Politecnico di Torino

Network Dynamics

Homework 2

Micol Rosini s302935

1 Exercise 1

Consider a single particle performing a continuous-time random walk in the network described by the graph in the Fig. 1 and with the following transition rate matrix:

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

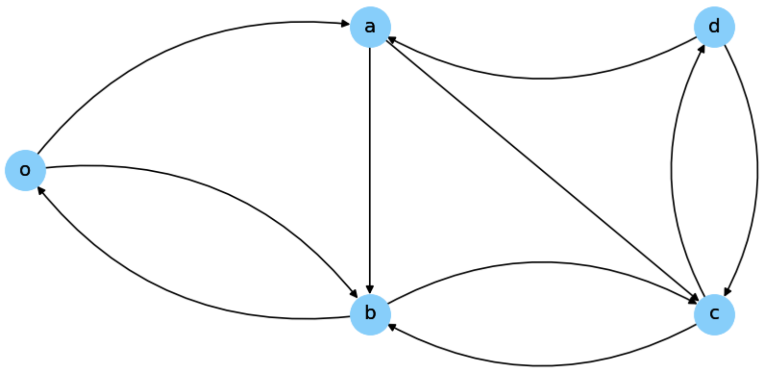

Figure 1: Graph of the exercise.

1.1 Average return-time

Given a directed graph $G = (\mathcal{V}, \mathcal{E})$, the *random walk* of a particle from one of its nodes to another can be interpreted as a Poisson process, $N_t = X(t)$, on the infinite set of non-negative integers, that starts in a at time $t = 0$, increases by one unit at every tick of the Poisson clock, and never decreases.
When the **Poisson clock ticks**, the particle randomly chooses the next node to move to among the out-neighbors of the current node with probability proportional to the row-stochastic matrix Q, which is defined in the following way:

$$Q_{ij} = \frac{\Lambda_{ij}}{\omega_*}, \quad i \neq j$$
$$Q_{ii} = 1 - \sum_{i \neq j} Q_{ij}$$

with $\omega = \Lambda \mathbf{1}$ and $\omega_* = \max_i \omega_i$.

The average time it takes a particle that starts in node a to leave the node and then return to it can be computed with the following function: `average_time_calculator`.

```python
def average_time_calculator(G,o,d,w_star,Q, num_simulations):
    nodes = list(G.nodes())
    o = nodes.index(o) # origin
    d = nodes.index(d) # destination
    returned_times = [] # returned_times will store the different time instants
         at which the particle returns in the starting node
    for simulation in range(num_simulations):
        pos = []
        pos.append(o) # we start from state a
        transition_times = 0 # simulation time
        t_next = -np.log(np.random.rand())/w_star
        i = 0
        returned = False
        while not returned:
            i += 1
            pos.append(np.random.choice(nstates, p=Q[pos[i-1],:]))
            transition_times += t_next
            if pos[i] == d : # arrived at destination
                returned_times.append(transition_times)
                if simulation == num_simulations/2:
                    pos_j = pos[1:]
                    pos_i = pos[:-1]
                    edges_visited = [(nodes[i],nodes[j]) for i,j in zip(pos_i,
                         pos_j))]
                returned = True
            t_next = -np.log(np.random.rand())/w_star
```

It simulate the movement of a particle 10000 times and then make the average of all the different return-time. With this function the average time to return in node a is around 6.75, i.e.

$$E_a[T_a^+] \approx 6.75$$

Fig. 2 shows us one possible way of a random walk that starts and returns in a.

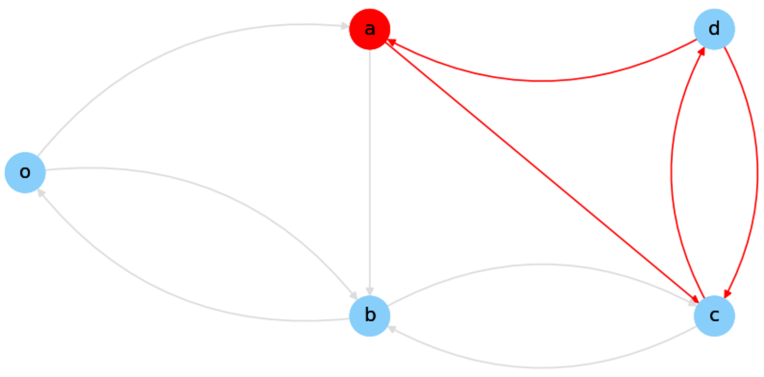

Figure 2: Random walk that starts and returns in a.

1.2 Theoretical return-time $E_a[T_a^+]$

The theory states that if the graph G is **strongly connected** the *expected return times* satisfy:

$$E_i[\overline{T}_i^+] = \frac{1}{\omega_i \overline{\pi}_i}$$

where

$$\overline{T}_i^+ = \inf\{t \geq 0 : X(t) = i \text{ and } X(s) \neq i \text{ for some } s \in (0,t)\}$$

Where π represents the *stationary* probability vector of the continuous-time Markov chain with transition rate matrix Λ.
The graph is strongly connected and the expected return-time is **6.75**

$$E_a[T_a^+] = 6.75$$

This result coincide with the average return-time computed by numerical simulations.

1.3 Average hitting time

The average time it takes to move from node o to node d can be computed by numerical simulations using the same function used in the subsection1.1: `average_time_calculator` and it is around **8.8**

$$E_o[T_d] \approx 8.8$$

Fig. 3 shows one possible random walk that starts in node o and ends in node d.

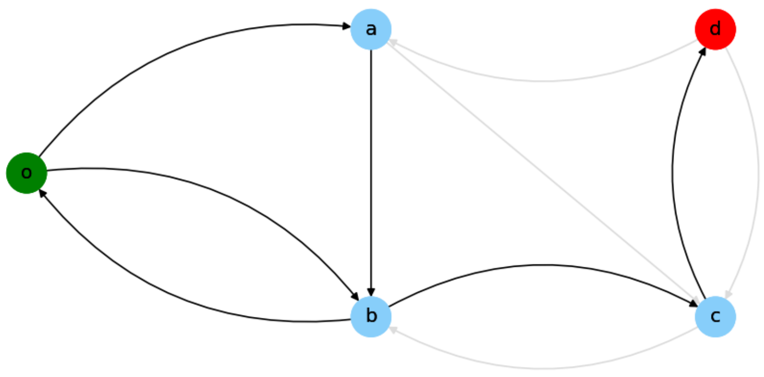

Figure 3: Random walk that starts in o and ends in d

1.4 Theoretical hitting-time $E_o[T_d]$

If d is the node to *hit*, the theoretical hitting-time $E_o[T_d]$ for a **strongly connected** graph is computed in the following way:

$$\begin{aligned}
\overline{\tau}_i^d &= \mathbf{E}_i[T_d] \\
\overline{\tau}_i^d &= 0 \quad \text{if} \quad i = d \\
\overline{\tau}_i^d &= \frac{1}{w_i} + \sum_j P_{i,j}\mathbf{E}_j[T_d] \quad \text{if} \quad i \neq d
\end{aligned}$$

P is the matrix of transition probabilities and for the node d it is defined in the following way:

$$P_{d_{ij}} = \begin{cases} P_{i,j} & \text{if } i \neq d \\ 0 & \text{if } i = d \end{cases}$$

$$\overline{\tau}_i^d = \frac{1}{w_d} + P_d\mathbf{E}[T_d]$$

where

$$\frac{1}{w_i} = 0 \quad \text{if } i = d$$

And so, the result states that the theoretical hitting time is:

$$E_o[T_d] = 8.7857$$

1.5 French-DeGroot dynamics

Interpreting the matrix Λ described in Eq. 1 as the weight matrix of a graph $G = (\mathcal{V}, \mathcal{E}, \Lambda)$, is possible to simulate the **French-DeGroot** dynamics on G with an arbitrary initial condition $x(0)$.

The theory states that the dynamic converges to the *consensus state* if:

1. Its condensation graph has 1 sink;
2. The sink component of the graph is aperiodic;

$$\Rightarrow \lim_{t \to +\infty} x(t) = \alpha \mathbf{1}$$

Fig. 4 shows the condensation graph of G.

Figure 4: Condensation graph of G.

If we 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. We can compute the variance of the consensus value, since the theory states that if ξ_i are independent with the same variance σ^2. In this case, the variance of the asymptotic consensus value X satisfies:

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

where π that is the dominant eigenvector is $= [0.13, 0.17, 0.26, 0.26, 0.17]$
And the theoretical computation of variance of consensus gives as result:

$$\sigma^2_{\bar{x}} \approx 0.0192249$$

This result can be compared with a numerical simulation: setting $\sigma^2 = 0.3$ and $\mu = 0.8$ for 1000 iterations. The simulation's result coincide with the theory:

$$\sigma^2_{\bar{x}} \approx 0,019$$

1.6 French-DeGroot with modified graph

If we remove the edges (d, a) and (d, c), the asymptotic behaviour of the dynamics can change. Indeed, the new graph $G1$ which is represented in Fig 5 is not strongly connected but its condensation graph has one sink.

In this case, the theory states that:

- The averaging dynamics converges to a consensus, whose value depends only on the initial state of nodes of the sink;
- The linear flow dynamics converges to the dominant eigenvector of P', which has support only on the nodes of the sink.

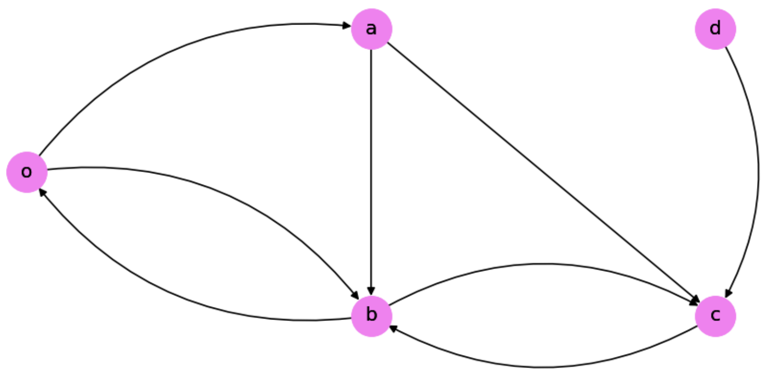

Figure 5: Graph without the edges (d,a) and (d,c).

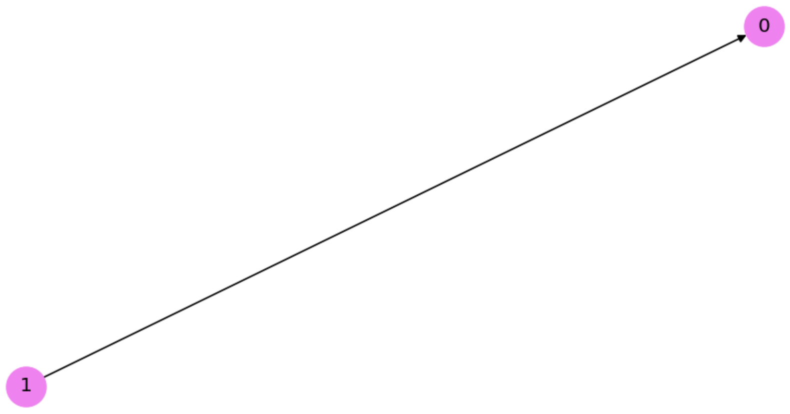

Figure 6: Condensation graph of G1.

In order to check if the graph satisfies the condition to consensus, first we have to check if the number of attracting components in the condensation graph is 1, because otherwise it will have more than 1 sink. Then, to check the aperiodicity of the sink component in $G1$, we apply the function `nx.is_aperiodic(G1)` on the attracting component of $G1$. With `utils_graph_modified` we can obtain the new weight matrix W of $G1$ and so compute the new P, in order to obtain the new different consensus values.

Assuming 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, the variance of consensus by numerical simulations is :

$$\sigma^2_{\bar{x}} \approx 0.1$$

The theoretical result is instead:

$$\sigma^2_{\bar{x}} = 0.09$$

with $\pi = [0, 0, 0, 0, 1]$.

Since the consensus value depends only on the initial state of the sink node d , the variance of the consensus equals the variance of the opinion of d.

Instead, considering a new graph $G2$ shown in Fig 7, obtained by removing the edges (c, b) and (d, a). We obtain a graph not strongly connected, with two component in the sink of the condensation graph: (c, d). Using

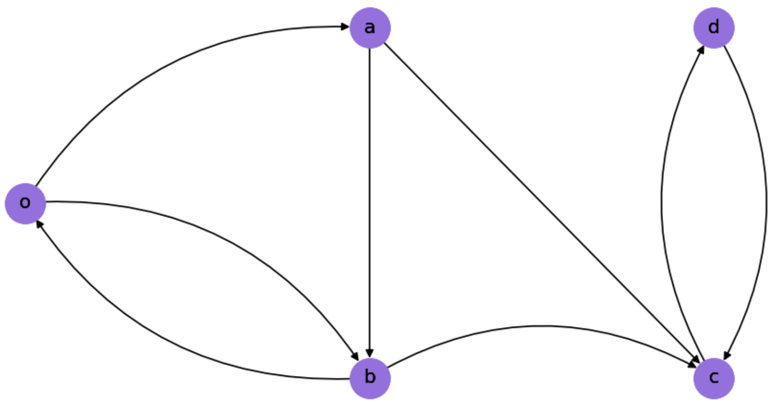

Figure 7: Graph without the edges (c,b) and (d,a).

the function `can_reach_consensus` we can see that the sink component of the graph is not aperiodic. If this happens, a consensus will (generally) not be reached. It could be reached **only** in the cases where the initial opinions of the sink components coincide,i.e,:

$$x_c(0) = x_d(0)$$

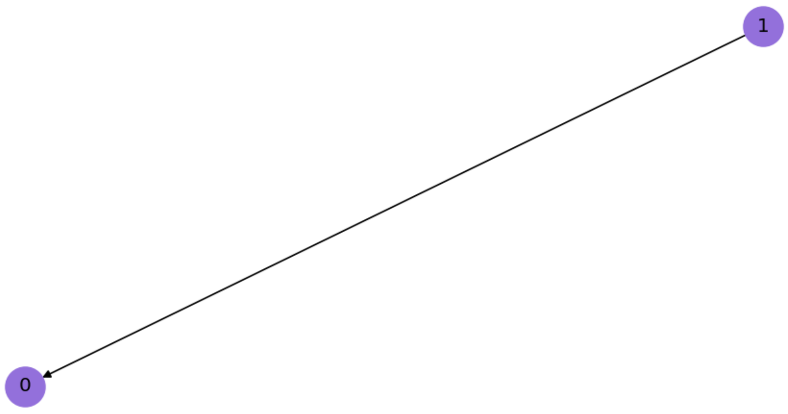

Figure 8: The condensation graph of the second modified graph.

2 Exercise 2

Consider the graph in Fig. 9 with weights matrix according to Λ (Eq. 1). If many particles move around in

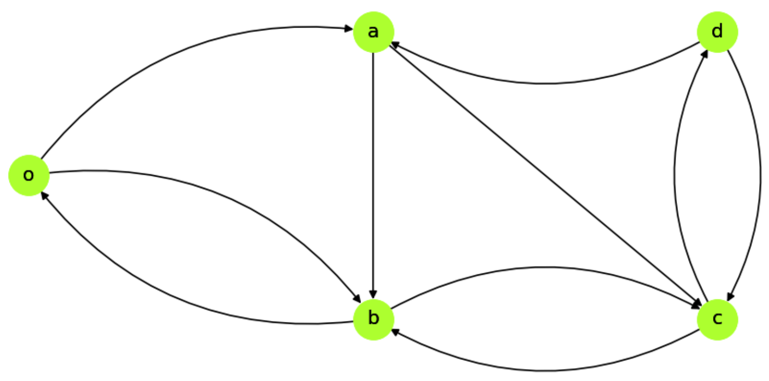

Figure 9: Graph of the exercise.

the network in continuous time, the time that each of them will stay in the node is exponentially distributed, and on average a particle stay $1/w_i$ 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 $\omega = \Lambda\mathbf{1}$. The simulation can be seen from 2 different perspective:

- The **particle perspective**, i.e. "follow the particle",
- The **node perspective**, i.e. "observe from the node".

2.1 Node perspective

If 100 particles all start in node a, the average time for a particle to return to node a is ≈ 6.75 computed with the function `multiple_avg_time`.
As the particles do not influence each other while moving, they take on average the average time taken by only one particle to return to the origin. Indeed, this can be demonstrated with the function used in the previous section 1.1 that compute the average return-time '`average_time_calculator`'.

Both results are really similar. Fig 10 shows the different return-times for each particle.

2.2 Particle perspective

If 100 particles start in node o, and the system is simulated for 60 time units, the average number of particles in the different nodes at the end of the simulation is:

$$\{o : 16, a : 10, b : 23, c : 23, d : 28\}$$

Fig 11 shows the graph with node size and colorproportionate to the number of particles inside them. Fig. 12 shows the number of particles in each node during the simulation time.
This result can be compared with the stationary distribution of the continuous-time random walk followed by the single particles, which is represented by the probability distribution $\bar{\pi}(t)$.

Given a finite state space $\mathcal{X}$, $\bar{\pi}_i(t)$ represents the probability that the chain $\mathbf{X(t)}$ is in the state $\mathbf{i}$ at time $\mathbf{t}$, i.e.:

$$\bar{\pi}_i(t) = P(X(t) = i), \quad i \in \mathcal{X}$$

The probability distribution $\bar{\pi}(t)$ for this graph is:

$$\{o : 0.18, a : 0.15, b : 0.22, c : 0.22, d : 0.22\}$$

It is possible to see that the stationary distribution of the continuous-time random walk is really similar to the average number of particles in the different nodes after 60 time units

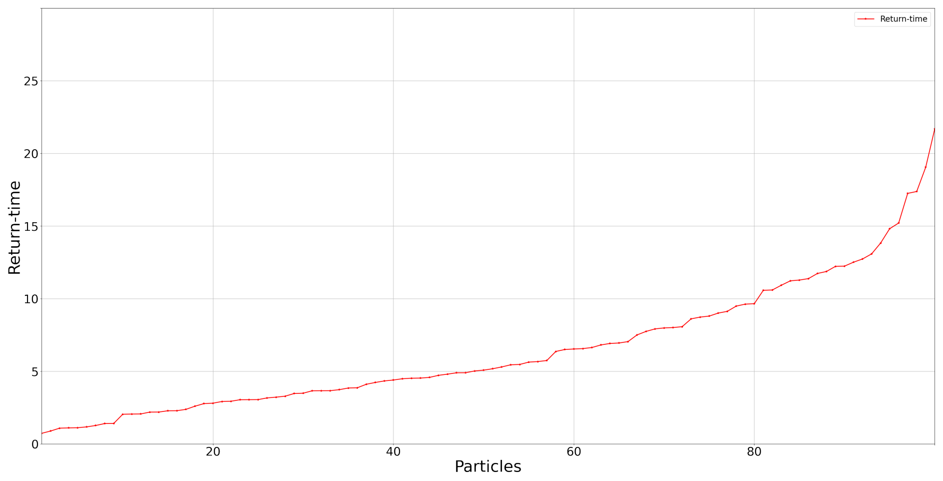

Figure 10: Plot of the different return-times for each particle.

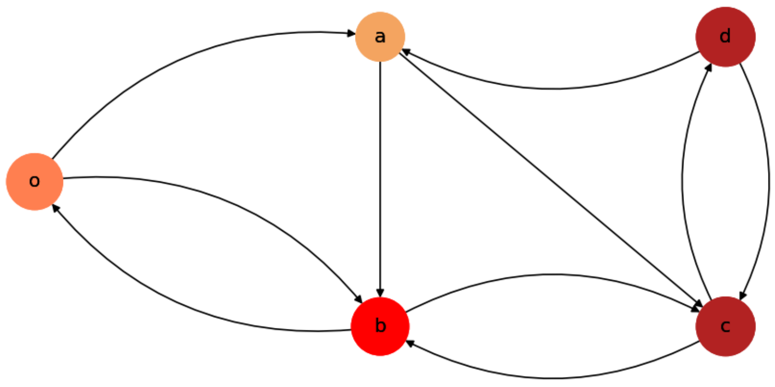

Figure 11: Graph with nodes proportionate to the number of particles inside them.

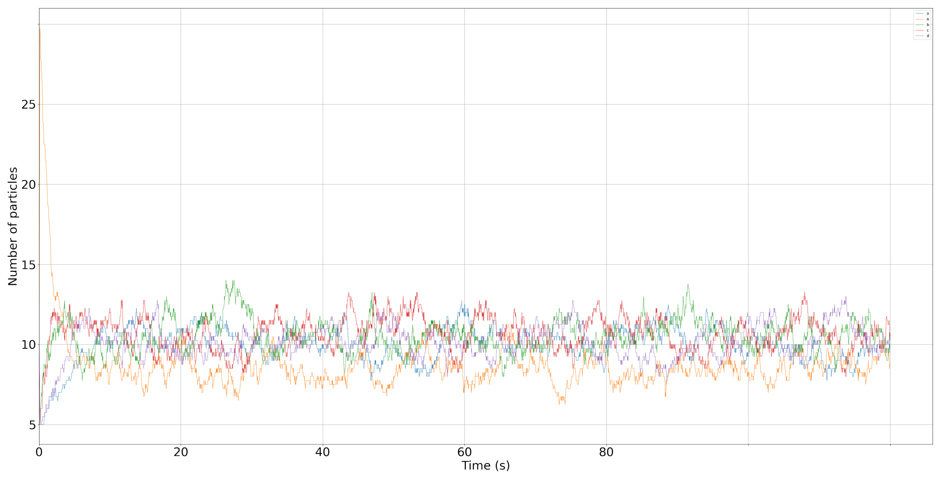

Figure 12: Plot of the number of particles in each node during the simulation time.

3 Exercise 3

Consider the open network in Fig. 13, with transition rate matrix Λ_{open}:

$$\Lambda = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{ccccc|c} o & a & b & c & d & \\ 0 & 3/4 & 3/8 & 0 & 0 & o \\ 0 & 0 & 1/4 & 1/4 & 2/4 & a \\ 0 & 0 & 0 & 1 & 0 & b \\ 0 & 0 & 0 & 0 & 1 & c \\ 0 & 0 & 0 & 0 & 0 & d \end{array}$$

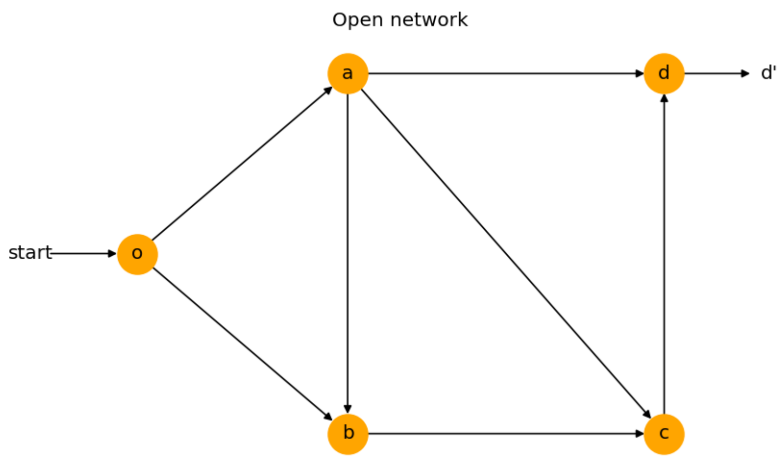

Figure 13: Open network.

Consider the particles enter into the system at node o according to a Poisson process with input rate $\lambda = 1$. Each node will then pass along a particle according to a given rate. Let $\omega = \Lambda\mathbf{1}$ and let $N(t)$ denote the vector of number of particles in each node at time **t**.

3.1 Proportional rate

In this scenario each node 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 $\omega_i N_i(t)$.

A particle will move from one node to another according to its rate:the time it takes to move is a Poisson process, and in this case the time is proportionate to the total number of particles at that time in the node times the ω_i.

Fig. 14 represents the distribution of particles among nodes during a simulation lasting 60 time units:

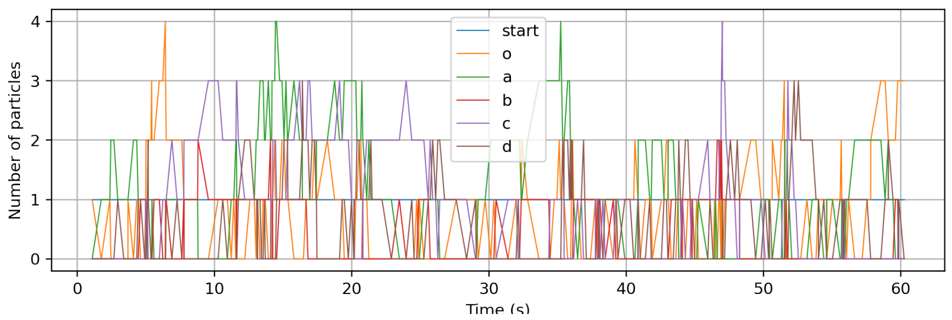

Figure 14: The distribution of particles among nodes.

Since the rate a node passes along particles is proportional to the number of particles in the node, the system can handle any input rate without blowing up. When there will be a great number of particles in the system, a lot of them will move from a node inside the graph and not from the input. For example, with an input rate of 1000 the distribution of particles is the one showed in Fig. 15:

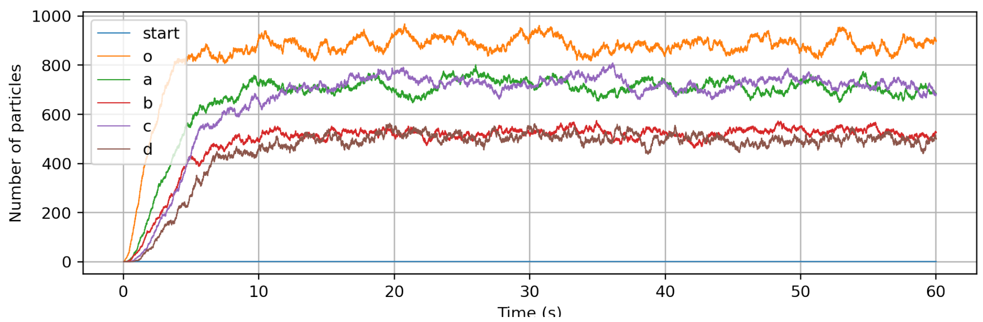

Figure 15: The distribution of particles among nodes with input rate = 1000.

3.2 Fixed rate

In this case, each node i will instead pass along particles with a fixed rate ω_i. Fig. 16 represents the distribution of particles among nodes during a simulation lasting 60 time units:

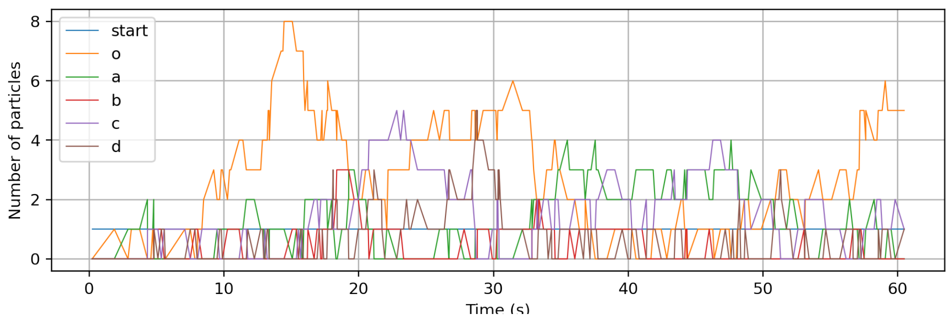

Figure 16: The distribution of particles among nodes.

In this case, the system can handle an input rate of one particle per time unit, because the rate each node passes along particles is greater than or equal to the rate new particles are injected inside the graph, so particles won't stuck on any node. The system blows up for each input rate greater than one, because the rate of node o would be smaller than the input rate, so particles will stuck on this node. For example, we can see how the system behaves with an input rate of 2 particles per time unit:

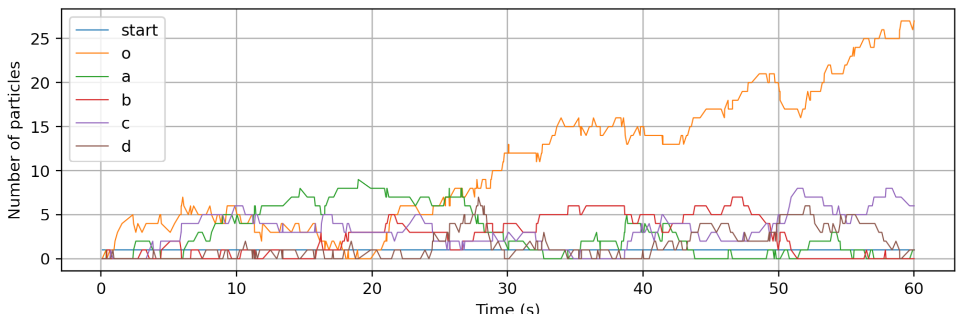

Figure 17: The distribution of particles among nodes with input rate = 2.