Network Dynamics: Homework II

Andrea Silvi
Politecnico di Torino
andrea.silvi@studenti.polito.it

1. Problem 1

In this first problem we study the trajectory of a single particle performing a continuous time random walk in the network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$ described by the transition rate matrix

$$\Lambda = \begin{array}{c} \\ o \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} o & a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 2/5 & 1/5 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 & 1/3 & 0 \end{pmatrix} \end{array} \tag{1}$$

1.1. Problems 1.a, 1.b

In order to estimate the return time of a particle starting from node a, defined as the first time when the process, starting from a, comes back to it after having left it for some time, we simulate the process for 1000000 steps and we keep track of how many times the particle comes back to a. Then we can just divide the total time (up to the last step of the particle being in a) and divide by this counter to obtain the average return time to a. We obtain a value of $\widehat{T}_a^+ \simeq 6.75332$. This is because the process, being Markovian, whenever it comes back to node a, is as if it was brand new, so it is like we are simulating a bunch of shorter processes each of which ends when the particle comes back to a.

If we want to compare it with the theoretical value of $\mathbb{E}_a[T_a^+]$, we can observe firstly that since the chain is strongly connected and the state space is finite, then a unique invariant distribution $\overline{\pi}$ exists. We know from Kac's formula that

$$\forall i \in \mathcal{V}, \mathbb{E}_i[T_i^+] = \frac{1}{\omega_i \overline{\pi}_i} \tag{2}$$

where $T_i^+ = \inf\{t \geq 0 : X(t) = i \text{ and } X(s) \neq i \text{ for some } s \in (0,t)\}$, $\overline{\pi}$ is the stationary distribution of the jump chain computed as $\overline{\pi} = \overline{P}'\overline{\pi}$, with

$$\begin{aligned} \overline{P}_{ij} &= \frac{\Lambda_{ij}}{\max_i \omega_i}, i \neq j \\ \overline{P}_{ii} &= 1 - \sum_{j \neq i} \overline{P}_{ij}, \end{aligned} \tag{3}$$

and $\omega = \Lambda \mathbb{1}$.

When we compute these values we get that $\overline{\pi}_a \simeq 0.14815$ and $\omega_a = 1$, so finally we get that $\mathbb{E}_i[T_i^+] \simeq 6.75$, which is fairly close to what we get from the simulations.

1.2. Problems 1.c, 1.d

We now want to estimate the hitting time from node o to node d, defined as the first time when the process, starting from o, reaches node d. This time we simulate 100000 times a random walk over graph $\mathcal{G}$ which starts from node o and ends as soon as it reaches d. We then average the different simulation times and get $\widehat{T}_{o,d} \simeq 8.78142$.

If we want to compare this estimate with the theoretical expected hitting time from o to d, we can define this as

$$v_o = \mathbb{E}_o[T_d] = \frac{1}{\omega_o} + \sum_{j \in \mathcal{Q}} \hat{P}_{oj} \mathbb{E}_j[T_d], \tag{4}$$

where $\mathcal{Q} = \mathcal{V} \setminus \{d\}$ and $\hat{P}$ is the transition matrix P of the jump chain restricted to the nodes in $\mathcal{Q}$. We then use this formula to define a vector v such that $v_i = \frac{1}{\omega_i} + \sum_{j \in \mathcal{Q}} \hat{P}_{ij} v_j$, $\forall i \in \mathcal{Q}$ and we rewrite the formula from (4) in vector form, getting

$$v = \frac{\mathbb{1}}{\hat{\omega}} + \hat{P}v, \tag{5}$$

and since $(\mathbb{I} - \hat{P})$ is invertible because $\hat{P}$ is strictly substochastic since $\{d\}$ is globally reachable, so not all rows of $\hat{P}$ sum up to 1, we solve the linear system as

$$v = (\mathbb{I} - \hat{P})^{-1} \frac{\mathbb{1}}{\hat{\omega}}. \tag{6}$$

From the calculations we obtain that $v_o = v(0) = \mathbb{E}_o[T_d] \simeq 8.78571$, which again is fairly close to the estimate we get from the simulations.

1.3. Problem 1.e

We now interpret the matrix Λ as the weight matrix of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$ and simulate the French-DeGroot dynamics on $\mathcal{G}$. We set the initial condition $x(0) = [1, 0, 0.3, 0.7, 0.18]^T$ and simulate for 100000 iterations the dynamics, defined as

$$x(t) = Px(t-1), \tag{7}$$

with P defined as the normalized weight matrix of $\mathcal{G}$. Note that the way the model is defined means that at each time step a node obtain its value as the average of the values of its outgoing neighbours. We know that $\mathcal{G}$ is strongly connected and node o is aperiodic, since $per_{\mathcal{G}}(o)$ is the greatest common divisor of the lengths of all circuits starting and ending in o, and two of these are $o \to a \to b \to o$ and $o \to b \to o$, so $per_{\mathcal{G}}(o) = 1$. Since $\mathcal{G}$ is strongly connected and o is aperiodic, we also know that $\mathcal{G}$ is itself aperiodic. Finally we know from theory that the dynamics, given aperiodicity and strong connectivity of $\mathcal{G}$, converges to the consensus value $\pi' x(0)$, where π is the invariant distribution of the normalized weight matrix P.

From all the previous point we know that the dynamics should converge to a consensus value $\pi' x(0) \simeq 0.4226$. From the simulation we get that after all the iterations, $x \simeq [0.4226, 0.4226, 0.4226, 0.4226, 0.4226]^T$, confirming our expectations.

1.4. Problem 1.f

We now want to observe what is the effect of agents with noisy initial conditions on their consensus value. We define $\mu = \frac{1}{2}$ as the true value and for each simulation we sample the initial errors vector η from a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.2$. We then define $x(0) = \mu \mathbb{1} + \eta$. So the agents know at the start only a noisy approximation of the true value μ.

We simulate again the French-DeGroot dynamics for 500 steps and do 1000 simulations of this model. For each simulation we save the errors squared as $e = (\bar{x}(t) - \mu)^2$ with $\bar{x}(t) = \sum_{i \in \mathcal{V}} x_i(t)/|\mathcal{V}|$ that coincides with the consensus value $\pi' x(0)$ if t is large enough that the dynamics has reached consensus.

We then compute the estimate of the consensus variance $\hat{\tau}^2$ as the average of all the squared errors e and we obtain $\hat{\tau}^2 \simeq 0.04298$, which is a lot lower than the variance $\sigma^2 = 0.2$ of the single agents: the linear averaging dynamics leads to an estimation of the true state μ which is strictly better than the original estimations of each node.

We can obtain this value also theoretically by considering the fact that we know that the dynamics, because of aperiodicity and strong connectivity of $\mathcal{G}$, converges to

$$q = \pi' x(0) = \sum_{i \in \mathcal{G}} \pi_i (\mu + \eta_i) = \mu + \sum_{i \in \mathcal{G}} \pi_i \eta_i \tag{8}$$

and the variance of this random variable q is

$$\sigma_q^2 = \sigma^2 \sum_{i \in \mathcal{G}} \pi_i^2. \tag{9}$$

Computing this value gives us $\sigma_q^2 \simeq 0.04272$, which is fairly close to the estimated variance.

1.5. Problem 1.g

We now define a new graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \Lambda')$, with $\mathcal{E}'$ obtained by removing edges (d, a) and (d, c) from $\mathcal{E}$, and consequently Λ' equal to the one defined in (1), except for the last row which is now the vector $\mathbf{0}$. This results in the state d being now a trapping state. The condensation graph of $\mathcal{G}'$ now results in the sink $\{d\}$ and the source $\mathcal{V} \setminus \{d\}$. We know that whenever a graph has one globally reachable component which is also aperiodic (and $\{d\}$ obviously is being a single node component), then the French-DeGroot dynamics converges to $\pi' x(0)$.

Note that since the invariant distribution π has support only on the trapping component, then the consensus value is a convex combination of the initial values $x_i(0), \forall i$ in the trapping component. In our case, since the trapping component is comprised of node d only, we expect the consensus value to be

$$\lim_{t \to \infty} x_i(t) = \pi' x(0) = x_d(0), \forall i. \tag{10}$$

Indeed, the simulation of the dynamics on graph $\mathcal{G}'$ shows that every node converges to the initial opinion of node d.

If we define again the same problem as in (1.4), with agents having noisy initial conditions of a real value μ, we can observe that using the formula defined in (9), and recalling that the invariant distribution is defined only on the trapping component (in our case $\{d\}$), then the variance of the crowd will be equal to the original variance of the noises of the agents. This is to be expected since we know that only the initial condition of node d has an effect on the consensus value, so the consensus value will be as wrong as the initial noisy opinion of node d is. The estimated variance obtained through the simulations also confirms these calculations.

1.6. Problem 1.h

We now reconsider the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$ and we define $\mathcal{E}''$ by removing from $\mathcal{E}$ the edges (c, b) and (d, a), obtaining a new graph $\mathcal{G}'' = (\mathcal{V}, \mathcal{E}'', \Lambda'')$ with the weight matrix defined as

$$\Lambda'' = \begin{array}{c} \\ o \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} o & a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 2/5 & 1/5 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 2/3 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix} \end{array} \tag{11}$$

We can see from the last two rows of Λ'' that now the nodes c and d form a trapping component (d can only reach c and vice versa). The set c, d is also globally reachable and the sole sink of the condensation graph of $\mathcal{G}''$ (with $\mathcal{V} \setminus \{c, d\}$ being the only source). Unfortunately if we analyze the sink component $\{c, d\}$, we can easily see from Λ'' that the trapping component is periodic, since $per_{\mathcal{G}''}(c) = per_{\mathcal{G}''}(d) = 2$. Then we know that the dynamics will never reach convergence. The simulations show that even after 1000000 iterations, the dynamics has not reached a consensus value. In particular it is interesting to note that, since the nodes c and d are influenced only by one another, this results in c getting at each time step the opinion of d at the previous time step and vice versa, continuously oscillating between their two initial conditions.

2. Problem 2

In this second problem we consider again the graph $\mathcal{G}$ defined in Section 1 with transition rate matrix Λ defined in (1). This time we consider a process consisting of 100 particles moving around the network in continuous time. We consider two different ways of looking at this stochastic process:

- from the *particle perspective*, where we follow each particle trajectory;
- from the *node perspective*, where we do not care about the single particles but instead we focus on the evolution of the number of particles in each node over time.

2.1. Problem 2.a

We firstly analyze the evolution of 100 particles starting from node a from the particles perspective, and we want, similarly to Section (1.1), to understand which is the average time it takes to the particles to come back to node a. In order to simulate this process, we consider one global Poisson Clock which ticks after a time which is distributed as an exponential random variable with rate 100, representing the time of the winner (more specifically the one that moves first) of the exponential race between the 100 particles. Then one of the 100 particles is randomly selected to be the one which is moving, and this then moves according to the i-th line of the probability matrix $\overline{P}$, where $\overline{P}$ is defined as in (3) and i is the node the chosen particle was in before the clock ticked.

We simulate this process up to time 60 for 10000 times and keep track of the return times to a of all the particles. We then average all the return times to get an estimate of the return time to node a. We obtain an estimated value

$$\widehat{T}_a^+ = \sum_{j=1}^{10000} \sum_{i=1}^{100} t_a^{+(i,j)} \simeq 6.74737, \qquad (12)$$

where $t_a^{+(i,j)}$ is the observed return time of particle i in the j-th simulation. As we can see, if we compare this result with the ones from Section 1.1, we get very similar results. This is because each particle is not influenced by the other 99 and moves independently inside the same network as in (1.1). In fact a simpler way of simulating this problem could have been to simulate the trajectory of each particle separately, which would have been the same as done in (1.1), and then average the return times.

2.2. Problem 2.b

We now follow the process from the nodes perspective, and we want to track what the average number of particles for each node at the end of a run is. Note that runs are again 60 time units long. We simulate 1000 runs by using one single Poisson Clock again with rate 100, but this time we get a node i (and not a particle) from which 1 particle is going to move from by picking it randomly, proportionally to the number of particles in the nodes at that time. We then subtract 1 from the number of particles in i. Then one particle moves from i to a node j, randomly chosen according to i-th row of the transition matrix $\overline{P}$, and we add 1 to the number of particles in j. Note that if a particle does not move (which can happen whenever $i = j$, which happens only if $\overline{P}_{i,i} > 0$), then we just add and subtract 1 from the number of particles in i, thus obtaining no change whatsoever.

We can observe that since the graph $\mathcal{G}$ is strongly connected, if we define a vector ϕ that represents the distribution of the particles over the nodes at each time step, we know that $\phi(t) \xrightarrow{t\to\infty} \overline{\pi}$, $\forall \phi(0) = p$, where in our case

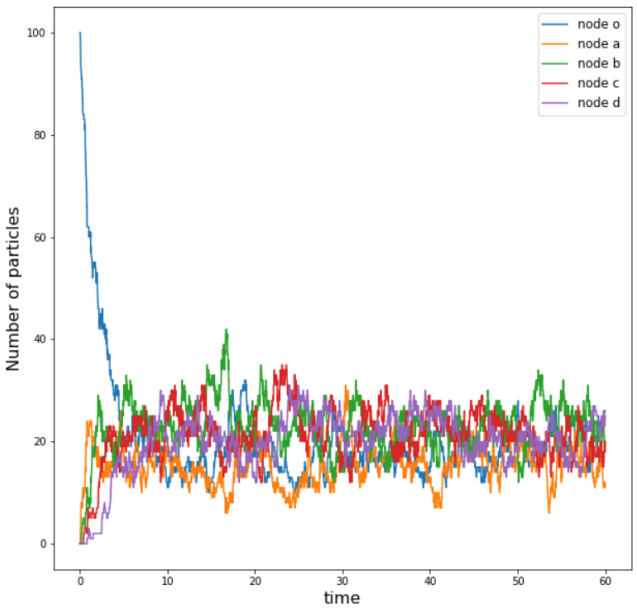

Figure 1: The evolution of the number of particles in each node over time, starting with 100 particles in node *o*.

$p = \delta^{(o)}$. So the normalized average number of particles in each node after the simulations should be fairly close to the stationary distribution $\overline{\pi}$, and indeed we obtain very close values, more specifically

$$\overline{\pi} \simeq [0.1852, 0.1481, 0.2222, 0.2222, 0.2222]^T \quad (13)$$

and by defining the estimator of the distribution

$$\hat{\overline{\pi}} = \frac{\sum_{i=1}^{1000} \phi(60)}{1000} \quad (14)$$

we get as a result from the simulations

$$\hat{\overline{\pi}} \simeq [0.1847, 0.1468, 0.2228, 0.2219, 0.2238]^T.$$

Note that we consider 60 units of time long enough to reach stationarity. We also do a large amount of simulations to get a more accurate estimate of the invariant distribution.

Finally in Figure 1 we plot the evolution of the number of particles in each node over time for one simulation. We can see that indeed they stationarize around the values of the invariant distribution multiplied by 100 (the number of particles).

3. Problem 3

In this last problem we consider how the movement of particles inside an open network affect each others. We simulate the process over the network described in Figure 2. As we can see the particles can enter the system in node o and can exit it through node d. A new transition matrix Λ is defined as

$$\Lambda = \begin{array}{c} \\ o \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} o & a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 2/4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (15)$$

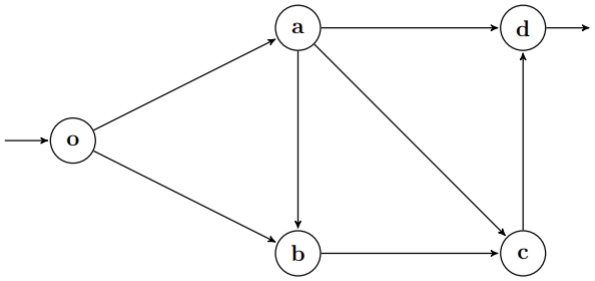

Figure 2: The open network.

In order to simulate the entrance of particles in the system from node o, we create a fictitious node $\bar{o}$ that is connected to o only and always has 1 particle available in it.

We can thus augment Λ in order to fit node $\bar{o}$ into the chain:

$$\Lambda_{aug} = \begin{array}{c} \\ \bar{o} \\ o \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccccc} \bar{o} & o & a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1/4 & 2/4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (16)$$

With this matrix we can then obtain $\overline{P}$ by using the same formula presented in (3).

We consider two scenarios that differ in the rate at which nodes pass particles along the system:

- *proportional rate*, where each node pass particles according to a Poisson process of rate proportional to the number of particles in it;
- *fixed rate*, where each node pass particles with a fixed rate of 1.

Note that the fictitious node $\bar{o}$ instead always fires particles with a fixed rate, which we call *input rate*. Whenever this node fires a particle in the system, we do not decrease its number of particles, which stays at 1 forever. Instead, whenever particles move from node d, we just decrease the number of particles in node d by 1.

3.1. Problem 3.a

We start by having nodes firing at a rate proportional to the number of particles inside them. As in the previous problems, in order to simulate an exponential race to decide at what time a node is going to fire first between the nodes that have at least 1 particle, we use 1 global Poisson Clock with a variable rate, which depends on the number of particles inside the system. In particular, the global rate at time t is

$$\gamma(t) = \lambda + n(t) \quad (17)$$

where $\gamma(t)$ is the global rate at time t, λ is the input rate (in this case equal to 1), and $n(t)$ is the number of particles inside the network at time t (not considering the fictitious node $\bar{o}$). In this way, we simulate an exponential race between the particle that can enter the system in o with rate $\lambda = 1$ and the other particles that are in the network.

After obtaining from the Poisson Clock the time at which the first particle moves, we choose randomly which node i fires a particle, proportionally to the number of particles in them, except for node $\bar{o}$ which can be randomly chosen proportionally to the input rate λ: whenever $\lambda > 1$, the

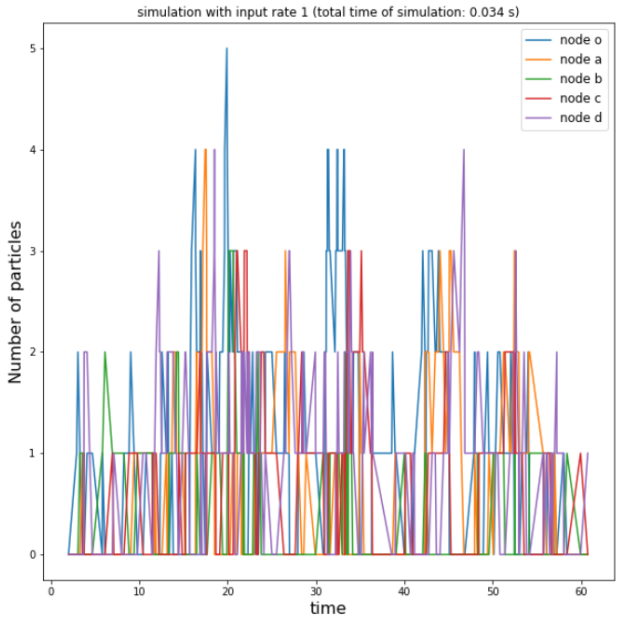

Figure 3: The evolution of the number of particles in each node of the open network over time, with node rates proportional to the number of particles in them, and input rate $\lambda = 1$. We also report the amount of time in seconds it took to complete the simulation.

particles enter the system at a higher and higher rate, so the probability of node $\bar{o}$ of firing into o increases with λ.

Then as usual the particle from the node i chosen moves randomly to another node, proportionally to the i-th row of matrix $\overline{P}$.

We simulate the process for 60 units of time and plot the evolution of the number of particles in the different nodes over time in Figure 3. We can see that the system is well balanced, since particles do not get stuck in the network.

We then try to increase the input rate, to see if this behaviour changes. Note that the simulation works exactly the same as before, but with a different input rate λ in both the calculation of the global rate defined in (17) and in the random choice of the node from which a particle moves.

The results for different values of λ are reported in Figure 5. As we can see, the number of particles in the system never blows up, but is consistent with the input rate λ: this is due to the fact that the nodes inside the network fire with rates which are proportional to the number of particles inside them, so the increasingly higher and higher input rate pumps more and more particles in the network, and these increasingly higher number of particles make the rates of the nodes higher and higher, thus making the system not break even at very high input rates. It is also very interesting to note, especially with the last two graphs of Figure 5, how the number of particles in each nodes tend to stationarize around different values, which suggest an inherent invariant distribution in this network to which the distribution of particles tend to stationarize. This can be considered possible because, even though the network is not strongly connected, actually whenever a particles exits from node d it is as if it returns to node $\bar{o}$, which has a fixed rate equal to λ, and so the graph can actually be considered somewhat connected and can have a stationary distribution.

3.2. Problem 3.b

We now consider the same problem as before, but with nodes firing at constant rates all equal to 1. We start again with an input rate $\lambda = 1$.

Now the global Poisson Clock is defined as

$$\gamma(t) = \lambda + \nu(t), \tag{18}$$

where $\gamma(t)$ and λ are again the global rate at time t and the input rate, while $\nu(t)$ is the number of active nodes, which are those nodes in which there is at least 1 particle. This is due to the fact that a node cannot participate in the exponential race which declares at which time the first node fires a particle, if it has no particles inside: for example, whenever no particles are in the system, we just have the node $\bar{o}$ that can fire at a rate λ, and since $\nu(\hat{t}) = 0$ in this case, $\gamma(t) = \lambda$.

The same thinking applies whenever we choose which node gets to fire the particle: we choose a node randomly between $\bar{o}$ and the nodes of the network that have at least 1 particle inside, proportionally to their rate, which are λ for node $\bar{o}$ and 1 for all the remaining active nodes. The nodes that are not active simply cannot be chosen. The node to

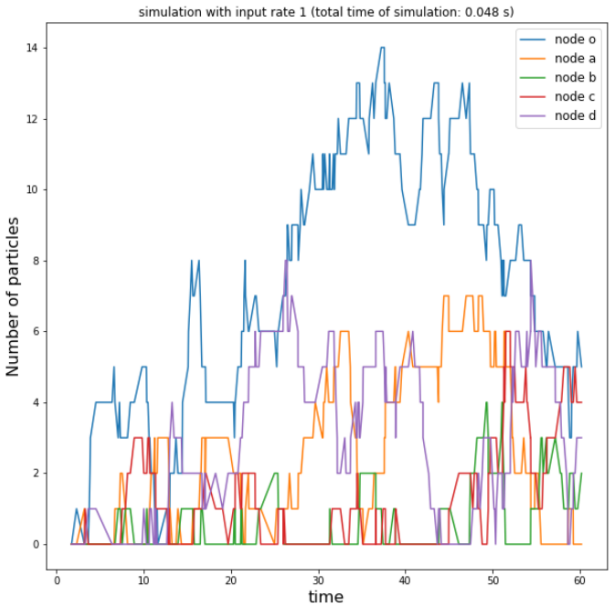

Figure 4: The evolution of the number of particles in each node of the open network over time, with fixed node rates equal to 1, and input rate $\lambda = 1$. We also report the amount of time in seconds it took to complete the simulation.

which a particle transitions is then chosen randomly again through the transition matrix $\bar{P}$.

Of course this time we expect the system to have a harder time at expelling the particles which are getting fired inside of it, but still with $\lambda = 1$ we expect the number of particles in the system not to blow up. The results are reported in Figure 4. Indeed we can see that some particles get stuck at some point in node o but the system is fast enough in eventually removing them and we get some kind of equilibrium between the number of particles entering and exiting the network.

If we try with higher input rates λ, now we expect the number of particles in the system to blow up. Indeed, the rates of the nodes being stuck at 1 means that a larger input rate is going to fire at a quick pace a lot of particles inside the network, while the rest of the network cannot keep up with it, resulting in a way higher number of particles entering the network with respect to the number of particles exiting it. The results are reported in Figure 6. Indeed already with input rate $\lambda = 2$ the system simply cannot keep up with the particles entering it in o and the number of particles inside it grows enormously. With λ increasing, the number of particles inside node o and inside the rest of the nodes differ by some orders of magnitude.

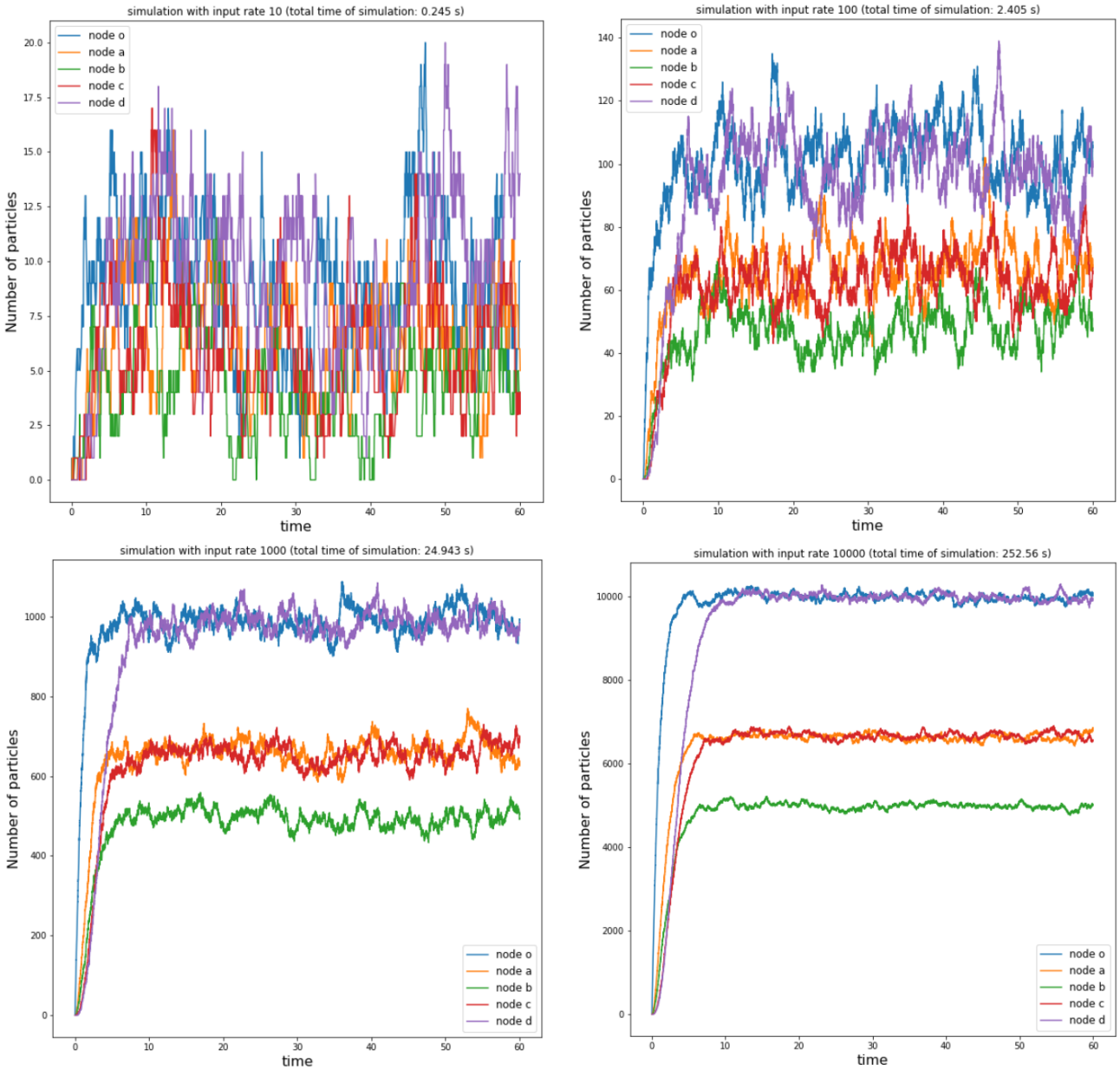

Figure 5: The evolution of the number of particles in each node of the open network over time, with node rates proportional to the number of particles in them, and different input rates $\lambda = \{10, 100, 1000, 10000\}$. We also report the amount of time in seconds it took to complete the simulations.

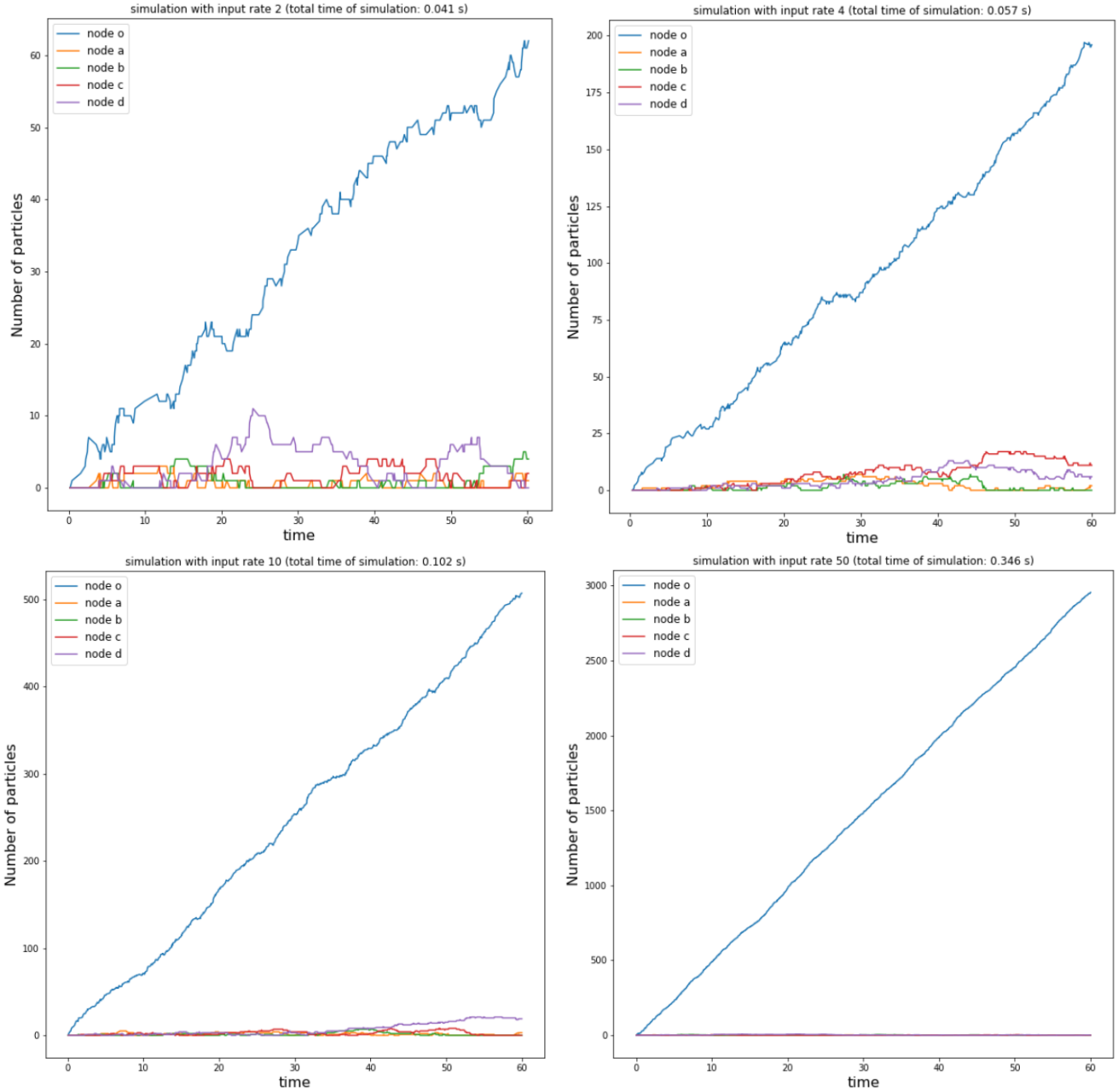

Figure 6: The evolution of the number of particles in each node of the open network over time, with fixed node rates equal to 1, and different input rates $\lambda = \{2, 4, 10, 50\}$. We also report the amount of time in seconds it took to complete the simulations.