Homework Assignment 1: Network Flows

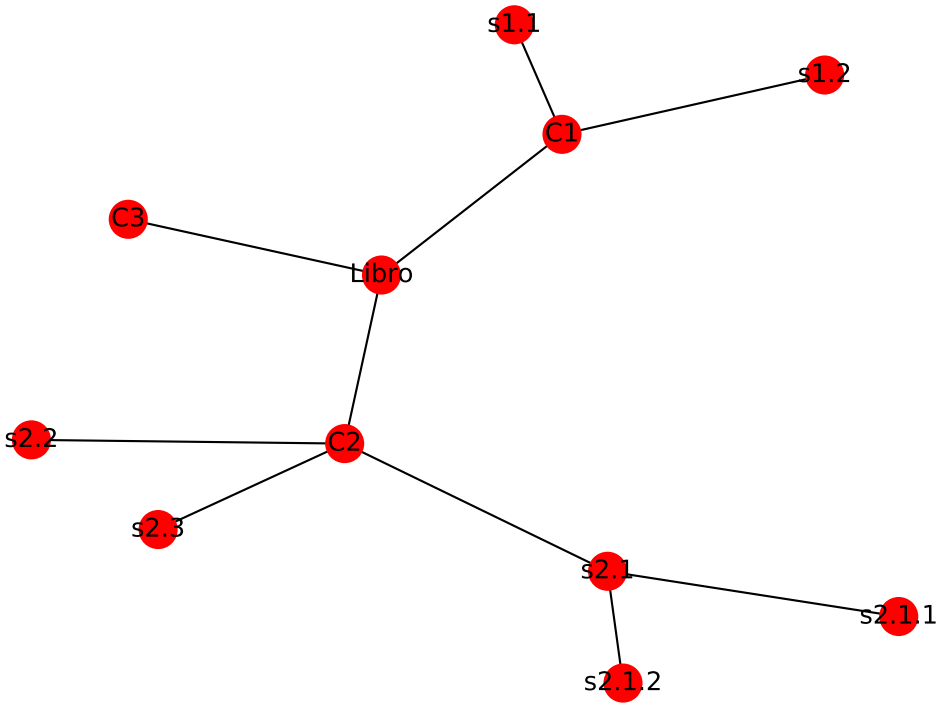
Oscar Alejandro Hernández López

Undirected acyclic graph

Consider the general index of a book. Such index is a tree and has a relation with the chapters and these with the sections. The root, the node *Book*, has three subtrees which are the chapters *C1*,*C2* and *C3*. This three nodes are the inheritance of the node named *Book*. Also we have two subtrees with non trivial relations, for example *C2* has another three subtrees which represent sections *s2.1*, *s2.2*, *s2.3*. [1]

```
import networkx as nx                                   #Library to create graphs
import matplotlib.pyplot as plt                         #Library to show graphs

G = nx.Graph()                                          #Create an empty graph

G.add_node('Libro')                                     #Add a simple node
G.add_nodes_from(['C1','C2','C3'])                      #Add a list of nodes
G.add_nodes_from(['s1.1','s1.2','s2.1','s2.2','s2.3'])
G.add_nodes_from(['s2.1.1','s2.1.2'])

G.add_edges_from([('Libro','C1',),('Libro','C2',),('Libro','C3')])   #Add a list of edges
G.add_edges_from([('C1','s1.1',),('C1','s1.2',)])
G.add_edges_from([('C2','s2.1',),('C2','s2.2',),('C2','s2.3')])
G.add_edges_from([('s2.1','s2.1.1'),('s2.1','s2.1.2')])

nx.draw(G,with_labels=True)                             #Draw the Graph named G
plt.savefig("Graph01.eps", format="EPS")                #Save figure in eps format
plt.show()                                              #Show (Print) Graph
```

01undirected_acyclic_graph.py

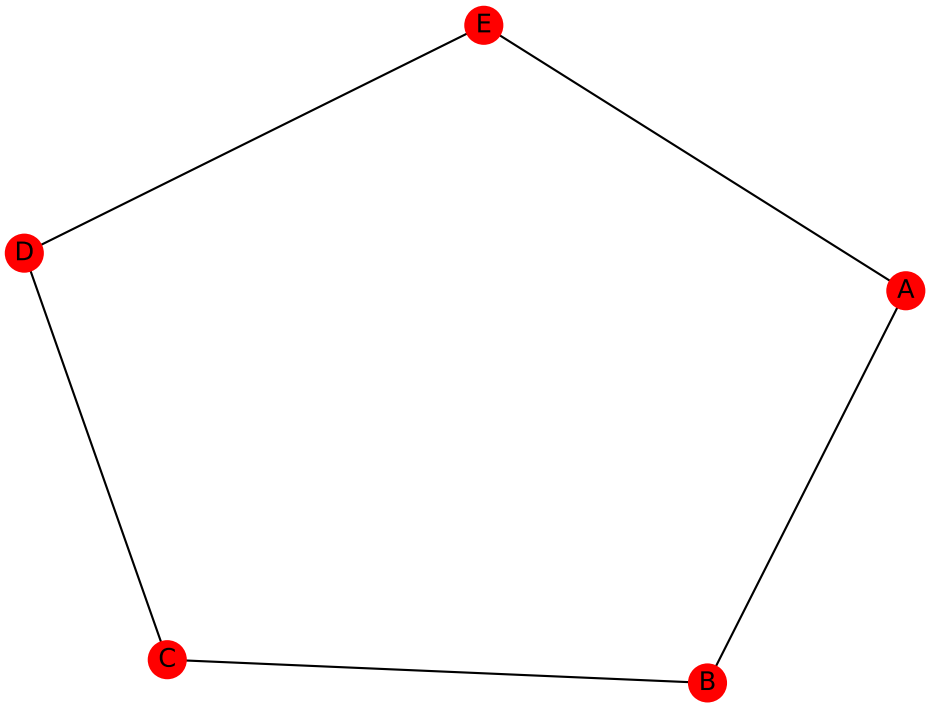


Undirected cyclic graph

In this situation we have five cities. Cities A and B are separated by a distance of 50 km, between cities B and C are 20 km and there is 30 km and 35 km between cities C and D, and D and E, respectively. The nearest distance is between cities A and E, with 15 km. [6]

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

G=nx.Graph()
G.add_nodes_from(['A','B','C','D','E'])

G.add_edges_from([('A','B'),('B','C'),('C','D'),('D','E'),('E','A')])

nx.draw(G, with_labels=True)
plt.savefig("Graph02.eps", format="EPS")
plt.show()
```

02undirected_cyclic_graph.py

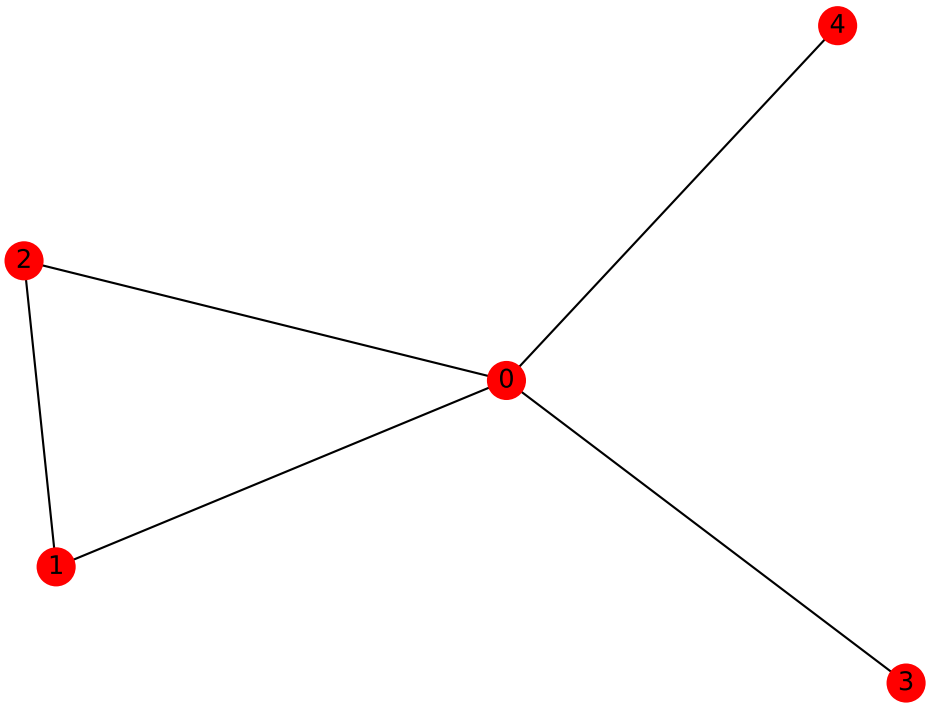


Undirected reflexive graph

The example shows relationships between republics, we can call it cluster (*0,1* and *2*). Clusters *3* and *4* are other republics that have significant ties with *0*, but almost no ties with anyone else. In case of republic *3* in addition to other relationships, can do other local relations or trades. [2]

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

H=nx.Graph()

H.add_nodes_from([0,1,2,3,4],)

H.add_edges_from([(0,1),(0,2),(1,2),(0,3),(0,4),(3,3)])

nx.draw(H, with_labels=True)
plt.savefig("Graph03.eps", format="EPS")
plt.show()
```

03undirected_reflexive_graph.py

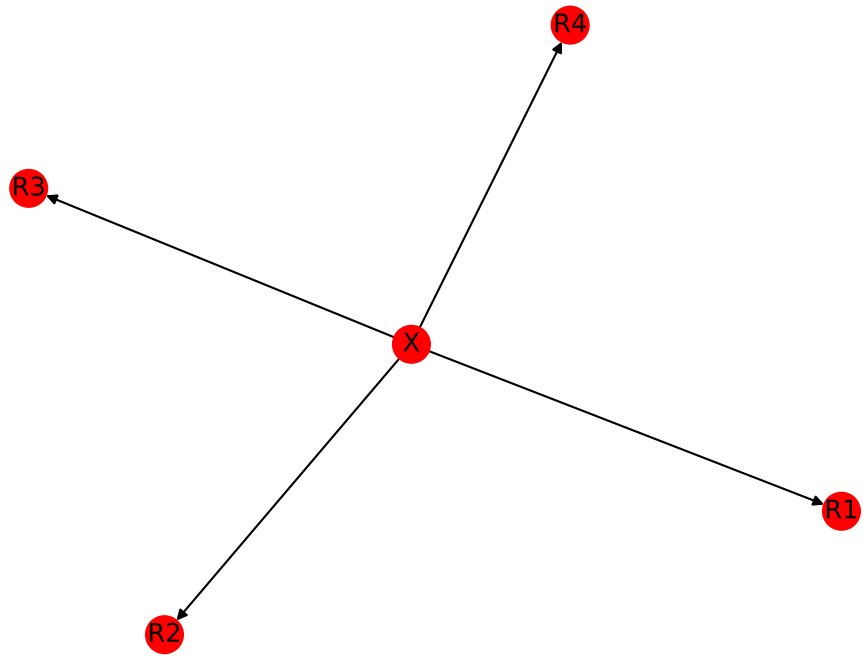


Directed acyclic graph

The example is about the Bayes's basic structure, which begins by a tree structure with its predictive variables and later connect the X variable with every single predictive variables R_n. In this acyclic directed graph every node represents a variable and every edge a probabilistic dependence, specifying the conditional probability of this variables with its sources. Every connections are identified by simple path that does not repeat vertices. [9]

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

G=nx.DiGraph()                                  #Create an empty directed graph
G.add_node("X")
G.add_nodes_from(["R1","R2","R3","R4"])

G.add_edges_from([("X","R1"),("X","R2"),("X","R3"),("X","R4")])
nx.draw(G,with_labels=True)
plt.savefig("Graph04.eps", format="EPS")
plt.show()
```

04directed_acyclic_graph.py

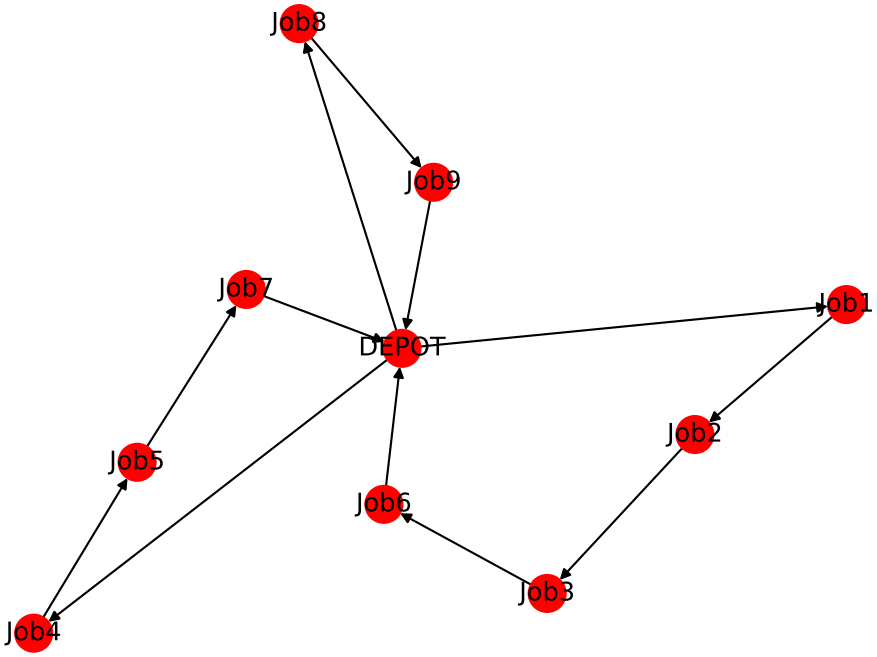


Directed cyclic graph

This situation consists in a fleet of homogeneous vehicles which need to distribute the jobs to respective customers. The routing graph is the set of nodes where a node *DEPOT* represents the main facility where the vehicles are loaded. The rest of nodes denote the rest of the customers. Each job corresponds to a unique customer. The number of vehicles is limited and each customer order needs to be delivered. Tours are allowed to visit multiple customers. [8]

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

G = nx.DiGraph()
G.add_node('DEPOT')
G.add_nodes_from(['Job1','Job2','Job3','Job6'])
G.add_nodes_from(['Job4','Job5','Job7'])
G.add_nodes_from(['Job8','Job9'])

G.add_path(['DEPOT','Job1','Job2','Job3','Job6','DEPOT'])  #Construct a path from the nodes in that order
G.add_path(['DEPOT','Job4','Job5','Job7','DEPOT'])
G.add_path(['DEPOT','Job8','Job9','DEPOT'])

nx.draw(G, with_labels=True)
plt.savefig("Graph05.eps", format="EPS")
plt.show()
```

05directed_ciclic_graph.py

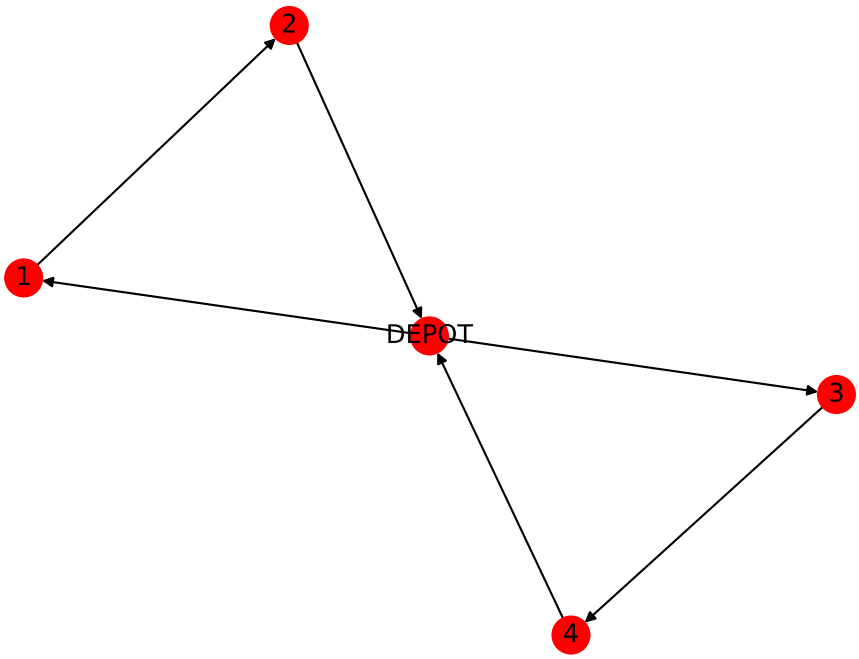


Directed reflexive graph

A global courier has to deliver some goods in 4 countries. For technical issues and distances between destinations a transport can only visit at most two destinations. Once it visit the costumers has to return to depot in order to begin maintenance for the next departure. In case an order had started from *DEPOT* and is cancelled the transport can return. [7]

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

R=nx.DiGraph()
R.add_nodes_from(["DEPOT","1","2","3","4"])

R.add_edges_from([("DEPOT","1"),("1","2"),("2","DEPOT")])
R.add_edges_from([("DEPOT","3"),("3","4"),("4","DEPOT"),("DEPOT","DEPOT")])

nx.draw(R, with_labels=True)
plt.savefig("Graph06.eps", format="EPS")
plt.show()
```

06directed_reflexive_graph.py

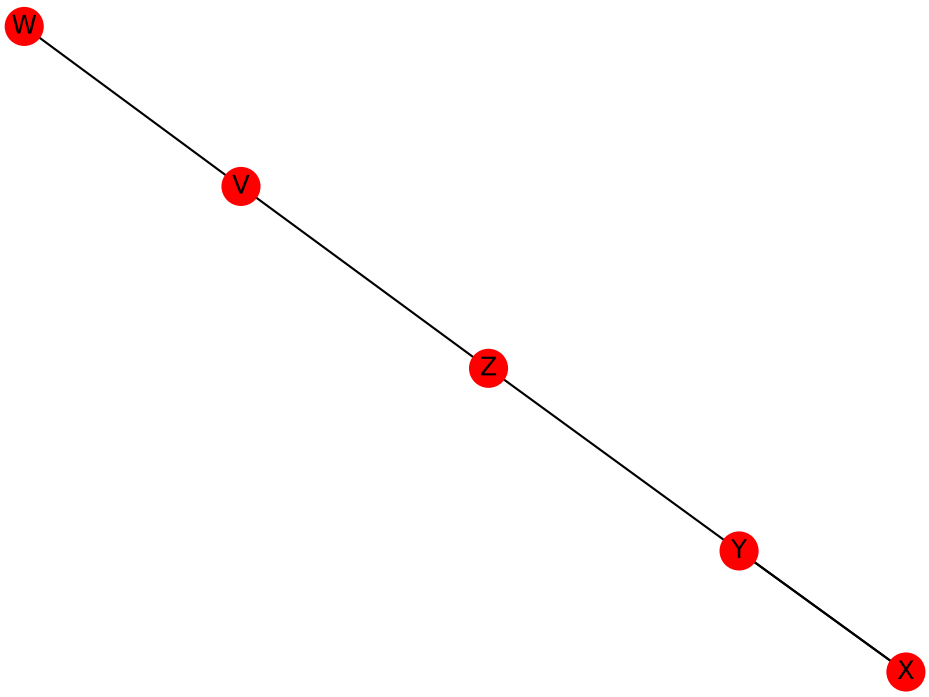


Undirected acyclic multigraph

Here we have a set of 5 tasks: X, Y, Z, V, W which need to be done to run a system. We also can consider we have 6 processors we need to assign to each task. Due to complexity of activities X and Y we shall assign two processors to execute at least one of them. This way we have two different relations between X and Y, and only one relation between the other tasks. [5]

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

M = nx.MultiGraph()                                    #Create an empty Multigraph

M.add_nodes_from(['X','Y','Z','V','W'])
M.add_edges_from([('X','Y'),('X','Y'),('Y','Z'),('Z','V'),('V','W')])

nx.draw(M, with_labels=True)
plt.savefig("Graph07.eps", format="EPS")
plt.show()
```

07undirected_acyclic_multigraph.py

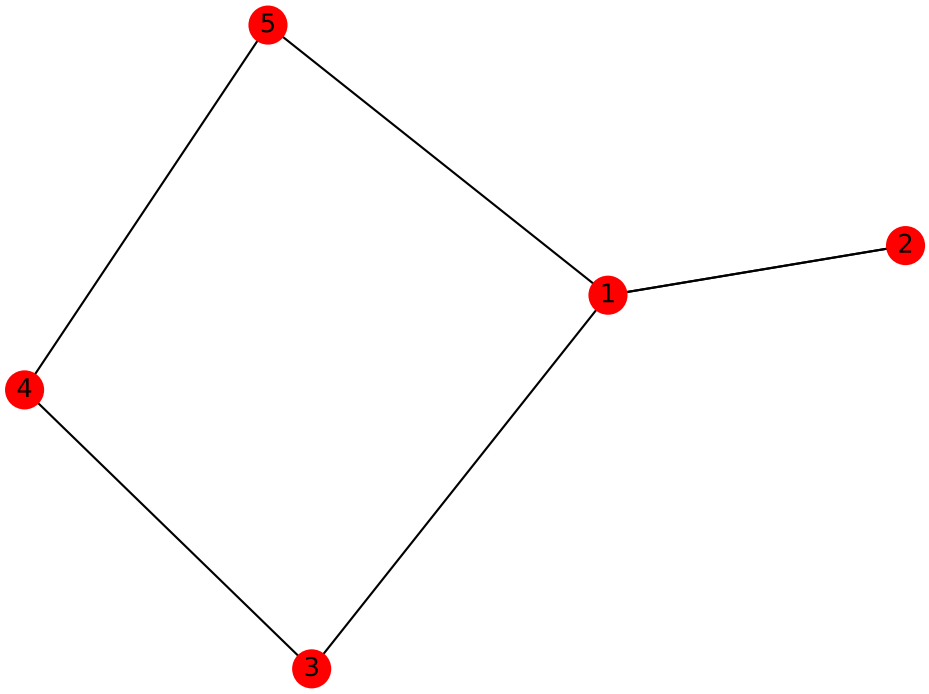


Undirected cyclic multigraph

This example is about a city divided in five zones. This five zones are united by six roads and people of the city, wants to cross each of the paths only once and end in the same place where they have started. [11]

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

M=nx.MultiGraph()

M.add_nodes_from([1,2,3,4,5])

M.add_edges_from([(1,2),(2,1),(1,3),(3,4),(4,5),(5,1)])

nx.draw(M, with_labels=True)
plt.savefig("Graph08.eps", format="EPS")
plt.show()
```

08undirected_cyclic_multigraph.py

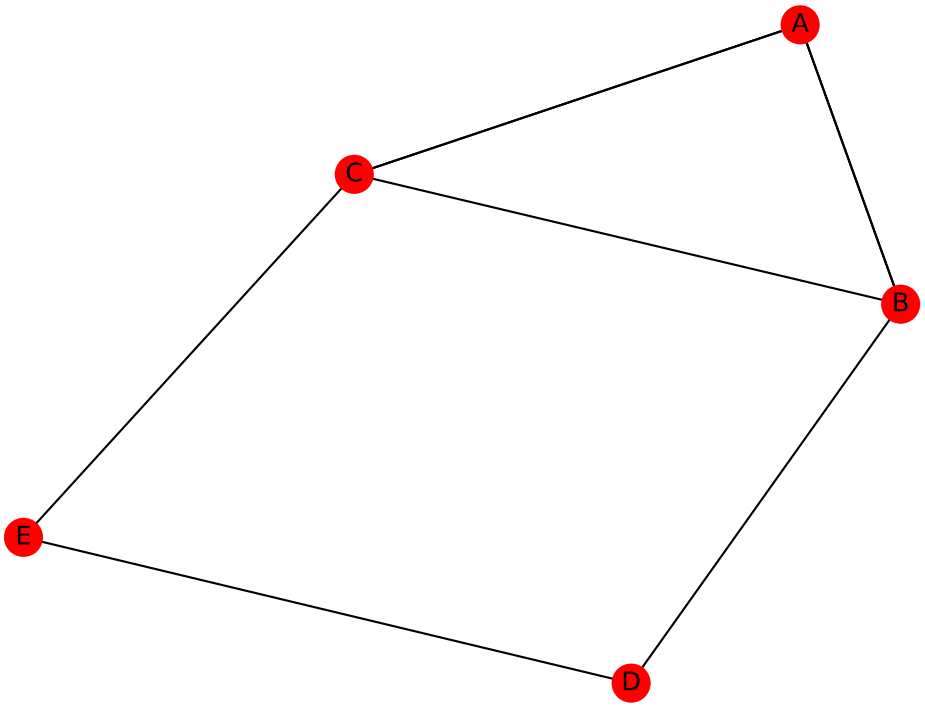


Undirected reflexive multigraph

Assume we have five workstations in a plant. The product needs to go through all the stations not matter the order or a secuence. We know station A has a section of quality control where the product is tested, and if it is not good enough needs to be reprocessed. Also relations in sections A-B or A-C can be choosing two paths, while relations B-C, B-D,C-E and D-E can only be performed by one path. [12]

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

M=nx.MultiGraph()
M.add_nodes_from(["A","B","C","D","E"])

M.add_edges_from([("A","A"),("A","B"),("A","B"),("A","C"),("A","C"),("B","C"),("B","D"),("C","
    E"),("D","E")])

nx.draw(M, with_labels=True)
plt.savefig("Graph09.eps", format="EPS")
plt.show()
```

09undirected_reflexive_multigraph.py

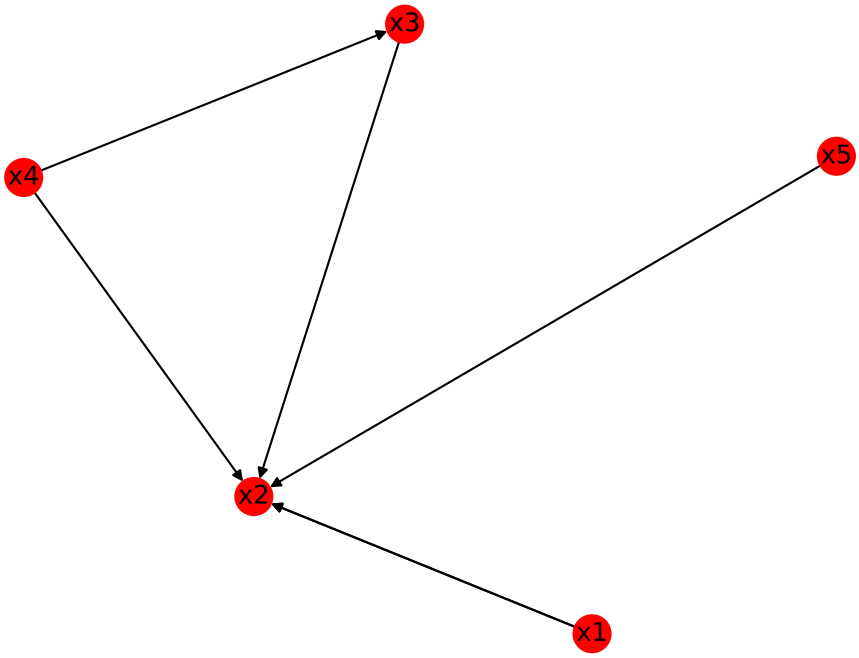


Directed acyclic multigraph

Certain company has 5 departments which some of them share relations in terms of information flow. Let say x_2 is the central deparment and needs to have a complete feedback. Deparments x_1 and x_5 has to share information only with the central deparment, the same as x_3, with the difference of two subjects. Whereas the department x_4 has to share information to departments x_3 and the central one. [3]

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

M=nx.MultiDiGraph()                          #Create an empty directed multigraph

M.add_nodes_from(["x1","x2","x3","x4","x5"])

M.add_edges_from([("x1","x2"),("x1","x2"),("x3","x2"),("x4","x3"),("x4","x2"),("x5","x2")])

nx.draw(M, with_labels=True)
plt.savefig("Graph10.eps", format="EPS")
plt.show()
```

10directed_acyclic_multigraph.py

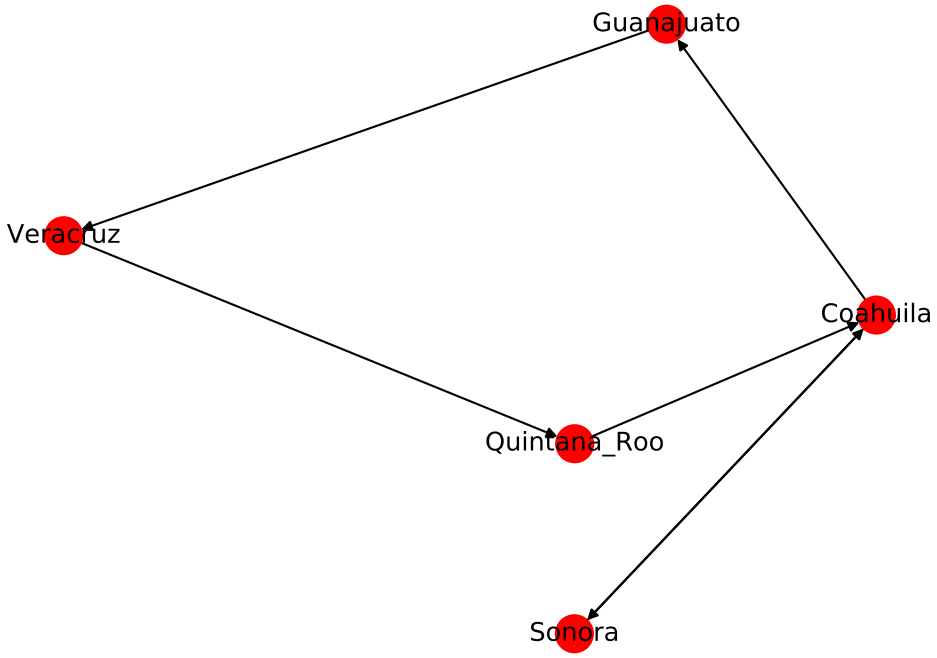


Directed cyclic multigraph

A student wants to spend his summer vacations in 5 states of Mexico: Coahuila, Sonora, Guanajuato, Veracruz and Quintana Roo. He will start in Coahuila, and then he will go to Sonora to visit his family. Then he will return to Coahuila to travel with his friends to Guanajuato, Veracruz and Quintana Roo in that order. He has the chance to return to Coahuila once he finish the visits. [4]

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

M=nx.MultiDiGraph()
M.add_nodes_from(["Coahuila","Sonora","Guanajuato","Veracruz","Quintana_Roo",])

M.add_edges_from([("Coahuila","Sonora"),("Sonora","Coahuila"),("Coahuila","Guanajuato"),("
    Guanajuato","Veracruz"),("Veracruz","Quintana_Roo"),("Quintana_Roo","Coahuila")])

nx.draw(M,with_labels=True)
plt.savefig("Graph11.eps", format="EPS")
plt.show()
```

11directed_cyclic_multigraph.py

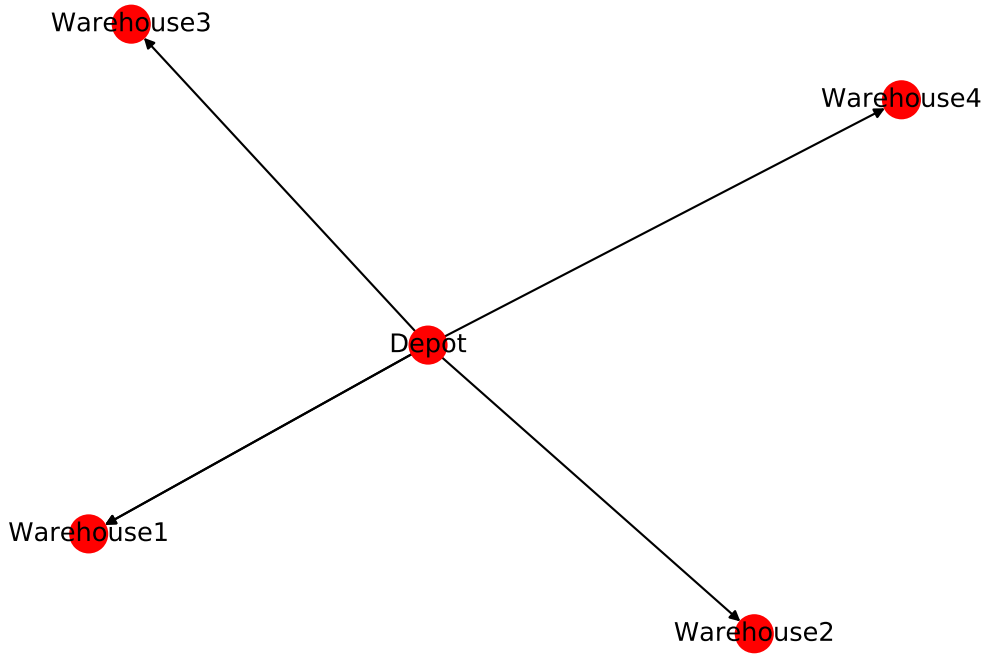


Directed reflexive multigraph

A logistic company has to deliver 4 orders of pallets from the depot to its warehouses. In order to speed operations up it will allocate 2 trucks to deliver in *warehouse1* and will use only one truck in the other warehouses. In order to keep stock, it will have to produce in-place a small quantity of pallets for inventory security and transport it to the small warehouse at the depot.[10]

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

M=nx.MultiDiGraph()

M.add_nodes_from(["Depot","Warehouse1","Warehouse2","Warehouse3","Warehouse4"])

M.add_edges_from([("Depot","Warehouse1"),("Depot","Warehouse1"),("Depot","Warehouse2"),("Depot
    ","Warehouse3"),("Depot","Warehouse4"),("Depot","Depot")])

nx.draw(M,with_labels=True)
plt.savefig("Graph12.eps", format="EPS")
plt.show()
```

12directed_reflexive_multigraph.py



References

[1] Ullman J.E. Aho A.V., Hopcroft J.E. *Estructura de Datos y Algoritmos*. Addison-Wesley, 1988.

[2] Maksim Tsvetovat Alexander Kouznetsov. Social network analysis for startups, 2019.

[3] Romel Félix Capcha Ventura. La teoría de grafos en la resolución de problemas aritméticos para estudiantes del laboratorio de investigación e innovación pedagógica de la universidad nacional daniel alcides carrión de pasco-2014. 2016.

[4] Norma Cabrera Luisa Gonzalez Carlos Garcia, Abel Rodriguez. Detection and correction of inconsistencies of cyclical references in database logical schemas. *Fac. Ing. Univ. Antioquia*, (55):165–173, 2010.

[5] Edgardo Ferro, Javier D Orozco, and Ricardo Cayssials. Asignación de tareas en un sistema distribuido de tiempo real duro. In *II Congreso Argentino de Ciencias de la Computación*, 1996.

[6] Zamantha Gonzalez. Grafos. pages 22–25, 2008.

[7] Jim Webber Ian Robinson and Emil Eifrem. *Graph Databases*. O'Reilly, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2 edition, 2015.

[8] Saadettin Erhan Kesen and Tolga Bektaş. Integrated production scheduling and distribution planning with time windows. In *Lean and Green Supply Chain Management*, pages 231–252. Springer, 2019.

[9] Carlos Arturo Vega Lebrun. Integracion de herramientas de tecnologias de informacion como soporte en la administracion del conocimiento, 2005.

[10] Jesus Garcia Miranda. *Matematica Discreta*. 2005.

[11] Sherman K Stein. The mathematician as an explorer. *Scientific American*, 204(5):148–161, 1961.

[12] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.