POLITECNICO DI TORINO

NETWORK DYNAMICS AND LEARNING

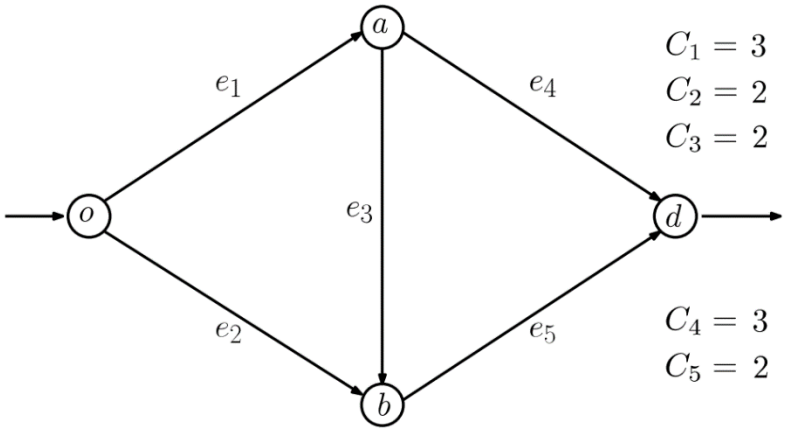
2020 / 2021

ALBERTO MARIA FALLETTA

S277971

HOMEWORK I

Problem 1



A)

Apart from the nodes of origin ‘o’ and destination ‘d’ there are two other nodes, this means there are 2^2 possible ways of splitting the nodes in two subsets for which the origin node and the destination one are always apart. For each of these cuts the sum of the capacity of the links going from a node belonging to the subset containing ‘o’ to one node belonging to the subset containing ‘d’ is to be calculated (for ‘o-b’ cut, in fact, the link e_3 is not to be considered since it is ingoing).

CUT	CUT OUTGOING LINKS	CUT CAPACITY
o	e_1, e_2	$C_1 + C_2 = 5$
o-a	e_2, e_3, e_4	$C_2 + C_3 + C_4 = 7$
o-b	e_1, e_5	$C_1 + C_5 = 5$
o-a-b	e_4, e_5	$C_4 + C_5 = 5$

As it is possible to see from this table there are three cuts with value of cut capacity five, this value being the minimum capacity value among cuts is also called *Min-Cut Capacity* and corresponds to the minimum number of capacity units to be removed from the cut with that capacity to be in the condition of no feasible unitary flow from ‘o’ to ‘d’.

B)

To maximize the feasible throughput from node ‘o’ to node ‘d’, in case there would be the possibility of adding a total of two units of additional capacity to any possible node, these additional capacity units are to be added with the goal of maximizing the Min-Cut Capacity choosing, among the links of the cut with value of capacity equal to the Min-Cut Capacity, the ones in common with the most possible number of cuts having capacity equal to the Min-Cut Capacity.

Therefore, as it is possible to see from the table above, there are three cuts with capacity five and one with capacity seven. Adding a first capacity unit to link e_1 not only makes the cut capacity increase to six but also increases by one unit the capacity of the ‘o-b’ cut (adding the capacity unit to e_2 in fact would have had a positive effect only on the ‘o’ cut).

At this point there is only one cut with capacity five ('o-a-b') so the last capacity unit is to be added to one of its outgoing links therefore link e_5 can be selected since adding a capacity unit there benefits also the 'o-b' cut.

CUT	CAPACITY ADDED	CUT CAPACITY
o	$e_1 + 1$	$C_1 + C_2 = 6$
o-a		$C_2 + C_3 + C_4 = 7$
o-b	$e_1 + 1, e_5 + 1$	$C_1 + C_5 = 7$
o-a-b	$e_5 + 1$	$C_4 + C_5 = 6$

C)

Considering the delay functions associated to each link and infinite capacity, we can compute the Wardrop equilibrium:

$$d_1(x) = d_5(x) = x + 1, \qquad d_3(x) = 1, \qquad d_2(x) = d_4(x) = 5x + 1 \qquad (1)$$

As it is possible to see from the graph there are three possible paths going from 'o' to 'd' ('o-a-d', 'o-a-b-d' and 'o-b-d'). Assuming unitary flow we can state that this can only be divided among these three paths and therefore, identifying z_i as the flow on path i, we can say:

$$z_1 + z_2 + z_3 = 1 \longrightarrow z_2 = 1 - z_1 - z_3 \qquad (2)$$

We then write the equations relative to the flow on each link given by the sum of flows going through that specific link so:

$$f_1 = z_1 + z_2, \qquad f_2 = z_3, \qquad f_3 = z_2, \qquad f_4 = z_1, \qquad f_5 = z_2 + z_3 \qquad (3)$$

We consequently write the delay on each link by combining the flow-on-link equations (3) with the delay functions (1), specifically $\tau_i = d_i(f_i)$:

$$\tau_1 = z_1 + z_2 + 1, \qquad \tau_2 = 5z_3 + 1, \qquad \tau_3 = 1, \qquad \tau_4 = 5z_1 + 1, \qquad \tau_5 = z_2 + z_3 + 1$$

We now write the delays-on-path equations given by the sum of the delay of the links belonging to the given path:

$\Delta_1 = \tau_1 + \tau_4 = 6z_1 + z_2 + 2$

$\Delta_2 = \tau_1 + \tau_3 + \tau_5 = z_1 + 2z_2 + z_3 + 3$

$\Delta_3 = \tau_2 + \tau_5 = 6z_3 + z_2 + 2$

Condition for Wardrop equilibrium is $z_i > 0 \Longrightarrow \Delta_i \leq \Delta_j, \forall j \neq i$ that means if there is flow on a given path then the delay of that path has to be smaller or equal to the delay of any other path. We can now test this condition for the three flows substituting one variable with a combination of the two others thanks to (2):

$z_1 > 0 \implies \Delta_1 \leq \Delta_2 \longrightarrow 6z_1 + z_2 + 2 \leq z_1 + 2z_2 + z_3 + 3 \longrightarrow z_1 \leq 2/6 = 1/3$

$z_1 > 0 \implies \Delta_1 \leq \Delta_3 \longrightarrow 6z_1 + z_2 + 2 \leq 6z_3 + z_2 + 2 \longrightarrow z_1 \leq z_3$

$z_2 > 0 \implies \Delta_2 \leq \Delta_1 \longrightarrow z_1 + 2z_2 + z_3 + 3 \leq 6z_1 + z_2 + 2 \longrightarrow z_1 \geq 2/6 = 1/3$

$z_2 > 0 \implies \Delta_2 \leq \Delta_3 \longrightarrow z_1 + 2z_2 + z_3 + 3 \leq 6z_3 + z_2 + 2 \longrightarrow z_3 \geq 2/6 = 1/3$

$z_3 > 0 \implies \Delta_3 \leq \Delta_1 \longrightarrow 6z_3 + z_2 + 2 \leq 6z_1 + z_2 + 2 \longrightarrow z_3 \leq z_1$

$z_3 > 0 \implies \Delta_3 \leq \Delta_2 \longrightarrow 6z_3 + z_2 + 2 \leq z_1 + 2z_2 + z_3 + 3 \longrightarrow z_3 \leq 2/6 = 1/3$

Solving the system made of these six equations along with (2) we find $z_1 = z_2 = z_3 = 1/3$, therefore the flow vector is $f^{(0)} = (f_1,\ f_2,\ f_3,\ f_4,\ f_5) = \left(\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}\right)$

D)

We can now compute the Social Optimum flow vector:

$$min \sum_{e \in \varepsilon} f_e \tau_e(f_e) = \min\{f_1\tau_1(f_1) + f_2\tau_2(f_2) + f_3\tau_3(f_3) + f_4\tau_4(f_4) + f_5\tau_5(f_5)\}$$

Applying once again equations (3) to the previous minimization formula we obtain:

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

And after a few computations and the use of (2):

$$\min\{5 - 3z_1 - 3z_3 + 6z_1^2 + 6z_3^2\}$$

To minimize this function (calling $f = 5 - 3z_1 - 3z_3 + 6z_1^2 + 6z_3^2$) we derive with respect to every variable and set the derivative equal to zero after checking the concavity with the help of the second derivative (which correctly give us a U shape per the quadratic function in analysis):

$$\frac{df}{dz_1} = -3 + 12z_1, \qquad \frac{df}{dz_3} = -3 + 12z_3$$

We now have the system:

$$\begin{cases} 12z_1 - 3 = 0 \\ 12z_1 - 3 = 0 \\ z_2 = 1 - z_1 - z_3 \end{cases} \longrightarrow \quad z_1 = z_3 = \frac{1}{4}, \qquad z_2 = \frac{2}{4}$$

And consequent Social Optimum flow vector:

$$f^* = (f_1, f_2, f_3, f_4, f_5) = \left(\frac{3}{4}, \frac{1}{4}, \frac{2}{4}, \frac{1}{4}, \frac{3}{4}\right)$$

E)

We can now compute the Price of Anarchy:

$$PoA = \frac{\sum_{e \in \varepsilon} f_e^{(0)} \tau_e(f_e^{(0)})}{\sum_{e \in \varepsilon} f_e^* \tau_e(f_e^*)}$$

Where $f^{(0)}$ is the Wardrop Equilibrium flow vector and $f^*$is the Social Optimum flow vector, so,

$$PoA = \frac{\frac{2}{3}\left(\frac{2}{3}+1\right)+\frac{1}{3}\left(5\frac{1}{3}+1\right)+1\frac{1}{3}+\frac{1}{3}\left(5\frac{1}{3}+1\right)+\frac{2}{3}\left(\frac{2}{3}+1\right)}{\frac{3}{4}\left(\frac{3}{4}+1\right)+\frac{1}{4}\left(5\frac{1}{4}+1\right)+1\frac{2}{4}+\frac{1}{4}\left(5\frac{1}{4}+1\right)+\frac{3}{4}\left(\frac{3}{4}+1\right)} = \frac{\frac{39}{9}}{\frac{68}{16}} = \frac{624}{612} \approx 1{,}0196$$

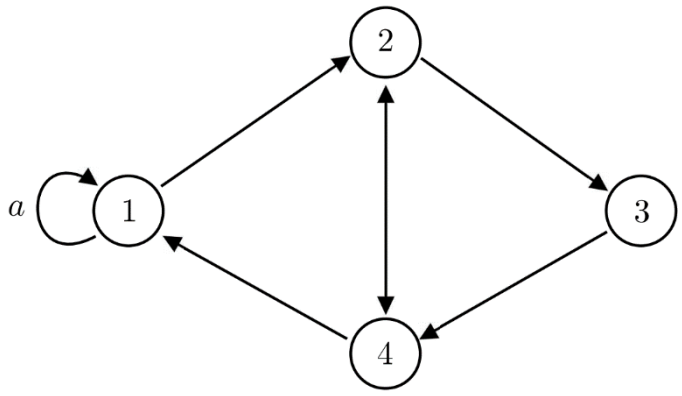
F)

We now compute the vector of tolls that reduces the Price of Anarchy to one using the formula:

$$\omega_e = f_e^* \tau_e'(f_e^*)$$

$$\omega_1 = \frac{3}{4}\cdot 1 = \frac{3}{4}, \qquad \omega_2 = \frac{1}{4}\cdot 5 = \frac{5}{4}, \qquad \omega_3 = \frac{2}{4}\cdot 0 = 0, \qquad \omega_4 = \frac{1}{4}\cdot 5 = \frac{5}{4}, \qquad \omega_5 = \frac{3}{4}\cdot 1 = \frac{3}{4}$$

Problem 2



A)

$$W = \begin{bmatrix} a & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} a/a+1 & 1/a+1 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix},$$

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

C)

Since the graph is strongly connected and aperiodic, and these properties are true also in the absence of a (i.e. $a = 0$), the opinion profile $x(t)$ converges to $\pi' x(0)$ as $t \longrightarrow +\infty$ for every value of $a \geq 0$ and every initial condition $x(0)$. Furthermore, since G is strongly connected and balanced the invariant distribution centrality π is unique.

D)

The graph under analysis is strongly connected (meaning π is unique) and balanced (so $\pi = \frac{w}{\mathbb{1}'W}$), consequently:

$$\pi = \frac{w}{\mathbb{1}'W} = (\frac{1}{6}, \frac{2}{6}, \frac{1}{6}, \frac{2}{6})$$

Having initial condition:

$$x(0) = (-1, 1, -1, 1)$$

The consensus value is therefore:

$$\pi' x(0) = \frac{1}{6}(-1) + \frac{1}{3} \cdot 1 + \frac{1}{6}(-1) + \frac{1}{3} \cdot 1 = -\frac{1}{6} + \frac{1}{3} - \frac{1}{6} + \frac{1}{3} = \frac{1}{3}$$

E)

The graph is still balanced and strongly connected even if $a = 0$, so:

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

$$x(0) = (-1,1,-1,1)$$

$$\pi' x(0) = \frac{1+a}{6+a}(-1) + \frac{2}{6+a} \cdot 1 + \frac{1}{6+a}(-1) + \frac{2}{6+a} \cdot 1 =$$

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

We now want to look for the minimum value of $a \geq 0$ for which $x(t) \leq 0$. Being $a \geq 0$ there is no condition to impose to the denominator, it is always positive, therefore, $x(t) \leq 0$ when $2 - a \leq 0$

$$\frac{2-a}{6+a} \leq 0 \Longrightarrow 2 - a \leq 0 \Longrightarrow a \geq 2$$

The smallest possible value of $a \geq 0$ to have $x(t) \leq 0$ is then $a = 2$.

F)

$$x_i(0) = N_i, \qquad \mathbb{E}[N_i] = 0, \qquad Var[N_i] = 1, \qquad \pi = (\frac{1+a}{6+a}, \frac{2}{6+a}, \frac{1}{6+a}, \frac{2}{6+a})$$

$$\pi' x(0) = \sum_i \pi_i N_i, Var[\pi' x(0)] = Var\left[\sum_i \pi_i N_i\right] = Var[N_i] \cdot Var[\sum_i \pi_i] = 1 \cdot \sum_i \pi_i^2 =$$

$$= \frac{(1+a)^2}{(6+a)^2} + \frac{2^2}{(6+a)^2} + \frac{1^2}{(6+a)^2} + \frac{2^2}{(6+a)^2} = \frac{a^2 + 2a + 10}{a^2 + 12a + 36}$$

To minimize this function, we derive and set the result equal to zero:

$$\frac{df}{da} = \frac{(2a+2)(a^2+12a+36) - (a^2+2a+10)(2a+12)}{(6+a)^4}$$

No problems on the denominator, we then set this derivative to zero:

$$\frac{(2a+2)(a^2+12a+36) - (a^2+2a+10)(2a+12)}{(6+a)^4} = 0 \Longrightarrow$$

$$(2a+2)(a^2+12a+36) = (a^2+2a+10)(2a+12) \Longrightarrow$$

$$2a^3+24a^2+72a+2a^2+24a+72 = 2a^3+12a^2+4a^2+24a+20a+120 \Longrightarrow$$

$$10a^2+52a-48 = 0 \Longrightarrow$$

$$a = \frac{-52 \pm \sqrt{2704+1920}}{20} = -6 \; or + \frac{4}{5}$$

Since $a \geq 0$ we can only accept the positive value.

Problem 3

A)

As it is possible to see, the graph is strongly connected (that tells us the invariant distribution is unique), undirected and therefore balanced (that tells us the invariant distribution is simply equal to the normalized degree vector) and finally aperiodic (that along with strong connectivity, or at least a single globally reachable sink component, is a requirement for convergence of the averaging dynamics). Before computing the consensus value, it is, however, useful to establish a relationship dictionary between vectors indices and Florentine families (the same notation will be used in the coding part):

families = {0:"Lamberteschi", 1:"Peruzzi", 2:"Bischeri", 3:"Guadagni", 4:"Castellani", 5:"Strozzi", 6:"Tornabuoni", 7:"Ridolfi", 8:"Albizzi", 9:"Ginori", 10:"Barbadori", 11:"Medici", 12:"Acciaiuoli", 13:"Salviati", 14:"Pazzi"}

Therefore:

$$\pi = \frac{w}{\mathbb{1}'W} = (\frac{1}{38}, \frac{3}{38}, \frac{3}{38}, \frac{4}{38}, \frac{3}{38}, \frac{4}{38}, \frac{2}{38}, \frac{2}{38}, \frac{3}{38}, \frac{1}{38}, \frac{2}{38}, \frac{6}{38}, \frac{1}{38}, \frac{2}{38}, \frac{1}{38})$$

$$x(0) = (0,0,0,0,0,-1,0,0,0,0,0,1,0,0,0)$$

$$consensus = \pi' x(0) = -1 \cdot \frac{4}{38} + 1 \cdot \frac{6}{38} = \frac{2}{38} = \frac{1}{19}$$

B)

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

families = {0:"Lamberteschi", 1:"Peruzzi", 2:"Bischeri", 3:"Guadagni", 4:"Castellani",
          5:"Strozzi", 6:"Tornabuoni", 7:"Ridolfi", 8:"Albizzi", 9:"Ginori",
          10:"Barbadori", 11:"Medici", 12:"Acciaiuoli", 13:"Salviati", 14:"Pazzi"}

pos = {0:[6,6], 1:[2,5], 2:[3,5], 3:[5,5], 4:[1,4],
      5:[2,4], 6:[4,4], 7:[3,3], 8:[5,3], 9:[6,3],
      10:[2,2], 11:[4,2], 12:[3,1], 13:[5,1], 14:[6,1]}

G = nx.Graph()
G.add_nodes_from(families.keys())
# Even if undirected the edges are added in both directions to avoid forgetting one
G.add_edges_from([(0, 3),  # Lamberteschi
                  (1, 4), (1, 5), (1, 2),  # Peruzzi
                  (2, 1), (2, 5), (2, 3),  # Bischeri
                  (3, 0), (3, 2), (3, 6), (3, 8),  # Guadagni
                  (4, 1), (4, 5), (4, 10),  # Castellani
                  (5, 4), (5, 1), (5, 2), (5, 7),  # Strozzi
                  (6, 3), (6, 11),  # Tornabuoni
                  (7, 5), (7, 11),  # Ridolfi
                  (8, 3), (8, 11), (8, 9),  # Albizzi
                  (9, 8),  # Ginori
                  (10, 11), (10, 4),  # Barbadori
                  (11, 6), (11, 8), (11, 13), (11, 12), (11, 10), (11, 7),  # Medici
                  (12, 11),  # Acciaiuoli
                  (13, 11), (13, 14),  # Salviati
                  (14, 13)])  # Pazzi

n_nodes = len(G)
print("Number of nodes:", n_nodes)

nx.draw(G, pos, with_labels=True)
n_iter = 20  # Number of iterations

# Stubborn and regular nodes
stubborn = [11, 5]
regular = [node for node in G.nodes if node not in stubborn]

u = [1, -1]  # Stubborn nodes input
ic = np.zeros(len(regular))  # Regular nodes initial condition

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

# Submatrices Q and E
```

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

# Initial condition for the dynamics
x = np.zeros((n_nodes,n_iter))
x[stubborn, 0] = u;
x[regular, 0] = ic;
print("Initial condition:", x[:,0], '\n')

# Evolve the opinion vector
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];

print("Nodes value:")
x_final = x[:,n_iter-1]
for node in G.nodes:
   print(families[node], ": ", round(x_final[node], 2))
fig = plt.figure(1, figsize=(17,10))
ax = plt.subplot(111)

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

ax.legend()
```

Final convergence values:
Lamberteschi: 0.27
Peruzzi: -0.64
Bischeri: -0.46
Guadagni: 0.27
Castellani: -0.46
Strozzi: -1.0
Tornabuoni: 0.63
Ridolfi: 0.0
Albizzi: 0.63
Ginori: 0.63
Barbadori: 0.27
Medici: 1.0
Acciaiuoli: 1.0
Salviati: 1.0
Pazzi: 1.0

$$convergence\ value = (0.27,\ -0.64,\ -0.46,\ 0.27,\ -0.46,\ -1,\ 0.63,\ 0,\ 0.63,\ 0.63,\ 0.27,\ 1,\ 1,\ 1,\ 1)$$

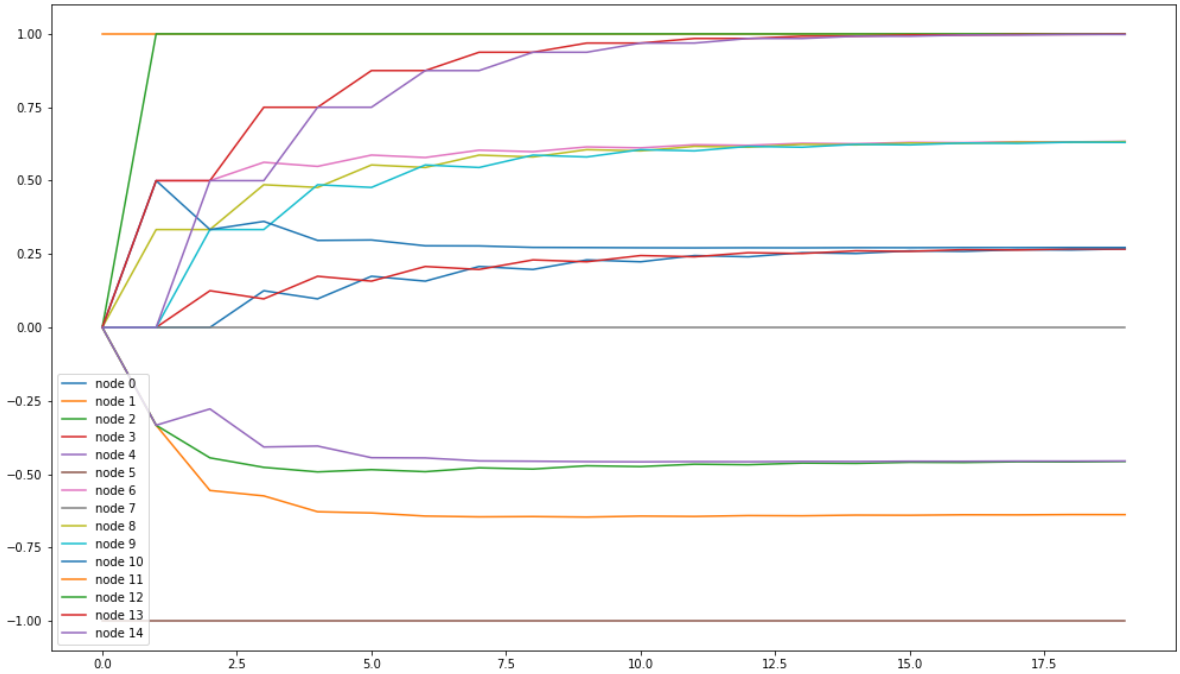
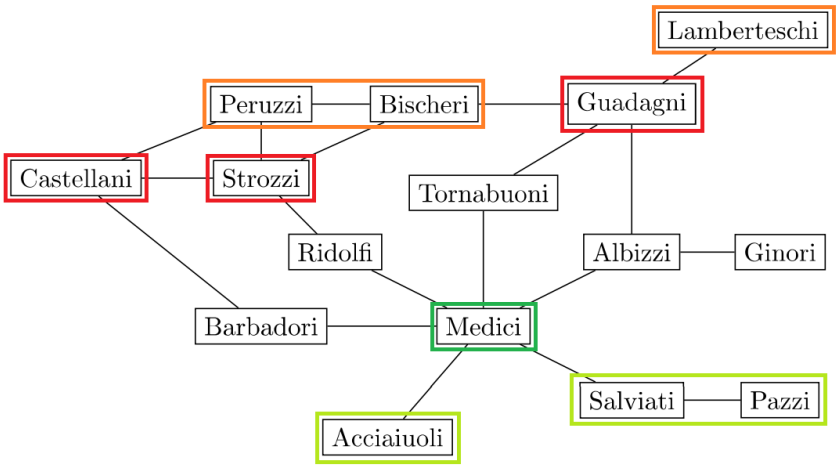


Figure: Node state trajectories

C)



We could compute analytically the equilibrium vector of the averaging dynamics with stubborn nodes ('Castellani', 'Strozzi', 'Guadagni', 'Medici') but what is even easier il looking at the graph to apply the gluing property of electrical networks. Identifying the -1 value stubborn nodes in red and +1 value stubborn node in dark green shows us that the nodes highlighted in orange will converge to -1 since they can only reach -1 value stubborn nodes, while, on the other hand, the light green group of nodes, being in connection to dark green nodes only, will converge to +1.
To be discussed is the convergence value of all the remaining nodes but as we can see, apart from 'Ginori' they are all equally influenced by the red group and by the dark green one so their value will stay constant from the start at zero. 'Ginori' node, being in connection to a node that keeps the value of zero will maintain the value of zero itself.

$$convergence\ value = (-1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 1, 1, 1, 1)$$

D)

As we know, Pagerank centrality is the case of Bonachich centrality ($x = (1 - \beta)P'x + \beta\mu$) when $\beta = 0{,}15$ and $\mu = 1$.

After the custom implementation, checking what we obtain from the library provided Pagerank algorithm (networkx.algorithms.link_analysis.pagerank_alg.pagerank) we can see the results are the same at most of an approximation:

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

families = {0:"Lamberteschi", 1:"Peruzzi", 2:"Bischeri", 3:"Guadagni", 4:"Castellani",
        5:"Strozzi", 6:"Tornabuoni", 7:"Ridolfi", 8:"Albizzi", 9:"Ginori",
        10:"Barbadori", 11:"Medici", 12:"Acciaiuoli", 13:"Salviati", 14:"Pazzi"}

pos = {0:[6,6], 1:[2,5], 2:[3,5], 3:[5,5], 4:[1,4],
     5:[2,4], 6:[4,4], 7:[3,3], 8:[5,3], 9:[6,3],
     10:[2,2], 11:[4,2], 12:[3,1], 13:[5,1], 14:[6,1]}

G = nx.Graph()
G.add_nodes_from(families.keys())
G.add_edges_from([(0, 3),  # Lamberteschi
            (1, 4), (1, 5), (1, 2),  # Peruzzi
            (2, 1), (2, 5), (2, 3),  # Bischeri
            (3, 0), (3, 2), (3, 6), (3, 8),  # Guadagni
            (4, 1), (4, 5), (4, 10),  # Castellani
            (5, 4), (5, 1), (5, 2), (5, 7),  # Strozzi
            (6, 3), (6, 11),  # Tornabuoni
            (7, 5), (7, 11),  # Ridolfi
            (8, 3), (8, 11), (8, 9),  # Albizzi
            (9, 8),  # Ginori
            (10, 11), (10, 4),  # Barbadori
            (11, 6), (11, 8), (11, 13), (11, 12), (11, 10), (11, 7),  # Medici
            (12, 11),  # Acciaiuoli
            (13, 11), (13, 14),  # Salviati
            (14, 13)])  # Pazzi

n_nodes = len(G)
print("Number of nodes:", n_nodes)
nx.draw(G, pos, with_labels=True)

W = nx.adjacency_matrix(G).toarray() # scipy.sparse.csr_matrix to numpy array
degrees = np.sum(W, axis=1)
D = np.diag(degrees)
P = np.linalg.inv(D) @ W

beta = 0.15
mu = 1
```

```
n_nodes = len(G)

x_0 = np.ones((n_nodes,1))/n_nodes  # initial condition: 1/n-uniform vector of size n_nodes
tol = 1e-5  # set a tolerance to assess convergence to the limit

x_old = x_0
while True:
   x_new = (1-beta) * P.T @ x_old + beta * mu
   if np.linalg.norm(x_new-x_old) < tol:
      break
   x_old=x_new
my_pagerank = x_new

my_pagerank = my_pagerank/np.sum(my_pagerank)  # Normalization

print("My Pagerank: \n")
for node in G.nodes:
  print(families[node], ": ", round(my_pagerank[node][0], 3))
print()

official_pagerank = nx.algorithms.link_analysis.pagerank_alg.pagerank(G)

print("Official Pagerank: \n")
for node in G.nodes:
  print(families[node], ": ", round(official_pagerank[node], 3))
```

Output:

My Pagerank:

Lamberteschi : 0.032
Peruzzi : 0.07
Bischeri : 0.072
Guadagni : 0.104
Castellani : 0.072
Strozzi : 0.092
Tornabuoni : 0.054
Ridolfi : 0.051
Albizzi : 0.082
Ginori : 0.033
Barbadori : 0.052
Medici : 0.154
Acciaiuoli : 0.032
Salviati : 0.063
Pazzi : 0.037

Official Pagerank:

Lamberteschi : 0.032
Peruzzi : 0.07
Bischeri : 0.072
Guadagni : 0.104
Castellani : 0.072
Strozzi : 0.092
Tornabuoni : 0.054
Ridolfi : 0.051
Albizzi : 0.082
Ginori : 0.033
Barbadori : 0.052
Medici : 0.154
Acciaiuoli : 0.032
Salviati : 0.063
Pazzi : 0.037

Problem 4

Graph 1

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

pos = {0:[1,1], 1:[3,1], 2:[4,2], 3:[2,2], 4:[1,6],
     5:[3,6], 6:[4,7], 7:[2,7]}

G = nx.Graph()
G.add_nodes_from([0, 1, 2, 3, 4, 5, 6, 7])
G.add_edges_from([(0, 1), (1, 2), (2, 3), (3, 0),  # Lower square
             (4, 5), (5, 6), (6, 7), (7, 4),  # Upper square
             (0, 4), (1, 5), (2, 6), (3, 7),])  # Connection between squares

n_nodes = len(G)
print("Number of nodes:", n_nodes)

nx.draw(G, pos, with_labels=True)

n_iter = 50  # Number of iterations

final_opinions = dict()
average_opinion = dict()

for node in G.nodes:
   # 0-stubborn node is fixed
   if node == 0:
      continue
```

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

    u = [0,1]  # Input to stubborn nodes
    ic = np.random.uniform(0,1,len(regular))  # random initial condition for regular nodes

    # P matrix
    W = nx.adjacency_matrix(G).toarray()  # scipy.sparse.csr_matrix to 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)]

    # 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[node] = x[:,n_iter-1]
    average_opinion[node] = np.average(final_opinions[node])
    print("Average opinion:", round(average_opinion[node], 3), "Stubborn nodes: ", stubborn)
```

Output:
Average opinion: 0.5 Stubborn nodes: [0, 1]
Average opinion: 0.5 Stubborn nodes: [0, 2]
Average opinion: 0.5 Stubborn nodes: [0, 3]
Average opinion: 0.5 Stubborn nodes: [0, 4]
Average opinion: 0.5 Stubborn nodes: [0, 5]
Average opinion: 0.5 Stubborn nodes: [0, 6]
Average opinion: 0.5 Stubborn nodes: [0, 7]

From the output of the numerical simulation we can see that all the positions lead to the same value of average asymptotic opinion.

Graph 2

```
pos = {0:[1,3], 1:[2,4], 2:[3,4], 3:[4,3], 4:[3,2], 5:[2,2], 6:[2,5], 7:[1,5],
      8:[3,5], 9:[5,3], 10:[5,4], 11:[3,1], 12:[2,1], 13:[1,1]}

G = nx.Graph()
G.add_nodes_from([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
G.add_edges_from([(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0),  # Central Exagon
                  (1, 6), (6, 7), (2, 8), (3, 9), (9, 10), (4, 11), (5, 12), (12, 13)])
```

```
n_nodes = len(G)
print("Number of nodes:", n_nodes)

nx.draw(G, pos, with_labels=True)
n_iter = 500  # Number of iterations

final_opinions = dict()
average_opinion = dict()

for node in G.nodes:
    # 0-stubborn node is fixed
    if node == 0:
        continue

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

    u = [0,1]  # Input to stubborn nodes
    ic = np.random.uniform(0,1,len(regular))  # random initial condition for regular nodes

    # P matrix
    W = nx.adjacency_matrix(G).toarray()  # scipy.sparse.csr_matrix to 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)]

    # 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[node] = x[:,n_iter-1]
    average_opinion[node] = np.average(final_opinions[node])
    print("Average opinion:", round(average_opinion[node], 3), "Stubborn nodes: ", stubborn)
```

Output:

Average opinion: 0.557 Stubborn nodes: [0, 1]
Average opinion: 0.536 Stubborn nodes: [0, 2]
Average opinion: 0.548 Stubborn nodes: [0, 3]
Average opinion: 0.536 Stubborn nodes: [0, 4]
Average opinion: 0.557 Stubborn nodes: [0, 5]
Average opinion: 0.331 Stubborn nodes: [0, 6]
Average opinion: 0.239 Stubborn nodes: [0, 7]

Average opinion: 0.337 Stubborn nodes: [0, 8]
Average opinion: 0.386 Stubborn nodes: [0, 9]
Average opinion: 0.296 Stubborn nodes: [0, 10]
Average opinion: 0.337 Stubborn nodes: [0, 11]
Average opinion: 0.331 Stubborn nodes: [0, 12]
Average opinion: 0.239 Stubborn nodes: [0, 13]

From the output of the numerical simulation we can see that the positions that maximize the value of average asymptotic opinion are the two adjacent nodes to the stubborn node 0.