1 City Connectivity (MATLAB)

In this section, a connected, undirected, unweighted graph representing connections between cities in the Netherlands (mainly Vesting Holland) has been supplied. The Bonacich, closeness, decay, and betweenness centralities for the cities of Amsterdam, Schiphol, Rotterdam and Utrecht will be calculated.

1.a

To calculate the Bonacich centrality π, the city connections first need to be represented in a graph. This graph is defined as the triple:

$$\mathcal{G} = \{\mathcal{V},\ \mathcal{E},\ W\},$$

of which the node set $\mathcal{V}$ and the edge set $\mathcal{E}$ can directly be computed from the assignment. Since the graph is unweighted, the adjacency matrix can be constructed from the edge set $\mathcal{E}$ as:

$$W_{ij} = \begin{cases} 1, & \forall e_{ij} \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases}$$

From this graph definition, the degree vector $w = W\mathbb{1}$, degree matrix $D = \text{diag}(w)$ and the normalised weight/adjacency matrix $P = D^{-1}W$ can be found.

To calculate the Bonacich connectivity, a vector π needs to be found which serves as an eigenvector for the normalised adjacency matrix P with corresponding eigenvalue of 1:

$$\pi = P^\top \pi$$

This vector is found using the `null` command to recover the kernel of $(P^\top - I)$. The values for the correct nodes (cities) are then selected from this vector and presented in table 1.

City	*Bonacich centrality*
Amsterdam	0.5145
Schiphol	0.3430
Rotterdam	0.3430
Utrecht	0.3430

Table 1: Bonacich centrality for the four nodes.

1.b

For the closeness centrality of node i, the following definition is used:

$$\text{closeness}(i) = \frac{n}{\sum_{j\in\mathcal{V}} \text{dist}(i,j)}, \quad i \in \mathcal{V} \tag{1}$$

Where the total number of nodes n is divided by the sum of the distances from node i to all other nodes in the set $\mathcal{V}$. This distance is taken to be the shortest distance from node i to node j and is computed using the `distances` command. The resulting closeness centralities for the four nodes are shown in table 2.

City	*Closeness centrality*
Amsterdam	0.6500
Schiphol	0.6190
Rotterdam	0.5000
Utrecht	0.5200

Table 2: Closeness centrality for the four nodes.

1.c

The decay centrality is calculated using the same distance measure as used for the closeness centrality only excluding the distance where node $i = j$. The decay centrality is then defined as:

$$\pi_i = \sum_{j\neq i}^{\mathcal{V}} \delta^{\text{dist}(i,j)}, \quad i,j \in \mathcal{V} \tag{2}$$

where δ is a positive real number between 0 and 1. Table 3 shows the decay centralities for different values of δ.

City	*Decay centrality*		
	$\delta = 0.25$	$= 0.5$	$= 0.75$
Amsterdam	1.7813	4.2500	7.5938
Schiphol	1.4531	3.8750	7.3594
Rotterdam	1.2539	3.3125	6.6914
Utrecht	1.3008	3.4375	6.8320

Table 3: Decay centrality for the four nodes for three different values of δ up to 4 digits.

The decay centrality uses the δ parameter which can be interpreted as the relative importance of 'close' nodes compared to 'distant' nodes.

A $\delta \to 1$ adds distant nodes with the same weighting factor as close nodes. Distant nodes are thus just as important. A $\delta \to 0$ lets the contribution of distant nodes go to 0 fairly quickly. The relative importance of distant nodes will thus be less.

1.d

The betweenness centrality is a related centrality that gives a measure of the importance of a node when travelling within the graph. Calculating this centrality measure is quite involved because the naive approach is to calculate all possible geodesic paths that the graph allows. Luckily, there are more efficient implementations, such as the one proposed in [1], that we have implemented for this exercise using queue and stack classes provided by [2]. We correct for the fact that their definition of the centrality does not include the scaling factor $\frac{1}{n^2}$, and that for undirected graphs the computed numbers must be divided by 2. The centralities of the desired cities are then presented in table 4:

City	*Betweenness centrality*
Amsterdam	0.20168
Schiphol	0.10848
Rotterdam	0.08136
Utrecht	0.03353

Table 4: Betweenness centrality for the four nodes.

1.e

From pure visual inspection of the graph, both Amsterdam and Schiphol are very well connected in the graph, and their connections are typically also well-connected. The Bonacich centrality is proportional to the degree of the nodes: Amsterdam with 5 edges is the highest, and the other cities all have 4 edges. Notable is that for every centrality but the betweenness centrality, Utrecht scores equal or more than Rotterdam. Rotterdam however scores higher for the betweenness centrality because it is the only connection to the leaf Dordrecht, and thus all geodesic paths connecting to Dordrecht must go through Rotterdam.

2 Graphs and Opinions

In this section, we will consider an unweighted directed graph consisting of 6 nodes. We will determine some if its basic properties and then evaluate some discrete-time opinion dynamics on it.

2.a

To check if $\mathcal{G}$ is balanced, the in-degree and out-degree of every node needs to be the same: $w_i^- = w_i$. Since graph $\mathcal{G}$ is undirected, this is always the case. This can be extended to regularity where $w_i = w_i^- = \overline{w}$, i.e. the in degree and out degree of all nodes is the same. This however is not the case since for example $w_3 = w_3^- = 1$ and $w_1 = w_1^- = 2$.
A graph is aperiodic when the maximum common divisor of all cycle lengths is 1. This is not the case since the length of the only cycle is 4.
The diameter of the graph is the largest shortest path between two possible nodes i, j. In graph $\mathcal{G}$, this is equal to 4 on the path (1-2-5-4), since the end node may be equal to the start node.

2.b

The weighted adjacency matrix W is given as:

$$W = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The out-degree vector w is given as:

$$w = W\mathbb{1} = \begin{bmatrix} 2 & 3 & 1 & 2 & 3 & 1 \end{bmatrix}^\top$$

The normalised weight matrix P is defined as:

$$P = D^{-1}W, \quad D = \text{diag}(w)$$

Which results in:

$$P = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Combining D and W to compute the Laplacian matrix L as:

$$L = D - W = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

2.c

The normalised Bonacich centrality vector π is defined as the eigenvector of $P^\top$ associated with the eigenvalue of 1. This vector is then normalised and given as:

$$\pi = P^\top \pi = \frac{1}{\sqrt{28}} \begin{bmatrix} 2 & 3 & 1 & 2 & 3 & 1 \end{bmatrix}^\top$$

2.d

The centralities are given in table 5.

Comparison	*Value pair*
Degree (1,5)	$(2,\ 3)$
Closeness (1,2)	$\left(\frac{2}{3},\ \frac{6}{7}\right)$
Betweenness (2,3)	$\left(\frac{5}{6^2},\ 0\right)$

Table 5

The degree centrality of node 5 is greater than the degree centrality of node 1, since node 5 is conencted to more neighbors. For the closeness centrality, node 2 is higher then node 1. This implies that node 2 has more nodes close by then node 1. For the betweenness centrality, it is clear node 3 has a value of 0 since no path is passing through this node other than those that start or end at this node. Node 2 thus has the higher betweenness centrality.

2.e

Because the graph has been augmented with self-loops on every node, it has now become aperiodic. This is because the maximum common divisor of all cycle lengths is 1 (since a self loop has length 1). These self-loops can be interpreted as a node's self-confidence, which keeps it from just blindly copying the value of its neighbours. This inhibits the emergence of oscillatory behaviour.

2.f

According to the de Groot's model of opinion dynamics, if a graph $\mathcal{G}$ is connected and converges, its limit for $t \to \infty$ is equal to the initial state vector weighted by the Bonacich centrality vector: $\overline{x} = \pi^\top x(0)$. P is based on the new weight/adjacency matrix incorporating the self loops:

$$W = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The out-degree vector w is then calculated as:

$$w = W\mathbb{1} = \begin{bmatrix} 3 & 4 & 2 & 3 & 4 & 2 \end{bmatrix}^\top$$

Which is used to calculate $D = \text{diag}(w)$. The normalised adjacency matrix P is then calculated as $P = D^{-1}W$ which results in:

$$P = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

The Bonacich centrality vector π is then calculated as an eigenvector of $P^\top$ corresponding to eigenvalue 1:

$$\pi = P^\top \pi = \frac{1}{\sqrt{58}} \begin{bmatrix} 3 & 4 & 2 & 3 & 4 & 2 \end{bmatrix}^\top$$

Since the consensus value $\overline{x}$ is defined as $\overline{x} = \pi^\top x(0)$, this is an fairly straight forward linear optimisation problem to maximise $\overline{x}$. The maximum can empirically be achieved by combining the largest entries in $\pi^\top$ with the largest values in $x(0)$ which results in two possible initial condition vectors for the maximum consensus in which just the two entries corresponding to the two 3s in π are swapped.

$$x_{0,1} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 1 \end{bmatrix}, \quad x_{0,2} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

Both these vectors result in an equilibrium value of $\overline{x} = 3.8079$, which is the maximum. A similar approach can be taken when finding the minimum value for $\overline{x}$ where the largest values in π are matched with the smallest values in $x(0)$. This again results in two possible initial value vectors:

$$x_{0,1} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 2 \end{bmatrix}, \quad x_{0,2} = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

Both these vectors result in an equilibrium value of $\overline{x} = 3.2827$.

2.g

In de Groot dynamics, stubborn agents are modelled as sinks. We therefore remove the outgoing connections of node 1 and 6 from the adjacency matrix:

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

From this adjacency matrix, the normalised adjacency matrix P can again be computed as described before. To model the dynamics of the regular nodes, the update equation for the regular node state vector $y(k)$ is defined as:

$$y(k+1) = Qy(k) + Bu \tag{3}$$

Here, Q is defined as the $\mathcal{R} \times \mathcal{R}$ sub matrix of P corresponding to the regular nodes.

$$Q = \sum_i \sum_j \delta^{(i)} \delta^{(j)\top} P_{i,j}, \; i,j \in \{2,3,4,5\}$$

In this case Q is the part of P corresponding to nodes 1 through 5: `Q = P(2:5,2:5)`. B are the columns of P corresponding to the stubborn agents affecting the regular nodes:

$$B = \sum_i \sum_j \delta^{(i)} \delta^{(j)\top} P_{i,j}, \; i \in \{1,6\}, \; j \in \{2,3,4,5\}$$

Which in this case results in: `B = P(2:5,[1 ... 6])`.

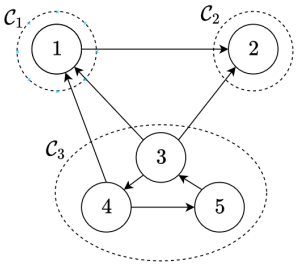
Since there are at least 2 stubborn nodes with different opinion, the de Groot's opinion dynamics will then converge to a disagreement between the nodes described by the equilibrium vector y given by:

$$y_{\text{eq}} = (I - Q)^\top Bu = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{2} \end{bmatrix} \tag{4}$$

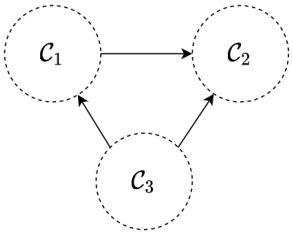
y_{eq} gives the asymptotic opinion value for all remaining nodes for stubborn node opinion vector $u = [0, \; 1]^\top$.

2.h

Graph A A connected component is defined as a maximum subset of node set $\mathcal{V}$ which is connected. Looking at graph A, it can be seen that there are three distinct connected components: $\mathcal{C}_1 : \{1\}, \quad \mathcal{C}_2 : \{2\}, \quad \mathcal{C}_3 : \{3,4,5\}$. This partitioning is graphically shown in the condensation graph in figure 1.



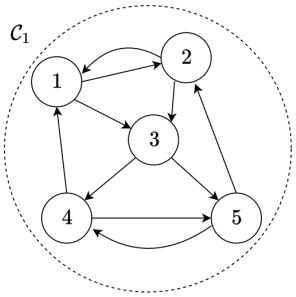
(a) $\mathcal{G}_A$ before condensation



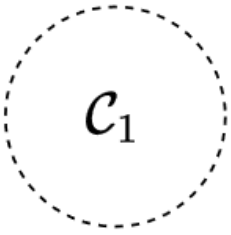
(b) $\mathcal{G}_A$ After condensation

Figure 1: Condensation graph for graph A

Graph B Since the entire graph in Graph B is connected, the number of connected components is equal to 1, namely $\mathcal{C}_1 : \{1, 2, 3, 4, 5\}$. This is also shown in figure 2.



(a) $\mathcal{G}_B$ before condensation



(b) $\mathcal{G}_B$ After condensation

Figure 2: Condensation graph for graph B

3 Influence on Twitter

3.a

The matrix W is not square due to it including some users that do not have any followers. Thus there are columns missing in the adjacency matrix. This can be solved by appending additional zero columns to the matrix until it is square. This is allowed since it does not add any connections, and thus still uniquely represents the graph.

3.b

To compute the PageRank iteratively, a distributed algorithm can be used. This algorithm uses the result of theorem 4[1], which states that for a graph with a nonempty sink set of which at least one sink is reachable, the opinion vector of the regular nodes y will converge to the following value:

$$\overline{y} = (I - Q^\top)^{-1}\lambda \tag{5}$$

This theorem can be implemented by setting the converged opinion vector $\overline{y}$ to the pageRank centrality vector:

$$(I - Q^\top)^{-1}\lambda = (I - (1-\beta)P^\top)^{-1}\beta\mu = \pi^{(\beta)} \tag{6}$$

The system that converges to this value can thus be recovered as:

$$y(t+1) = (1-\beta)P^\top y(t) + \beta\mu, \tag{7}$$

where $\mu = \mathbb{1}$ and $\beta = 0.15$ as typical values. This will converge to the PageRank centrality. After implementing this, the 5 most central nodes can be extracted and are presented in table 6. The number of iterations necessary for convergence was empirically determined to be around 75.

Node	*PageRank centrality*
2	122.5
1	106.9
112	82.3
9	71.1
26	62.7

Table 6

[1] From the lecture notes

3.c

To simulate the discrete time consensus algorithm using two stubborn nodes, first the node set needs to be checked for already present stubborn nodes. It turns out node 1 does not follow anyone, probably due to the way the crawling process does not incorporate the users node 1 follows. This node thus already is a sink. The second sink is chosen from the nodes with the largest PageRank as node 112.

Node 1 will be given opinion 1, node 112 will be given opinion 0. The dynamics from Equation 7 on 5 random nodes in the network over 150 time steps is shown in Figure 3.

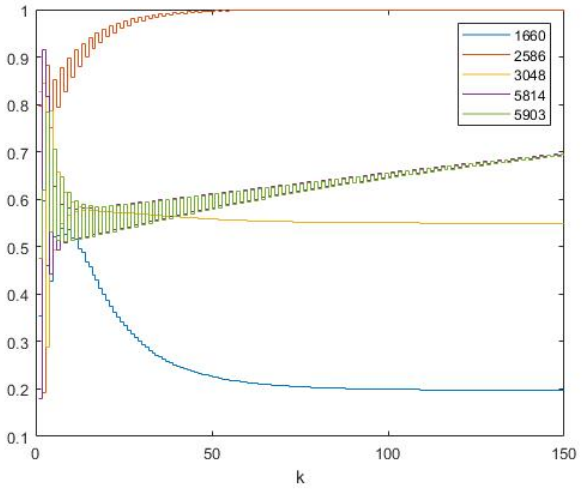


Figure 3: Opinion dynamics of 5 random nodes over 150 time steps.

3.d

Two things seem important to the steady-state consensus value: the relative PageRank of the sinks, and their opinion. In Figure 4, we have investigated strong-strong, strong-weak, weak-strong, and weak-weak sink pairs for an opinion vector $u = \begin{bmatrix}1 & 0\end{bmatrix}^\top$. Since the distribution of opinions is continuous, we have chosen to display them using a boxplot.

It can be seen that pair 1 converges to a low value, even if node 1 has a higher PageRank than node 112. For pair 2, node 2 manages to convince more

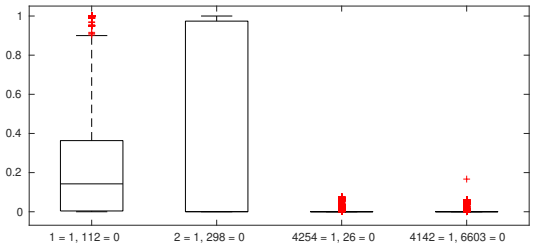


Figure 4: Boxplot of opinion distribution for 4 distinct stubborn node pairs based on their PageRank.

nodes when faced with a the weaker node 298. Keep in mind that here, node 1 is still stubborn! It will still influence the dynamics depending on what opinion value it was initialized as. This also counts for the final two pairs. For node pair 3, the stronger point was assigned opinion 0, and it seems it has managed to drive down the consensus to very low values. The same has happened for node pair 4 with two weak nodes. The visible outlier at ~0.2 is the always stubborn node 1, which will always influence the opinion dynamics since it is a sink.

4 Flows flows flows

4.a

Let us make use of the max-flow min-cut theorem, which states:

$$\tau^*_{o,d} = C^*_{o,d}$$

Since the max-flow problem is a linear programming problem, we can be sure that the optimum lies on at least one active constraint of the problem, namely the capacity constraints, mass conservation constraints and throughput nonnegativity constraints. We can therefore start by finding a flow that maximizes the throughput. From C_9 and C_{10}, we can deduce that an upper bound on this throughput is 12. By working backwards, it is possible to construct a feasible flow vector that realizes this throughput:

$$f^* = [5,\ 6,\ 1,\ 1,\ 9,\ 4,\ 2,\ 0,\ 10,\ 2]^\top$$

Therefore, the min-cut capacity is also 12. Any min-cut of the network would therefore also need

to have a throughput of 12. This can be done by cutting edges e_9 and e_{10}. No other cuts are possible that have a throughput of 12.

4.b

Since we have already constructed a feasible flow vector in subsection 4.a for a throughput of 12, it is trivial to modify it for a throughput of 11, e.g. by setting f_3^* to 0.

4.c

Recall that the maximum possible throughput of a network is limited by the min-cut capacity as bottleneck. In order for no feasible flow of throughput 8 to exist, it suffices to simply remove capacity from the edges cut by the min-cut. Since the current max-flow is 12, it suffices to remove $12 - 8 + 1 = 5$ units of capacity from C_9 and C_{10}, e.g. by setting C_9 to 5.

4.d

Let $\mathcal{M}$ denote the set of sets of edges cut by the min-cuts of the network, with capacity c. In the original network, $\mathcal{M}_{12} = \{\{e_9,\ e_{10}\}\}$. Since the throughput of the min-cut is the bottleneck of the network, we should expend 1 unit of extra capacity to increase the throughput of this min-cut. This can be done by either adding it to e_9 or e_{10}. Our max-flow is now 13, and there are two min-cuts with a throughput of 13 that are edge-independent:

$$\mathcal{M}_{13} = \{\{e_2,\ e_3,\ e_4,\ e_6\},\ \{e_9,\ e_{10}\}\}$$

The maximum throughput of the network τ^* is now limited by two disjoint sets of edges. Since we have only 1 extra unit of capacity left, the min-cut capacity of the network can *not* be increased any further, since there would always remain at least one min-cut with a throughput of 13. Therefore, it can be spent wherever one would like, or saved for some other time. If we are allowed to expend our one unit of capacity to also construct a new edge, we can create a new edge between o and d, such that is must be included in both min-cuts.

4.e

For the second part of question 4, the network in figure 5 will be used. Here, the edges and node names are already defined.

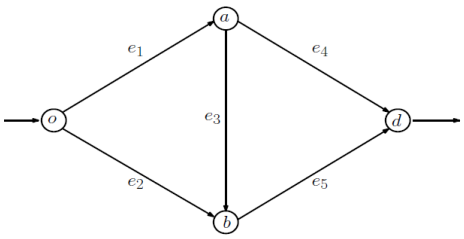


Figure 5: Network for second part of problem 4.

The flow capacities of the edges are assumed to be infinite. The total throughput is defined as τ and the delays for the edges are defined as:

$$\begin{aligned} d_1(x) &= d_5(x) = x + 1 \\ d_3(x) &= 1 \\ d_2(x) &= d_4(x) = 5x + 1 \end{aligned} \tag{8}$$

With x being the flow on the respective link.

The path set that is investigated is the set of paths between node o and node d. For this (o, d) pair, the link-path incidence matrix can be set up containing all 3 paths from o to d:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{9}$$

The flow vector f can be found by multiplying the link-path incidence matrix with a vector describing the aggregate flows over the separate paths $z = \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix}^\top$ as:

$$f = Az = \begin{bmatrix} z_1 + z_3 \\ z_2 \\ z_3 \\ z_1 \\ z_2 + z_3 \end{bmatrix} \tag{10}$$

Where $z_1 + z_2 + z_3 = \tau$ and $z \geq 0$.

The delays for the separate edges can then be found by substituting this flow vector into the delay functions from 8.

$$\begin{aligned} d_1(z) &= z_1 + z_3 + 1 \\ d_2(z) &= 5z_2 + 1 \\ d_3(z) &= 1 \\ d_4(z) &= 5z_1 + 1 \\ d_5(z) &= z_2 + z_3 + 1 \end{aligned}$$

Using this, the total delay for every path can be computed:

$$\begin{aligned} d_{p_1} &= d_1 + d_4 &&= 6z_1 + z_3 + 2 \\ d_{p_2} &= d_2 + d_5 &&= 6z_2 + z_3 + 2 \\ d_{p_3} &= d_1 + d_3 + d_5 = &&\ z_1 + z_2 + 2z_3 + 3 \end{aligned}$$

From the definition of the Wardrop equillibrium, if $z_i > 0$, the delay over d_{p_i} must be equal or less than the delays over the other paths. Therefore, if we assume some z_i to be greater than 0, we can infer constraints on the other path flows such that we satisfy the main constraint.

We will first assume $z_1 > 0$, and compare the delay on path 1 to the delay on path 2:

$$\begin{aligned} d_{p_1} = 6z_1 + \cancel{z_3 + 2} &\leq 6z_2 + \cancel{z_3 + 2} = d_{p_2} \\ z_1 &\leq z_2 \end{aligned} \tag{11}$$

We can then compare path 1 to path 3, by making use of the fact that $z_1 + z_2 + z_3 = 1$:

$$\begin{aligned} d_{p_1} = 6z_1 + z_3 + 2 &\leq z_1 + z_2 + 2z_3 + 3 = d_{p_3} \\ z_1 &\leq 1/3 \end{aligned} \tag{12}$$

The same can be done when assuming $z_2 > 0$, and comparing the delay on path 2 to the delay on path 1 and path 3. This results in the following constraints:

$$z_2 \leq z_1 \tag{13}$$

$$z_2 \leq 1/3 \tag{14}$$

If we combine constraints 11 and 13, we infer $z_1 = z_2$, and from constraints 12 and 14, we infer z_2, $z_2 \in (0, \frac{1}{3})$. Therefore, z_3 is at least $1 - \max(z_1) - \max(z_3) = 1/3$. If compare the delay on path 3 with the delay on path 1 with that fact in mind:

$$\begin{aligned} z_1 + z_2 + \cancel{2z_3} + 3 &\leq 6z_2 + \cancel{z_3} + 2 \\ 1 - \cancel{z_3} + \cancel{z_3} + 3 &\leq 6z_1 + 2 \\ 1/3 &\leq z_1 \end{aligned} \tag{15}$$

Since we know z_3 must always be greater or equal than 1/3, and thus $1/3 \geq z_1 \geq 1/3$, we arrive at the conclusion that $z_1 = z_2 = z_3 = 1/3$. The final user optimum vector $z^{(0)}$ is thus equal to:

$$z^{(0)} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}^\top$$

The Wardrop optimum flow vector can then be computed by substituting this in equation 10:

$$f^{(o)} = \begin{bmatrix} 2/3 & 1/3 & 1/3 & 1/3 & 2/3 \end{bmatrix}^\top$$

4.f

To find the social optimal flow vector f^*, the total cost function is defined as the sum of the cost at every edge multiplied with the flow on that edge subject to positive flow and the geography of the graph:

$$\begin{aligned} M(\nu) = \min_{f \geq 0} \quad & \sum_{e \in \epsilon} f_e \cdot d_e(f_e) \\ \text{s.t.} \quad & \\ & Bf = \nu \end{aligned} \tag{16}$$

Where:

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix},$$

$$\nu = \underbrace{\begin{bmatrix} \tau \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau \end{bmatrix}}_{\lambda - \mu} = \begin{bmatrix} \tau \\ 0 \\ 0 \\ -\tau \end{bmatrix}$$

We can use the constraint that $Bf = \nu$ to make some substitutions:

$$f_2 = 1 - f_1 \quad f_5 = f_4 - 1 \quad f_3 = 1 - f_2 - f_4$$

We can then express the optimization problem as

$$\begin{aligned} M(\nu) = \min_{f \geq 0} \quad & 6f_2^2 - 3f_2 + 6f_4^2 - 3f_4 + 5 \\ \text{s.t.} \\ & 1 - f_2 \geq 0, \quad f_2 \geq 0, \quad 1 - f_2 - f_4 \geq 0 \\ & f_4 \geq 0, \quad 1 - f_4 \geq 0 \end{aligned}$$

We're lucky and the global optimum lies within the constraints:

$$\begin{aligned} \nabla(M(\nu)) = \begin{bmatrix} 12f_2 - 3 \\ 12f_4 - 3 \end{bmatrix} = 0 \\ f_2 = f_4 &= \tfrac{1}{4} \\ \implies \quad f_1 = f_5 &= \tfrac{3}{4} \\ f_3 &= \tfrac{1}{2} \end{aligned}$$

Therefore the social optimum flow vector f^* is

$$f^* = [3/4,\ 1/4,\ 1/2,\ 1/3,\ 3/4]$$

4.g

The price of Anarchy (PoA) associated to a Wardrop equilibrium $f^{(o)}$ is the ratio between the total delay at that Wardrop equilibrium and the total delay at the social equilibrium subject to the network structure. Assuming the throughput $\tau = 1$:

$$\begin{aligned} PoA(0) &= \frac{\sum_{e \in \epsilon} f_e^{(o)} \cdot d_e(f_e^{(o)})}{\sum_{e \in \epsilon} f_e^* \cdot d_e(f_e^*)} \\ &= \frac{52}{51} \end{aligned}$$

Since $\text{PoA}(0) \geq 1$, the Wardrop optimum is clearly less optimal then the social optimum.

4.h

To obtain a $\text{PoA}(\omega)$ of 1, the cost function for the Wardrop equilibrium needs to be realigned with the cost function for the social optimum, which intuitively makes the users optimise the global cost while thinking they're optimising their selfish cost function. This can be done by choosing the toll cost vector ω to be the optimal flow at every edge times the derivative of the delay function evaluated at the optimal flow of that edge:

$$\omega_e = f_e^* d_e'(f_e^*) \tag{17}$$

Implementing this, we find the toll vector:

$$\omega = \begin{bmatrix} \frac{3}{4} & \frac{5}{4} & 0 & \frac{5}{4} & \frac{3}{4} \end{bmatrix}^\top$$

Which results in the optimal Wardrop flow vector:

$$f(\omega^*) = \begin{bmatrix} 2/3 & 1/3 & 1/3 & 1/3 & 2/3 \end{bmatrix}^\top$$

This coincides with the social optimal flow vector, which thus reduces the price of anarchy to 1.

5 Traffic in Rio

In this question, we will work with the following delay function:

$$d_e(f_e) = \frac{l_e}{1 - \frac{f_e}{C_e}} \tag{18}$$

5.a

To find the shortest path, the minimum traveltimes along the edges are taken as weights. The graph and these weights are used by `shortestpath()` to determine the shortest time path in the graph. The resulting shortest time path is represented by vertex set:

$$p_t^* = \{e_1,\ e_2,\ e_3,\ e_4,\ e_7,\ e_{12},\ e_{13}\}$$

With accompanying total travel time: $T = 23$.

5.b

For the maximum flow in the network, the capacities of the links are assigned as weights. Using these capacities, the function `maxflow()` determines the min cut and its respective max flow at 5400 between node 1 and 13.

5.c

The in- and outflow at each node can be found by using the node-incidence matrix B as:

$$Bf = \nu \tag{19}$$

Where f is the given flow vector, and ν represents the net in or outflow at every node. A negative flow represents an outflow, positive an inflow.

Node 1:	3329	Node 8:	-3507
Node 2:	9654	Node 9:	0
Node 3:	4552	Node 10:	-6725
Node 4:	0	Node 11:	-2539
Node 5:	0	Node 12:	-18
Node 6:	9897	Node 13:	-11760
Node 7:	-2883		

Table 7: External in and outflow at each node

5.d

The given cost function can be rewritten slightly for better performance:

$$\sum_{e\epsilon\mathcal{E}} \frac{l_e C_e}{1 - f_e/C_e} \tag{20}$$

The following CVX statements will then compute the social optimum:

```
cvx_begin
  variable f(M)
  minimize sum((l.*c).* ...
    inv_pos(One-f.*(1.*inv_pos(c))))
  subject to
    B*f == lambda - mu
    0 ≤ f ≤ c
cvx_end
```

We will take τ, the traffic flow that wants to go from node 1 to node 13, to be equal to the net inflow at node 1 (3329). We then find the following social optimum:

$$\begin{aligned} f^* = 10^3 \cdot [&3.329,\ 2.536,\ 0.793,\ 1.321,\\ &1.216,\ 0.069,\ 2.045,\ 0.000,\\ &0.000,\ 1.216,\ 0.000,\ 1.216,\\ &0.000,\ 0.021,\ 1.216,\ 0.048,\\ &2.045,\ 0.000,\ 1.237,\ 2.2093]^\top \end{aligned} \tag{21}$$

with a total delay of 187.0191. The throughput is lower than the maximum flow! The flow is visualised in Figure 6:

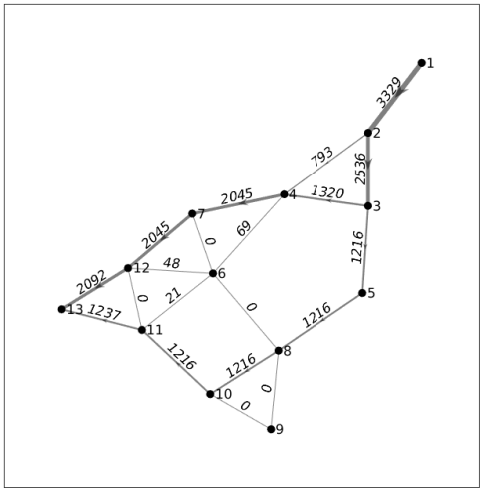


Figure 6: Network flows superimposed on edges. The edges are labelled with the flow on each edge, rounded to the nearest integer.

It can be seen that the flow is around the edges of the graph, with almost no flow going through the central node 6, which corresponds to the large city Tijuca. Node 9, Botafogo, is also bypassed.

5.e

We will now again use CVX to find the Wardrop equilibrium f_W, using the supplied cost function

$$J_W(f_e) = \sum_{e\epsilon\mathcal{E}} \int_0^{f_e} d_e(x)dx = \sum_{e\epsilon\mathcal{E}} \int_0^{f_e} \frac{l_e}{1 - x/C_e} dx$$

Since by definition $0 \le f_e < C_e$, we can evaluate the integral and arrive at the following convex cost function, cancelling the constant additive terms:

$$J_W(f_e) = \sum_{e\epsilon\mathcal{E}} C_e l_e (\cancel{\ln(C_e)} - \ln(C_e - f_e)) \tag{22}$$

Note that we will have a problem if $f_e^W = C_e$. After evaluation, we then obtain the Wardrop

equilibrium:

$$f^o = 10^3 \cdot [3.329,\ 2.253,\ 0.077,\ 2.478\\ 0.774,\ 0.000,\ 2.555,\ 0.000\\ 0.000,\ 0.774,\ 0.000,\ 0.774\\ 0.000,\ 0.000,\ 0.774,\ 0.000\\ 2.555,\ 0.000,\ 0.774,\ 2.554]^\top$$

with a total delay of 188.7019. This is not much worse than the social optimum! The flow is visualised in Figure 7:

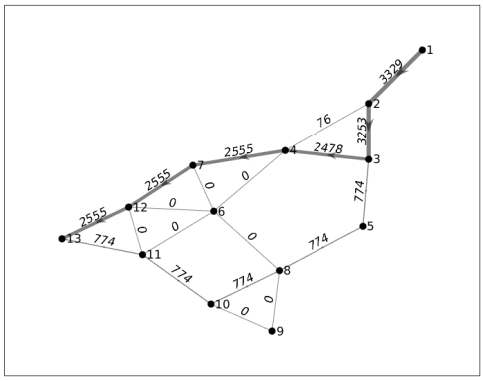


Figure 7: Network flows superimposed on edges. The edges are labelled with the flow on each edge, rounded to the nearest integer.

People now completely avoid node 6 (Tijuca) and solely travel on the ringroad.

5.f

We now introduce tolls on the system, such that the toll on link e is $\omega_e = f_e^* d_e'(f_e^*)$, where f_e^* is the flow at the social optimum as found in Equation 21. Then the delay on link e is

$$d_e(f_e) + f_e^* d_e'(f_e^*) = \frac{l_e}{1 - f_e/C_e} + \frac{l_e C_e f^*}{(C_e - f^*)^2}$$

This is again evaluated using CVX, resulting in the following Wardrop optimum:

$$f^W = 10^3 \cdot [3.329,\ 2.536,\ 0.793,\ 1.321\\ 1.216,\ 0.069,\ 2.045,\ 0.000\\ 0.000,\ 1.216,\ 0.000,\ 1.216\\ 0.000,\ 0.021,\ 1.216,\ 0.048\\ 2.045,\ 0.000,\ 1.237,\ 2.093]^\top$$

Comparing this optimum to the social optimum computed in 21 we can see that there is still a very slight difference assuming the two vectors have double precision. The difference vector is given as:

$$f^W - f^* = [0.000,\ -0.012,\ 0.012,\ 0.025\\ -0.036,\ 0.037,\ 0.001,\ 0.000\\ 0.000,\ -0.036,\ 0.000,\ -0.036\\ 0.000,\ 0.028,\ -0.0363,\ 0.009\\ 0.001,\ 0.000,\ -0.008,\ 0.008]^\top$$

The order of this error is much smaller then the order of the flow vectors themselves. This leads to believe that this error might be due to numerical inaccuracies or the convergence bound in the optimisation algorithm.

5.g

The new social optimum f^* using the new cost function is equal to:

$$f^* = 10^3 \cdot [3.329,\ 2.891,\ 1.438,\ 0.837,\\ 1.054,\ 1.204,\ 1.071,\ 0.243,\\ 0.000,\ 1.054,\ 0.231,\ 0.824,\\ 0.231,\ 0.404,\ 1.054,\ 0.557,\\ 1.314,\ 0.077,\ 1.535,\ 1.794]^\top \tag{23}$$

To obtain a Wardrop equilibrium that coincides with the social optimum, the appropriate delay vector needs to be computed. This delay vector is taken as:

$$w^* = c_e'(f_e^*) - d_e(f_e^*)$$

With:

$$c_e(f_e^*) = f_e^*(d_e(f_e^*) - l_e)$$

Taking the derivative of $c_e(f_e^*)$ with respect to f_e^* and substituting this in the equation for w^* yields:

$$\begin{aligned} w^* &= f_e^* d_e'(f_e^*) - l_e \\ &= f_e^* \frac{C_e l_e}{(C_e - f_e^*)^2} - l_e \end{aligned} \tag{24}$$

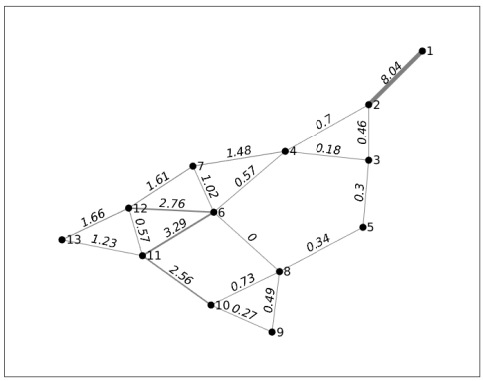


Figure 8: Delay with respect to the traveltime with 0 flow superimposed on the graph, rounded to the nearest one hundreth.

6 Drugs

6.a

The system can be described by a linear compartmental system model of the form $\dot{x} = -L^\top x + \lambda$, where L is the Laplacian of the graph that represents the system:

$$\begin{aligned} \dot{x}_D(t) &= \lambda(t) + k_{SD}x_S(t) \\ &\quad - (k_{DU} + k_{DS} + k_{DM})x_D(t) \\ \dot{x}_S(t) &= k_{DS}x_D(t) - k_{SD}x_S(t) \\ \dot{x}_M(t) &= k_{DM}x_D(t) - k_{MU}x_M(t) \end{aligned} \tag{25}$$

Which we rewrite to

$$\dot{x}(t) = \underbrace{\begin{bmatrix} -(k_{DU} + k_{DS} + k_{DM}) & k_{SD} & 0 \\ k_{DS} & -k_{SD} & 0 \\ k_{DM} & 0 & -k_{MU} \end{bmatrix}}_{-L^\top} x(t) + \begin{bmatrix} \lambda(t) \\ 0 \\ 0 \end{bmatrix} \tag{26}$$

6.b

Via Theorem 3 from the reader, let a graph $\mathcal{G}$ contain a non-empty set $\mathcal{S} \subseteq \mathcal{V}$ of sinks such that from every node $i \in \mathcal{V}$ at least one sink $s \in \mathcal{S}$ is reachable. Our compartmental system does not contain a sink, but we can easily augment it with one extra compartment U which represents the sum of drug and metabolite output in urine, without regard of units:

$$\mathcal{V}_{\text{aug}} = \mathcal{V} \cup \mathcal{S} = \mathcal{V} \cup \{U\}$$

This augmented system is graphically represented in Figure 9.

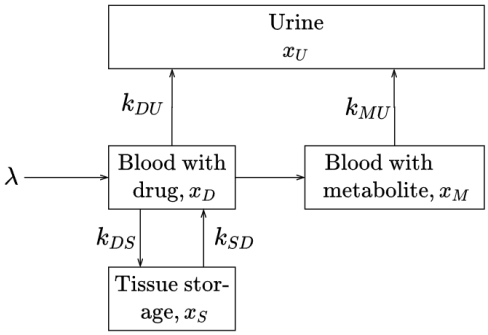


Figure 9: Augmented graph with sink U.

Then the set $\mathcal{R} = \mathcal{V}_{\text{aug}} \setminus \mathcal{S}$ is simply the original set of compartments $\mathcal{V}$, and from every compartment $i \in \mathcal{R}$ we can reach the virtual compartment $U \in \mathcal{S}$. Note that in this case, the $\mathcal{R} \times \mathcal{R}$ block M of the Laplacian of the augmented system is simply the Laplacian L of the original system, and the vector $y(t) \in \mathbb{R}^{\mathcal{R}}$ is simply the original state vector $x(t) \in \mathbb{R}^{\mathcal{V}}$. We can then state that for any initial condition $x(0) \in \mathbb{R}^3$,

$$\lim_{t \to +\infty} x(t) = (L^\top)^{-1}\lambda \tag{27}$$

with λ constant. For the given values this results in

$$\frac{1}{3}\begin{bmatrix} 10 \\ 20 \\ 20 \end{bmatrix} \tag{28}$$

Interpreting the graph with the provided flows as an LTI system, one can also apply LTI techniques to find the steady state concentrations. The general solution to an LTI system is given by equation 29 as presented in [3].

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{29}$$

The state of the system is taken as $x = [x_D,\ x_S,\ x_M]^\top$. Assuming a nonzero initial condition and a constant step input from $t = 0$ of $u(t) = \lambda = 2$, and state matrices:

$$A = \begin{bmatrix} -(k_{DU} + k_{DS} + k_{DM}) & k_{SD} & 0 \\ k_{DS} & -k_{SD} & 0 \\ k_{DM} & 0 & -k_{MU} \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Using this and following the approach on on p.150 of [3], equation 29 can be written as:

$$\begin{aligned} x(t) &= e^{At}x(0) + 2\int_0^t e^{A(t-\tau)}d\tau B \\ &= e^{At}x(0) + 2\int_0^t e^{A\sigma}d\sigma B \\ &= e^{At}x(0) + 2(A^{-1}e^{A\sigma}B)\Big|_{\sigma=0}^{\sigma=t} \\ &= \underbrace{e^{At}x(0)}_{\text{initial}} + \underbrace{2A^{-1}e^{At}B}_{\text{transient}} - \underbrace{2A^{-1}B}_{\text{steady-state}} \end{aligned} \tag{30}$$

Now taking $\lim_{t\to\infty} x(t)$, it is clear the e^{At} term in the transient and initial condition response exponentially goes to 0 since A has strictly negative eigenvalues for the given parameters. The steady state response thus becomes:

$$\overline{x} = -2A^{-1}B = \frac{1}{3}\begin{bmatrix} 10 \\ 20 \\ 20 \end{bmatrix}$$

Which is equal to the limit concentrations of each compartment.

7 Playing with graphs 1/2

We will first state the following properties:

$$\begin{aligned} \text{Even} \times \text{Even} &= \text{Even} \\ \text{Even} \times \text{Odd} &= \text{Even} \\ \text{Odd} \times \text{Odd} &= \text{Odd} \end{aligned}$$

Statement. For a k-regular undirected graph, where k is odd, the total number of edges m must be divisible by k.

Proof. Let $v \in \mathcal{V}$, $\deg(v) = k \in \mathbb{N}$ for any v, k is odd, $|\mathcal{E}| = m \in \mathbb{N}$, $|\mathcal{V}| = n \in \mathbb{N}$. Since the graph is k-regular, every node v connects to k edges. Every edge connects to 2 nodes. Therefore:

$$m = \frac{nk}{2} \leftrightarrow 2m = nk.$$

Since $2m$ is divisible by two, it is even, and this implies the product nk is even. Since k is odd, n must be even such that the product is even as stated before. Therefore by definition, n is integer divisible by 2 and:

$$\frac{m}{k} = \frac{n}{2} \in \mathbb{N}$$

Therefore, m must be divisible by k. □

If k were even, we would have the intermediate result nk is even. Since k is even, n can be either even or odd. Therefore, the condition that n must be even is not necessary.

8 Playing with graphs 2/2

Let E_λ denote the eigenspace associated with eigenvalue λ of a matrix A. Then:

$$E_\lambda = \ker(A - \lambda I).$$

And trivially, $\dim(E_\lambda) = \dim(\ker(A - \lambda I))$. We also call $(A - \lambda I)$ the eigenproblem of A for a fixed λ. Finally, the algebraic multiplicity of an eigenvalue λ is also equal to $\dim(E_\lambda)$. An eigenvalue λ and its algebraic multiplicity m can be written as $(\lambda)^m$ shorthand.

Let $\mathbb{E}_n$ be the $n \times n$ matrix containing all 1s, that is, $\mathbb{E}_{i,j} = 1,\ i, j \in (1, n)$. Since all rows and columns are identical, the rank of $\mathbb{E}_n$ is 1. Therefore, the kernel has dimension $n - 1$. Therefore, it has eigenvalue 0 with algebraic multiplicity $n - 1$.

Since the sum over the diagonal of a matrix is equal to the sum of the spectrum of the matrix,

$$\sum \Lambda(\mathbb{E}_n) = \mathrm{tr}(\mathbb{E}_n) = n \implies \sum^{n-1} 0 + \lambda_n = n$$

the last eigenvalue must be n. Therefore the spectrum of $\mathbb{E}_n = \{(n)^1,\ (0)^{n-1}\}$.

Let $\mathcal{C}(n)$ be a complete graph with n nodes. Then by definition, $A_{i,j} = 1,\ (i \neq j) \in \{1, 2, ..., n\}$. We can therefore represent $A_{\mathcal{C}(n)}$ as

$$A_{\mathcal{C}(n)} = \mathbb{E}_n - I \tag{31}$$
$$= \begin{bmatrix} 0 & 1 & \cdots & & 1 \\ 1 & 0 & & & \\ \vdots & & \ddots & & \vdots \\ & & & & 1 \\ 1 & & \cdots & 1 & 0 \end{bmatrix}$$

Note that the matrix $\mathbb{E}_n$ is associated with the eigenvalue problem of the matrix A for $\lambda = -1$:

$$\mathbb{E}_n = (A - (-1)I).$$

Since $\mathbb{E}_n$ has an $(n-1)$-dimensional kernel, A has eigenvalue $(-1)^{n-1}$. Then again, $\sum \Lambda(A) = \mathrm{tr}(A) = 0$. Therefore the last eigenvalue is $n-1$, and the spectrum of $A_{\mathcal{C}(n)} = \{(-1)^{n-1},\ (n-1)^1\}$.

We can also relate $L_{\mathcal{C}(n)}$ to $\mathbb{E}_n$:

$$\begin{aligned} D = \mathrm{diag}(A\mathbb{1}) \quad &= (n-1)I \\ L = D - A \quad &= (n-1)I - (\mathbb{E}_n - I) \\ = nI - \mathbb{E}_n & \end{aligned}$$

Therefore, $\mathbb{E}_n$ is associated with the eigenvalue problem of L for the eigenvalue of n:

$$\mathbb{E}_n = (nI - L) = -1 \cdot (L - nI).$$

Note that we can scale with a constant and keep the kernel intact. Following previous arguments, L has eigenvalue n with multiplicity $n-1$, and via the trace, the final eigenvalue is 0. Therefore the spectrum of $L = \{(n)^{n-1},\ (0)^1\}$. This is also because $L\mathbb{1} = D\mathbb{1} - A\mathbb{1} = 0$, therefore 0 is an eigenvalue of L.

$$L_{\mathcal{C}(n)} = \begin{bmatrix} n-1 & -1 & \cdots & & -1 \\ -1 & n-1 & & & \\ \vdots & & \ddots & & \vdots \\ & & & & -1 \\ -1 & & \cdots & -1 & n-1 \end{bmatrix} \tag{32}$$

References

[1] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, p. 163–177, 2001.

[2] H. Kavitz, "Queue and stack classes," Dec 2011. [Online]. Available: https://nl.mathworks.com/matlabcentral/fileexchange/33876-queue-and-stack-classes

[3] K. Åström and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010. [Online]. Available: https://books.google.nl/books?id=cdG9fNqTDS8C