Network Dynamics and Learning: Homework 2

Fabio Tatti
Politecnico di Torino
s282383@studenti.polito.it

1. Exercise 1

We have to simulate a particle moving in the network described from the matrix Λ that is the transition rate matrix.

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

1.1. Question A

We compute the expected returning time in node a for a particle in the network by simulation. We build a Poisson Clock class that returns the tick of the clock: this will be a random variable with rate r, returns a value

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

where u is a uniformly distributed random variable. This class is be implemented in order to simulate the time that the particle spends on the nodes of the graph: each time that the clock ticks, the particle will change node according the transition probability described by the matrix Λ. To get the distribution we will take the row relative to the current node, but, as we can see the current matrix Λ can not properly represent a distribution, since not all the rows sum up to 1. To fix this we will build the normalized adjacency matrix P, that, since it is normalized, can be used to describe the probability. We define a function *hit_ret* that takes as input the normalized adjacency matrix P, the starting node for the particle and the ending one. Since in this case we want the returning time for the node a origin and ending nodes will both be a. The simulation will stop each time the particle gets back to the node a, returning the total time - obtained by summing cumulatively t_{next} for each node - and the total number of steps that the particle took to return to the origin. This will result in having the simulation of just one run of the particle over the network, but we want the average returning time $\mathbf{E}_a[T_a^+]$, so we simulate it for $n\_iterations = 10^5$ times, a number big enough to converge given the simplicity of the graph. The resulting expected return time is

$$\mathbf{E}_a[T_a^+] = 6.758 \tag{1}$$

with a variability at each different run of the algorithm given by the aleatory nature of the clocks. We also can see that the average steps needed are 5.

1.2. Question B

We now compute the theoretical expected returning time $\bar{\mathbf{E}}_a[T_a^+]$ in order to compare it with the simulated value. To compute it we will need this equation:

$$\bar{\mathbf{E}}_a[T_a^+] = \frac{1}{\omega_a \bar{\pi}_a} \tag{2}$$

where ω_a is the a-th entry of the vector $\omega = \Lambda\mathbb{1}$ and $\bar{\pi}_a$ is

$$\bar{\pi}_a = \frac{\pi_a/\omega_a}{\sum_j \pi_j/\omega_j} \tag{3}$$

To obtain the vector $\bar{\pi}_a$ we exploit the equation

$$\bar{P}^T\bar{\pi} = \frac{\Lambda^T + diag(\omega_*\mathbb{1}) - diag(\omega))}{\omega_*}\bar{\pi} \tag{4}$$

since the $\bar{\pi}$ is the invariant distribution probability of the matrix $\bar{P}$ that we can obtain from the last equation 3. Moreover, as result of the Perron - Frobenious theorem, since the matrix $\bar{P}$ is stochastic and non-negative, it has a unique non-negative eigenvector and it is the related to the maximum value eigenvalue. This eigenvalue results being $\bar{\pi}$, that now we can obtain exploiting the iterative methods such as the power method. Finally, the result we get when computing the expected returning time theoretically is

$$\mathbf{E}_a[T_a^+] = 6.75 \tag{5}$$

As we can see the theoretical value well approximates the simulated one, and vice versa, obviously taking into account a little error given by the variations given by the Poisson Clocks.

1.3. Question C

We want to compute the expected hitting time by simulation for a particle starting in node o and hitting node d $\mathbf{E}_o[T_d]$. We exploit the same function of the *Question A hit_ret*, but this time we change the starting node and the ending one with respectively o and d. The function will work exactly as in the previous use: we get, for *n_iterations* $= 10^5$, an average hitting time

$$\mathbf{E}_o[T_d] = 8.765 \tag{6}$$

with a number of average steps taken equal to 7.

1.4. Question D

We compute the theoretical expected hitting time for particle starting in node o and hitting node d in order to compare it with the previously simulated one $\bar{\mathbf{E}}_o[T_d]$. To do it we solve the system $Ax = b$. The matrix A is

$$A = \mathbb{1} - P$$

where from the matrix P we exclude the d's row and column since is our ending point.Then

$$b = \frac{\mathbb{1}}{w}$$

where w is the vector containing the sum over the row of matrix Λ. We solve this linear system and then take the first value of the resulting vector x that will be the expected hitting time for a particle starting in o and hitting d.

1.5. Question E

We now build the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Lambda)$ in order to simulate the French-DeGroot dynamics on it, given an arbitrary initial condition $x(0)$. The result is shown in Figure 1.

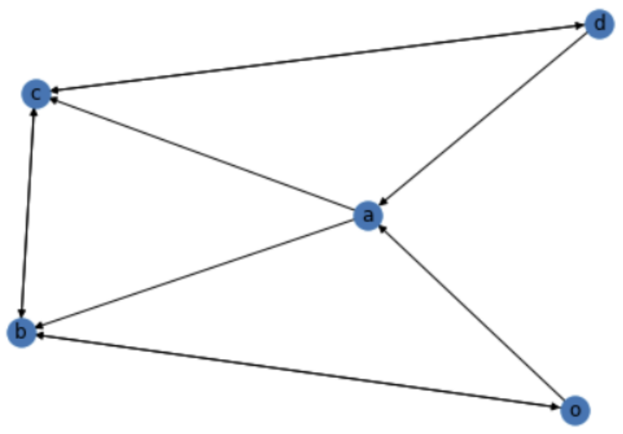


Figure 1: The graph built from Λ matrix for Exercise 1 Question E.

We used for the initial condition $x(0)$ a random vector that is normalized in order to properly represent a probability distribution. For this simulation we used $n\_iterations = 1000$: each of them will compute the product $P \times x$, where x is the vector on which we iterate and has initial state $x = x(0)$. This will converge such that all the entries of the vector will have the same numerical value. This means that the dynamics converges to a consensus state: this is consequence of the graph being strongly connected and aperiodic.

1.6. Question F

We assume that the initial state for each node i is given by $x_i(0) = \xi_i$, where each ξ_i is a random variable, independent from the others and identically distributed with variance σ^2. In this case we will use for generic ξ a normally distributed random variable, with the centre of the distribution $\mu = 0.5$ and variance $\sigma^2 = 0.5$, consequently the standard deviation is σ. We want to compare the variance of the nodes at the consensus state with the variance of the nodes at the initial states, both theoretically and by simulation. We first simulate it, computing many iterations in order to have a good approximation of the returned value since each time the initial vector $x(0)$ changes due to the aleatory nature of ξ. We simulate for $n\_iterations = 10'000$ and we get a variance in the consensus state

$$\sigma^2_{consensus} = 0.10915 \tag{7}$$

given an initial variance $\sigma^2_{init} = 0.5$ defined by us. To get the theoretical value we exploit again the eigenvectors and eigenvalue of the matrix P, in order to get the invariant vector π. The theoretical crowd variance will be equal to

$$\bar{\sigma}^2 = \sigma^2_{init}\pi \times \pi = 0.107 \tag{8}$$

We can see that the simulation results approximates quite precisely the theoretical result; moreover, if we want to improve the simulated value respect to the theoretical one, we can increase the number of iterations taken to compute it, that will, of course, also increase the computation time.

1.7. Question G

We now delete edge (d, a) and (d, c) from $\mathcal{G}$ getting the graph showed in Figure 2. This will result in a new transition rate matrix

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

We put 1 in the 5^{th} entry of the last row because now the node d behaves like a sink, so we can use the self loop to model that. As in the previous point, we are interested

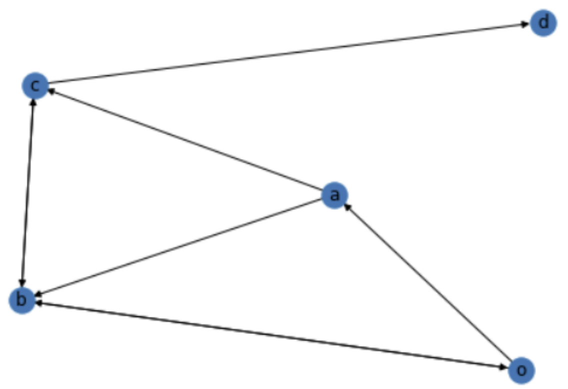


Figure 2: The graph built from Λ' matrix for Exercise 1 Question G.

in computing the variance of the consensus value, given the initial state vector such that $x(0)_i = \xi_i$ where ξ_i is a random variable uniformly distributed with variance σ^2. Again we perform $n\_iterations = 1000$ in order to simulate the dynamic consensus. As we can see in Figure 3, the whole system will converge to the consensus: we can notice by plotting the initial state versus the consensus state that all the nodes will converge to the initial value of node d. This is a straight consequence of the topology of the graph, since the node's d opinion remains unchanged over the iterations, while the rest of the graph, starting from node c, is influenced by it. Of course this will result in a consensus where only d's opinion exists over the graph. Then we want to compute the variance of the consensus value, we already showed how we do it. The result, for initial variance $\sigma^2_{initial} = 0.5$, is

$$\sigma^2_{consensus} = 0.492 \tag{9}$$

versus a theoretical value

$$\bar{\sigma}^2 = 0.5 \tag{10}$$

1.8. Question H

We remove from the graph described by matrix Λ edges (c, b) and (d, c) obtaining the transition rate matrix

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

```
The initial value of node d is 0.22675455316211743
The dynamics with 1000 iterations is [0.22675455 0.22675455 0.22675455 0.
22675455 0.22675455]
```

Figure 3: The result of the consensus in Question G.

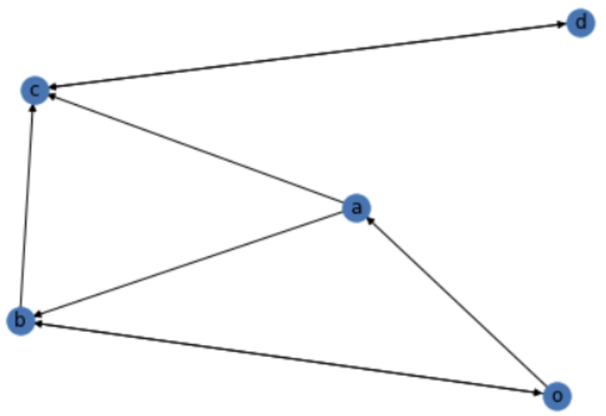


Figure 4: The graph built from Λ" matrix for Exercise 1 Question H

```
The initial distribution is [0.07317855 0.24472451 0.27009027 0.31214905
0.09985762]
The dynamics with 1000 iterations is [0.21416839 0.22233345 0.14884795 0.
31214905 0.09985762]
Mean of c - d initial values: 0.20600333572550894
```

Figure 5: The result of the consensus in Question H.

We simulate the dynamic consensus of the graph depending on the initial condition $x(0)$, initialized as a random and normalized distribution. What we get is a result of the type shown in Figure 5: given the initial distribution c and d will maintain same value (for a even number of iteration). This is a consequence of the fact they constitute a connected component where at each iteration the value held from the two nodes is being swapped, with no influence from outside the connected component. Moreover this partially explain why the other nodes of the graph have values that are near to the mean of the initial of c and d: they obtain, from node c the two values at alternate times. This means that the connected component provides for $n\_iterations \to +\infty$ a value equal to the mean between the initial values of c and d.

2. Exercise 2

Using the same graph of the previous Exercise, with transition rate matrix Λ, simulating multiple particles moving along its nodes.

2.1. Question A: Particle perspective

We compute the average time for a particle to return to the initial node a given a number of 100 particles all starting in the node at the same time. We simulated it with $n\_iterations = 10^3$, and, since we are simulating from the particle perspective, we used the same approach of the previous exercise, with the difference that in that case we do it for each particle. Since there is no interaction between particles we can say that what we are doing is equivalent, from the conceptual point of view, to simulate $n\_iterations = 10^5$ but for one particle only in the system. This is due to the fact that we can just compute the simulation iteratively, since the particles do not interact among them. What we get is

$$Average\_return\_time = 6.757 \qquad (11)$$

Compared with the answer from Exercise 1 Questions A-B we can see that the result well fits the concept expressed above: the two simulations are equivalent, and that is support by the numerical result, both simulated and theoretical. Even the average number of steps necessary to return to a are the same, that lead us to say that our concept well model this situation.

2.2. Question B:Node perspective

We simulate from the node perspective 100 particles, all starting in node o for 60 time units: what we first want to obtain are the average number of particles for at the end of the simulation. To do that, we build a new function *nodes_perspective* that accepts the matrix P, the number of particles, the origin node and the number of time units that the simulation has to last. The function will return a list of list object containing all the configuration that the system had along the simulation, ordered by time. This will allow us to also plot the chart of the number of particles per each node across the time as shown in Figure 6. To simulate from the node point of view we exploit a single Poisson clock for the whole network that will tick with a rate equal to the total number of particles, 100 in this case. This will return the time passed before that one of 100 present will pass to a different node according the transition probability matrix. To model this we select, for each tick of the clock, the node from which the particle will leave: since the total number of particles is fixed in time we can obtain easily the distribution of particles that represent, if normalized, the probability of each node of being selected. Then, according to the relative row in P, we select the node where the particle will move to. We simulate all this process for the given time of 60 units: all those steps are iterated for $n\_iterations = 10^3$ in order to get the average number of particles for each node at time 60 units. The results are showed in Table 1: as we can notice the values sum up to 99, meaning that, due to the rounding of the originally float numbers, it is like we have ”lost” a particle. Then we want to compare the results of our simulation with the stationary distribution of the continuous time random walk followed by the single particles $\bar{\pi}$. In order to properly compare the two arrays we store at each iteration the last configuration and then sum node-wise the number of particles obtained. Then we divide by the number of particles and finally normalize in order to obtain a distribution. As we can see in Figure 7 the results approximate the ones obtained in the previous Exercise, meaning that we managed to properly describe the Node perspective.

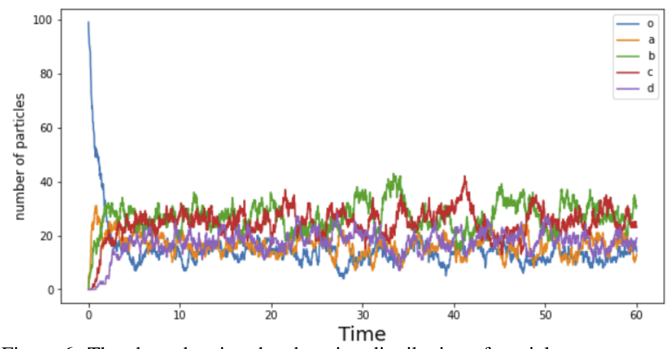


Figure 6: The chart showing the changing distribution of particles among the nodes of the network in a simulation, for Exercise 2 Question B.

3. Exercise 3

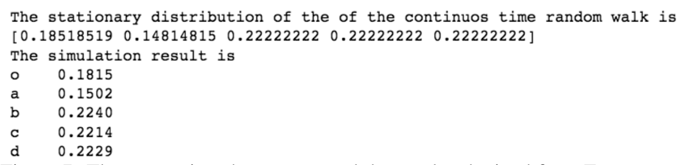
We consider an open network with transition rate matrix

$$\Lambda_{open} = \begin{bmatrix} 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 2/4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Each particle will enter the system from node o according to a Poisson clock with rate $\lambda = 1$. To simulate this we will consider two different scenarios: one with a *proportional rate* for the clocks of nodes, while the other one will have a *fixed rate*.

3.1. Question A: Proportional rate

We build a function $prop\_fixed\_rate$ that accepts as input the transition probability matrix P, obtained from matrix Λ_{open}, the rate of the input Poisson clock (in this case fixed to one), the total time units the simulation is going to last and a flag value *fixed* that has default value *False* (see next Question). Inside the function we set a fixed Poisson clock for the input rate: this will tick each time a particle


```
The stationary distribution of the of the continuos time random walk is
[0.18518519 0.14814815 0.22222222 0.22222222 0.22222222]
The simulation result is
o    0.1815
a    0.1502
b    0.2240
c    0.2214
d    0.2229
```
Figure 7: The comparison between $\bar{\pi}$ and the results obtained from Exercise 2 Question B

o	a	b	c	d	Tot
18	15	22	22	22	99

Table 1: Average number of particles per node at after 60 time units

enters the system from node o. The initial configuration of the system is empty, then, once at least one particle is inside, we take also in account the clocks of the nodes: each node will have its own clock, with a rate equal to the number of particles in that node (no tick for $n\_particles = 0$). The first clock that ticks among the nodes and input ones decides what will happen next to the network configuration. If first comes the input tick, we only have to add a new particle in node o. Instead, if one or more nodes tick before the input, the first one will lose a particle that will move to another node according to the relative row of the matrix P. This simulation modeling is possible thanks to the Markov property, that in this case allows us to take always the first ticking clock. The resulting chart is shown in Figure 8. In order to understand if the system can handle a bigger number of particles we try also with a series of bigger values for λ as shown in Figure 9, 10 and 11. As we can see, the graphs tell us that the system can handle any kind of number of particles (at least if we don't have a threshold maximum capacity associated to each node) because after an initial phase it sets around stable values. This is a consequence of the proportional rate: the more are the particles in, the more will be inside the nodes and higher will be the rates of the Poisson clocks of the nodes, meaning that more particles will move inside the open network. Since all the particles have to pass in node d to leave the network we see that in all the charts there is a relationship between it and the input node o, while all the others have a lower asymptotic value due to the topology of the graph.

3.2. Question B: Fixed rate

In this case the rate of the Poisson clock of each node has value one. We can still exploit the same function as before, $prop\_fixed\_rate$, just by passing a different value for the flag $fixed =$ **True**. This will change the value of the clocks of the nodes from proportional to fixed with value equal to one. The resulting plot in Figure 12 for $\lambda = 1$ shows that the system is less stable, due to the fixed rate of the nodes. This makes it more subject to overloads, as it is showed in Figures 13, 14 and 15: increase the in rate λ, even of a small number if compared to the proportional rate case, leads the system to overload. The reason is simple: the input clock ticks a number of times proportional to λ, while all nodes maintain rate one. As consequence, node o's number of particles will increase almost linearly in time, as shown in Figure 15 and higher λ will be, the more will be true.

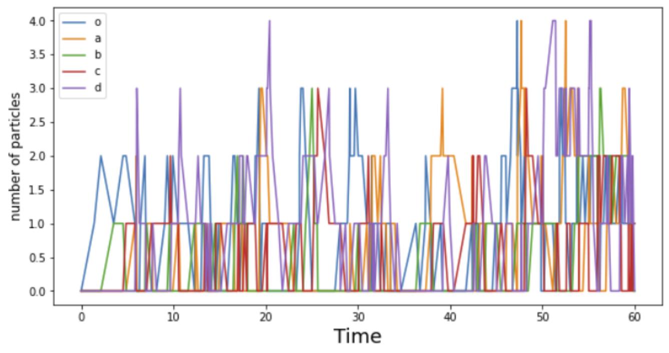


Figure 8: Number of particles for each node during the simulation, with proportional rate and $\lambda = 1$.

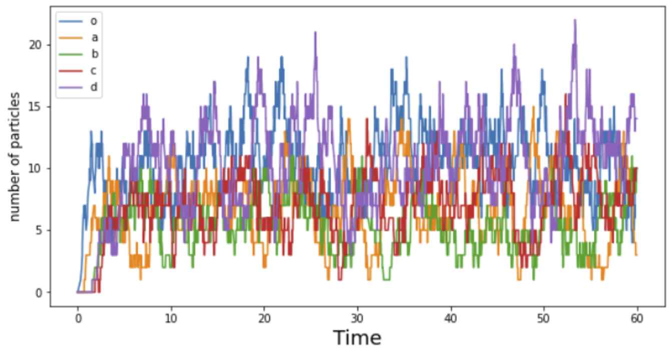


Figure 9: Number of particles for each node during the simulation, with proportional rate and $\lambda = 10$.

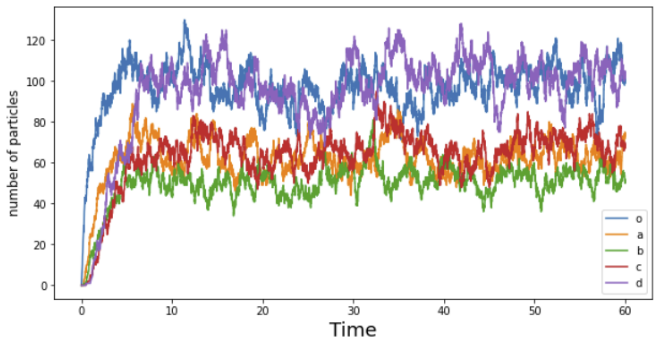


Figure 10: Number of particles for each node during the simulation, with proportional rate and $\lambda = 10^2$.

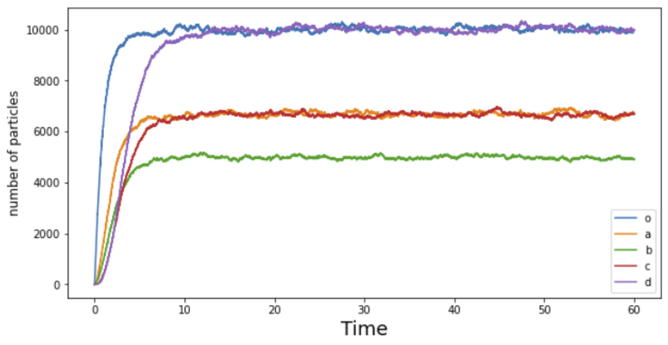


Figure 11: Number of particles for each node during the simulation, with proportional rate and $\lambda = 10^4$.

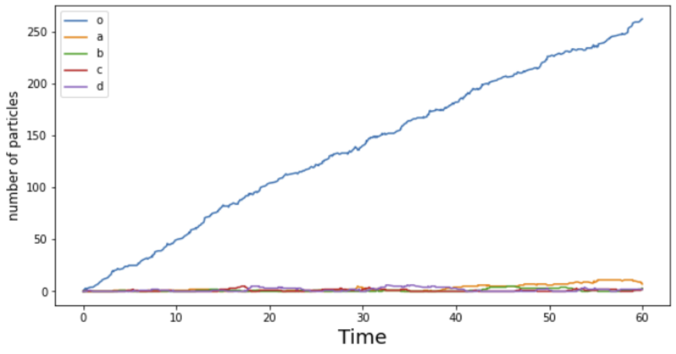


Figure 14: Number of particles for each node during the simulation, with fixed rate and $\lambda = 5$.

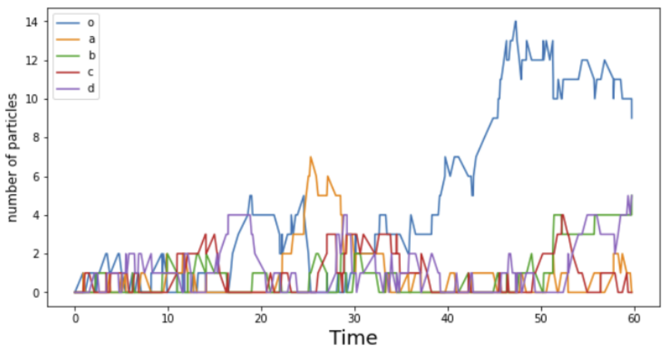


Figure 12: Number of particles for each node during the simulation, with fixed rate and $\lambda = 1$.

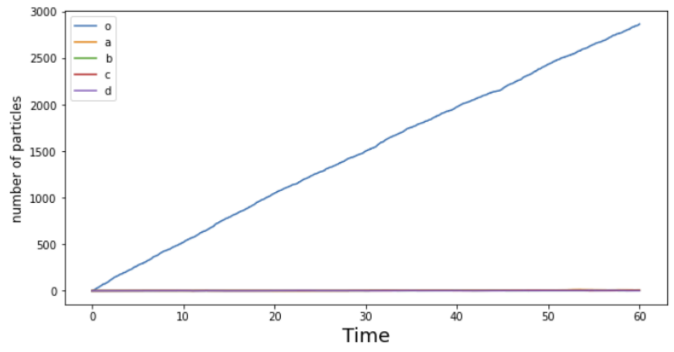


Figure 15: Number of particles for each node during the simulation, with fixed rate and $\lambda = 50$.

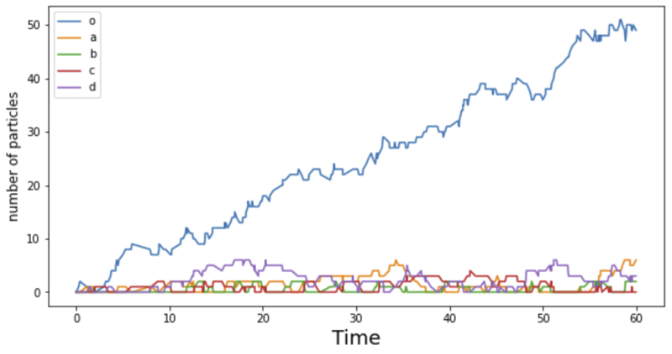


Figure 13: Number of particles for each node during the simulation, with fixed rate and $\lambda = 2$.