Homework II

M.Sc. Data Science and Engineering - AY 2022-2023

Network Dynamics and Learning

Professors: F. Fagnani, G. Como, L. Cianfanelli

Pietro Cagnasso
s300801
pietro.cagnasso@studenti.polito.it

Nicolò Vergaro[1]
s295633
nicolo.vergaro@studenti.polito.it

Exercise 1

This first exercise requires to study a continous time random walk of a particle on a graph as the one in Fig. 1 using the associated transition matrix:

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

Rows and columns refer in order to nodes o, a, b, c, d, respectively.
Once we studied this random walks the exercise moves into studying opinion dynamics on the same graph also with some small modifications to its topology (1.g, 1.h).

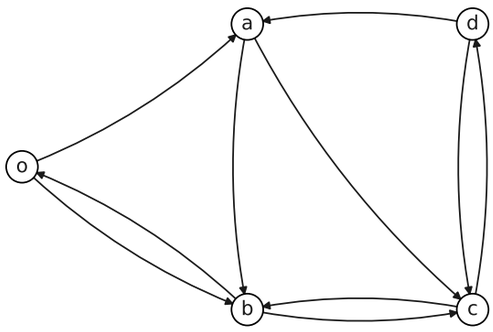

Figure 1: Graph described in the Exercise 1 on which we are going to study some dynamics.

Theoretical basis

Definition 1 (Return time) *We define the return time of a particle in a node j as the first time in which the particle, starting in that node, returns to that same node:*

$$T_i^+ = inf\{t \geq 1 \ s.t. \ X(t) = j\}.$$

Definition 2 (Random walk) *Random walks are Markov chains whose state space is identified with the node set of a graph: the links of this graph in turn represent the possible state transitions. The walk can happens moving to neighbor nodes with a probability that is defined by the matrix $P = diag(\omega)^{-1}\Lambda$, where $\omega = \Lambda\mathbb{1}$.*

[1] With whom I collaborated on the entire homework.

Theorem 1 (Kac's formula) *Given a random walk on a strongly connected graph $\mathcal{G}$ having unique invariant distribution π, the theoretical return time for each node i is defined as*

$$E_i[T_i^+] = \frac{1}{\pi_i}.$$

Definition 3 (Hitting time) *We define the hitting time of a particle in a node j starting from node i as the first time in which the particle arrives to that node:*

$$T_j = inf\{t \geq 0 \ s.t. \ X(t) = j\}.$$

If the node i coincide with node j this hitting time is null.

Theorem 2 (Recursive relations for expected hitting times) *In continuous time Markov chains the expected hitting time for the set of node $\mathcal{S}$ can be defined as:*

$$E_i[T_{\mathcal{S}}] = \frac{1}{\omega_i} \sum_j P_{ij} E_j[T_{\mathcal{S}}].$$

As in the discrete time case, considering $Q = P|_{\mathcal{R}}$ and calling $z_i = E_i[T_{\mathcal{S}}]$ we can write this relation as $(I - Q)z = \mathbb{1}$ and so, if $(I - Q)$ is invertible, we have a direct form for z. This is the case if $\mathcal{S}$ is globally reachable because this implies Q to be sub-stochastic and so having no eigenvalue equal to 1.

$$\mathcal{S} \text{ globally reachable} \implies z = (I - Q)^{-1}\mathbb{1}.$$

Definition 4 (French-DeGroot dynamic) *The French-DeGroot dynamic is a learning model in which at each step agents modify their opinion averaging the opinions of neighbors. The relative influence between nodes is defined by the normalized weight matrix $P = diag(w^{-1})W$, producing a system dynamic as follows:*

$$x(t) = P^t x(0).$$

Having as stationary distribution π we know that the long term trend of the dynamic on a strongly connected and aperiodic graph will converge to $\bar{x} = \pi^T x(0)$.

Definition 5 (Wisdom of crowds) *Assuming an underlying state of the world denoted as μ we can imagine each client seeing it in a noisy version $x_i(0) = \mu + \xi_i$. From the French-DeGroot dynamic we know that the final state will be $\bar{x} = \mu + \sum_i \pi_i \xi_i$. Considering all the noises ξ_i as independent measurements drawn from the same distribution with variance σ^2, the variance of the final state will be:*

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

1.a

To simulate the expected time required to a particle to return to the node a (Def. 1) we defined a function called `simulateWalkTime`. This function simulates, for a given number of times, a random walk between the initial point and the destination point. In this function we used a global clock with rate $\omega^* = \max_i \omega_i$, where $\omega = \Lambda\mathbb{1}$. And as transition probability matrix $P = diag(\omega^{-1})\Lambda$.

In this problem the initial and the destination points coincide in a. This function returns the average time required by the particle across all the simulations. This works based on the law of large numbers, for which the average of n i.i.d. measurements tends to the expected value of the random variable as n grows. Since we are talking about random walks on a graph we can consider these measurements as i.i.d., so this technique holds.

Results

The result we obtained with 100000 simulations is an expected return time of 6.7804.

1.b

To compute the thoeretical return time of a particle in a we can exploit the Kac's theorem (Theo. 1). Once we computed the stationary distribution π we have the return time of a as:

$$E_a[T_a^+] = \frac{1}{\pi_a}.$$

Results

The theoretical return time obtained for a is 6.75. The result we got from simulations differs by 0.4506% from the thoeretical one. We can affirm that the amount simulations we performed (100000) was enough to correctly approximate this quantity.

1.c

To simulate the expected time we need to reach d from o (Def. 3) we used once again the function `simulateWalkTime`, this time using as initial point o and destination point d. Once again having measurements from random walks we can consider them as i.i.d. and so the average of them tends to the expected hitting time.

Results

The result we obtained from 100000 simulations is an expected hitting time of 8.7968.

1.d

To compute theoretically the hitting time of d from o we used the recursive relation for expected hitting times (Theo. 2, but extended to continuous time). To exploit this theorem we set as stubborn node $\mathcal{S} = \{d\}$, which is the node we would like to reach. Since the node d is globally reachable the matrix $(I - Q)^{-1}$ is defined and a unique $z = (I - Q)^{-1}\frac{\mathbb{1}}{\bar{\omega}}$ exists ($\bar{\omega} = \omega|_{\mathcal{R}}$). The theoretically expected hitting time is z_d.

Results

The theoretical hitting time for d starting in o is 8.7857. The result we got from simulations differs by 0.1263% from the thoeretical one. We can affirm that the amount simulations we performed (100000) was enough to correctly approximate this time.

1.e

Interpreting the matrix Λ as a weight matrix for the graph $\mathcal{G}$ we can now simulate the French-DeGroot dynamic (Def. 4) on the system. To generate an arbitrary initial condition $x(0)$ we used a normal random generation between with mean $\mu = 0$ and standard deviation $\sigma = 1$.

Results

Running this kind of dynamic on a system that has no stubborn nodes produces the convergence of all nodes to a consensus for any possible initial condition since the graph is *strongly connected and aperiodic*. As we can see in Fig. 2 in just 25 iterations of the dynamic there is a clear trend towards the consensus. The theoretical consensus value is 0.68448 and the average difference from it at the 25^{th} iteration is just 0.0003.

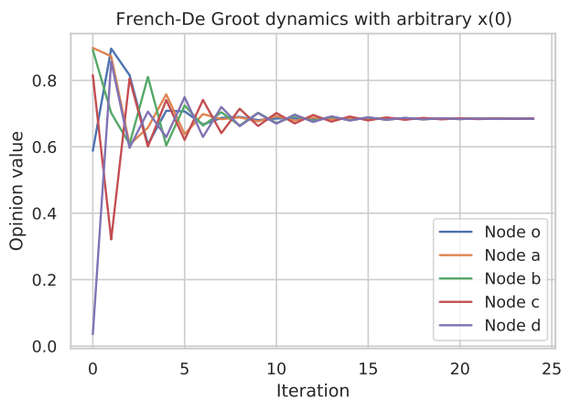

Figure 2: French-DeGroot dynamic on the graph $\mathcal{G}$ with a uniform random initial condition $x(0)$ and no stubborn nodes.

1.f

To define a noisy initial condition with a fixed mean and variance we used a random generation from a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. We have as theoretical result

from the wisdom of crowds (Def. 5) that the variance after passing through the system will be defined as:

$$\mathrm{Var}(\sum_k \pi_k N_k) = \sigma^2 \sum_k \pi_k^2.$$

To obtain a similar result through simulations we can repeat n times the dynamic $x(t) = P^t x(0)$ over the network. Then we compute for each experiment i the variance as $\sigma_i^2 = (\mu - x^{(i)}(t))^2$, the average of these experimental variances, for large n, should approach the theoretical result.

Results

The theoretical variance we should reach is 0.21361. From 1000 simulations of the system running for 100 iterations we got as expected variance 0.21224, differing by just 0.63964% from the theoretical result.

1.g

By removing the edges (d, a) and (d, c) we get that the node d becomes a sink node. This new graph would have a null determinant adjacency matrix, to use the methods we used previously we need to add a self-loop on node d. With this 2 steps we get the graph in Fig. 3.

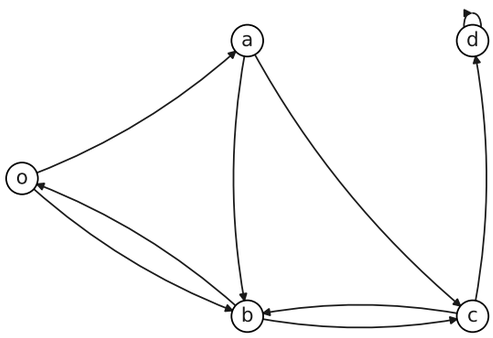

Figure 3: The graph $\mathcal{G}_g$ used to complete the exercise 1.g.

Now the node d cannot see the outside world and so it will behave as a stubborn node. We expect the entire system converging to the initial opinion of d.

To compute theoretical and simulated variances we can use the same method used above.

Results

As we can see in Fig. 4 the behavior of the system is, as expected, a trend towards a consensus on the opinion of the node d.

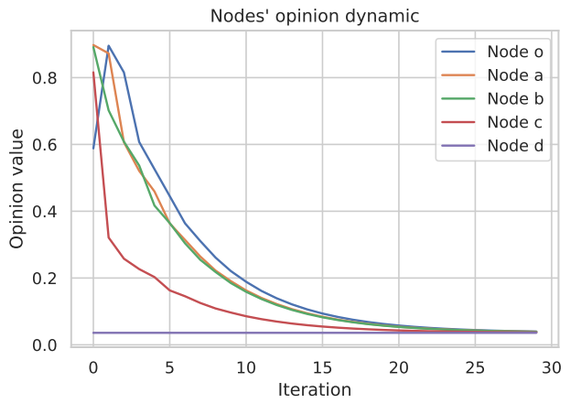

Figure 4: Opinion dynamic on the new graph $\mathcal{G}_g$ starting from an initial condition drawn from a normal distribution with mean 0 and variance 1.

The theoretical expected variance is 1 since the node d is a sink. The variance we got from 1000 experimental results letting the system run for 100 iterations each time is 0.97679.

1.h

By removing the edges (c, b) and (d, a) we get the graph in Fig. 5. We can clearly notice the presence of a trapping component composed by nodes c and d. This topological trait leads us to the conclusion that, if these two nodes will not have the same opinion, the system will not converge to a consensus.
In fact node c will copy at each time the opinion of node d and vice versa. The rest of the system can just see node c and so the system as a whole will oscillate never reaching a consensus.

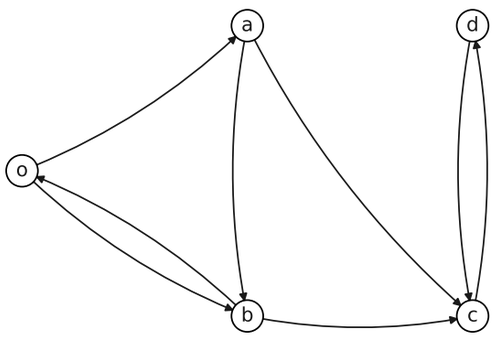

Figure 5: The graph $\mathcal{G}_h$ used to complete the exercise 1.h.

Results

As expected the system oscillates in a range of opinions that is bounded by the opinions of nodes c and d, Fig. 6 shows this trend.

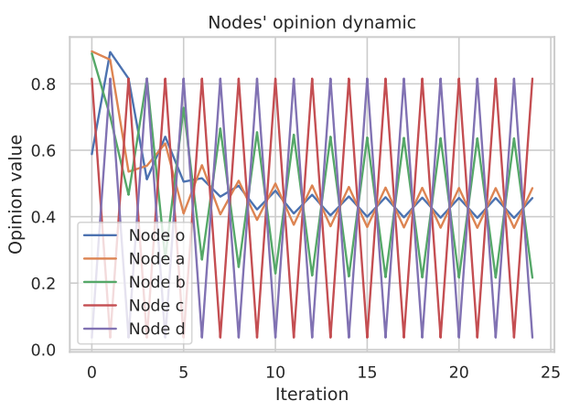

Figure 6: Opinion dynamic on the new graph $\mathcal{G}_h$ starting from an initial condition drawn from a normal distribution with mean 0 and variance 1.

Exercise 2

In this second exercise we consider the same network already used in the previous one and presented in Fig. 1. Now we will consider the dynamic of multiple particles moving around on this graph, doing this both from particles' and nodes' perspective.

Theoretical basis

The theory behind this exercise is the same we used in exercise 1.

2.a Particles' perspective

In this exercise we hypothesize to have 100 particles all in the node a. To compute the average time each particle takes to return to that node (Def. 3) we can let the system run and record the return times.

To do this we set up a global clock with a rate equal to $100 \cdot \omega^*$ where the factor 100 comes from the number of particles. Each time the clock ticks we randomly select a particle from the system and move it on the graph according to the transition probability matrix $\bar{P}_{ij} = \frac{\Lambda_{ij}}{\omega^*}$ for $i \neq j$ and $\bar{P}_{ii} = 1 - \sum_{j \neq i} \bar{P}_{ij}$. To be sure that the time recorded is a legit return time we just need to check

that the particle does not come from the node a. What happens after the first return time is not something we need to consider in this case, so, after the first return in a, the movements of the particle is not followed anymore but we still let the particle move around and be selected whenever the clock ticks.

Results

Running the system with 100 particles the results extremely noisy. In fact depending on the seed we use we can get very different results, for example with seed 0 we get 7.9577 while with seed 20 we get as average 5.9291. This is expected since the number of particles is very low to reach stable results, this is equivalent to run the function `simulateWalkTime` for just 100 simulations.

To achieve more accurate results we would need to repeat this experiment several times and average the results. Since each particle is independent in its movements to achieve the same result we could also increase the number of particles in the system and adjust the global clock rate accordingly. For example running a system with 100000 particles we got as expected return time 6.73, which is a better approximation for the theoretical return time of 6.75.

2.b Node perspective

If we hypothesize to have 100 particles all starting in node o and let the system run for 60 time units we can study how the particles move around looking at the amount of particles in each node at each clock tick.

To simulate this trend we can use a global Poisson clock with rate $100 \cdot \omega^*$ and as transition probability matrix the matrix $\bar{P}$. We do not have any particle entering or exiting the system and we are interested in studying the position of all the particles throughout the entire simulation. Each time the clock ticks we select a node with a probability proportional to the amount of particles in the node and move one of its particles to a neighbor node accordingly to $\bar{P}$.

Results

We have simulated this process and the resulting number of particles in each node through the simulation is represented in Fig. 7.

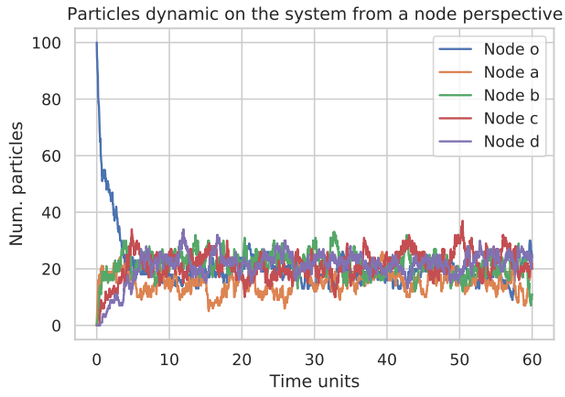

Figure 7: Number of particles in each node throughout the simulation on 60 time units.

Retrospectively looking to the average amount of particles in each node we see an average amount of

$$\bar{x}_{60} = (20.758, 15.267, 21.602, 21.253, 21.118)$$

which is a gross approximation for the stationary distribution that 100 particles would have followed

$$100\pi = (18.518, 14.814, 22.222, 22.222, 22.222).$$

This is the case since the amount of particles and time units is very limited.

Letting the system run for example for 10000 time units we can see a much more refined approximation for that vector:

$$\bar{x}_{10000} = (18.583, 14.843, 22.235, 22.211, 22.125).$$

This is the case because with a large amount of time units the contribution of the initial transitional phase becomes negligible.

Exercise 3

We have a new slightly modified graph in this exercise, as represented in Fig. 8.

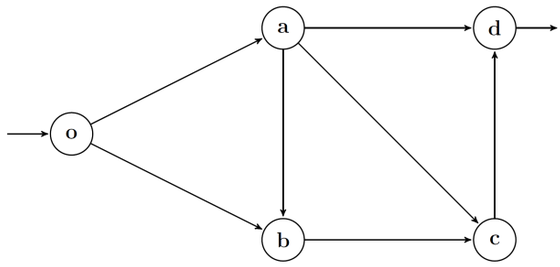

Figure 8: Graph $\mathcal{G}_2$ on which we study the dynamics required by exercise 3.

In this case the graph is open, meaning that particles can enter from outside in node o and exit the system from node d. We have also a new transition matrix:

$$\Lambda_{open} = \begin{bmatrix} 0 & 3/4 & 3/8 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 2/4 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 1/3 & 0 & 1/3 & 0 \end{bmatrix}.$$

We considered $\omega_d = 2$, following the requirements of the text.

Theoretical basis

The theory behind this exercise is the same we used in exercise 1.

3.a Proportional rates

To simulate this system for 60 time units we found out two methods.

The first uses a clock per node and a clock for a dummy node from which the particles come from. The rate for each node at time t is $n_i(t)\omega_i$ ($n_i(t)$ is the amount of particles in node i at time t), while for the dummy node we considered a rate r. At each iteration we let all clocks tick and select the node with faster clock. From that node we move a particle to a neighbor with probability given by the matrix P. If the selected node is the dummy node a particle is added to node o, while if the selected node is d a particle is removed from the system.

The second method uses a global clock to control the system with rate $N(t)\omega^*$, where $N(t)$ is the total amount of particles in the system. And a second clock just regulating the entrance of particles in the system. We generate in advance all the entrance times and save them into a list. Then we let the system run for the 60 time units. Each time the global clock ticks we let some particles enter the system (if the entrance clock has ticked between the previous and the current ticks of the global clock), then we select a node with probability proportional to $n_i(t)$. The node to which the particle moves is selected with probability given by the matrix $\bar{P}$, of course if the selected node is d we just remove the particle from the system.

Results

We ran the system with several input rates r: 1, 10, 20, 100, 200, 1000 (Fig. 9). As we can notice the system does not blow up even with very large input rates like 1000 (bottom right). This is the case since the moving rate from each node is proportional to the amount of particles in that node, so the system after an initial transitional phase stabilizes around a certain amount of particles in each node.

We have tested both the methods obtaining similar results. The main difference between these methods is the time required to run: the first is much quicker.

3.b Fixed rates

To simulate the behavior of the system described previously, but with fixed rates we used the first method described in exercise 3.a. We just needed to modify the rate of nodes' clocks from $n_i(t)\omega_i$ to ω_i. So we selected again the node that ticked the earlier. If the node is the dummy node a particle enters the system in o, if it is the node d the particle just exits from the system, otherwise the particle is moved to a neighbor with probabilities given by the matrix P.

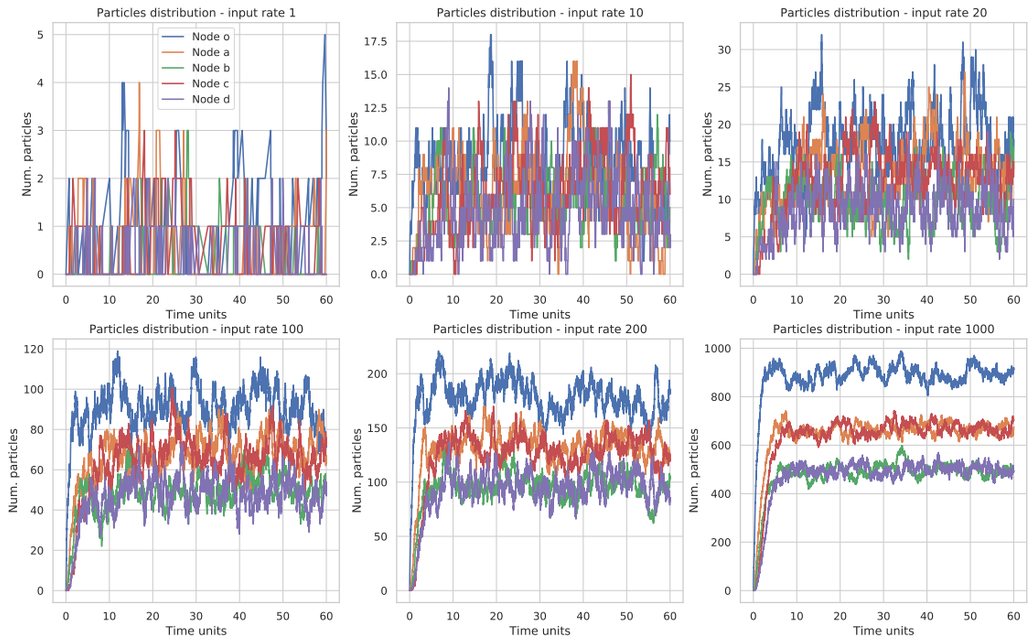

Figure 9: Number of particles per node moving around with proportional rates, charts for different input rates.

Results

We have tested several input rates: 0.5, 1, 1.25, 2, 5, 10 as shown in Fig. 10. As the input rate grows we can clearly see the amount of particles in node o exploding. For very large input rates like 10 (bottom right) we can notice that the number of particles in node o tends to follow a line with slope (input rate $- \omega_o$).

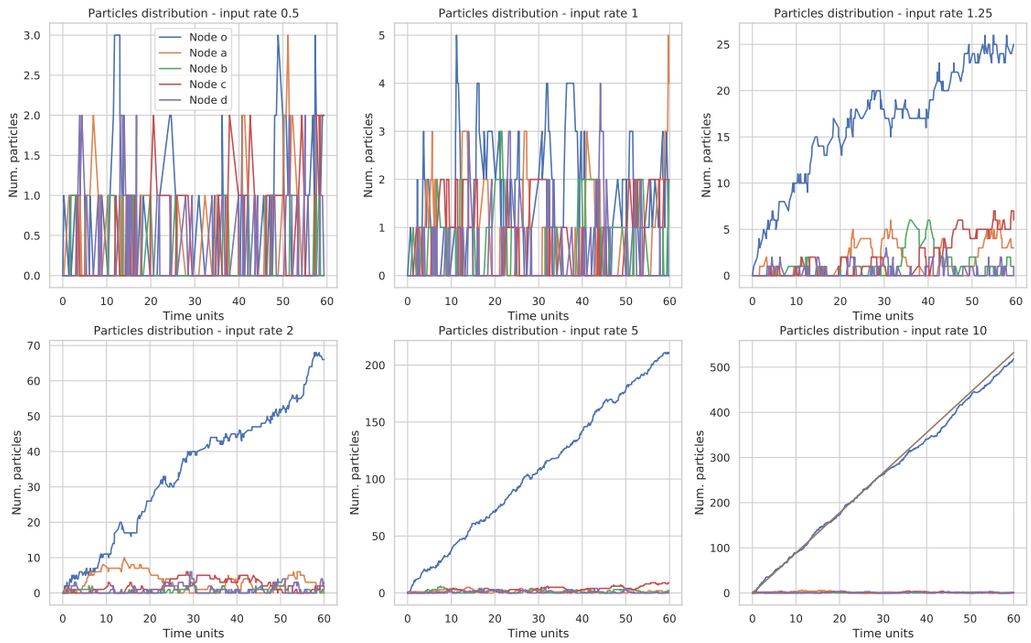

Figure 10: Number of particles per node moving around with fixed rates, charts for different input rates.