Network Dynamics and Learning

Homework 2

Alessandro Ferì (s305949) Mattia Sabato (s305849)
Marco Tasca (s285174) Riccardo Moroni (s320002)

December 17, 2023

Abstract

The following report is the result of the collaborative work between Alessandro Ferì, Mattia Sabato, Marco Tasca and Riccardo Moroni. When convenient, some of the numerical results have been omitted from this document and reported only in the accompanying Notebooks.

Exercise 1

The first assignment regards the study of a single particle performing a continuous-time random walk in the network described by the graph in Figure 1 and with the following transition rate matrix:

$$\Lambda = \begin{pmatrix} 0 & 2/5 & 1/5 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 & 1/3 & 0 \end{pmatrix}$$

1. *What is, according to the simulations, the average time it takes a particle that starts in node b to leave the node and then return to it?*
2. *How does the result in 1. compare to the theoretical return-time* $\mathbb{E}_b[T_b^+]$*?*
3. *What is, according to the simulations, the average time it takes to move from node o to node d?*
4. *How does the result in 3. compare to the theoretical hitting-time* $\mathbb{E}_o[T_d]$*?*
5. *Interpret the matrix* Λ *as the weight matrix of a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$*, and simulate the French-DeGroot dynamics on* $\mathcal{G}$ *with an arbitrary initial condition* $x(0)$*. Does the dynamics converge to a consensus state for every initial condition* $x(0)$*?*

6. *Assume that the initial state of the dynamics for each node $i \in \mathcal{V}$ is given by $x_i(0) = \xi_i$, where $\{\xi_i\}_{i\in\mathcal{V}}$ are independent random variables with variance*

$$\sigma_a^2 = \sigma_b^2 = \sigma_c^2 = 1, \quad \sigma_o^2 = \sigma_d^2 = 2$$

Compute the variance of the consensus value, and compare the results with the numerical simulations.

7. *Remove the edges (d, a) and (d, c). Describe and motivate the asymptotic behaviour of the dynamics. If the dynamics converges to a consensus state, how is the consensus value related to the initial condition $x(0)$? Assume that the initial state of the dynamics for each node $i \in \mathcal{V}$ is given by $x_i(0) = \xi_i$, where $\{\xi_i\}_{i\in\mathcal{V}}$ are i.i.d. random variables with variance σ^2. Compute analytically the variance of the consensus value.*

8. *Consider the graph $(\mathcal{V}, \mathcal{E}, \Lambda)$, and remove the edges (c, b) and (d, a). Analyse the French-DeGroot dynamics on the new graph. In particular, describe how the asymptotics behaviour of the dynamics varies in terms of the initial condition $x(0)$, and motivate your answer.*

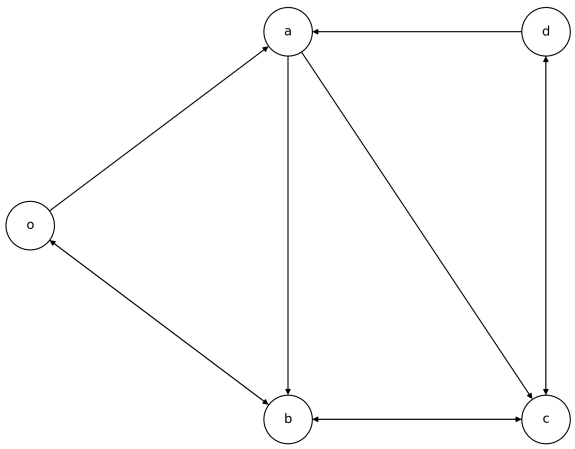


Figure 1

Exercise 1.1

When considering a single particle performing a continuous-time random walk, modeled by a continuous-time Markov chain $X(t)$, it proves necessary to consider a *Poisson process*. Even though randomness is core, it is still possible to compute many interesting deterministic insights providing useful information. For what is required in the next exercises, let us define now $\omega = \Lambda\mathbb{1}$, where Λ is the transition-rate matrix, and $P = diag(\omega)^{-1}\Lambda$, imposing self-loops whenever a node has no out-neighborhood. Moreover, let us define $\mathcal{G}_\Lambda = (\mathcal{X}, \mathcal{E}, \Lambda)$ to be a weighted directed graph having as the set of nodes the state space $\mathcal{X}$ of the Markov chain, representing the position of the particle at a given instant of time t, and as the weights

the transition rate matrix Λ; trivially, $(i,j) \in \mathcal{E} \iff \Lambda_{ij} \neq 0$. Once in state i, the particle waits a certain amount of time and moves to the adjacent state j if and only if the rate-Λ_{ij} exponential random variable corresponding to link (i,j), and representing a waiting time, ticks before any other exponential random variable associated to edges $\{(i,k) \in \mathcal{E} \mid k \in \mathcal{N}_i\}$. In the following, by denoting with r a generic rate, and by considering a uniformly distributed random variable $u \sim \mathcal{U}(0,1)$, we simulate the waiting time as

$$t_{next} = \frac{-\ln(u)}{r}$$

Notice that once in state i, each edge $\{(i,k) \in \mathcal{E} \mid k \in \mathcal{N}_i\}$ will be characterized by a $t_{ij,next}$, and the particle will move to state j for which $t_{ij,next}$ is minimum. In order to compute the empirical average return time $\bar{T}_b^+$, we have simulated 50.000 random walks, all starting from node b. Then, the first time the particle returns back to the initial position, the random walk stops and we store the corresponding total time needed. Finally, we compute the empirical average of such values considering all the simulation, which results to be 4.60 t.u.

Exercise 1.2

When comparing the empirical average return time with the expected one, it proves necessary to first compute $\bar{\pi}_i(t) = \mathbb{P}(X(t) = i), \forall i \in \mathcal{X}$, representing the time-t marginal probability distribution for the continuous-time Markov chain $X(t)$. Notice this distribution is in general different from $\pi(t) \xrightarrow{d} \pi$, as $t \to \infty$, from the discrete-time case, so handy formulas as Kac's one cannot be used. Nevertheless, we are still able to leverage several results from a theorem that applies whenever $\mathcal{G}_\Lambda$ is strongly connected, as in our case. First of all, there exists a unique Laplace-invariant probability distribution $\bar{\pi}(t)$ that satisfies $L'\bar{\pi} = 0, \mathbb{1}'\bar{\pi} = \mathbb{1}$. Moreover, for any initial probability distribution $\bar{\pi}(0)$, $\bar{\pi}(t) \xrightarrow{d} \bar{\pi}$, as $t \to \infty$. Then, for any node, the expected return times is given by

$$\mathbb{E}_i[T_i^+] = \frac{1}{\omega_i \bar{\pi}_i}, \quad i \in \mathcal{X} \tag{1}$$

In order to compute such a value for node b, we have first solved the convex homogeneous system regarding the Laplace-invariant distribution using the library `CVXPY`, thus obtaining $\bar{\pi}$. We have then simply computed the expected theoretical value using (1), which gave us exactly 4.60 t.u, according to numerical simulations.

Exercise 1.3

Differently from **Exercise 1.1**, we now want to compute the average hitting time to reach node d when starting from node o. Nevertheless, an identical line of reasoning applies here too, with the only exception that now the random walk stops whenever the particle reaches node d. Also in this case, we have run 50.000 experiments, recorded the time needed at each simulation, and computed the average value which is resulted to be equal to 10.79 t.u.

Exercise 1.4

An additional result concerning hitting times, provided by the theorem mentioned in **Exercise 1.2**, claims that the expected hitting times

$$\bar{\tau}_i^{\mathcal{S}} = \mathbb{E}_i[T_{\mathcal{S}}], \quad i \in \mathcal{X}$$

are the unique solutions of the following system of equations

$$\bar{\tau}_s^{\mathcal{S}} = 0, \quad s \in \mathcal{S}, \qquad \bar{\tau}_i^{\mathcal{S}} = \frac{1}{\omega_i} + \sum_{j \in \mathcal{X}} P_{ij} \bar{\tau}_i^{\mathcal{S}}, \quad i \in \mathcal{X} \setminus \mathcal{S}$$

By defining $\mathcal{S} = \{d\}$, and by removing from P the row and the column corresponding to d, we have solved the following linear system of equations using `CVXPY`, where we introduce the notation ω^{-1} to refer to a vector whose components are the reciprocal of ω after having excluded the one associated with d

$$\bar{\tau}^{\mathcal{S}} = \omega^{-1} + P\bar{\tau}^{\mathcal{S}}$$

Finally, the component $\bar{\tau}_o^{\mathcal{S}}$, which corresponds to the theoretical hitting-time $\mathbb{E}_o[T_d]$, is resulted to be equal to 10.77 t.u., in line with the numerical simulations.

Exercise 1.5

Let us now interpret the transition rate matrix Λ of the previous exercises as the weight matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$. We are interested in studying the *French-DeGroot* dynamics, considering an arbitrary initial condition $x(0)$ representing the initial opinions of the agents, modeled as nodes, and its evolution over time. In our convention, we assume that edges (i, j) represents the fact that i has access to the information of j, and can thus eventually update his one. We recall here that, whenever a strongly connected graph has just one sink component, which is also aperiodic, then the opinion dynamics model is guaranteed to converge, and, in particular, the following holds

$$x(t) \xrightarrow{t \to \infty} \alpha \mathbb{1}, \quad \alpha = \pi' x(0) \tag{2}$$

This fundamental result tells us that, no matter what the initial opinions are, whenever the aforementioned conditions are satisfied the opinion dynamics will converge to a convex combination of the initial opinions weighted by the corresponding value of the invariant distribution, the last reported in the following

$$\pi = [0.16, 0.19, 0.27, 0.23, 0.15]$$

A graphical representation of the convergence is shown in Figure 2.

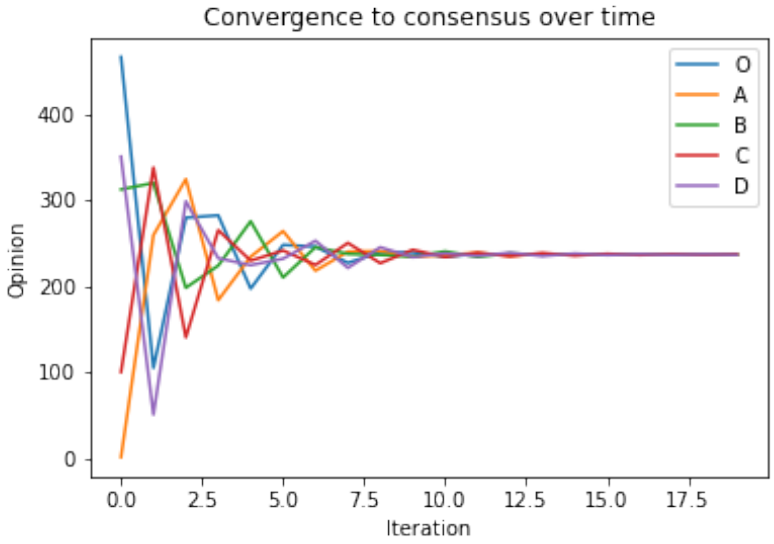


Figure 2: Node opinion vs Simulation Step

Exercise 1.6

In the context of *social learning*, we usually assume that each individual has a certain initial opinion $x_i(0)$ defined as follows

$$x_i(0) = \theta + \xi_i$$

where θ represents the true unknown state, and ξ_i represents a zero-mean random variable representing noise in the opinion of each agent. Considering only the scalar consensus value $\bar{x}$ toward which all the agents will eventually converge to, we are able to rewrite (2) as

$$\bar{x} = \sum_{k \in \mathcal{V}} \pi_k x_k(0) = \theta + \sum_{k \in \mathcal{V}} \pi_k \xi_k$$

If we had the same independent ξ_i for every agent, computing the variance of the consensus would be immediate, resulting to be

$$\sigma_{\bar{x}}^2 = \sigma^2 \sum_i \pi_i^2 \tag{3}$$

However, in our case, we have a slightly different situation, where the initial opinion of each agent is a random variable ξ_i, with a certain non uniform variance $\mathbb{V}[\xi_i]$. In particular, notice that $\mathbb{V}[\xi_i] = \mathbb{V}[\xi_j]$ only when considering certain couples of nodes, so (3) cannot be directly used. Nevertheless, by leveraging fundamental probability properties, we have the following

$$\mathbb{V}[\bar{x}] = \mathbb{V}\left[\sum_i \pi_i \xi_i\right] = \mathbb{V}\left[\pi_o \xi_o + ... + \pi_d \xi_d\right] = \pi_o^2 \mathbb{V}[\xi_o] + ... + \pi_d^2 \mathbb{V}[\xi_d] =$$
$$= 1(\pi_a^2 + \pi_b^2 + \pi_c^2) + 2(\pi_o^2 + \pi_d^2) = 0.26$$

which results to be equal to the variance computed through numerical simulations.

Exercise 1.7

Once established the convergence for the network in Figure 1, we now want to study the asymptotic behaviour of the dynamics associated to graph $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}}, \Lambda)$, with $\bar{\mathcal{E}} = \mathcal{E} \setminus \{(d,a),(d,c)\}$. Notice that the erased edges were the ones that prevented d to become a sink component, which actually now plays a crucial role in determining the consensus. In particular, after having added a self-loop to d, his opinion will represent the limit toward which the state of all the other agents will converge. This is due to (2), in which we claimed that the invariant distribution plays a paramount importance when determining a consensus, if any, starting from the initial conditions. Let's recall that, given a graph, there are as many linearly independent invariant distributions as sinks in the condensation graph, and each of them has support only over the nodes belonging to sink component considered. In our case, having just one sink component, which is moreover composed by only one node, it comes with no news that the unique invariant distribution will simply be $\pi = [0,0,0,0,1]$. Trivially, by computing (2), we have

$$\alpha = \pi' x(0) = \begin{bmatrix} 0,0,0,0,1 \end{bmatrix} \begin{bmatrix} x_o(0) \\ x_a(0) \\ x_b(0) \\ x_c(0) \\ x_d(0) \end{bmatrix} = x_d(0)$$

The last result tells us that every agent, after a certain amount of iterations, will end up having an opinion equal to the initial opinion of node d. If we further assume that each agent has an initial opinion equal to *i.i.d.* random variables ξ_i, the variance of the consensus can simply be computed with (3). However, since, again, π is non-zero only for nodes belonging to the sink component, the variance of the consensus will correspond to the variance of the opinion of node d.

Exercise 1.8

After having verified that the dynamics does indeed converge also when removing certain edges, we now want to check whether this is also the case when considering a new graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \Lambda)$, with $\mathcal{E}' = \mathcal{E} \setminus \{(c,b),(d,a)\}$. Notice the network is again characterized by a sink component $\mathcal{S}' = \{c,d\}$, so one would expect to again find a consensus which is a convex combination of the initial opinions of such nodes. However, the lack of aperiodicity in the sink component causes a weird behaviour in the evolution of the opinions, which keeps severely oscillating. This is due to the fact that agents in the sink component will keep copying each other's opinion in a never ending loop, with no chance to find an agreement. Since c and d are the only ones that could eventually play a role in determining the consensus, all the other nodes are not able to reach an equilibrium as well. This means that, no matter what are the initial conditions, in the graph $\mathcal{G}'$ it is not possible to reach a consensus if the agents belonging to the sink component do not already start at one. A graphical represention of such severe oscillations is shown in Figure 3.

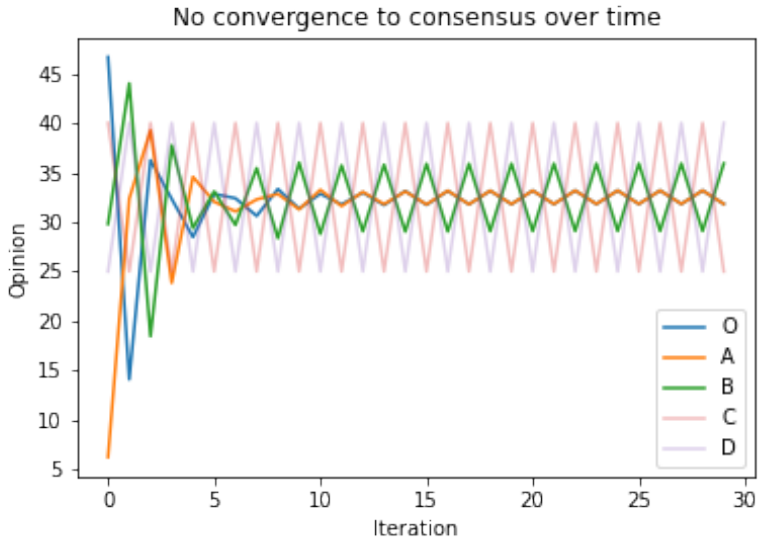


Figure 3: Node opinion vs Simulation Step

Exercise 2

In this part we will again consider the network of Figure 1, with weights provided by the associated Λ. However, now we will simulate many particles moving around in the network in continuous time. Each of the particles in the network will move around just as the single particle moved around in Problem 1: the time it will stay in a node is exponentially distributed, and on average it will stay $1/\omega_i$ time-units in a node i before moving to one of its out-neighbors. The next node it will visit is based on the probability matrix $P = diag(\omega)^{-1}\Lambda$, where $\omega = \Lambda\mathbb{1}$. Your task is to simulate this system from two different perspectives: the *particle perspective*, i.e. "follow the particle", and the *node perspective*, i.e. "observe from the node". Simulating the system from a particle perspective is exactly as in Problem 1, but here you have to follow many particles instead. To simulate the system from the node perspective you instead have to observe the particles from the node. When doing this you do not have to care about each single particle, but only about the number of particles in the node. Note that at node i, each particle in that node will stay there on average $1/\omega_i$ time units. Therefore, the node will pass along particles at a rate proportional to the number of particles in the node. In fact, if at time t the number of particles in node i is $n_i(t)$, it will pass along particles at a rate of $n_i(t)\omega_i$. The departure times of the node can thus be seen as a Poisson process with rate $n_i(t)\omega_i$. At each tick of the Poisson clock of the node, it will move a particle to a neighboring node. The node to which the particle will move is again based on the normalized transition rate matrix P. Simulate the system from the two perspectives, and then answer the following questions:

1. **Particle perspective:**

   - *If 100 particles all start in node b, what is the average time for a particle to return to node b?*

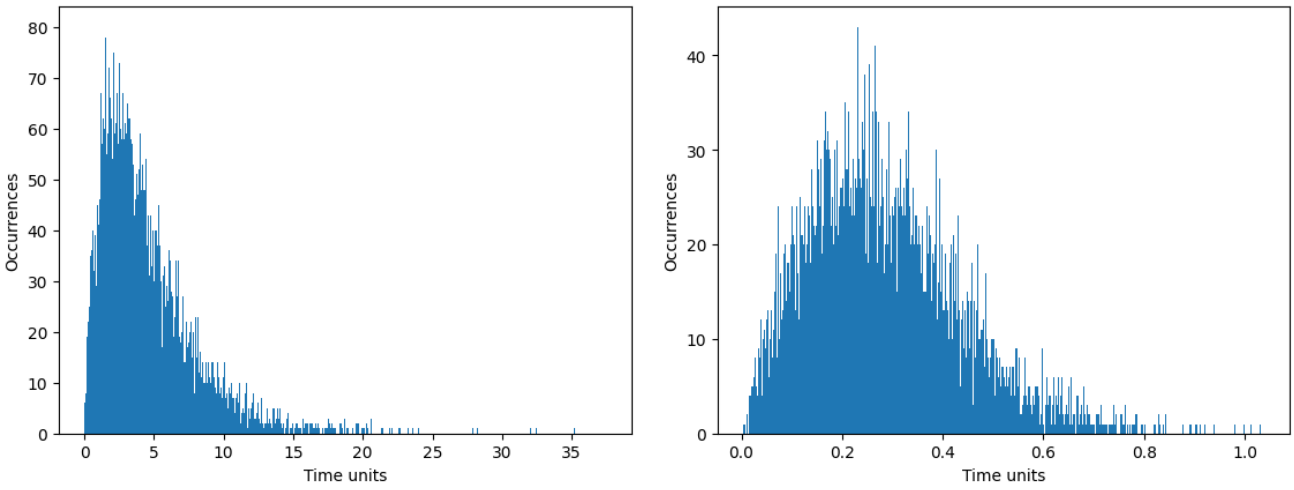
- *How does this compare to the answer in Problem 1, why?*

2. **Node perspective:**

   - *If 100 particles start in node o, and the system is simulated for 60 time units, what is the average number of particles in the different nodes at the end of the simulation?*
   - *Illustrate the simulation above with a plot showing the number of particles in each node during the simulation time*
   - *Compare the simulation result in the first point above with the stationary distribution of the continuous-time random walk followed by the single particles.*

Exercise 2.1

In order to compute the average time for a particle to return to node b, having a total of 100 particles, we proceeded in a way similar to the previous exercises, imposing Poisson clocks over the edges. In particular, we have run 100 simulations, each composed of 100 1-particle random walks, and we have stored for each the return times. Then, for every simulation, we have kept just the lowest value, i.e. the first particle to return, and we have computed the average over all simulations, which is resulted to be 0.286 t.u. Notice how this value is sensibly smaller than the results of **Exercise 1.1-1.2**. This is due to the fact that we are no more computing the average return time for a single particle, but instead we are averaging only the left-most points of the time distributions associated to each simulation, as shown in Figure 4.



(a) T_b^+ distribution of 10k particles.

(b) T_b^+ distribution of the first arrivals of 10k simulations.

Figure 4: Comparison of the two distributions we are computing the average of in, respectively, Exercise 1.1 and in Exercise 2.1 .

Exercise 2.2

We are requested now to find the average number of particles in each node for a single simulation, consisting of 100 particles performing a random walk starting from o and lasting 60 t.u. To do so, we have changed the way Poisson clocks were managed and we ended up trying four different implementations. In this report we discuss only the one responsible for the results we are going to mention, while the other three can be found extensively commented in the Notebook. The consistency of the results obtained from all four implementations is assessed in the Appendix at the end of the Notebook, as well as their efficiency in terms of execution time.

Since the goal is now to observe the system with an higher level of abstraction, in the implementation we are discussing (for reference: `particles_in_nodes_local_poisson()`, which also happens to be the fastest one), a Poisson clock is placed on every node. Once the network has been initialized to its starting state, every node will tick at a certain time; the minimum tick time belongs to the node from which a particle will move, while the node it will jump to is assessed as in the exercises above. Every time a particle jumps, all the ticking times have to be updated by subtracting from them the minimum ticking time (this is how we move forward in time). This is done because all the Poisson clocks are independent from each other, so they never see the other nodes ticking and simply continue with their ticking time. But since the Poisson clocks depends on the amount of particles they host, the rates of the Poisson clocks on the nodes implied in the exchange need to be updated, specifically: the clock on the sender will be reset with the new (slower) rate, while the tick time of clock on the receiver (to be intended as the amount of time units to wait for it to tick) will be updated to the minimum between the previous tick time (already updated by subtracting the sender tick time) and the time to stay in the node for the new particle. The result is identical to simply updating the rate of the node, and then recalculating the ticking time according to the new exponential distribution.

Any time a clock ticks we store the updated network state and the ticking time.

To empirically find the average number of particles in each node at the end of the simulation we did 1.000 simulations and we computed the average. The result is shown in Table 1. The dynamic of a single simulation is represented in Figure 5, while the average number of particles in each node during simulation time can be seen in Figure 6.
We believe Figure 6 to be the most informative since it not only shows what the asymptotic stationary distribution $\bar{\pi}$ is, but gives also an idea about how fast each node in the network gets there. Finally, by recalling here that $\bar{\pi}$ can also be computed analytically starting from the Laplacian matrix, as already done previously in **Exercise 1.2**, we confront the last with the empirical one computed in our experiments. It arises that the two appear to be consistent, as shown in the following

Node	Average n. of particles
o	21.6
a	14.9
b	26.1
c	18.6
d	18.8

Table 1

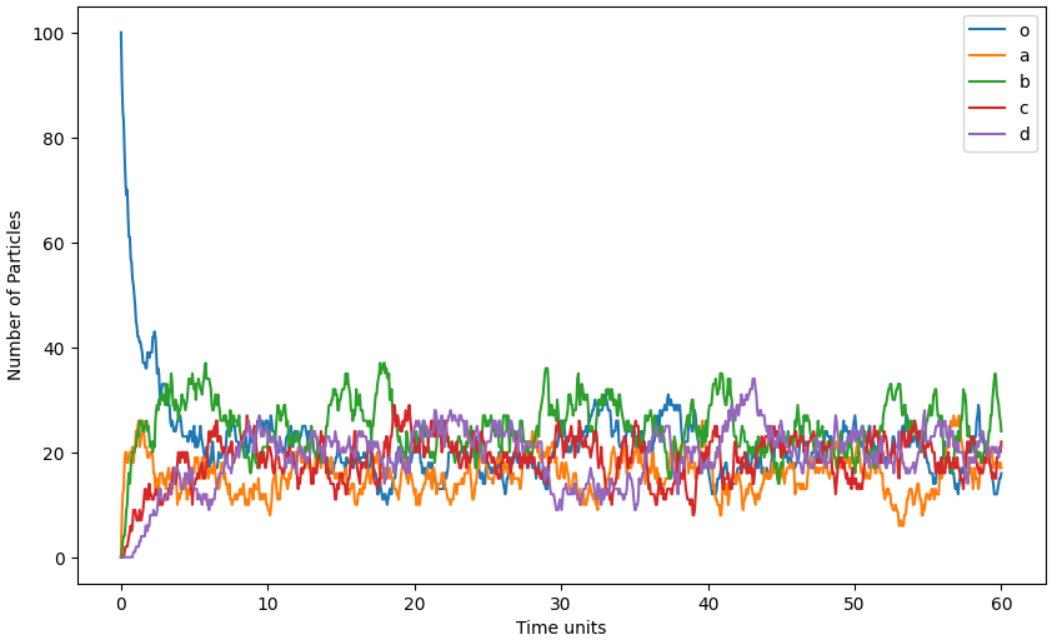


Figure 5: Number of particles in nodes during a single simulation

$$\bar{\pi} = [0.217, 0.149, 0.261, 0.186, 0.186]$$
$$\bar{\pi}_{\text{empirical}} = [0.216, 0.149, 0.261, 0.186, 0.188]$$

Notice the components of $\bar{\pi}$ does not add up to 1 only because of rounding errors.

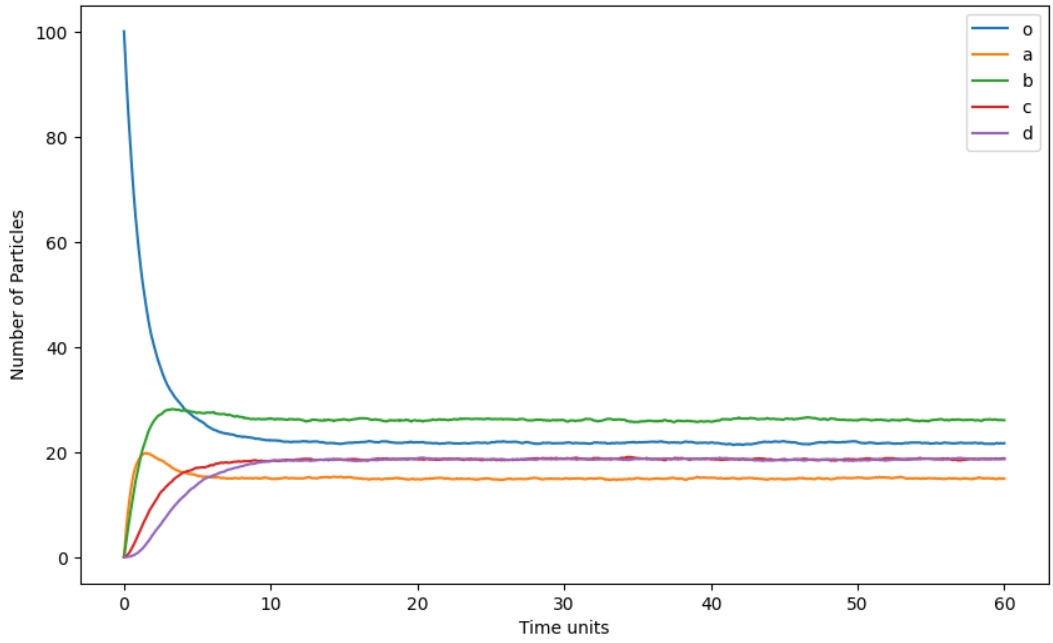


Figure 6: Average number of particles in nodes during simulation time over 1.000 simulations of 100 particles and 60 time units each.

Exercise 3

In this part we consider the open network defined by the following transition rate matrix

$$\Lambda = \begin{pmatrix} 0 & 3/4 & 3/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For this system, particles will enter the system at node o according to a Poisson process with input rate λ. Each node will then pass along a particle according to a given rate, similar to what you did in Problem 2 with the "node perspective". Let $\omega = \Lambda\mathbb{1}$ and let $N(t)$ denote the vector of number of particles in each node at time t.
You will simulate two different scenarios that differ by what rate the nodes will pass along particles: i) *proportional rate*, and ii) *fixed rate*. In scenario i), each node i will pass along particles according to a Poisson process with rate equal to the number of particles in the node times the rate of the local Poisson clock of node i, i.e., the node i will pass along particles rate with rate $\omega_i N_i(t)$. In scenario ii), each node i will instead pass along particles with a fixed rate ω_i.
Since node d does not have a node to send its particles to, we assume that $\omega_d = 2$. When the Poisson clock of node d ticks, you could simply decrease the number of particles in the node by one (if there are any particles in the node). Equivalently think of another node d'

connected to node d, such that at every tick of the Poisson clock of d, it sends a particle to node d'. The goal is to simulate the two systems and answer the following questions:

1. **Proportional rate:**
   - *Simulate the system for 60 time units and plot the evolution of the number of particles in each node over time with input rate* $\lambda = 100$.
   - *What is the largest input rate that the system can handle without blowing up?*
2. **Fixed rate:**
   - *Simulate the system for 60 time units and plot the evolution of the number of particles in each node over time with input rate* $\lambda = 1$.
   - *What is the largest input rate that the system can handle without blowing up? Motivate your answer.*

Exercise 3.1

In order to simulate the system, having, for each node, a rate proportional to the number of particles at each time t, we have introduced two additional nodes to the original graph: a node s, with only o as out-neighbor for particles intake, and a node d', with only d as in-neighbor representing a sink component with a self-loop serving as an exit node. We then conducted the simulations adapting the rate of each node i through $\omega_i N_i(t)$, where $N_i(t)$ represents the number of particles in node i at time t. As illustrated in Figure 7, the system has shown to be resilient and able to manage the flow within the network without blowing up and reaching convergence. This is due to the fact that we can model the number of

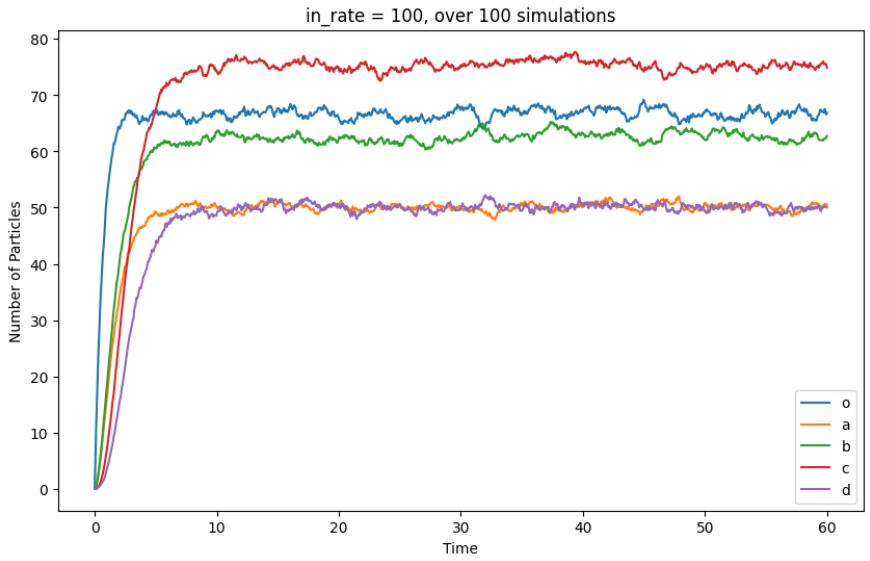


Figure 7: Simulation with a proportional rate

particles i in a node as a continuous-time birth and death chain, for which i represents the

state, λ represent the arrival rate and μ represents the departure rate. Defining $\rho = \frac{\lambda}{\mu}$, the stationary probability distribution $\bar{\pi}$ proves then to be:

$$\bar{\pi}_i = \frac{\rho^i / i!}{\sum_{j=0}^{n} \rho^j / j!}, \quad i = 0, 1, \dots$$

Finally, notice how, thanks to Erlang's formula, we can also compute the probability of finding the queue full as

$$\bar{\pi}_n = \frac{\rho^n / n!}{\sum_{j=0}^{n} \rho^j / j!}$$

By observing that $\bar{\pi}_n \overset{n \to \infty}{\longrightarrow} 0$, we have a theoretical confirmation that the system will never blow up.

Exercise 3.2

Differently from the previous exercise, we now consider the rate of each node to be fixed and not dependent on $N_i(t)$. Moreover, we consider a global rate-ω^* Poisson clock, with ω^* being the largest ω_i. When considering an input rate $\lambda = 1$ the system does not blow up, as shown in Figure 8, where we plot the average number of particle per node over 1.000 simulations. Notice that this behaviour is retained even when increasing the time.

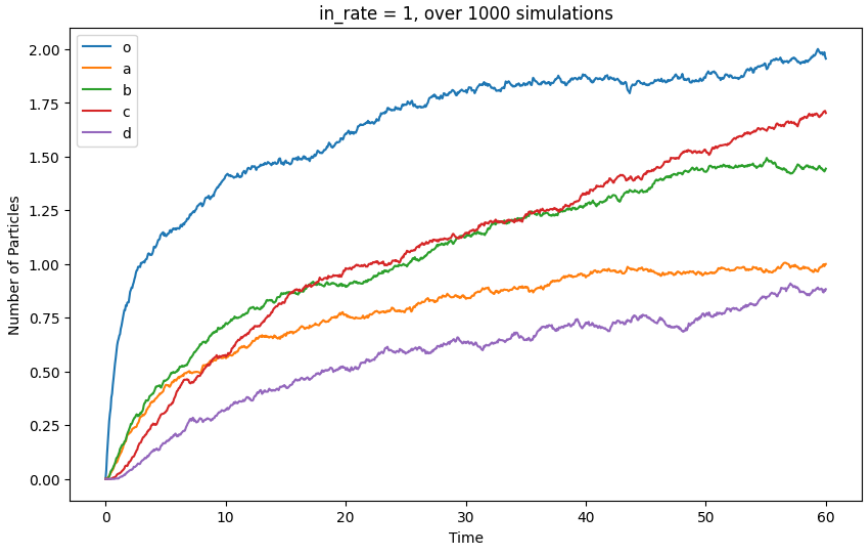


Figure 8

In order to establish what is the largest input rate the system can handle, it is necessary to focus on the arrival and departure rate of each node, as before modeled as continuous-time birth and death chains. Notice the system trivially blows up when $\lambda > 1.50$, being greater than the departure rate of the input node o. Once established a first upper-bound, we are interested in studying whether every smaller value is allowed. Analyzing the graph,

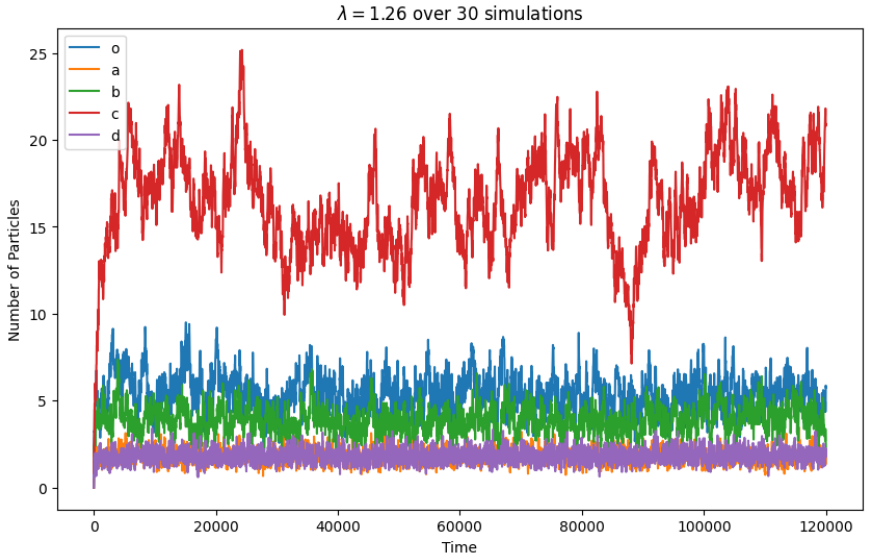


Figure 9

we immediately notice how node c is the only one characterized by an arrival rate greater than its departure rate. Being $\mu_c = 1.00$ and $\lambda_c = 1.25$, one could naively assume that for λ equal or greater than the last, but also for any values greater than 1, the system will actually blow up. However, this line of reasoning is only partially correct since we are assuming λ to be directly connected with c, which is not the case. Having some of the nodes more than one outgoing edge, we can expect just a percentage of the original input rate λ to be considered as the arrival rate of the nodes as we go deeper in the graph. Considering probability, we can compute the arrival rate of each node as just a portion of the input rate. Accordingly, we study the expected arrival rate $\mathbb{E}[\lambda_i]$ for each node to deduce their respective limitations. In the following, let us consider P_{ij} to be the probability, given node i, that a particle will move to node j, proportional to the rate Λ_{ij}. At a given instant of time, we may assume the expected arrival rate of a node and its departure rate to be computed as

$$\mathbb{E}[\lambda_j] = \sum_{i \in \mathcal{N}_j^-} P_{ij} \mathbb{E}[\lambda_i], \qquad \mu_j = \omega_j$$

with the convention that $\mathbb{E}[\lambda_o] = \lambda$. In this way, we are able to compute, for each node, how much in expectation the input rate impacts on the arrival rate. Notice, however, these are just expectations and so, as further confirmed by the experiments, fluctuations may have a severe impact on the results of each simulation.

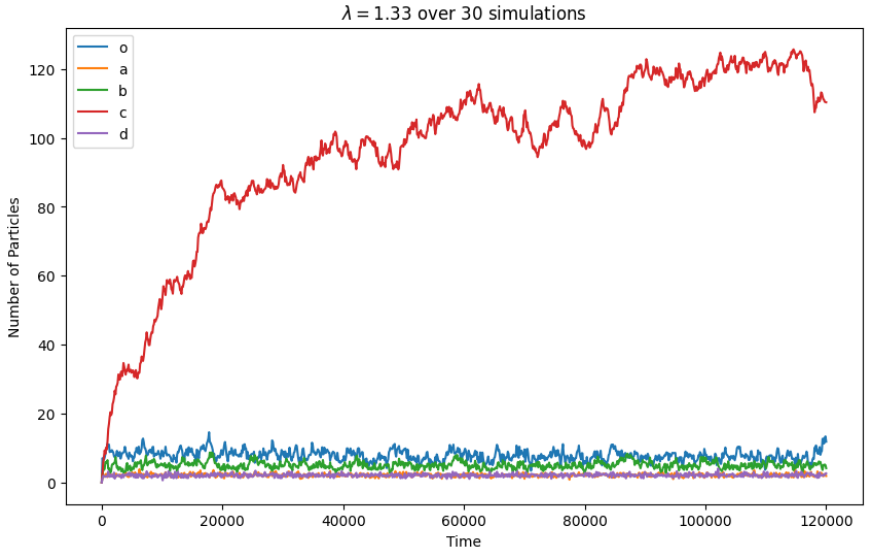


Figure 10

In order to find the bottleneck node of the graph, which is more likely to make the system blow up, we have computed the expected values as follows

$$\begin{aligned}
\mathbb{E}[\lambda_o] &= \lambda & \lambda_o \leq \mu_o \Rightarrow \lambda \leq \frac{3}{2} \\
\mathbb{E}[\lambda_a] &= \frac{1}{2}\mathbb{E}[\lambda_o] = \frac{1}{2}\lambda & \lambda_a \leq \mu_a \Rightarrow \lambda \leq 2 \\
\mathbb{E}[\lambda_b] &= \frac{1}{2}\mathbb{E}[\lambda_o] + \frac{1}{4}\mathbb{E}[\lambda_a] = \frac{5}{8}\lambda & \lambda_e \leq \mu_e \Rightarrow \lambda \leq \frac{8}{5} \\
\mathbb{E}[\lambda_c] &= \mathbb{E}[\lambda_b] + \frac{1}{4}\mathbb{E}[\lambda_a] = \frac{6}{8}\lambda & \lambda_c \leq \mu_c \Rightarrow \lambda \leq \frac{8}{6} \\
\mathbb{E}[\lambda_d] &= \mathbb{E}[\lambda_c] + \frac{1}{2}\mathbb{E}[\lambda_a] = \lambda & \lambda_d \leq \mu_d \Rightarrow \lambda \leq 2
\end{aligned}$$

Notice the strongest constraint is related to node c, for which $\lambda \leq \frac{8}{6} \approx 1.33$. Whenever λ is close to 1.33 the network is more likely to congest, with quick and unavoidable divergence for values above it. Instead, for smaller values, the system seems to be resilient and able to digest the particles without creating bottlenecks and uncontrolled accumulations. In order to test all our previous claims, we have run several simulations over 120.000 t.u. for different values of λ. As can be seen in Figure 9, where we considered $\lambda = 1.26$, the system does not blow up, confirming the hypothesis it could manage a rate $\lambda > \omega_c$.

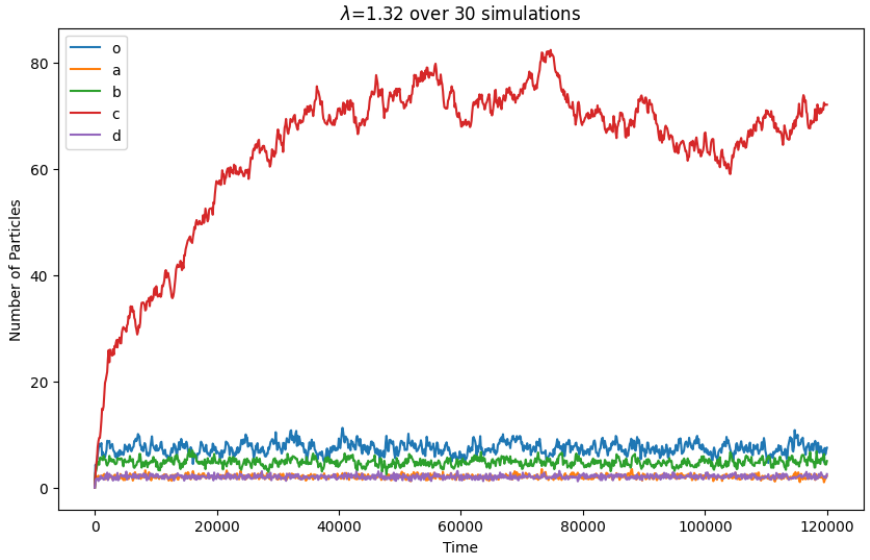


Figure 11

Whenever, instead, we fix a rate close or equal to 1.33, particles are more likely to accumulate in node c, as shown in Figure 10. Notice how, when taking $\lambda = 1.34$ as in Figure 12, where we plot also some single simulations along with the average, the system blows up very quickly and keeps diverging with much more probability. Instead, for smaller values, as in Figure 11, the system is characterized by a number of particles in node c that seems to stabilize after a while; this behaviour is expected to be more evident as the input rate decreases further.

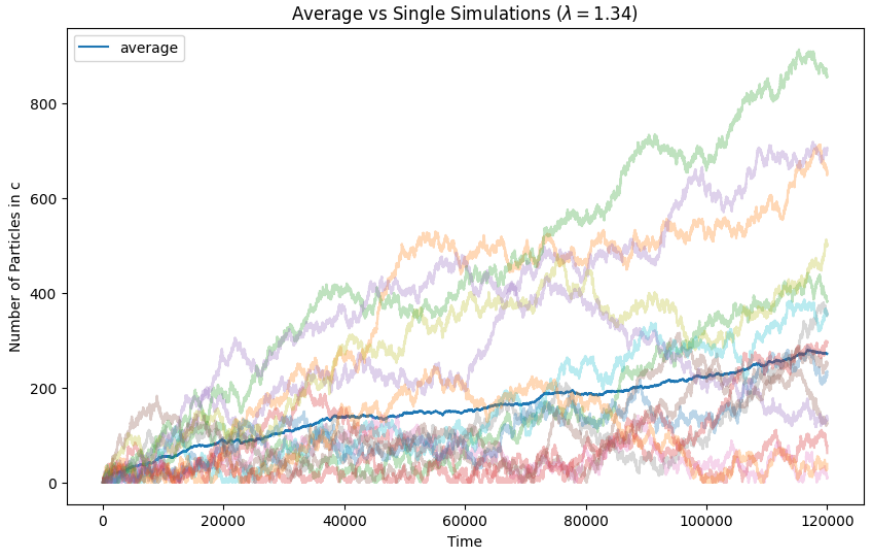


Figure 12