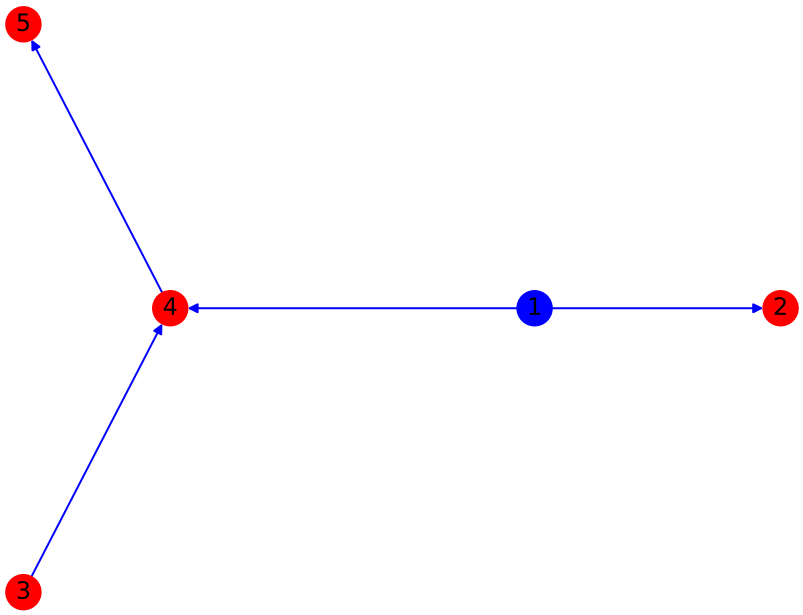

Figura 6: Grafo simple dirigido reflexivo (el nodo 1 tiene una arista reflexiva).

3.2. Multigrafo no dirigido acíclico

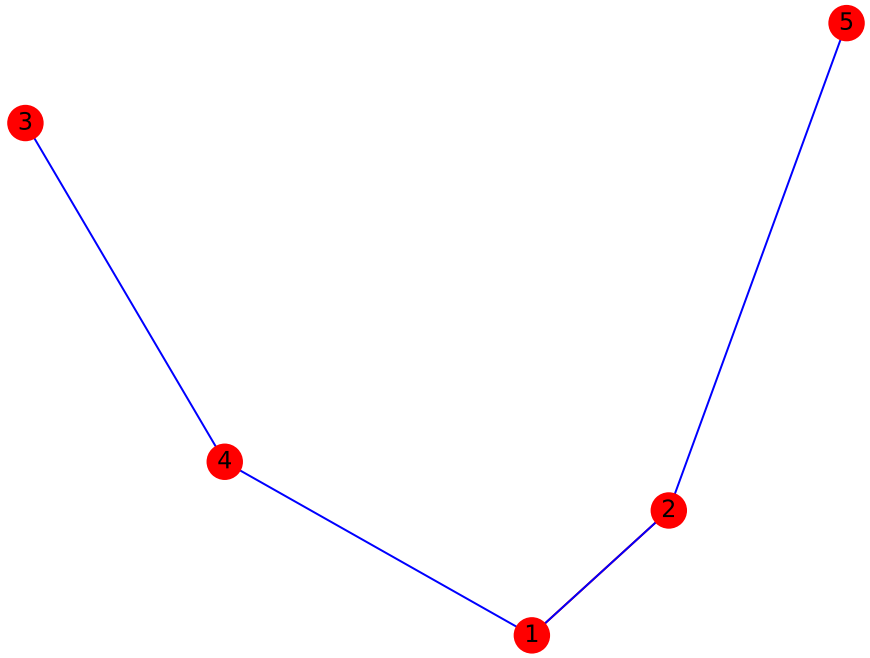

Figura 7: Multigrafo no dirigido acíclico (Los nodos 1 y 2 tienen múltiples aristas)

3.3. Multigrafo dirigido cíclico

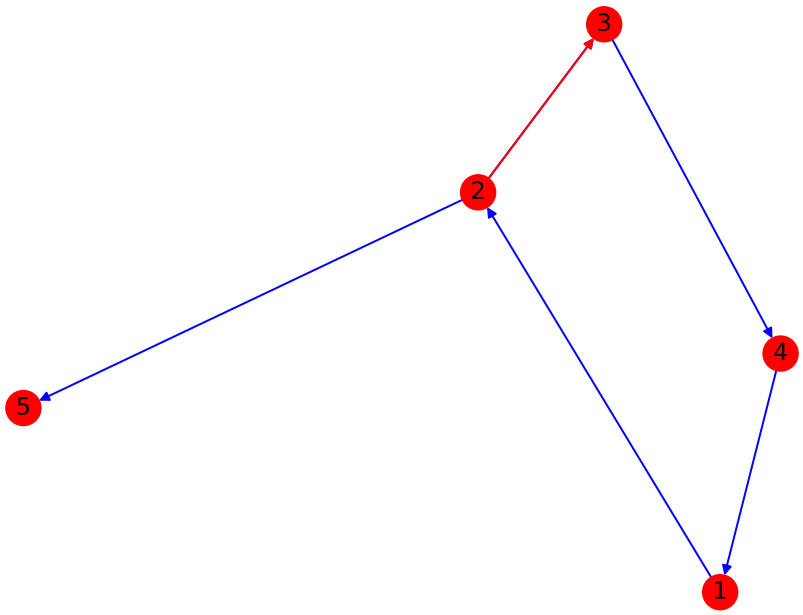

Figura 8: Multigrafo dirigido cíclico (los nodos 2 y 3 tienen múltiples aristas).

3.4. Multigrafo dirigido reflexivo

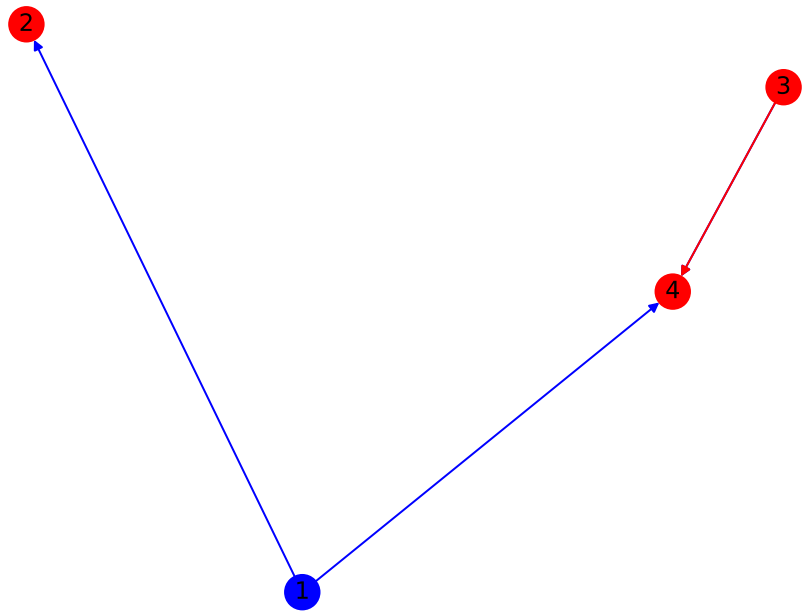

Figura 9: Multigrafo dirigido reflexivo (los nodos 3 y 4 tienen múltiples aristas, el nodo 1 tiene una arista reflexiva).

4. Algoritmo *Spring*

Basado en la física, el algoritmo *spring* trata de mantener que la suma de las fuerzas entre los nodos (los pesos de las aristas) sea de 0. Se basa esencialmente en la ley de Coulomb y en la métrica euclidiana para obtener estos valores, teniendo una complejidad computacional de $O(n^2)$ [2].

4.1. Grafo simple no dirigido reflexivo

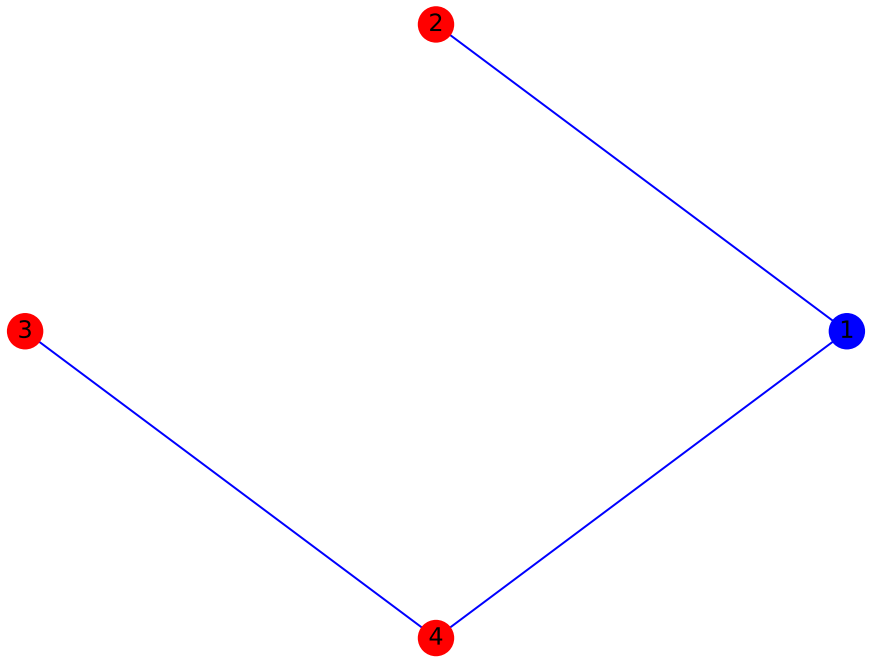

Figura 10: Grafo simple no dirigido reflexivo (el nodo 1 tiene una arista reflexiva).

4.2. Grafo simple dirigido acíclico

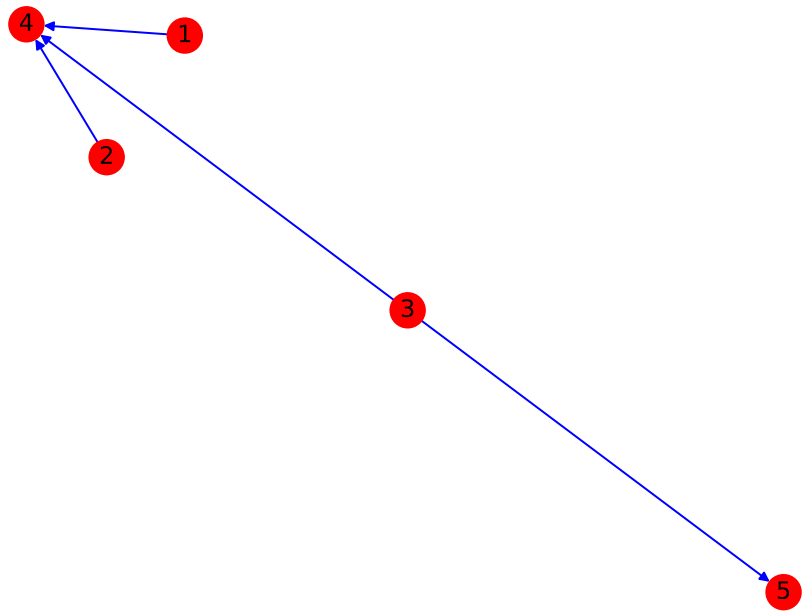

Figura 11: Grafo simple dirigido acíclico

5. Algoritmo *Shell*

Similar al algoritmo circular, el algoritmo *shell* realiza agrupaciones de nodos en círculos, con la diferencia que los nodos se pueden separar en distintos círculos a lo largo del grafo, por lo que la complejidad de este algoritmo es igual al algoritmo circular de $O(n)$ siendo n el número de aristas [8].

5.1. Multigrafo no dirigido reflexivo

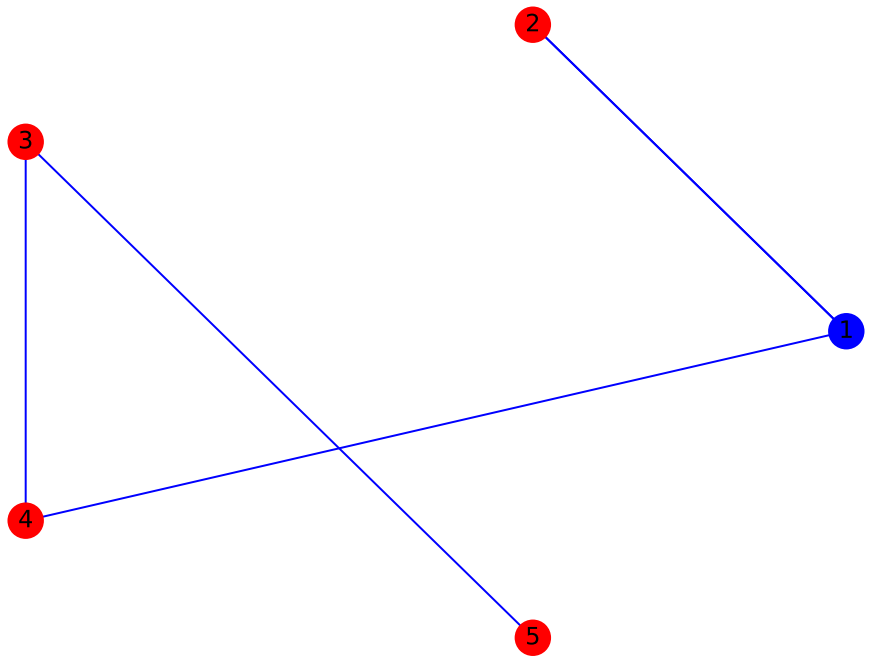

Figura 12: Multigrafo no dirigido reflexivo (los nodos 1 y 2 tienen múltiples aristas, el nodo 1 tiene una arista reflexiva).

6. Conclusiones entre algoritmos

Los algoritmos estudiados tienen distintos propósitos, siendo el tipo de visualización que se desea del grafo para determinar el algoritmo más adecuado. Por ejemplo, para simples pruebas de visualización se utilizaría el algoritmo *random* por su complejidad más simple entre los algoritmos, mientras que para otros tipos de grafos se utilizarían los otros tipos de algoritmos según el caso.

Referencias

[1] Python Software Foundation Versión 3.7.2. https://www.python.org/. Copyright ©2001-2019.

[2] Artículo: "Praktikum Algorithmen-Entwurf". http://wwwmayr.informatik.tu-muenchen.de/lehre/2012WS/algoprak/uebung/tutorial11.english.pdf.

[3] NetworkX developers con última actualización el 19 de Septiembre 2018. https://networkx.github.io/documentation/stable/index.html. Copyright ©2004-2018.

[4] NetworkX developers Versión 2.0. https://networkx.github.io/. Copyright ©2004-2018.

[5] The Matplotlib development team Versión 3.0.2. https://matplotlib.org/. Copyright @2002-2012.

[6] Artículo: "Spectral Graph Drawing". http://www.cis.upenn.edu/ cis515/ -15-graph-drawing.pdf.

[7] Plotting NetworkX graph in Python Pregunta en Stackoverflow. https://stackoverflow.com/questions/44692644/plotting-networkx-graph-in-python. Copyright ©2019.

[8] Documentación oficial Networkx: Shell Layout. https://networkx.github.io/documentation/networkx-1.10/$_m odules/networkx/drawing/layout.html shell_l ayout$.

[9] Documentación oficial Rogue Wave Software. https://docs.roguewave.com/visualization/views/6.1/views.htmlpage/ Options/layouts.51.19.html.

[10] Jesús Angel Patlán Castillo. Repositorio Optimización Flujo en Redes. https://github.com/JAPatlanC/Flujo-Redes.

[11] "How to set colors for nodes in NetworkXRespuesta de Abdallah Sobehy. https://stackoverflow.com/questions/27030473/how-to-set-colors-for-nodes-in-networkx-python.

[12] Janet M Six and Ioannis G Tollis. A framework for circular drawings of networks. In *International Symposium on Graph Drawing*, pages 107–116. Springer, 1999.