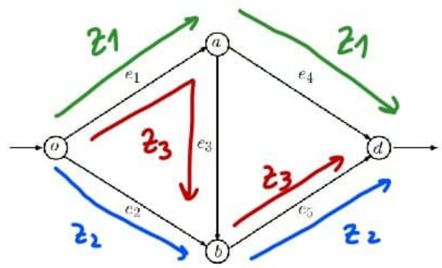
Network Dynamics and Learning, Homework 1

Silvia Giammarinaro

Exercise 1

$C_1 = C_4 = 3, \qquad C_2 = C_3 = C_5 = 2.$



a) We assume $C \in \mathbb{N}$.
For no feasible unitary flows we have to remove at least 5 units of capacity, this quantity is equal to the min-cut capacity.

b) Add 1 unit of capacity on e1 and e5.

c) delay function

$d_1(x) = d_5(x) = x+1 \qquad d_3(x) = 1$

$d_2(x) = d_4(x) = 5x+1$

- 3 paths from o to d:
  - $\gamma_1 : o \to a \to d$
  - $\gamma_2 : o \to b \to d$
  - $\gamma_3 : o \to a \to b \to d$
- flows on paths: $z = (z_1, z_2, z_3)$

  $z_1 + z_2 + z_3 = 1$
- delay on paths:

$$\Delta_1 = z_1 + z_3 + 1 + 5z_1 + 1 = 6z_1 + z_3 + 2$$
$$\Delta_2 = 5z_2 + 1 + z_2 + z_3 + 1 = 6z_2 + z_3 + 2$$
$$\Delta_3 = z_1 + z_3 + 1 + 1 + z_2 + z_3 + 1 = z_1 + z_2 + 2z_3 + 3$$

$\Delta_1 = 6z_1 + z_3 + 2$
$\Delta_2 = 6z_2 + z_3 + 2$
$\Delta_3 = z_1 + z_2 + 2z_3 + 3$

if $z_1 > 0 \rightarrow \Delta_1 \leq \Delta_2, \quad \Delta_1 \leq \Delta_3$

$6z_1 + \cancel{z_3} \cancel{+2} \leq 6z_2 + \cancel{z_3} \cancel{+2} \qquad z_1 = z_2$

if $z_2 > 0 \rightarrow \Delta_2 \leq \Delta_1, \quad \Delta_2 \leq \Delta_3$

$6z_2 + z_3 + 2 \leq z_1 + 5z_2 + 2z_3 + 3$
$\cancel{6z_2} + z_3 + 2 \leq \cancel{6z_2} + 2z_3 + 3$
$-z_3 \leq 1 \qquad z_3 \geq -1$

if $z_3 > 0 \rightarrow \Delta_3 \leq \Delta_1, \quad \Delta_3 \leq \Delta_2$

$z_1 + z_2 + 2z_3 + 3 \leq 6z_1 + z_3 + 2$

$-4z_1 + z_3 \leq -1$

$$\begin{cases} 4z_1 - z_3 = 1 & \longrightarrow 4z_1 - 1 + 2z_1 = 1 \\ z_1 = z_2 & \uparrow \\ z_1 + z_2 + z_3 = 1 & \longrightarrow z_3 = 1 - 2z_1 \end{cases}$$

$6z_1 = 2 \qquad z_1 = \frac{1}{3} = z_2 \qquad z = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$

$z_3 = 1 - 2 \cdot \frac{1}{3} = \frac{1}{3}$

$W.E. = 6 \cdot z_1 + z_3 + 2 = 6 \cdot \frac{1}{3} + \frac{1}{3} + 2 = \frac{13}{3}$

$UO_1 = z_1 + z_3 = \frac{2}{3} \qquad UO_4 = z_1 = \frac{1}{3}$

$UO_2 = z_2 = \frac{1}{3} \qquad UO_5 = z_2 + z_3 = \frac{2}{3}$

$UO_3 = z_3 = \frac{1}{3}$

user optimum flow vector $UO = \left[\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}\right]$

d) social optimum

$$\min \ (z_1+z_3)(z_1+z_3+1) + z_2(5z_2+1) + z_3 + z_1(5z_1+1) + (z_2+z_3)(z_2+z_3+1)$$

$$\text{st} \quad z_1+z_2+z_3=1$$

$$\min \ z_1^2+2z_1z_3+z_1+z_3^2+z_3+5z_2^2+z_2 + z_3 + 5z_1^2+z_1 + z_2^2+2z_2z_3+z_3^2+z_2+z_3$$

$$\text{st.} \quad z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+2z_3^2+2z_1+2z_2+3z_3 + 2z_1z_3+2z_2z_3+1 \quad \text{st.} \ z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+2z_3^2 + 3(z_1+z_2+z_3) - z_1 - z_2 + 2z_1z_3 + 2z_2z_3$$

$$\text{st.} \quad z_1+z_2+z_3=1 \quad \downarrow \ z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+2z_3^2+3-z_1-z_2 + 2z_1z_3+2z_2z_3 \quad \text{st} \ z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+3-z_1-z_2-z_3 + z_3 + 2z_3(z_1+z_2+z_3) \quad \text{st} \ z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+3-1+3z_3 \quad \text{st} \ z_1+z_2+z_3=1$$

$$\min \ 6z_1^2+6z_2^2+2+3-3z_1-3z_2$$

$$\frac{\partial f(z_1,z_2)}{\partial z_1} = 12z_1-3 = 0 \qquad z_1=\frac{1}{4}$$

$$\frac{\partial f(z_1,z_2)}{\partial z_2} = 12z_2-3=0 \qquad z_2=\frac{1}{4}$$

$$z_3=\frac{1}{2}$$

$$z=\left[\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right]$$

total cost:

$$\left(\frac{1}{4}+\frac{1}{2}\right)\left(\frac{1}{4}+\frac{1}{2}+1\right) + \frac{1}{4}\left(\frac{5}{4}+1\right) + \frac{1}{2} + \frac{1}{4}\left(\frac{5}{4}+1\right) + \left(\frac{1}{4}+\frac{1}{2}\right)\left(\frac{1}{4}+\frac{1}{2}+1\right) = \frac{17}{4} < \frac{13}{3}$$

$SO_1 = z_1 + z_3 = \frac{1}{4} + \frac{1}{2}$ $\quad SO_4 = z_1 = \frac{1}{4}$

$SO_2 = z_2 = \frac{1}{4}$ $\quad SO_5 = z_2 + z_3$

$SO_3 = z_3 = \frac{1}{2}$

Social optimum flow vector $SO = \left[\frac{3}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}\right]$

e)

$$\text{price of anarchy} = \frac{13/3}{17/4} = \frac{52}{51}$$

f) tolls that reduce the price of anarchy to 1

We replace the delay functions $d_e(f_e)$ are replaced with new delay functions $w_e + d_e(f_e)$, where w_e is the vector of tolls. Doing so we want to set the Wardrop Equilibrium $f^{(w)}$ equal to social optimum f^*

$$w_e = c'_e(f_e^*) - d_e(f_e^*)$$

$$c_e(f_e^*) = f\, d_e(f_e^*)$$

$$w_e = d_e(f_e^*) + f_e^* d'_e(f_e) - d_e(f_e^*) =$$
$$= f_e^* d_e'(f_e^*)$$

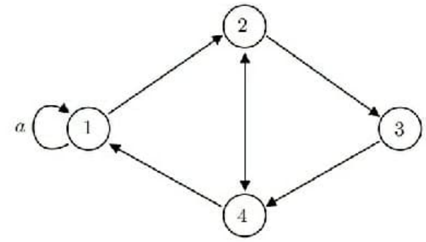
$w_{e_1} = z_1 + z_3 = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$ $\quad w_{e_4} = 5 \cdot \frac{1}{4}$

$w_{e_2} = 5 \cdot z_2 = 5 \cdot \frac{1}{4} = \frac{5}{4}$ $\quad w_{e_5} = z_2 + z_3 =$

$w_{e_3} = 0$ $\quad = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$

$$w_e = \left(\frac{3}{4}, \frac{5}{4}, 0, \frac{5}{4}, \frac{3}{4}\right)$$

$\boxed{\text{Exercise 2}}$



1) determine W, P, L

$$W = \begin{pmatrix} a & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad D = diag([a+1, 2, 1, 2])$$

$$P = D^{-1}W = \begin{pmatrix} \frac{a}{a+1} & \frac{1}{a+1} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

$$L = D - W = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & 0 & 2 \end{pmatrix}$$

b-c) French - DeGroot $x(t+1) = Px(t)$
for which value of $a \geq 0$ $x(t)$ converge to some limit as $t \to +\infty$ for every initial condition $x(0)$

the graph G is strongly connected and aperiodic, so the convergence $x(t+1) = Px(t)$ to $\mathbb{1}\pi' x(0)$ as $t \to +\infty$ is guaranteed $\forall\ a \geq 0$

d) $a=0$ $\quad x_1(0) = -1 \quad x_3(0) = -1$

$x_2(0) = 1 \quad x_4(0) = 1$

determine the limit option profile
$\lim_{t\to+\infty} x(t)$

if $a=0$, $\mathcal{G}$ is still strongly connected and aperiodic, so we can apply the following theorem:

$$\lim_{t\to\infty} x(t) = \mathbb{1}\pi' x(0)$$

$$w = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} \qquad \pi = \frac{w}{\mathbb{1}'w} = \begin{bmatrix} 1/6 \\ 1/3 \\ 1/6 \\ 1/3 \end{bmatrix}$$

$$\lim_{t\to+\infty} x(t) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & \frac{1}{6} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

e) min a s.t. $a \geq 0$, $\lim_{t\to+\infty} x_1(t) \leq 0$

$$\lim_{t\to+\infty} x_1(t) = \pi' x(0)$$

$$\pi = \begin{bmatrix} \frac{a+1}{a+6} \\ \frac{2}{a+6} \\ \frac{1}{a+6} \\ \frac{2}{a+6} \end{bmatrix}$$

$$\lim_{t\to+\infty} x_1(t) = \begin{bmatrix} \frac{a+1}{a+6} & \frac{2}{a+6} & \frac{1}{a+6} & \frac{2}{a+6} \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} =$$

$$= \frac{-a-1+2-1+2}{a+6} = \frac{-a+2}{a+6}$$

$$\frac{-a+2}{a+6} \leq 0 \qquad \begin{cases} -a+2 \leq 0 \\ a \leq -6 \quad \text{NO} \end{cases} \qquad a \geq 2 \downarrow a = 2$$

f) $x_i(0) = X_i \qquad E(X_i)=0$

$Var(X_i)=1$

determine the value of $a \geq 0$ that minimizes the variance of $\lim_{t\to\infty} x_i(t)$

$$\lim_{t\to\infty} x_i(t) = \pi' x(0) = \sum \pi_i X_i$$

$$Var(\sum \pi_i X_i) = 1 \cdot \sum \pi_i^2 =$$

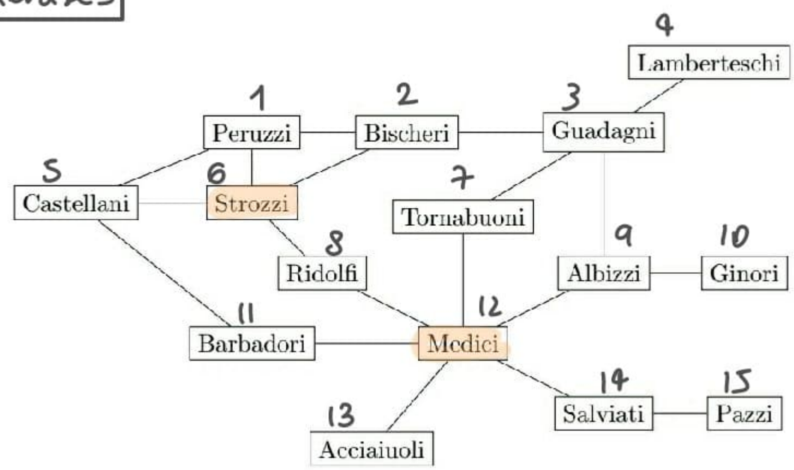
$$= \frac{a^2+2a+1+4+1+4}{(a+6)^2} = \frac{a^2+2a+10}{(a+6)^2}$$

$$f'(a) = \frac{2(5a-4)}{(a+6)^2}$$

$$f'(a) = 0 \qquad a = \frac{4}{5}$$

$$a = -6 \quad \text{NO}$$

Exercise 3



a) initial states:

$x_{Medici}(0) = 1$ $\quad$ $x_{Strozzi}(0) = -1$

0 for all others nodes

the graph $\mathcal{G}$ is strongly connected and aperiodic, so the convergence of $x(t+1) = Px(t)$ to $\mathbb{1}\pi' x(0)$ as $t \to +\infty$ is guaranteed $\forall\, a \geq 0$

$$w = [\,3 \;\; 3 \;\; 4 \;\; 1 \;\; 3 \;\; 4 \;\; 2 \;\; 2 \;\; 3 \;\; 1 \;\; 2 \;\; 6 \;\; 1 \;\; 2 \;\; 1\,]^T$$

$$\mathbb{1}' w = 38$$

$$\pi = \frac{1}{38}[\,3 \;\; 3 \;\; 4 \;\; 1 \;\; 3 \;\; 4 \;\; 2 \;\; 2 \;\; 3 \;\; 1 \;\; 2 \;\; 6 \;\; 1 \;\; 2 \;\; 1\,]^T$$

$$x(0) = [\,0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; -1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0\,]^T$$

$$\lim_{t \to \infty} x(t) = \mathbb{1}\pi' x(0) = \frac{1}{19}$$

c) equilibrium vector

$$w = \begin{bmatrix} 3 & 3 & 4 & 1 & 3 & 4 & 2 & 2 & 3 & 1 & 2 & 6 & 1 & 2 & 1 \end{bmatrix}^T$$

$$P = \begin{array}{c c}
\begin{array}{ccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} & \\
\begin{bmatrix}
0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & \frac{1}{6} & 0 & \frac{1}{6} & \frac{1}{6} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array}
\end{array}$$

$S = \{3, 5, 6, 12\}$ $\qquad R = V \setminus S$

$$P = \begin{array}{cc} \begin{array}{cc} R & S \end{array} & \\ \begin{bmatrix} Q & E \\ F & G \end{bmatrix} & \begin{array}{c} R \\ S \end{array} \end{array}$$

shape $(Q) = 11 \times 11$
shape $(E) = 4 \times 11$

$$
Q=\begin{bmatrix}
0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\quad
E=\begin{bmatrix}
0 & \frac{1}{3} & \frac{1}{3} & 0 \\
\frac{1}{3} & 0 & \frac{1}{3} & 0 \\
0 & 1 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} \\
0 & 0 & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{2} \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0
\end{bmatrix}
$$

$$u = [-1, -1, -1, 1]^T$$

$$\bar{x} = (I-Q)^{-1} E u =$$

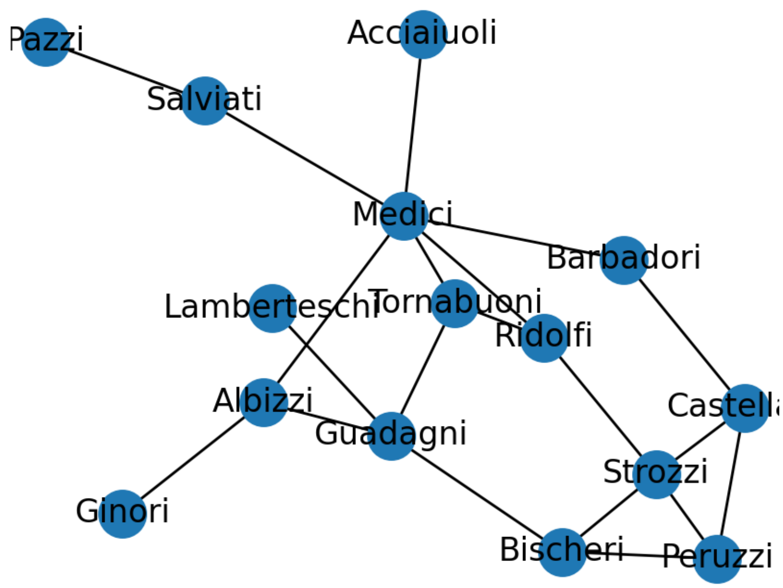
$$= [-1 \ -1 \ -1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1]$$

Peruzzi, Bischeri, Lamberteschi	-1
Acciaiuoli, Salviati, Pazzi	1
Others	0

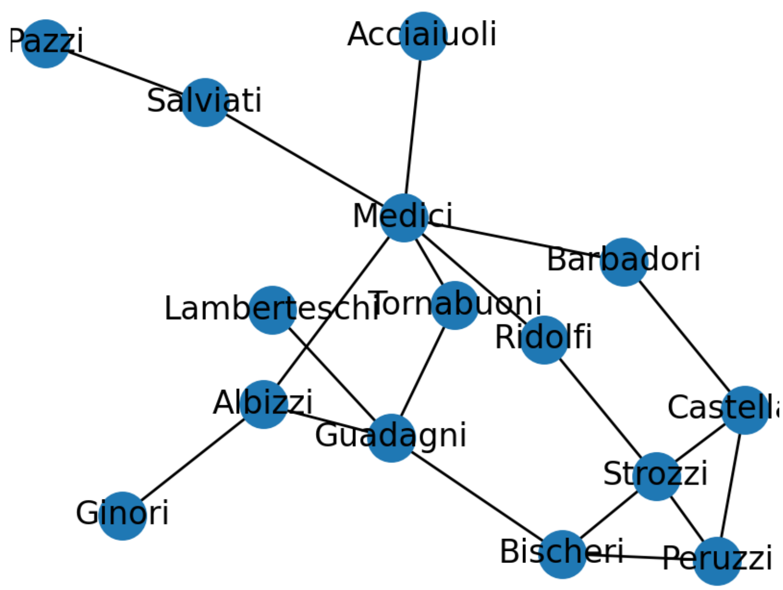
Exercise 3

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
```

```
G = nx.generators.social.florentine_families_graph()
pos = nx.spring_layout(G)
plt.figure(figsize=(4,3), dpi = 200)
nx.draw(G, pos, with_labels=True)
```




```
# the graph reported in our exercise do not have an edge between Ridolfi and␣
↪Tornabuoni, let's remove it
G.remove_edge('Ridolfi', 'Tornabuoni')
plt.figure(figsize=(4,3), dpi = 200)
nx.draw(G, pos, with_labels=True)
```



b. Write down a Python code to simulate the averaging dynamics with stubborn nodeset S={Medici,Strozzi} and opinions u_{Medici}= 1 and $u_{Strozzi}$=1. Plot the trajectories of the different states and deduce the equilibrium state vector

```
n = len(G)
indices = dict()
for i in range(n):
    indices[list(G.nodes)[i]] = i
print("Dictionary (name, index): ", indices)

iters = 50
stubborn_nodes = ['Medici', 'Strozzi']
stubborn_id = [indices.get(key) for key in stubborn_nodes]
regular = [node for node in G.nodes if node not in stubborn_nodes]
regular_id = [id for id in range(n) if id not in stubborn_id]

#initial opinions
u = [1, -1]



W = nx.adjacency_matrix(G)
W = W.toarray()
w = np.sum(W,axis=1)
D = np.diag(w)
P = np.linalg.inv(D) @ W

Q = P[np.ix_(regular_id, regular_id)]
```

```
E = P[np.ix_(regular_id, stubborn_id)]

ic = np.random.uniform(0,1,len(regular))

x = np.zeros((n,iters))
x[stubborn_id,0] = u;
x[regular_id,0] = ic;
print("Initial condition:", x[:,0])

for t in range(1,iters):
    x[regular_id, t] = Q @ x[regular_id, t-1] + E @ x[stubborn_id, t-1]
    x[stubborn_id, t] = x[stubborn_id, t-1];

x_final = x[:,iters-1]
```

```
Dictionary (name, index):  {'Acciaiuoli': 0, 'Medici': 1, 'Castellani': 2,
'Peruzzi': 3, 'Strozzi': 4, 'Barbadori': 5, 'Ridolfi': 6, 'Tornabuoni': 7,
'Albizzi': 8, 'Salviati': 9, 'Pazzi': 10, 'Bischeri': 11, 'Guadagni': 12,
'Ginori': 13, 'Lamberteschi': 14}
Initial condition: [ 0.87223244  1.          0.49179762  0.59653522 -1.
0.71759565
  0.02066915  0.81551509  0.49797775  0.87397153  0.10735041  0.18263217
  0.03880702  0.49492624  0.19995445]
```

```
results = pd.DataFrame(list(zip(G.nodes, x_final )), columns =['Node',
 ↪'Equilibrium vector'])
results
```

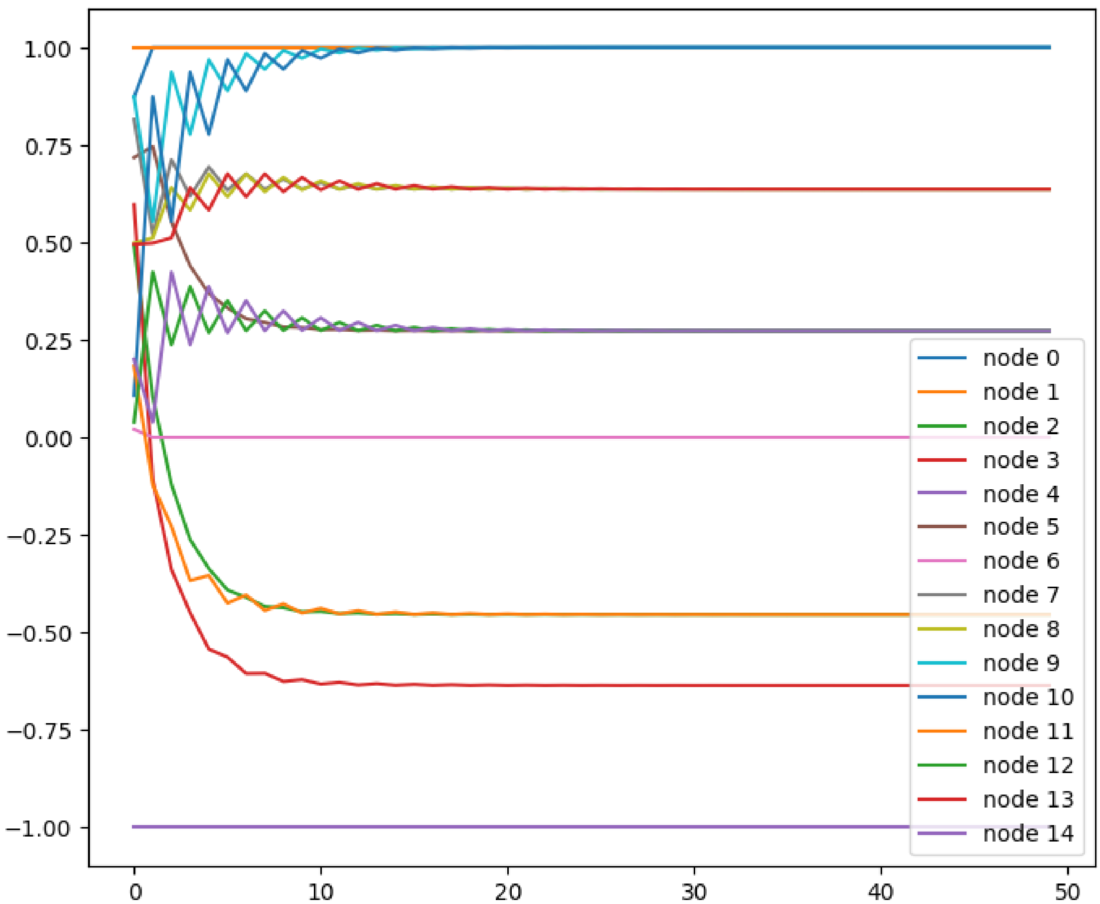
```
            Node  Equilibrium vector
0     Acciaiuoli            1.000000
1         Medici            1.000000
2     Castellani           -0.454545
3        Peruzzi           -0.636362
4        Strozzi           -1.000000
5      Barbadori            0.272728
6        Ridolfi            0.000000
7     Tornabuoni            0.636364
8        Albizzi            0.636364
9       Salviati            1.000000
10         Pazzi            1.000000
11      Bischeri           -0.454545
12      Guadagni            0.272734
13        Ginori            0.636371
14  Lamberteschi            0.272728
```

```python
fig = plt.figure(figsize=(8,7), dpi=100)
ax = plt.subplot(111)

for node in range(n):
    trajectory = x[node,:]
    ax.plot(trajectory, label='node {0:d}'.format(node))

ax.legend()
```

```
States trajectories
```

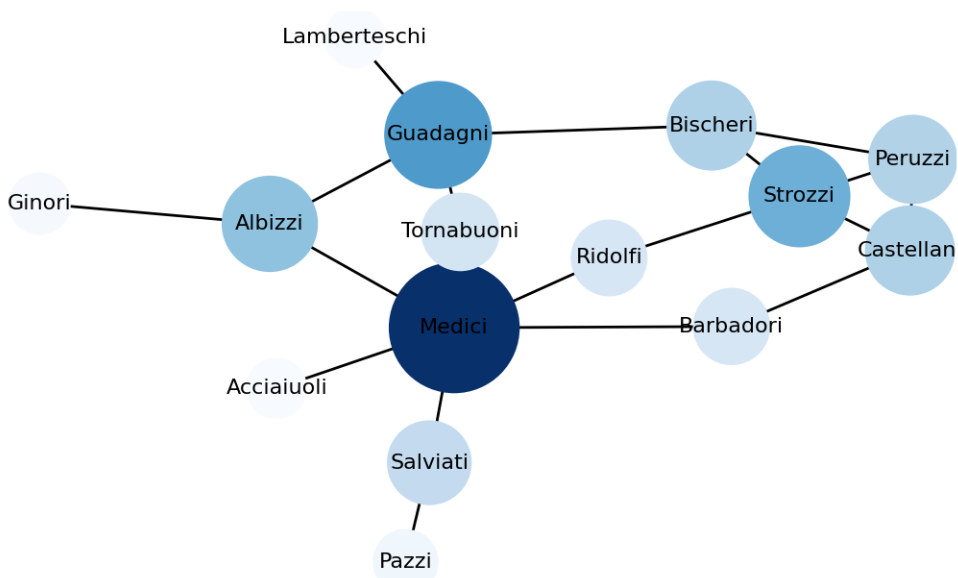


d. Write down a Python code for the iterative distributed computation of the PageRankcentrality in the network with = 0.15 and uniform input.

```python
def iterative_page_rank(tol, max_iter):
  beta = 0.15
  mu = np.ones(n)
```

```
    # Bonanich centrality
    # the pagerank centrality is a special case of the Bonacich centrailty with␣
↪beta = 0.15 and mu = 1
    x_0 = np.ones(n)/n
    x_old = x_0
    for i in range(max_iter):
        x_new = (1 - beta)* P.T @ x_old + beta*mu
        if np.linalg.norm(x_new-x_old) < tol:
            break
        x_old=x_new
    return x_new
```

```
tol = 1e-6
max_iter = 100
x = iterative_page_rank(tol, max_iter)
x_res = x/np.sum(x)
# alpha = 1-beta
pr = nx.algorithms.link_analysis.pagerank_alg.pagerank(G, alpha=0.85, tol=tol)
plt.figure(figsize=(5,3), dpi = 200)
nx.draw(G,pos,
         with_labels=True,
         nodelist=G.nodes(),
         node_size = [d*1000 for d in x],
         node_color=x,
         font_size=8,
         cmap=plt.cm.Blues)
```



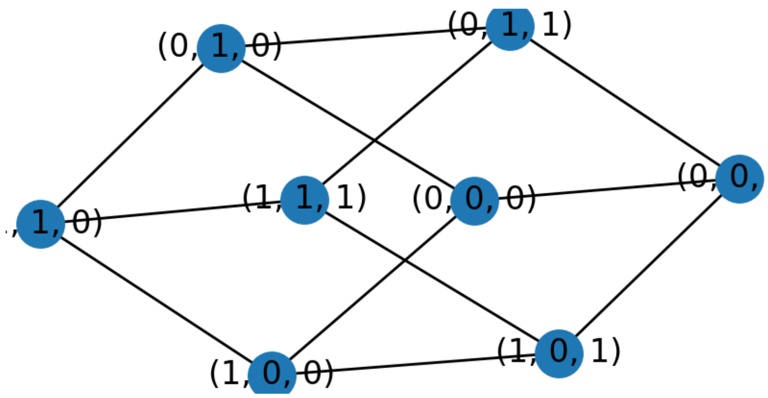
```
results = pd.DataFrame(list(zip(G.nodes, x_res, list(pr.values()) )), columns␣
↪=['Node', 'Iterative Page Rank', ' NX Page Rank'])
results
```

	Node	Iterative Page Rank	NX Page Rank
0	Acciaiuoli	0.031824	0.031824
1	Medici	0.154051	0.154050
2	Castellani	0.071658	0.071658
3	Peruzzi	0.070187	0.070187
4	Strozzi	0.092318	0.092318
5	Barbadori	0.052127	0.052127
6	Ridolfi	0.051442	0.051442
7	Tornabuoni	0.053848	0.053848
8	Albizzi	0.082126	0.082127
9	Salviati	0.063129	0.063130
10	Pazzi	0.036830	0.036830
11	Bischeri	0.071527	0.071528
12	Guadagni	0.103640	0.103640
13	Ginori	0.033269	0.033269
14	Lamberteschi	0.032024	0.032024

Exercise 4

Consider the two simple graphs below where the red node is to be interpreted as a stubborn node 0 with opinion $x_0 = 0$.
Find the positions for a second stubborn node with opinion x_s= 1 in such a way that, given x the asymptotic opinion profile relative to the averaging dynamics model, the quantity
$H(s) = \frac{1}{n} \sum_{i \in V} x_i$ is maximed.

```
G = nx.hypercube_graph(3)
pos = nx.spring_layout(G)
plt.figure(figsize=(4,2), dpi = 200)
nx.draw(G, pos, with_labels=True)
```




```
n = len(G)
max = 0
indices = dict()
for i in range(n):
    indices[list(G.nodes)[i]] = i
print("Dictionary (name, index): ", indices)

final_opinions = dict()
average_opinion = dict()

iters = 50
for (i,j,z) in G.nodes:
    if (i,j,z)==(0,0,0):
        continue

    # Stubborn and regular nodes
    stubborn = [(0,0,0), (i,j,z)];
    stubborn_id = [indices.get(key) for key in stubborn]
    regular = [node for node in G.nodes if node not in stubborn]
```

```
    regular_id = [id for id in range(n) if id not in stubborn_id]
    print("Stubborn nodes:", stubborn)

    # Input to stubborn nodes
    u = [0,1]

    W = nx.adjacency_matrix(G)
    W = W.toarray()
    w = np.sum(W,axis=1)
    D = np.diag(w)
    P = np.linalg.inv(D) @ W

    Q = P[np.ix_(regular_id, regular_id)]
    E = P[np.ix_(regular_id, stubborn_id)]

    ic = np.random.uniform(0,1,len(regular))

    x = np.zeros((n,iters))
    x[stubborn_id,0] = u;
    x[regular_id,0] = ic;

    for t in range(1,iters):
        x[regular_id, t] = Q @ x[regular_id, t-1] + E @ x[stubborn_id, t-1] #
        x[stubborn_id, t] = x[stubborn_id, t-1];

    final_opinions[(i,j,z)] = x[:,iters-1]
    average_opinion[(i,j,z)] = np.average(final_opinions[(i,j,z)])
    if average_opinion[(i,j,z)] > max:
      max = average_opinion[(i,j,z)]
      max_node = (i,j,z)
```

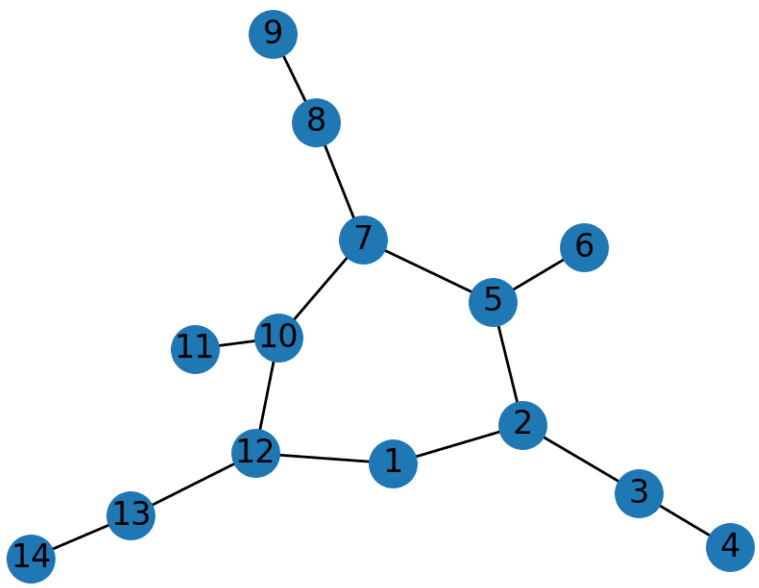
```
Dictionary (name, index):  {(0, 0, 0): 0, (0, 0, 1): 1, (0, 1, 0): 2, (0, 1, 1):
3, (1, 0, 0): 4, (1, 0, 1): 5, (1, 1, 0): 6, (1, 1, 1): 7}
Stubborn nodes: [(0, 0, 0), (0, 0, 1)]
```

```
results = pd.DataFrame(list(zip(average_opinion.keys(),  list(average_opinion.
 ↪values()) )), columns =['Stubborn Node', 'H(s)'])
results
```

Stubborn Node	H(s)
(0, 0, 1)	0.500002
(0, 1, 0)	0.500001
(0, 1, 1)	0.500000
(1, 0, 0)	0.500000
(1, 0, 1)	0.500000
(1, 1, 0)	0.500000
(1, 1, 1)	0.500000

After the simulation, we can see that the second stubborn node which maximises the quantity H(s) could be any node in the graph.

```
G = nx.Graph()
G.add_nodes_from(range(1,14))
G.add_edges_from([(1,2), (2,3), (3,4), (2,5), (5,6), (5,7), (7,8),
                  (8,9), (7,10), (10,11), (10,12), (12,13), (13,14), (12,1)])
pos = nx.spring_layout(G)
plt.figure(figsize=(4,3), dpi = 200)
nx.draw(G,pos, with_labels=True)
```




```
n = len(G)
max = 0
indices = dict()
for i in range(n):
    indices[list(G.nodes)[i]] = i
print("Dictionary (name, index): ", indices)

final_opinions = dict()
average_opinion = dict()

iters = 50
for i in G.nodes:
    if i == 1:
        continue

    # Stubborn and regular nodes
```

```python
    stubborn = [1, i];
    stubborn_id = [indices.get(key) for key in stubborn]
    regular = [node for node in G.nodes if node not in stubborn]
    regular_id = [id for id in range(n) if id not in stubborn_id]
    print("Stubborn nodes:", stubborn)

    # Input to stubborn nodes
    u = [0,1]

    W = nx.adjacency_matrix(G)
    W = W.toarray()
    w = np.sum(W,axis=1)
    D = np.diag(w)
    P = np.linalg.inv(D) @ W

    Q = P[np.ix_(regular_id, regular_id)]
    E = P[np.ix_(regular_id, stubborn_id)]

    ic = np.random.uniform(0,1,len(regular))

    x = np.zeros((n,iters))
    x[stubborn_id,0] = u
    x[regular_id,0] = ic

    for t in range(1,iters):
        x[regular_id, t] = Q @ x[regular_id, t-1] + E @ x[stubborn_id, t-1] #
        x[stubborn_id, t] = x[stubborn_id, t-1];

    final_opinions[i] = x[:,iters-1]
    average_opinion[i] = np.average(final_opinions[i])
    if average_opinion[i] > max:
      max = average_opinion[i]
      max_node = i

print("Best node is {} with H(s) equals to {} ".format(max_node, max))
```

```
Dictionary (name, index):  {1: 0, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, 8: 7, 9:
8, 10: 9, 11: 10, 12: 11, 13: 12, 14: 13}
Stubborn nodes: [1, 2]
Best node is 12 with H(s) equals to 0.556031944076116
```

```
results = pd.DataFrame(list(zip(average_opinion.keys(),  list(average_opinion.
 ↪values()) )), columns =['Stubborn Node', 'H(s)'])
results
```

```
    Stubborn Node      H(s)
              2  0.555539
              3  0.331714
              4  0.261775
              5  0.535765
              6  0.341601
              7  0.547629
              8  0.386400
              9  0.298307
             10  0.535193
             11  0.338967
             12  0.556032
             13  0.340451
             14  0.250603
```

In this case, the second stubborn can be choosen among two nodes: node 2 and node 12.