Network Dynamics and Learning: Homework 2

Matteo Merlo s287576

December 12, 2021

0 Introduction

This is the report of homework 2 of Network Dynamics and Learning. This part is intended to represent the results and give a theoric comment on how they are computed, every passage has been clarified after into the code execution as markdown or comment. The result are confronted with Andrea Tampellini and Matteo Giardino

1 Exercise 1

In this task, we analyze how to simulate continuous-time Markov chains (CTMC). We are given a weighted directed graph and one or more particles moving in the graph. Each particle moves to a node to one of its neighbors with a probability proportional to the weight of the link between the two. In the case of continuous-time Markov chains. The Markov chain has a finite state space which is our set of nodes.
In the continuous case, the time at which a jump is executed is determined by a Poisson process with rate r. We call it a *Poissonclock* and simulate its ticking. We use a Poisson clock because the time between two "ticks" is an independent random variable with a Poisson distribution.

1.1 Problem 1 - A

In the first point we simulate the time between two consecutive ticks, t_{next}. It is computed as follows

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

where u is a random variable with uniform distribution $u \in \mathcal{U}(0,1)$.
Each node has his proper rate r = ω_i The result of 10000 simulations was averaged, returning an average time equal to:

$$\bar{\tau}_{a,a} = 6.65$$

1.2 Problem 1 - B

To compute analytically the return time we can use several approaches. Since the graph is strongly connected we can use the **theorem 7.2** from pag 136 of the book by appling the formula from point (iv):

$$E_i[T_i^+] = \frac{1}{\omega_i \bar{\pi}_i} = \frac{1}{1 * 0.148} = 6.750$$

Another approach is to solve recursevely using the equation from point (v):

$$E_a[T_a^+] = \frac{1}{\omega_a} + \sum_j P_{a,j} E_j[T_a]$$

To compute this formula, we know that $E_a[T_a] = 0$ and the $E_j[T_a] = \frac{1}{\omega_j} + \sum_j P_{a,j} E_j[T_a]$is the expected hitting time to the set $S = a$ starting from j. To accomplish this in python, we need to write

the previous statements in a matrix formulation. We can index the matrix P and vector ω so that $\hat{P} = P_{RxR}$ and $\hat{\omega} = \omega_R$ with $R = V/S$. If we can solve the linear system:

$$\hat{x} = (I - \hat{P})^{-1} z$$

with $z_i = \frac{1}{\hat{\omega}_i}$ and then compute the average return times. The result obtained with the simulations is close to that computed analytically. If we average more simulations, the results are more accurate and stable

1.3 Problem 1 - C

In this point, we can use the same function of the point A. In this case, we need to specify as a origin the node o and as destination the node d. 1000 simulations are performed and the average time it takes to move from node a to node d is 8.13.

1.4 Problem 1 - D

In this case we can apply the same reasoning of point B in order to compute the hitting times but without having to compute the return times. The theoretical hitting-time can be written as:

$$E_j[T_d] = \frac{1}{\omega_j} + \sum_j P_{j,d} E_j[T_d]$$

The theoretical solution 8.785 is again close to that obtained with the simulations. The error measure between the two solution, theoretical and experimental, is 0.64

1.5 Problem 1 - E

First of all I set the arbitrary initial condition $x(0)$ to [1, 0, 0, 0, 1]. Simulating French - DeGroot dynamics on G, it converges to the consensus state [0.304 0.304 0.304 0.304 0.304]. As a matter of fact, the graph G is strongly connected and the condensation graph is composed of a single sink which is aperiodic. For this reason using **Corollary 5.1** from pag 89 of the book we can affirm that:

$$\lim_{x \to \infty} x(t) = \alpha 1, \quad \alpha = \pi^{'} x(0)$$

In Figure 1 we can see how the evolution of the simulated dynamics converge to the theoretical value of the consensus.

1.6 Problem 1 - F

The theoretical value of the variance obtained with equation

$$\sigma_x^2 = \sigma^2 \sum_{k \in G} \pi_k^2$$

is $\sigma^2 = 0.0178$

We compute 200 numerical simulations simulating the French - De Groot dynamics on the given system. the variance of the consensus state obtained is $\sigma^2 = 0.021$ that is similar to the theoretical value σ^2. Here it is also important to note that the variance of the consensus is less than the variance of the original opinion, which means that the crowd is wiser than any individual. This means that the crowd is smarter than any individual. As long as the graph describing the influence between individuals is contiguous and the individuals all have the same measurement capabilities (e.g., the same error variance), linear averaging always leads to an estimate of the true state Θ that is strictly better than its original estimate. This case is called the wisdom of the crowd.

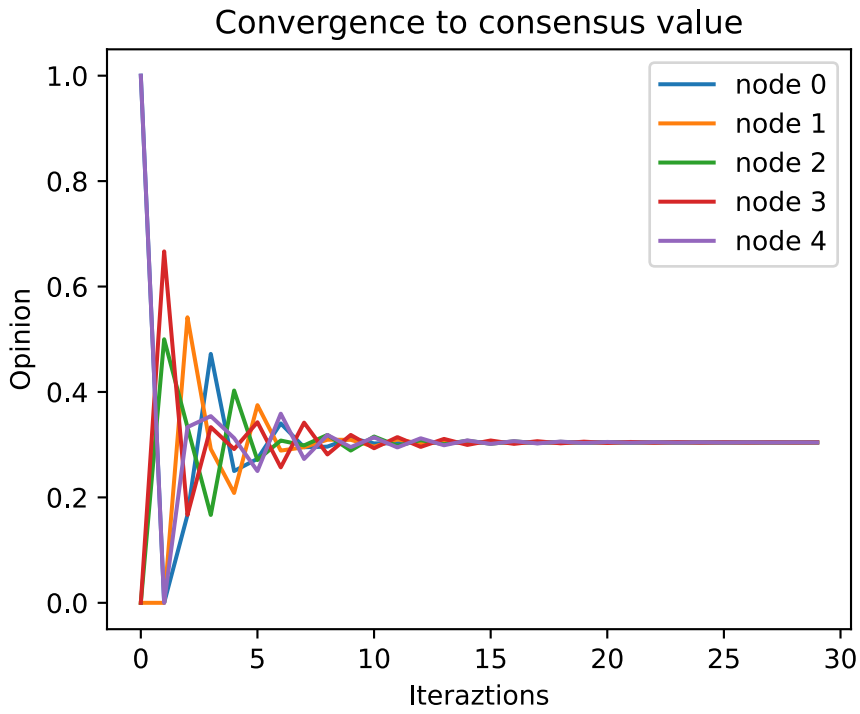

Figure 1: Convergence to consensus value

1.7 Problem 1 - G

Removing edges (d, a) and (d, c) the node d become a sink node. To simulate the French - De Groot dynamics a self-loop is added to the node d. Doing so, the node d will not be influenced by other agents in the network, but will act as a stubborn node. While, considering the initial opinions as random variables and computing the variance on the consensus value with 1000 simulations we obtain $\sigma_x^2 = 0.0785$, much more closer to the variance of the of the initial opinion vector. In fact at each simulation the consensus will be the value of initial state of node d and that value is chosen among a uniform distribution $U(0, 1)$ that has variance $\sigma^2 = 1/12$.

1.8 Problem 1 - H

Again we start considering the same graph and remove the edges (c, b) and (d, a). I simulate the French-De Groot dynamics on the new graph and I see that it does not converge to a consensus state. If we observe the graph, we see that after removing edges, the graph is no longer connected. Since we can only reach node c from node d and vice versa, in a trapping set. If there is a periodic trapping set, the system cannot reach the consensus state. Moreover, if I try to add a self loop in c or d, the trapping set becomes aperiodic and the system converges.

2 Exercise 2

The second exercise is based on the same system of the previous exercise. This time, multiple particles are in the system at the same time. We simulate the system from two different perspectives: the particle perspective and the node perspective. The simulation starts with 100 particles in node o and the system is simulated for 60 times units.

2.1 Problem 2 - Particle Perspective

Since all the particles are independent and identically distributed, the simulation of a random walk with 100 particles is equal to simulating a random walk with one particle 100 times. But, instead of computing the average for each particle (1000 simulation for each one) and then the average over the 100 particles, we can just compute one hundred 1000 simulations (in exercise 1.a there were performed 1000 simulations for 1 particle). Here the number of simulations is equal to the number of particles (100)

The average return time on the node a over 100 particles is $T_a^+ = 6,76$ and we can see how close this value is to the theoretical return time found in Problem 1 Point b $T_a^+ = 6,75$

2.2 Problem 2 - Node Perspective

At the starting 100 particles are setting on the node o. When the clock ticks (with a rate equal to the number of particles in the system), the starting node is chosen proportionally to the number of particles that are on the nodes. As for the terminal node, we can use the matrix as in exercise A instead. This is done until we reach 60 time units. At the end of the simulation, we can average the distribution of particles in each node throughout the experiment. The average number of particles obtained for each node was:

$$o = 21.62 \quad a = 14.04 \quad b = 21.87 \quad c = 21.22 \quad d = 21.24$$

The plot in figure 2 shows the particle distribution in the nodes during the simulation.

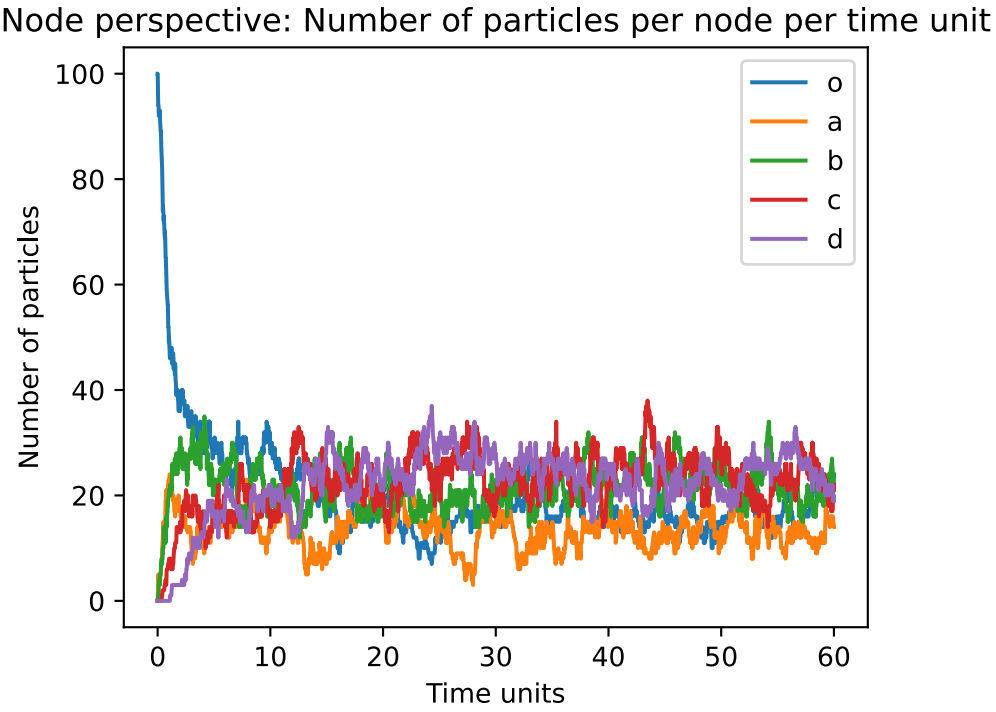

Figure 2: Node perspective

3 Exercise 3

3.1 Problem 3 - Proportional rate

The system has been simulated for 60 time units giving as result the evolution in Figure (3), then the input rate λ has been increased at 5, 10, 100 and 200 obtaining the evolution in Figures (4) (5) (6)

(7). We can see the system does not blow up, in particular, particles do not accumulate in node 'o' and they move around the graph. This is due to the proportional rate of the global clock, the more particles are in the graph, the more the rate is higher. So each node is able to pass along its particles to other nodes.

3.2 Problem 3 - Fixed rate

In this part the rate of the Poisson clock of each node is fixed, and equal to one. Again we simulate it for 60 time units obtaining the distribution of particles in Figure (11). The input rate has been tested also with 0.5, 0.6, 0.7 and 2 with the results in Figures (8) (9) (10) (12). For rates greater than one, the number of nodes explodes, while, if the rate is 1, it is unstable and the system could still blow up in some simulations. For smaller rates, such as 0.6 - 0.7, the handling of the particle distribution in the nodes is better.
We can notice that the particles are all accumulated in the starting node o and have difficulty to distribute across other nodes, in particular we have 0 particles in node d. The system has blown up now. The main difference from the previous point is that the rate is fixed, so even if the particles increase, it will stay the same. If too many particles will enter in the system, they will leave it with more difficulty than the previous point.

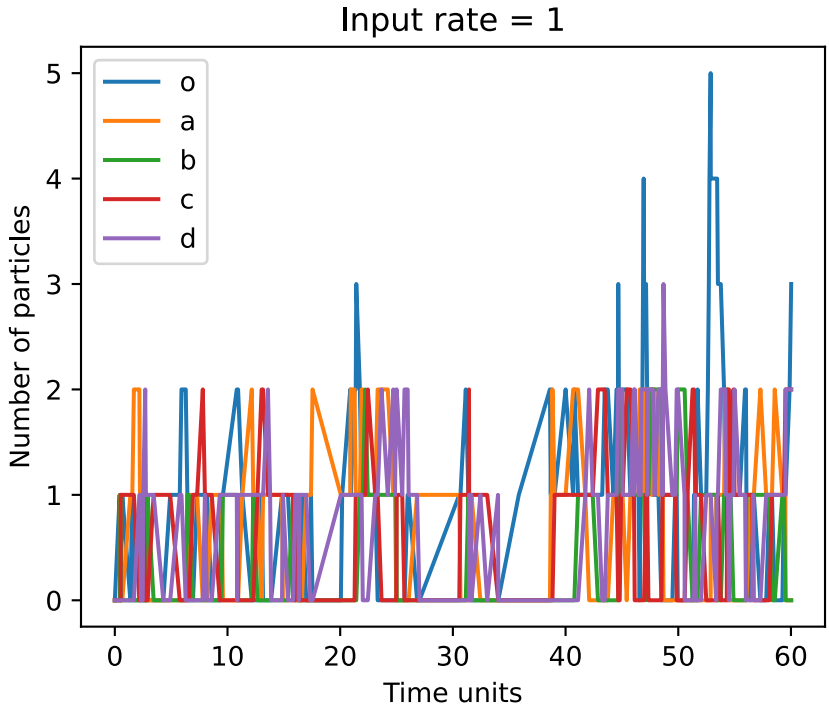

Figure 3: Proportional rate: Input Rate 1

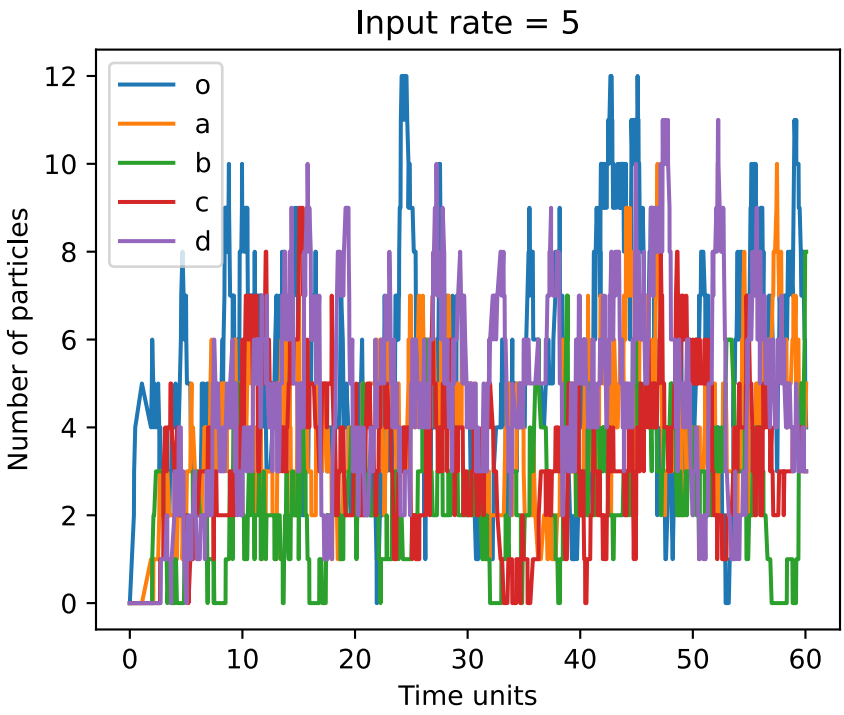

Figure 4: Proportional rate: Input Rate 5

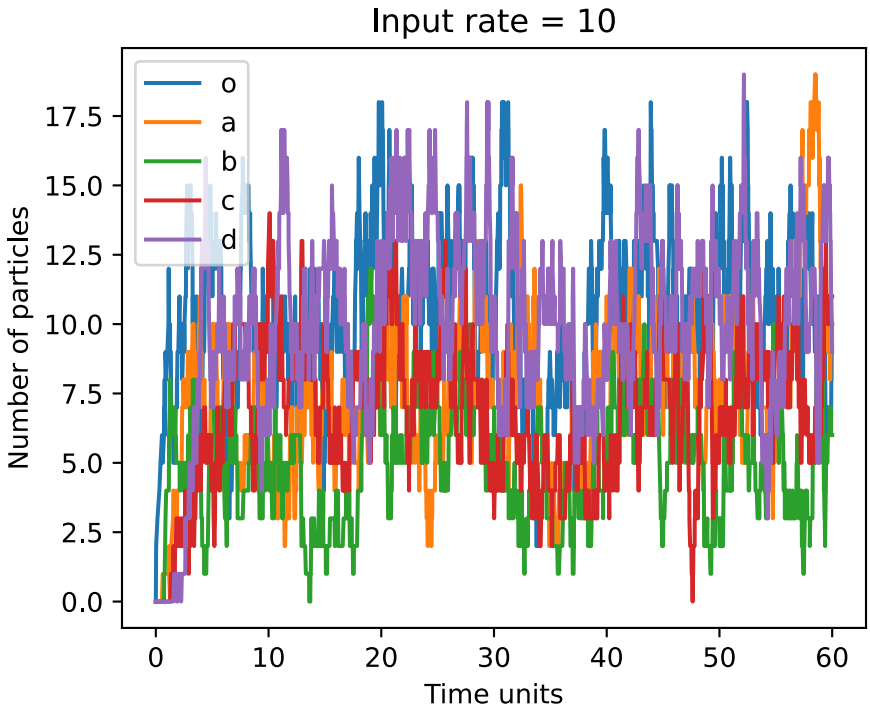

Figure 5: Proportional rate: Input Rate 10

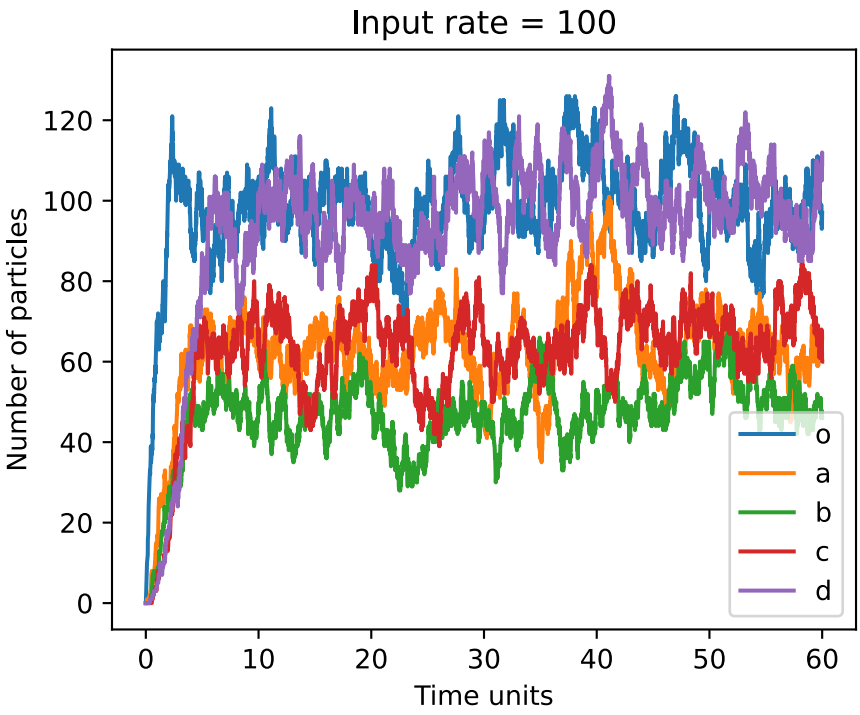

Figure 6: Proportional rate: Input Rate 100

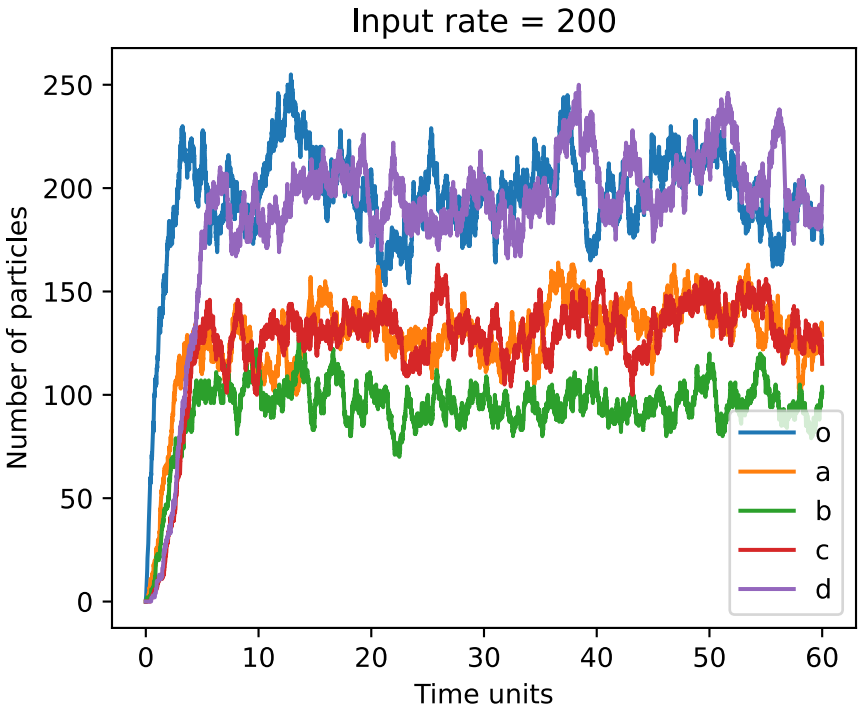

Figure 7: Proportional rate: Input Rate 200

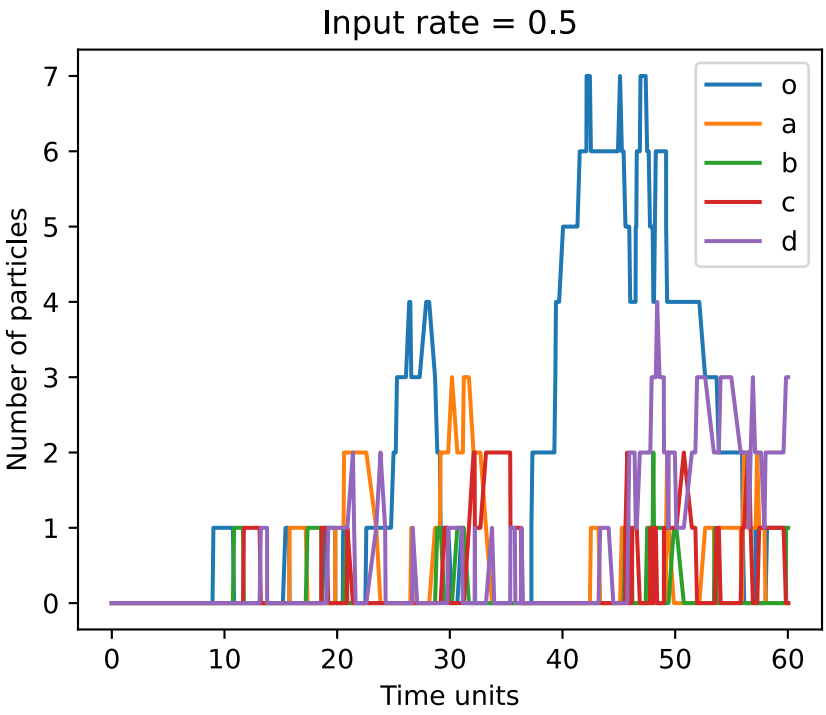

Figure 8: Fixed rate: Input Rate 0.5

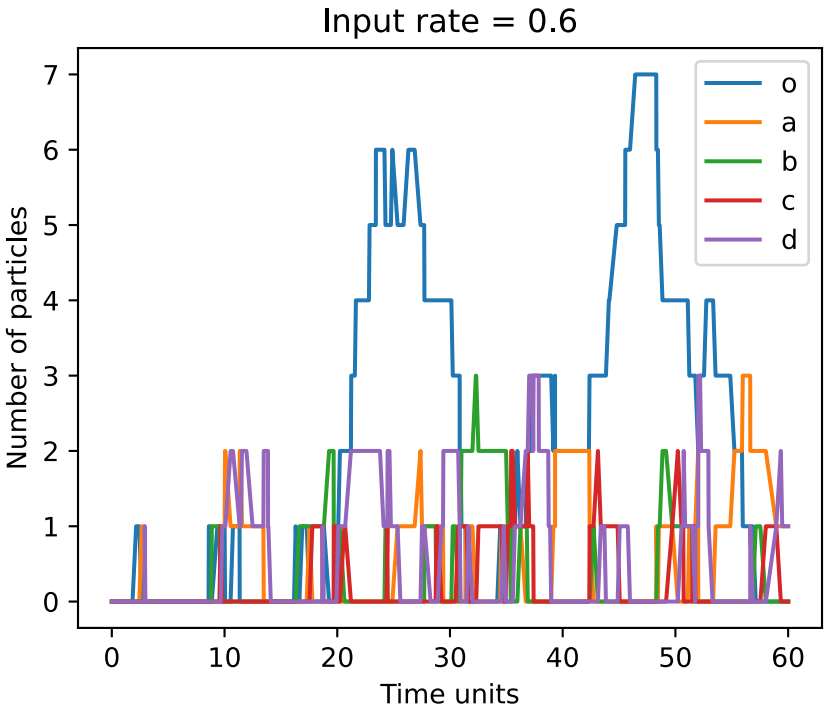

Figure 9: Fixed rate: Input Rate 0.6

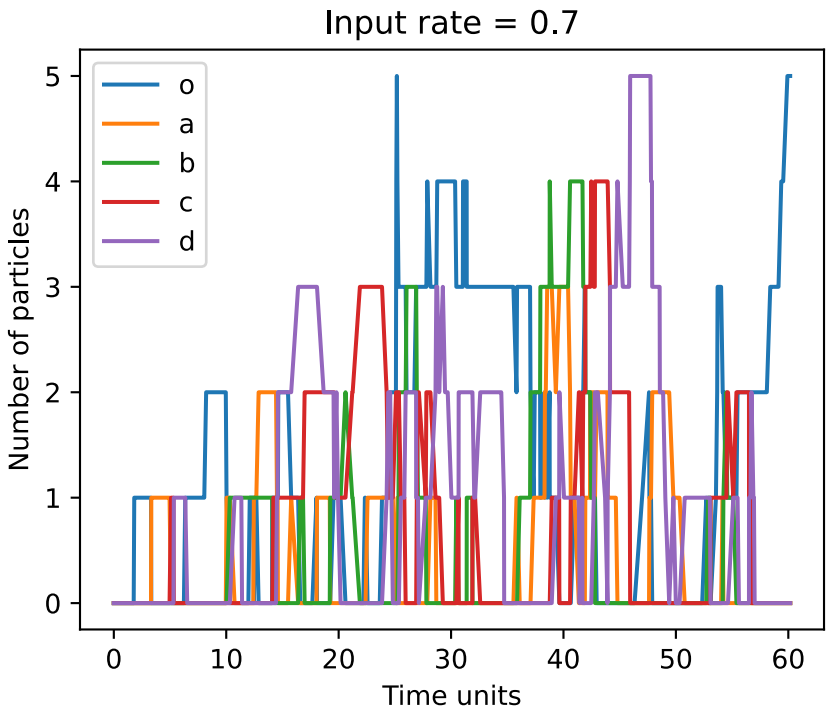

Figure 10: Fixed rate: Input Rate 0.7

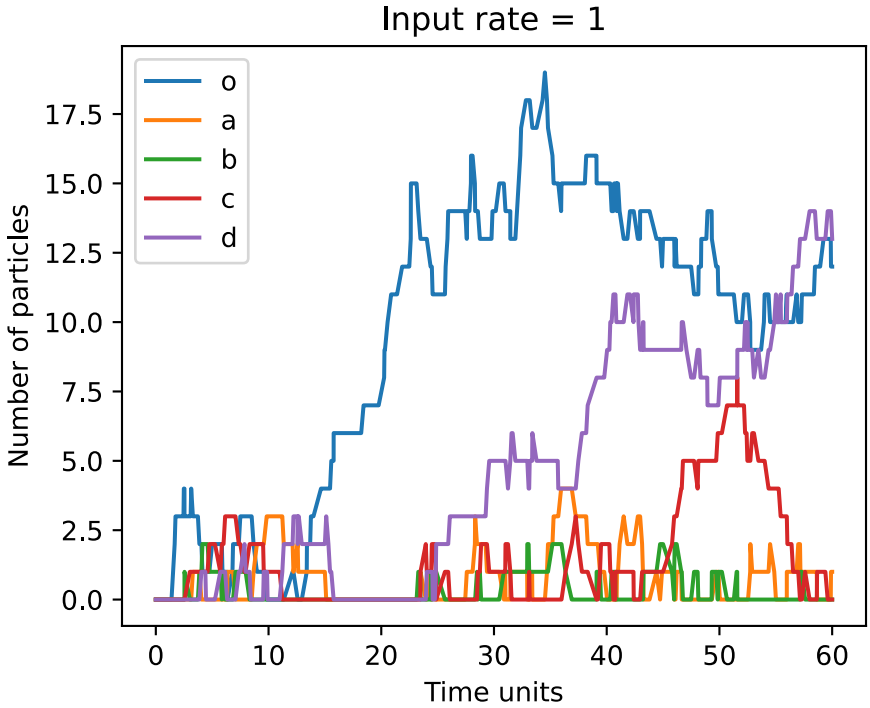

Figure 11: Fixed rate: Input Rate 1

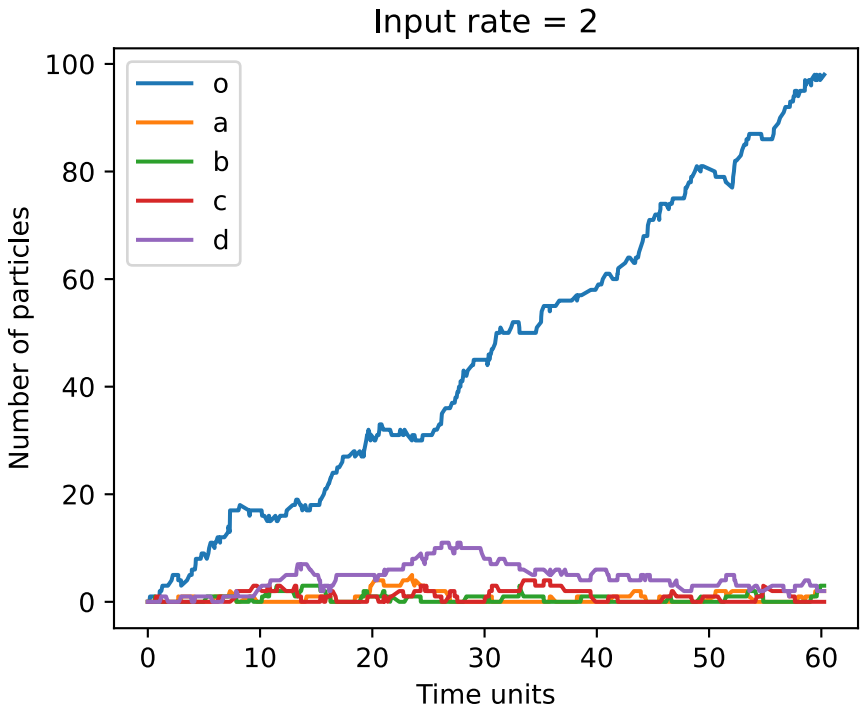

Figure 12: Fixed rate: Input Rate 2