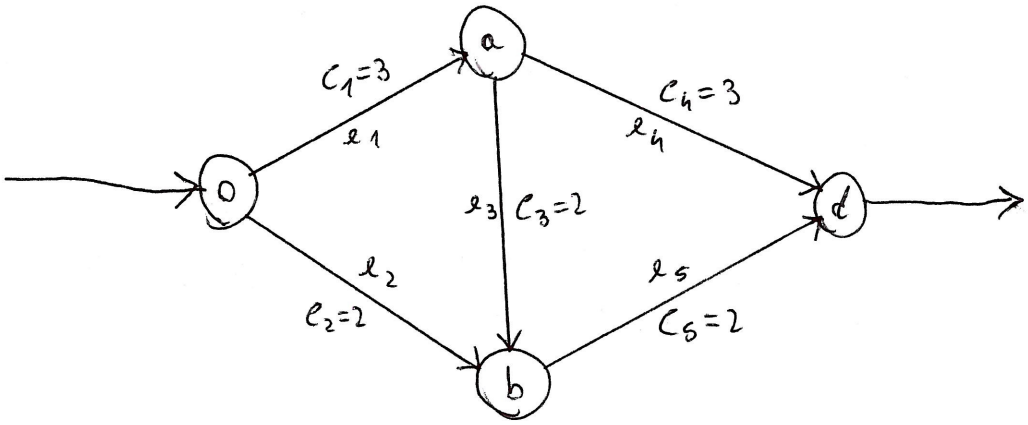
# NETWORKS DYNAMICS AND LEARNING

HOMEWORK 1

Student: s278727

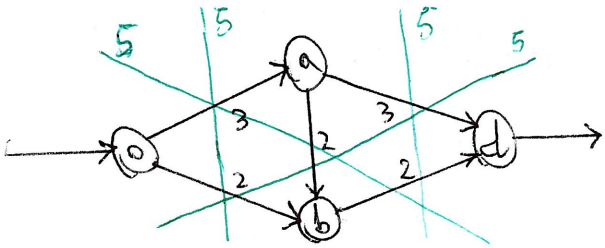
## PROBLEM 1




Let us consider unitary o-d network flows on the graph $G=(V,E)$ in figure.

(a) What is the infimum of the total capacity that needs to be removed for no feasible unitary flows from o to d to exist?

First, I compute the min-cut capacity:




All the o-d cuts have capacity 5, so the min-cut capacity is $c^*_{o,d}=5$. This is, by theorem, equal to the max-throughput $\tau^*_{o,d}$.

In order for no feasible o-d unitary flows to exist, we must impose a max-throughput $\tau_{MAX}<1$, i.e. only o-d flows associated to a throughput $<1$ may be feasible.

This can be done by removing a total capacity $c_R$ from the links of one of the o-d cuts, so that the new min-cut capacity becomes $c^*_{o,d}-c_R$. Mathematically:

$$\tau_{MAX}<1 \overset{\text{MAX FLOW MIN CUT TH.}}{\Longleftrightarrow} c^*_{o,d}-c_R<1 \iff c_R>c^*_{o,d}-1=4$$

Therefore, $c_R \in (4, +\infty)$, and the infimum of this interval is $\inf(4, +\infty) = 4$.

(b) Where should 2 units of additional capacity be allocated in order to maximize the feasible throughput from o to d?

To maximize $\tau^*_{o,d}$, I can add two units of capacity in the following way:

$$\left.\begin{array}{l} c_5 \leftarrow c_5 + 1 = 3 \Longrightarrow c_{\{o,b\}} = c_{\{o,a,b\}} = b \\ c_2 \leftarrow c_2 + 1 = 3 \Longrightarrow c_{\{o\}} = c_{\{o,a\}} = b \end{array}\right] \Longrightarrow c^*_{o,d} = b \overset{TH}{=} \tau^*_{o,d}$$

$\tau^*_{o,d}$ has increased from 5 to 6, the maximum possible increase given two additional units of capacity.
The same increase can be achieved by adding two additional units of capacity to nodes 1 and 4.

Now, assume the link capacities are all infinite and let the delay functions on the links be given by

$$\begin{cases} d_1(x) = d_5(x) = x + 1 \\ d_3(x) = 1 \\ d_2(x) = d_4(x) = 5x + 1 \end{cases}$$

c) Compute the user optimum (i.e., the Wardrop Equilibrium) flow vector.

Denote by $p^{(1)}$ the path $o \to a \to d$, by $p^{(2)}$ the path $o \to b \to d$, by $p^{(3)}$ the path $o \to a \to b \to d$.

Let $z = (z_1, z_2, z_3) \geq 0$ with $z_1 + z_2 + z_3 = 1$ the flows on path $p^{(1)}$, $p^{(2)}$, $p^{(3)}$ respectively. Notice that, even if not explicitly specified in the text of this point of the problem, I assume a throughput of 1, i.e. unitary o-d flows.

Let the total delay on path $p^{(1)}$, $p^{(2)}$, $p^{(3)}$ respectively:

$$\Delta_1 = z_1 + z_3 + 1 + 5z_1 + 1 = 6z_1 + z_3 + 2$$

$$\Delta_2 = 5z_2 + 1 + z_2 + z_3 + 1 = 6z_2 + z_3 + 2$$

$$\Delta_3 = z_1 + z_3 + 1 + 1 + z_2 + z_3 + 1 = z_1 + z_2 + z_3 + 3$$

By definition of Wardrop Equilibrium $f^{(0)}$, the flow configuration $z$ must respect these conditions:

$$z_1 > 0 \implies \Delta_1 \leq \Delta_2 \wedge \Delta_1 \leq \Delta_3$$

$$z_2 > 0 \implies \Delta_2 \leq \Delta_1 \wedge \Delta_2 \leq \Delta_3$$

$$z_3 > 0 \implies \Delta_3 \leq \Delta_1 \wedge \Delta_3 \leq \Delta_2$$

Let's assume $z > 0$, i.e. $z_1, z_2, z_3 > 0$. In that case, for $f^{(0)}$ to be the Wardrop Equilibrium flow vector:

$$\Delta_1 = \Delta_2 = \Delta_3$$

Therefore, in order to find $f^{(0)}$ we have to solve the following system:

$$\begin{cases} \Delta_1 = \Delta_2 \\ \Delta_1 = \Delta_3 \\ z_1 + z_2 + z_3 = 1 \\ z_1, z_2, z_3 > 0 \end{cases} \iff \begin{cases} 6z_1 + \cancel{z_3} + \cancel{2} = 6z_3 + \cancel{z_3} + \cancel{2} \\ 6z_1 + z_3 + 2 = z_1 + z_2 + 2z_3 + 3 \\ z_1 + z_2 + z_3 = 1 \\ z_1, z_2, z_3 > 0 \end{cases} \iff$$

$$\iff \begin{cases} z_1 = z_2 \\ 6z_1 + (1 - z_1 - z_2) + 2 = z_1 + z_2 + 2(1 - z_1 - z_2) + 3 \\ z_3 = 1 - z_1 - z_2 \\ z_1, z_2, z_3 > 0 \end{cases} \iff$$

$$\iff \begin{cases} z_1 = z_2 \\ 6z_1 = 2 \\ z_3 = 1 - 2z_1 \\ z_1, z_2, z_3 > 0 \end{cases} \iff z = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) \implies \boxed{f^{(0)} = \left(\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}\right)}$$

Moreover, since the delays are all non-decreasing functions, the Wardrop Equilibrium is unique, and it is the $f^{(0)}$ found.

(d) Compute the social optimum flow vector, i.e. the flow vector which minimizes the average delay from o to d.

The social optimum flow vector $f^*$ is defined as the solution of the following optimization problem:

$$\begin{aligned} &\min_{f \geq 0} && \sum_{e \in E} f_e \cdot d_e(f_e) := C \\ &Bf = \delta(o) - \delta(d) \end{aligned}$$

Let's denote the objective function by $C$.

$$C = f_1(f_1+1) + f_2(5f_2+1) + f_3 + f_4(5f_4+1) + f_5(f_5+1) =$$
$$= f_1^2 + f_1 + 5f_2^2 + f_2 + f_3 + 5f_4^2 + f_4 + f_5^2 + f_5$$

Since $f_1 = z_1 + z_3$, $f_2 = z_2$, $f_3 = z_3$, $f_4 = z_1$, $f_5 = z_2 + z_3$ and $z_1 + z_2 + z_3 = 1$, we can reduce the dimensionality of the problem to just two variables (e.g. $z_1$ and $z_2$)

$$C = (1-z_2)^2 + (1-z_2) + 5z_2^2 + z_2 + 1 - z_1 - z_2 + 5z_1^2 + z_1 +$$
$$+ (1-z_1)^2 + (1-z_1)$$

Requiring the gradient of $C$ to be zero gives:

$$\frac{\partial C}{\partial z_1} = 10z_1 + 2(1-z_1)\cdot(-1) - 1 = 12z_1 - 3 = 0 \iff z_1 = \frac{1}{4}$$

$$\frac{\partial C}{\partial z_2} = 2(1-z_2)\cdot(-1) - 1 + 10z_2 = 12z_2 - 3 = 0 \iff z_2 = \frac{1}{4}$$

$$z_3 = 1 - z_1 - z_2 = \frac{1}{2}$$

$\Longrightarrow z = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$ solves the optimization problem,

therefore $f^* = \left(\frac{3}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}\right)$ is the social optimum flow vector.

(e) Compute the Price of Anarchy

The Price of Anarchy $PoA(0)$ is:

$$PoA(0) = \frac{\sum_{e \in E} f_e^{(0)} \cdot d_e(f_e^{(0)})}{\sum_{e \in E} f_e^* \cdot d_e(f_e^*)} = \frac{\frac{2}{3}\left(\frac{2}{3}+1\right) + \frac{1}{3}\left(5\cdot\frac{1}{3}+1\right) + \frac{1}{3} + \frac{1}{3}\left(5\cdot\frac{1}{3}+1\right) + \frac{2}{3}\left(\frac{2}{3}+1\right)}{\frac{3}{4}\left(\frac{3}{4}+1\right) + \frac{1}{4}\left(5\cdot\frac{1}{4}+1\right) + \frac{1}{2} + \frac{1}{4}\left(5\cdot\frac{1}{4}+1\right) + \frac{3}{4}\left(\frac{3}{4}+1\right)}$$

$$= \frac{13}{3} \cdot \frac{4}{17} = \boxed{\frac{52}{51}}$$

(f) Find a vector of tolls on the links that reduce the price of anarchy to 1

The tolls to implement in order to reduce the PoA to 1 are the marginal cost tolls: $w^* = (w_1^*, w_2^*, w_3^*, w_4^*, w_5^*)$ so that

$$w_e^* = f_e^* \cdot d_e'(f_e^*) \quad \forall e$$

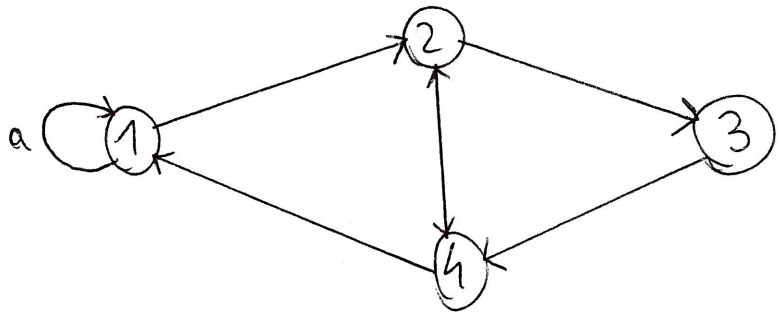
In our setting:

$w_1^* = \frac{3}{4} \cdot 1 = \frac{3}{4}$ / $w_2^* = \frac{1}{4} \cdot 5 = \frac{5}{4}$ / $w_3^* = \frac{1}{2} \cdot 0 = 0$ /

$w_4^* = \frac{1}{4} \cdot 5 = \frac{5}{4}$ / $w_5^* = \frac{3}{4} \cdot 1 = \frac{3}{4}$

$\Longrightarrow \boxed{w^* = \left(\frac{3}{4}, \frac{5}{4}, 0, \frac{5}{4}, \frac{3}{4}\right)}$

# PROBLEM 2




All link weights are unitary except for the selfloop at node 1, which has weight $a \geq 0$

Let $\mathcal{G}$ be the graph displayed in the figure above

(a) Determine its weight matrix $W$, normalized weight matrix $P$, Laplacian matrix $L$.

$$W = \begin{bmatrix} a & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \Longrightarrow P = \begin{bmatrix} \frac{a}{1+a} & \frac{1}{1+a} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 1+a & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \Longrightarrow L = D - W = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & 0 & 2 \end{bmatrix}$$

(b) Consider the French-DeGroot opinion dynamics on $\mathcal{G}$: $x(t+1) = P\,x(t)$.

(c) For which values of $a \geq 0$ does the opinion profile $x(t)$ converge to some limit as $t \to +\infty$ for every initial condition $x(0)$?

This question is equivalent to asking: for which values of $a \geq 0$ $\exists \lim_{t \to +\infty} x(t) = c$ for some $c \in \mathbb{R}^4$ $\forall x(0)$?

- For $a>0$, $\mathcal{G}$ is strictly connected and aperiodic, therefore

$$\lim_{t\to+\infty} x(t) \overset{TH}{=} \pi' x(0)\mathbb{1} \quad \forall x(0)$$

- For $a=0$, $\mathcal{G}$ loses the self-loop on node 1, but it is still strongly connected and aperiodic, so again the above formula holds.

(d) For $a=0$, determine, if it exists, the limit opinion profile $\lim_{t\to+\infty} x(t)$ when the initial opinions are $x_1(0)=-1$, $x_2(0)=1$, $x_3(0)=-1$, $x_4(0)=1$

$\mathcal{G}$ is balanced $\xrightarrow[\text{FROBENIUS TH}]{\text{PERRON}}$ $w = P'w \Rightarrow \pi = \frac{w}{\mathbb{1}'w} = \left(\frac{1}{6}, \frac{1}{3}, \frac{1}{6}, \frac{1}{3}\right)'$

($\mathcal{G}$ strongly connected $\xrightarrow[\text{FROBENIUS}]{\text{PERRON}}$ $\lambda_P = 1$ simple eigenvalue of $P$ $\Rightarrow$
$\Rightarrow$ the invariant distribution $\pi$ is unique)

$$\lim_{t\to+\infty} x(t) = \pi' x(0)\mathbb{1} = \left(\frac{1}{6}, \frac{1}{3}, \frac{1}{6}, \frac{1}{3}\right)\cdot(-1,1,-1,1)'\cdot(1,1,1,1)' =$$

$$= \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)'$$

(e) Determine, if it exists, the minimum value of $a \geq 0$ such that $\lim_{t\to+\infty} x_1(t) \leq 0$ with the same initial conditions as in (d)

- $a=0 \Rightarrow \lim_{t\to+\infty} x_1(t) = \pi' \cdot x(0) = \frac{1}{3} > 0$

- So we consider $a>0$

$\mathcal{G}$ is still balanced, so again it's easy to find $\pi$:

$$\pi = \frac{w}{\mathbb{1}'w} = \left(\frac{a+1}{a+6}, \frac{2}{a+6}, \frac{1}{a+6}, \frac{2}{a+6}\right)$$

$$\lim_{t\to+\infty} x_1(t) = \pi' x(0) = \left(\frac{-(a+1)+2-1+2}{a+6}\right) = \frac{-a+2}{a+6} \leq 0 \iff$$

$$\iff -a+2 \leq 0 \iff a \geq 2$$

Therefore, $\underline{a=2}$ is the minimum value of $a \geq 0$ such that $\lim_{t\to+\infty} x_1(t) \leq 0$ (for initial conditions as in (d)).

(g) For initial condition such that $X_i(0)$ are i.i.d $\forall i = 1,2,3,4$ random variables, with $E[X_i(0)] = 0$ and $V[X_i(0)] = 1$, determine, if it exists, the value of $a \geq 0$ that minimizes the variance of $\lim_{t \to +\infty} X_1(t)$

Let's denote the requested value of $a$ as $a^*$:

$$a^* = \underset{a \geq 0}{\operatorname{argmin}} \; V\left[\lim_{t \to +\infty} X_1(t)\right]$$

$$V\left[\lim_{t \to +\infty} X_1(t)\right] \overset{TH}{=} V\left[\pi' X(0)\right] = V\left[\frac{a+1}{a+b} X_1(0) + \frac{2}{a+b} X_2(0) + \frac{1}{a+b} X_3(0) + \frac{2}{a+b} X_4(0)\right] =$$

$$= \left(\frac{a+1}{a+b}\right)^2 + \frac{4}{(a+b)^2} + \frac{1}{(a+b)^2} + \frac{4}{(a+b)^2} = \frac{a^2+2a+10}{(a+b)^2}$$

Let's find the point of minimum $a^*$ of this quantity

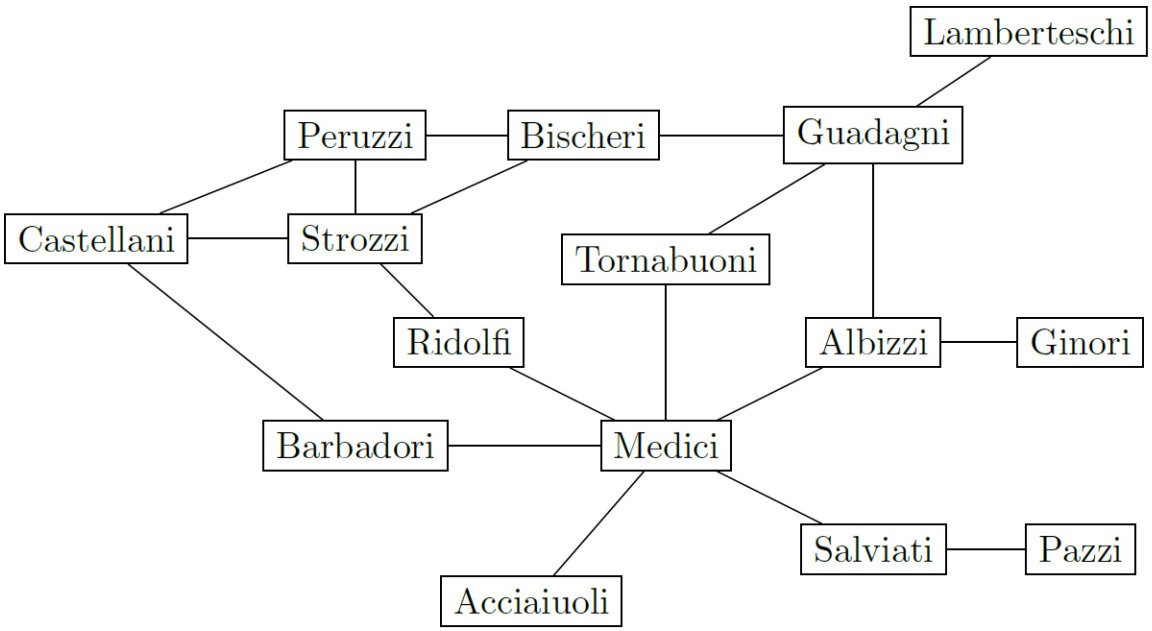
$$\frac{\partial}{\partial a} V\left[\lim_{t \to +\infty} X_1(t)\right] = \frac{\partial}{\partial a} \frac{a^2+2a+10}{(a+b)^2} = \frac{(2a+2)(a+b) - (a^2+2a+10) \cdot 2 \cdot (a+b)}{(a+b)^4} = 0 \iff$$

$$\iff 2a^2 + 12a + 2a + 12 - 2a^2 - 4a - 20 = 0 \iff 10a = 8 \iff a = \frac{4}{5}$$

(To verify that $a = \frac{4}{5}$ is indeed a point of minimum (and not of maximum) it suffices to verify that the sign of the the derivative is $\leq 0$ when $a \leq \frac{4}{5}$ and $\geq 0$ when $a \geq \frac{4}{5}$)

Therefore $\underline{a^* = \frac{4}{5}}$

# PROBLEM 3




Consider the network of figure above, illustrating the marriage-relationships between some of the most influential families in 14-th century Florence

(a) Compute analitically the limit of the distributed averaging dynamics on the graph when the initial states are equal to +1 for the Medici, $x_{MEDICI}(0) = 1$, -1 for the Strozzi, $x_{STROZZI}(0) = -1$, and 0 for all the other nodes

Let's consider the distributed averaging dynamics $x(t+1) = P\,x(t)$
G is strongly connected and aperiodic, so $\lim_{t\to+\infty} x_i(t)$ is guaranteed to converge to $\pi' x(0)$, for every node.
Moreover, the fact that G is strongly connected and balanced implies (by Perron-Frobenius theorem) that the invariant distribution $\pi$ is unique and $\pi = \frac{w}{\mathbf{1}'w}$, where w is the out-degrees vector.

Let's compute $\pi' x(0)$:

$$\pi^1 X(0) = \sum_k \pi_k x_k(0) = \pi_{MEDICI} \cdot (1) + \pi_{STROZZI} \cdot (-1)$$

$$\left.\begin{aligned} \pi_{MEDICI} &= \frac{w_{MEDICI}}{|\mathcal{E}|} = \frac{6}{38} = \frac{3}{19} \\ \pi_{STROZZI} &= \frac{w_{STROZZI}}{|\mathcal{E}|} = \frac{4}{38} = \frac{2}{19} \end{aligned}\right] \Rightarrow \pi^1 X(0) = \frac{3}{19} - \frac{2}{19} = \frac{1}{19}$$

(N.B: $\mathcal{G}$ unweighted $\Rightarrow \pi^1 w = \sum_i w_i = |\mathcal{E}|$, where $|\mathcal{E}|$ is the number of directed links of $\mathcal{G}$)

$$\Rightarrow \lim_{t \to +\infty} x_i(t) = \frac{1}{19} \quad \forall i$$

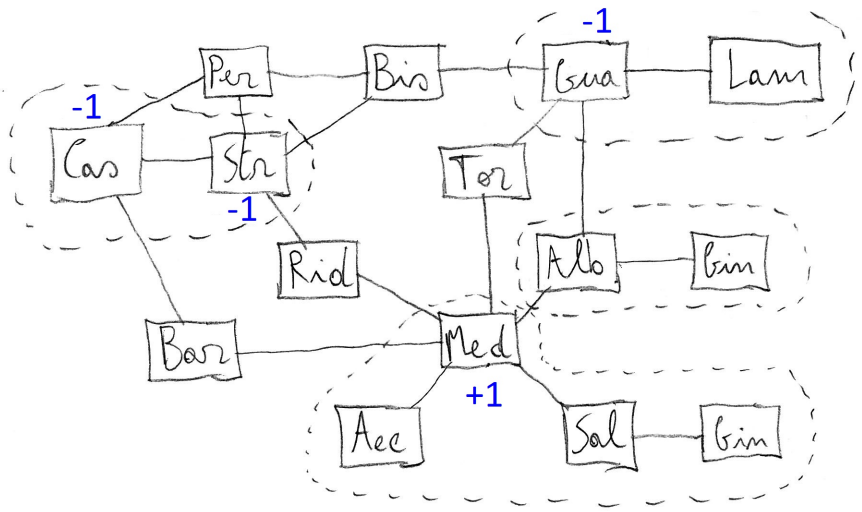
(b) Write down a Python code to simulate the averaging dynamics with stubborn node set $S = \{$Medici, Strozzi$\}$ and opinions $u_{MEDICI} = 1$ and $u_{STROZZI} = -1$; plot the trajectories of the different states; deduce the equilibrium state vector

SEE CODE AT THE END OF THIS PDF

(c) Compute analitically the equilibrium vector of the averaging dynamics with stubborn node set $S = \{$Medici, Strozzi, Castellani, Guadagni$\}$ and opinions $u_{MEDICI} = 1$, $u_{STROZZI} = u_{CASTELLANI} = u_{GUADAGNI} = -1$.

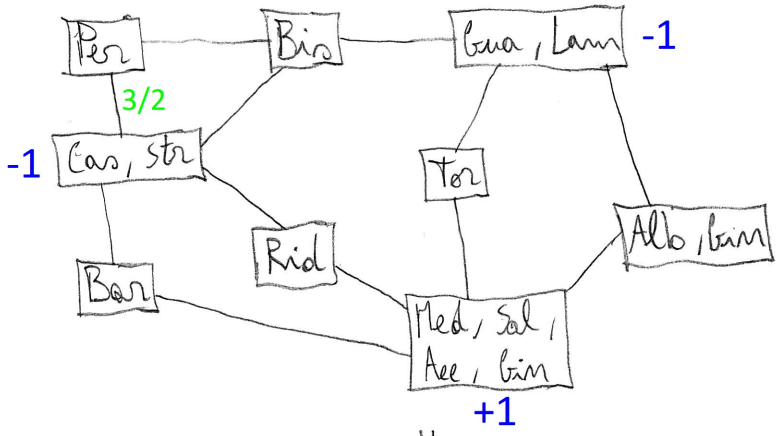
$\mathcal{G}$ is undirected, so the asymptotic opinion of each node can be computed using the rules commonly used for solving electrical circuits (because the asymptotic opinion of each node can be interpreted as a voltage of the electrical network where the stubborn nodes correspond to nodes with assigned voltage).

Therefore, I reduce the graph using 'electrical tricks' as follows:
(N.B: I only write the first 3 letters for each family, for readability)

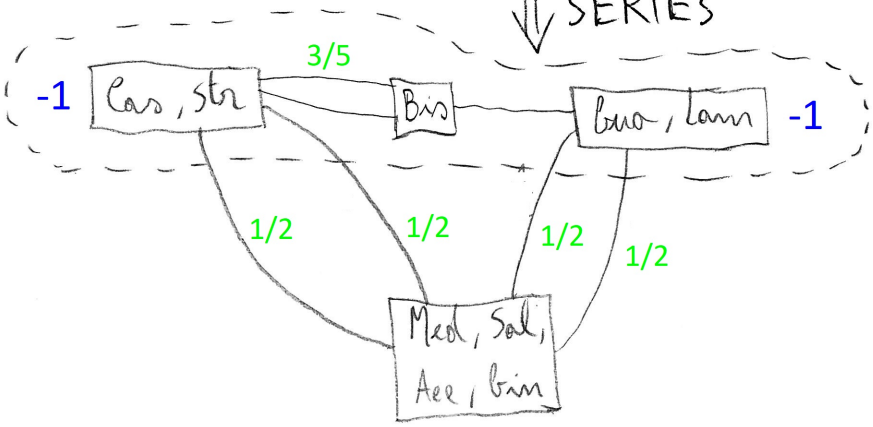



- Values in blue are the opinions of the stubborn nodes ("voltages")
- Values in green are the "conductances" of the links (1 if not specified)

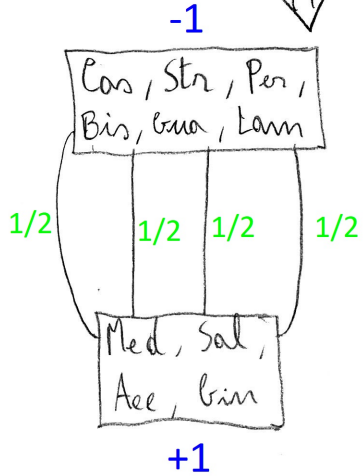
GLUEING + PARALLEL




SERIES




PARALLEL + SERIES + GLUEING



Interpreting this as an electrical network, across each link in the simplified network there is an electric current:

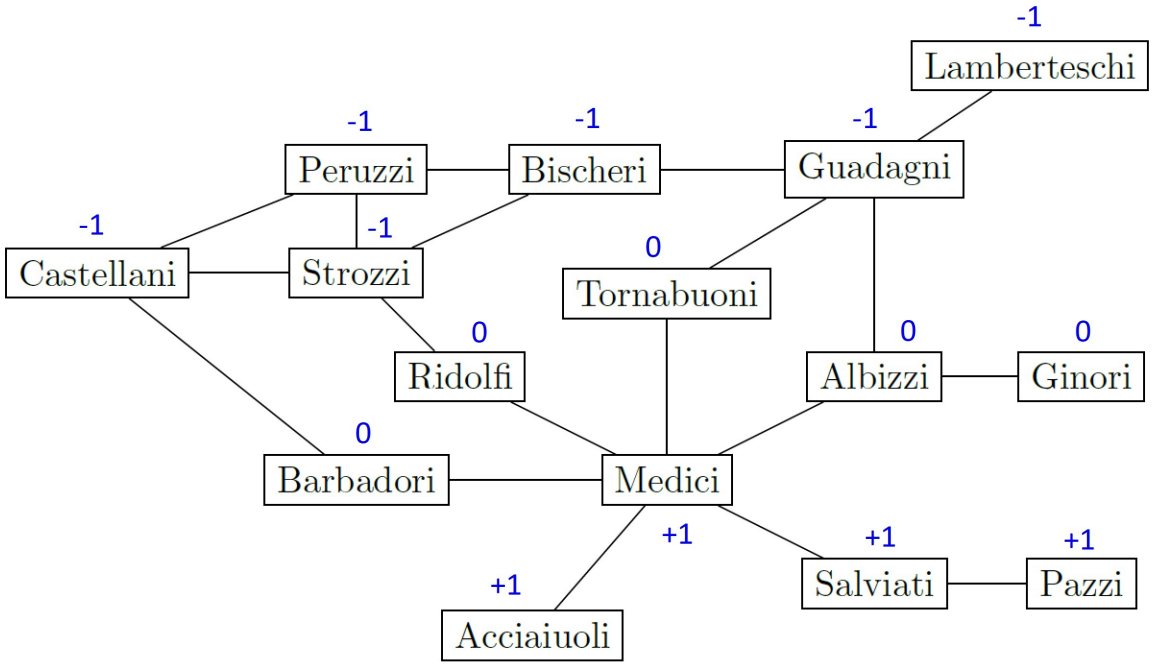
$$\phi = \frac{1}{2} \cdot (1 - (-1)) = 1$$

Using Ohm's law, we can compute the voltages of Barbadori, Ridolfi, Tornabuoni, Albizzi, Ginori:

$$\phi_{ij} = C_{ij} \cdot [x_i - x_j]_+ \Rightarrow 1 = 1 \cdot (1 - x_j) \Rightarrow x_j = 0$$

$$\forall j \in \{\text{Barbadori, Ridolfi, Tornabuoni, Albizzi, Ginori}\}$$

The final opinions, so the equilibrium vector of the averaging dynamics, are displayed back on the original graph:




(d) Write down a Python code for the iterative distributed computation of the PageRank centrality in the network with $\beta = 0.15$ and uniform input

SEE CODE AT THE END OF THIS PDF

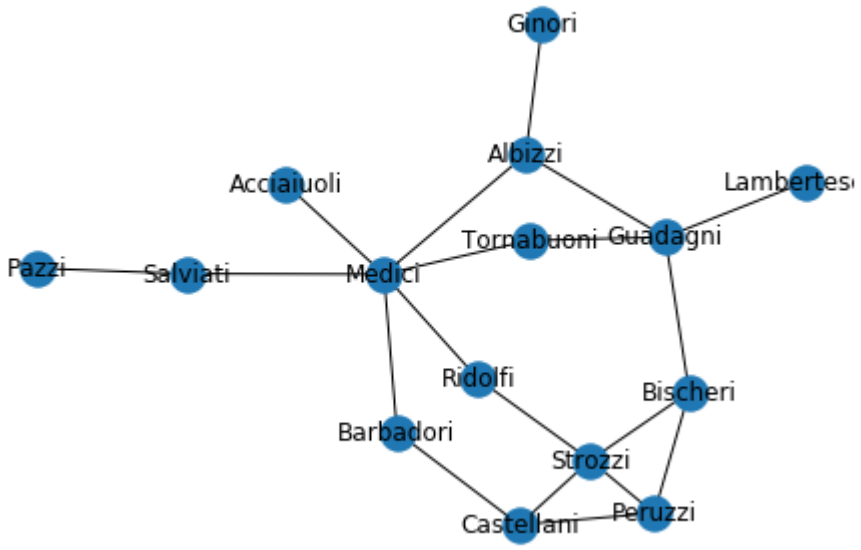
# Problem 3

### Graph implementation

In [5]:
```
import networkx as nx
import numpy as np

G = nx.florentine_families_graph()
G.remove_edge('Ridolfi', 'Tornabuoni')

n_nodes = len(G)
nx.draw(G,with_labels=True)
```




## 3 (b)

### Simulate the averaging dynamics with stubborn nodes and deduce the equilibrium state vector

In [6]:
```
# Construct a dictionary that maps the label of nodes (from "Acciaiuoli" to "L
amberteschi") to their index (from 0 to n_nodes-1)
indices = dict()
for i in range(n_nodes):
    indices[list(G.nodes)[i]] = i

# Number of iterations (i.e. the final value of t)
n_iter = 50;

# Stubborn and regular nodes
stubborn = ["Medici", "Strozzi"];
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_nodes) if id not in stubborn_id]

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


# P matrix
A = nx.adjacency_matrix(G) # -> return type is scipy.sparse.csr_matrix
A = A.toarray() # convert A to a numpy array
degrees = np.sum(A,axis=1)
D = np.diag(degrees)
P = np.linalg.inv(D) @ A

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

# Sample a random initial condition for regular nodes (doesn't matter which on
e, because there are stubborn nodes)
ic = np.random.uniform(0,1,len(regular))

# Set the initial condition for the dynamics
x = np.zeros((n_nodes,n_iter))
x[stubborn_id,0] = u;
x[regular_id,0] = ic;
print("Initial condition:", x[:,0], "\n")

# Evolve the opinion vector
for t in range(1,n_iter):
    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];

# Print the final opinions vector
x_final = x[:,n_iter-1]
print("Equilibrium state vector:")
for key in indices.keys():
    print(key, x_final[indices[key]])
```

```
Initial condition: [ 0.51933878  1.          0.58761043  0.23273592 -1.
0.98933672
  0.15493033  0.11515023  0.75079834  0.15335904  0.10372332  0.15784684
  0.53018453  0.61126294  0.30202046]

Equilibrium state vector:
Acciaiuoli 1.0
Medici 1.0
Castellani -0.45454387960344433
Peruzzi -0.6363619077955383
Strozzi -1.0
Barbadori 0.27272786589268255
Ridolfi 0.0
Tornabuoni 0.6363698417160129
Albizzi 0.6363720957143226
Salviati 0.9999999732888735
Pazzi 0.9999999495362665
Bischeri -0.45454014507399015
Guadagni 0.2727333712249387
Ginori 0.636370008389607
Lamberteschi 0.27273968343202576
```

**Plot the trajectories of the different states**

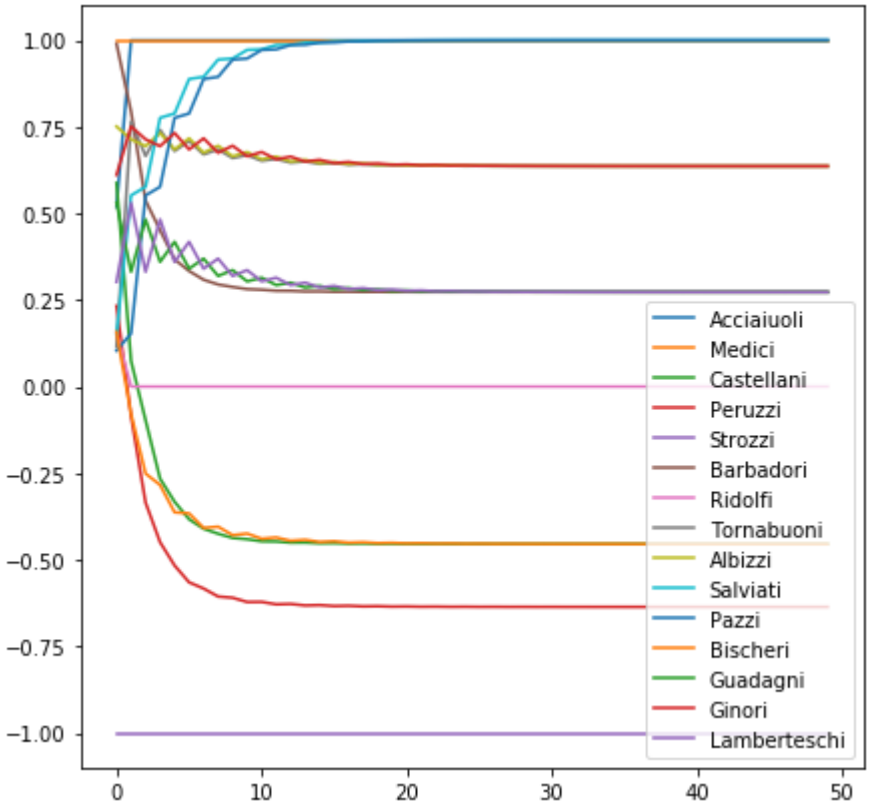
In [7]:
```
import matplotlib.pyplot as plt

fig = plt.figure(1, figsize=(7,7))
ax = plt.subplot(111)

for node in range(n_nodes):
    trajectory = x[node,:]
    ax.plot(trajectory, label=f'{list(G.nodes)[node]}')

ax.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x279708e3f48>
```




## 3 (d)

In [10]:
```
# Bonacich centrality with beta = 0.15 and mu = (1/n,...,1/n) (AKA PageRank ce
ntrality)
beta = 0.15
mu = np.ones(n_nodes) / n_nodes # vector mu = (1/n, ..., 1/n)

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

# Solve the equation iteratively
# initial condition: vector z_0 = (1/n, ..., 1/n)
z_0 = np.ones(n_nodes) / n_nodes
tol = 1e-6   # tolerance to assess convergence to the actual centrality

z_old = z_0
while True:
    z_new = (1-beta) * (P.T @ z_old) + beta*mu
    if np.linalg.norm(z_new-z_old) < tol:
        break
    z_old=z_new
pagerank_centrality = z_new
print("PageRank centrality z: \n", dict(zip(G.nodes, pagerank_centrality)))

print("\n")

# NB: This is the same as
# pagerank_centrality = nx.algorithms.link_analysis.pagerank_alg.pagerank(G)
# print("PageRank centrality z: \n", pagerank_centrality)
```

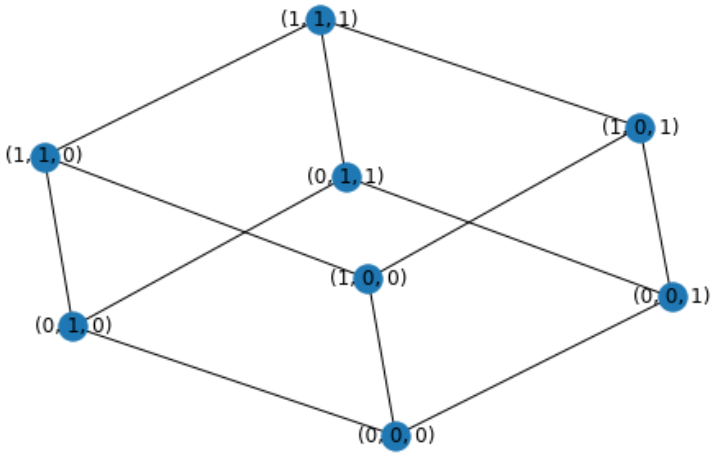
```
PageRank centrality z:
 {'Acciaiuoli': 0.03182343758844888, 'Medici': 0.1540483123469975, 'Castellan
i': 0.07165629904618817, 'Peruzzi': 0.07018523977288754, 'Strozzi': 0.0923160
4649492077, 'Barbadori': 0.05212598201573548, 'Ridolfi': 0.05144052388396614,
'Tornabuoni': 0.05384654657581029, 'Albizzi': 0.08212479959629101, 'Salviat
i': 0.06312845617354695, 'Pazzi': 0.036829519781271525, 'Bischeri': 0.0715259
3955359711, 'Guadagni': 0.10363850509777912, 'Ginori': 0.033268619164787516,
'Lamberteschi': 0.03202310898736141}
```

# Problem 4

### First graph

In [53]:

```
G = nx.generators.lattice.hypercube_graph(3)
n_nodes = len(G)
nx.draw(G, with_labels=True)
```




In [24]:

```
# Construct a dictionary that maps the label of nodes (from (0,0,0) to (1,1,1)) to their index (from 0 to
n_nodes-1 = 7)
indices = dict()
for i in range(n_nodes):
    indices[list(G.nodes)[i]] = i

# Number of iterations
n_iter = 100;

# We will store final opinion vectors and
# average of final opinions in dictionaries
# where the key is the position (i,j) of the
# 1-stubborn agent
final_opinions = dict()
average_opinion = dict()


for (i,j,k) in G.nodes:
    # Position (0,0,0) is occupied by the 0-stubborn node
    if (i,j,k)==(0,0,0):
        continue

    # Stubborn and regular nodes
    stubborn = [(0,0,0), (i,j,k)];
    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_nodes) if id not in stubborn_id]
    print("Stubborn nodes:", stubborn)

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


    # P matrix
    W = nx.adjacency_matrix(G) # -> return type is scipy.sparse.csr_matrix
    W = W.toarray() # convert W to a numpy array
    degrees = np.sum(W,axis=1)
    D = np.diag(degrees)
    P = np.linalg.inv(D) @ W

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

    # Sample a random initial condition for regular nodes
    ic = np.random.uniform(0,1,len(regular))
```

```
    # Set the initial condition for the dynamics
    x = np.zeros((n_nodes,n_iter))
    x[stubborn_id,0] = u;
    x[regular_id,0] = ic;

    for t in range(1,n_iter):
        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,k)] = x[:,n_iter-1]
    average_opinion[(i,j,k)] = np.average(final_opinions[(i,j,k)])
    print("Average opinion:", average_opinion[(i,j,k)])
```

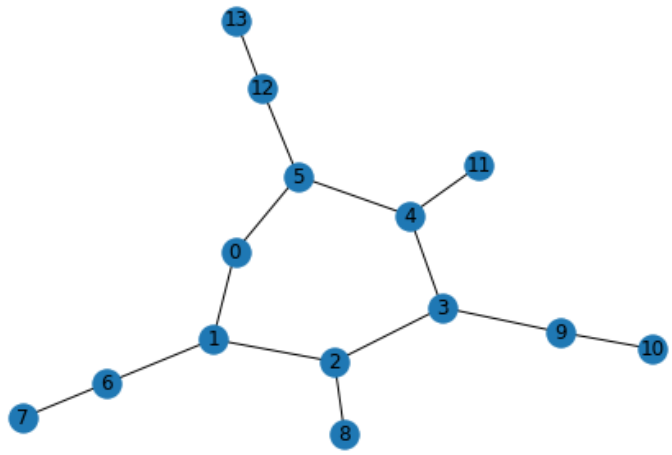
```
Stubborn nodes: [(0, 0, 0), (0, 0, 1)]
Average opinion: 0.4999999999850473
Stubborn nodes: [(0, 0, 0), (0, 1, 0)]
Average opinion: 0.4999999999628718
Stubborn nodes: [(0, 0, 0), (0, 1, 1)]
Average opinion: 0.49999999999997796
Stubborn nodes: [(0, 0, 0), (1, 0, 0)]
Average opinion: 0.5000000000117133
Stubborn nodes: [(0, 0, 0), (1, 0, 1)]
Average opinion: 0.5000000000000009
Stubborn nodes: [(0, 0, 0), (1, 1, 0)]
Average opinion: 0.49999999999998923
Stubborn nodes: [(0, 0, 0), (1, 1, 1)]
Average opinion: 0.4999999999999999
```

**So it doesn't matter where we position the second stubborn node $s$ with $x_s = 1$: the average asymptotic opinion $H(s)$ will be equal to $0.5$**

#### Second graph

In [33]:

```
G = nx.generators.classic.cycle_graph(6)
G.add_nodes_from(range(6,14))
G.add_edges_from([(1,6),(6,7),(2,8),(3,9),(9,10),(4,11),(5,12),(12,13)])
n_nodes = len(G)
nx.draw(G, with_labels=True)
```




In [35]:

```
# Number of iterations
n_iter = 1000;

# We will store final opinion vectors and average of final opinions
# in dictionaries where the key is the 1-stubborn node
final_opinions = dict()
average_opinion = dict()


for i in G.nodes:
    # Node 0 is the 0-stubborn node
    if i==0:
        continue

    # Stubborn and regular nodes
    stubborn = [0, i];
    regular = [node for node in G.nodes if node not in stubborn]
```

```
    print("Stubborn nodes:", stubborn)

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


    # P matrix
    W = nx.adjacency_matrix(G) # -> return type is scipy.sparse.csr_matrix
    W = W.toarray() # convert W to a numpy array
    degrees = np.sum(W,axis=1)
    D = np.diag(degrees)
    P = np.linalg.inv(D) @ W

    # Submatrices
    Q = P[np.ix_(regular, regular)]
    E = P[np.ix_(regular, stubborn)]

    # Sample a random initial condition for regular nodes
    ic = np.random.uniform(0,1,len(regular))

    # Set the initial condition for the dynamics
    x = np.zeros((n_nodes,n_iter))
    x[stubborn,0] = u;
    x[regular,0] = ic;

    for t in range(1,n_iter):
        x[regular, t] = Q @ x[regular, t-1] + E @ x[stubborn, t-1]
        x[stubborn, t] = x[stubborn, t-1];

    final_opinions[i] = x[:,n_iter-1]
    average_opinion[i] = np.average(final_opinions[i])
    print("Average opinion:", average_opinion[i])
```

```
Stubborn nodes: [0, 1]
Average opinion: 0.557142857142857
Stubborn nodes: [0, 2]
Average opinion: 0.5357142857142857
Stubborn nodes: [0, 3]
Average opinion: 0.5476190476190477
Stubborn nodes: [0, 4]
Average opinion: 0.5357142857142856
Stubborn nodes: [0, 5]
Average opinion: 0.5571428571428572
Stubborn nodes: [0, 6]
Average opinion: 0.3311688311688311
Stubborn nodes: [0, 7]
Average opinion: 0.2394957983193278
Stubborn nodes: [0, 8]
Average opinion: 0.3367346938775511
Stubborn nodes: [0, 9]
Average opinion: 0.38571428571428573
Stubborn nodes: [0, 10]
Average opinion: 0.2959183673469388
Stubborn nodes: [0, 11]
Average opinion: 0.336734693877551
Stubborn nodes: [0, 12]
Average opinion: 0.33116883116883117
Stubborn nodes: [0, 13]
Average opinion: 0.23949579831932777
```

**Therefore, the second stubborn node $s$ with $x_s = 1$ which would maximize the average asymptotic opinion $H(s)$ is equivalently node 1 or node 5, i.e. the nodes adjacent to the stubborn node $0$ with opinion $x_0 = 0$;**

**if one of these two nodes were stubborn with opinion $1$, then $H(s) \simeq 0.55714$**