Report Homework II

Network Dynamics and Learning

Benfenati Luca
Politecnico di Torino
Torino, Italy
s286582@studenti.polito.it

De Cristofaro Carmine
Politecnico di Torino
Torino, Italy
s291129@studenti.polito.it

Scarano Nicola
Politecnico di Torino
Torino, Italy
s287908@studenti.polito.it

Vacca Alessio
Politecnico di Torino
Torino, Italy
s288240@studenti.polito.it

***Abstract*—The following report addresses three different problems related to random walk and flow dynamics. The aim of this work is to propose solutions to specific problems, dealing with both the theoretical and the practical side of the subject, in order to present an approach which remains as general and broad as possible.**

INTRODUCTION

Before starting with the practical part, we want to provide the theory elements which will be used in the following exercises, so to be, once again, as complete and exhaustive as possible. For each section, a initial part regarding theory is proposed, for then proceed with the specific exercise.

I. PROBLEM 1

In the first part of this problem (section I-A) we deal with a single particle performing a continuous-time random walk. From the theoretical point of view we need to define what is a random walk and all its characteristics.

In the second part of the first problem (section I-B), we are asked to deal with average dynamics, addressing how opinions flow through the network, if and how consensus is reached with different configurations of the given network, checking also the asymptotic behavior of such dynamics.

A. Continuous-time Markov chains

A **Markov chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. It can be designed as discrete-time Markov chain (DTMC), where the chain moves state at discrete time steps, or as continuous-time Markov chain (CTMC). This is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable, ***rate-r Poisson clock*** (equation 1), and then move to a different state as specified by the probabilities of a stochastic matrix.

$$\begin{aligned} &T_0 = 0, \\ &T_k = \sum_{1 \le j \le k} S_j, \qquad k = 1, 2, ... \end{aligned} \tag{1}$$

In other words, a Poisson clock is characterized by the property that the time elapsed between any two of its consecutive ticks is an independent random variable with rate-r exponential distribution. So, the Poisson process can be explained as:

$$N_t := \sup\{k \ge 0 : T_k \le t\}, \qquad t \ge 0 \tag{2}$$

where N_t stands for the number of ticks of the Poisson clock occurred by time t.

Eventually, more in general, given a finite state space *X* and a square non-negative matrix $\Lambda = (\Lambda_{ij})_{i,j \in X \neq 0}$ with entries indexed by elements of *X*, diagonal entries $\Lambda_{i,j} = 0$, $w_i = \sum_{j \neq i} \Lambda_{ij}$ and $w^* = max_{i \in X}\{w_i\}$, we define a continuous-time Markov chain *X*(t) with transition rate matrix Λ and rate-w^* in the following ways:

- given current state i, wait rate w_i-exponential time, then jump to new state j chosen with probability $P_{ij} = \Lambda_{ij}/w_i$
- links are equipped with independent rate-Λ_{ij} Poisson clocks; if (i,j)'s clock ticks at time t, and X(t^-) = i, then jumps to X(t) = j
- rate-w^* Poisson clock $T_0 \le T_1 \le ...$, with Poisson process N_t independent jump chain $U(k), k = 0, 1, ...,$ transition probabilities

$$\begin{aligned} &\bar{P}_{ij} = \frac{\Lambda_{ij}}{w^*}, \qquad i \neq j, \\ &\bar{P}_{ii} = 1 - \sum_{j \neq i} \bar{P}_{ij}. \end{aligned} \tag{3}$$

For the first part of this problem, we are given a directed and weighted graph G shown in Figure 1, and the transition rate matrix Λ, in equation 4 and we are required to simulate a continuous random walk of a particle in the network according to the given Λ matrix.

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

Point a)*: return time*

In this section we compute the average time a particle takes to leave node a and then return to it. This information is obtained by running a simulation of continuous-time Markov chain with 10000 steps. The CTMC in this problem has been modeled as follow:

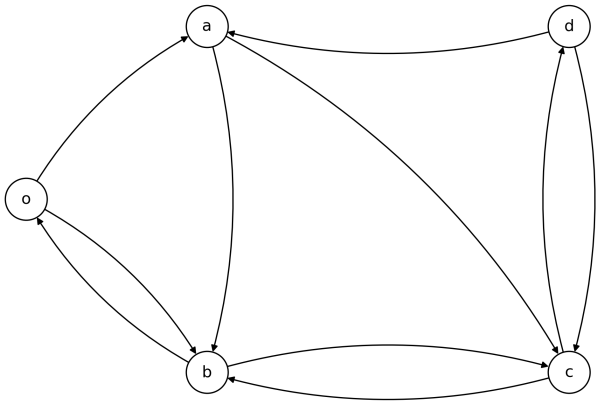

Fig. 1. Initial closed network

1) a "global" Poisson clock with an appropriate rate $\omega^* = \max_i(\omega_i)$ is defined. Such ω_i is defined as the sum over each column of the transition rate matrix, $\omega_i = \sum_{j \neq i} \Lambda_{ij}$ with $i \in \xi$,
2) when we are at node i and the global clock ticks, the transition probabilities to jump to a neighbor j are $Q_{ij} = \Lambda_{ij}/\omega^*$

We set *a* as starting node of the simulation and we take record of the transition time whenever the particle is in node *a*. Then, we compute the difference between two following transition times to get the travel time of the particle and eventually we derive the mean average equal to **6.74 time units**.

***Point b)**: theoretical return time*

The theoretical return-time $E_a[T_a^+]$ is defined as the first time $t \geq 1$ that $\chi(t)$ hits *j*. More in general, for a subset of states $S \subseteq \chi$ we can define the return time on S as

$$T_s^+ := \inf\{t \geq 1 : \chi(t) \in S\} = \min_{s \in S}\{T_s^+\} \tag{5}$$

For any state $i \in \chi$, the expected return times satisfy the equation

$$E_i[T_i^+] = 1 + \sum_{j \in \chi} P_{ij} E_j[T_i] \tag{6}$$

In particular, $E_i[T_i^+]$ is finite if and only if state *i* is reachable from every state $j \neq i$ such that $P_{ij} > 0$. However, since we are dealing with a strongly connected graph, we can exploit the theorem that states that the expected return times satisfies: The theoretical return-time of node *a* has been computed solving

$$E_a[T_a^+] = \frac{1}{\omega_a \bar{\pi}_a} \tag{7}$$

where $\bar{\pi}_a$ is the eigenvalue associated to the node *a*. The result is equal to **6.75 time units**. As we can see this last result is so close to the one computed before because the number of steps for the simulation is large enough to guarantee this alignment.

***Point c)**: from node o to node d*

Point (c) of the assignment focuses on hitting time, in details, it asks to compute the average time that a particle takes to go from node *o* to node *d*. As before we design the CTMC following the approach in section I-A with 10000 steps for the simulation. In this case, every time the particle is in node d, we save the travel time and we let the particle start again from node o. Eventually we compute the mean of the stored times. The result obtained is **8.792 time units**.

***Point d)**: theoretical hitting-time*

The hitting-time represents, in a random walk, the time to go from a node to another one and it is defined on a subset of states $S \subseteq X$ as:

$$T_s := \inf\{t \geq 0 : \chi(t) \in S\} = \min_{s \in S}\{T_s\} \tag{8}$$

The following result asserts the finiteness, both with probability one and in expectation, of the hitting times on every set of states that is reachable from the rest of the states. More precisely, if S is globally reachable, then

$$\begin{aligned} &P_i(T_S < +\infty), \\ &E_i[T_S] < +\infty, \qquad \forall i \in X. \end{aligned} \tag{9}$$

To compute the theoretical hitting time $\mathbb{E}_o[T_d^+]$, we need to compute the expected hitting times to $S = \{o, d\}$, $\mathbb{E}_i[T_S]$, $\forall i \in R = V \setminus S$. Thus, we need to solve the system of equations:

$$\hat{x} = \mathbb{1} + \hat{P}\hat{x} \tag{10}$$

where $\hat{P}$ is obtained from P (normalized weight matrix of the graph) by removing the rows and columns corresponding to the nodes in the set S.

More explicitly, the expected hitting times can be expressed as

$$\hat{x} = (I - \hat{P})^{-1}\mathbb{1} \tag{11}$$

Once we have found those expected hitting times, we finally need to apply the linear relation:

$$\mathbb{E}_o[T_d^+] = \frac{1}{\omega_o} + \sum_j P_{oj}\mathbb{E}_j[T_d^+] \tag{12}$$

The obtained result is equal to **8.78 time units** that is close to the result of point (c) as expected for the high number of steps of the simulation.

B. Linear average flow dynamics

Let us firstly introduce, once again, the theory required to solve the second part of the first problem. In particular, we study and report the theory necessary to deal with opinion dynamics. Consider a network described as a graph $G = (V, E, W)$, where the node set V represents a population of agents, the link set E represents interactions among agents, and the entries W_{ij} of the weight matrix quantify the strength of the influence that agent j has on agent i. Every agent i has a state $x_i(t) \in \mathbf{R}$ that is updated in response to the current states

of his out-neighbors in G according to the linear averaging rule.

$$x_i(t+1) = \sum_{j \in V} P_{ij} x_j(t), i \in V. \tag{13}$$

where P is the normalized adjacency matrix. The equation show how the new state of the agent i, $x_i(t+1)$ coincides with the weighted average of the current opinions $x_j(t)$, of his out-neighbors, each weighted proportionally to the link's weight matrix W.

The equation can be written more compactly as

$$x(t+1) = Px(t) \qquad t = 0, 1, ... \tag{14}$$

An important application of linear averaging dynamics is the opinion dynamics, known as **French-De Groot**, which will largely studied in the following points. According to connectivity and other mild conditions, it can be shown that the dynamics converges to a consensus, i.e. a configuration of states all equal to each others. In particular, if we want to characterize this consensus we need to introduce π as the normalized left dominant eigenvector of P such that $P'\pi = \pi$, which also represent the invariant distribution centrality. The consensus point α can be expressed as a convex combination of the initial states where the weights of this combination are the invariant distribution centralities of the graph itself. The following equation represents this consensus value.

$$\pi' x(0) = \pi' \lim_{t \to +\infty} x(t) = \alpha \pi' \mathbb{1} = \alpha. \tag{15}$$

For the subsequent points, we will assume by convention that the opinion of node i is influenced by the opinion of node j if $P_{ij} > 0$, i.e., the link (i, j) is to be interpreted as i watching j and updating her opinion based on opinion of j. Now that we have addressed the theory necessary, we can proceed to solve the next points of the problem.

***Point e)**: French-DeGroot dynamics*

Considering the matrix Λ, shown in equation 4, as the weight matrix of the graph $G = (V, E, \Lambda)$, we are asked to simulate the French-DeGroot dynamics on G with an arbitrary initial condition x(0), and find out if the dynamics converges to a consensus state. We know that the theorem for the convergence states that if we assume:

- that the condensation graph of G has only 1 sink ($s_G = 1$);
- the sink component is aperiodic;

then we can say that:

$$\lim_{t \to +\infty} x(t) = \alpha \mathbf{1}, \tag{16}$$

i.e. the dynamics converges to consensus. Considering now our graph G, we note that the is aperiodic and strongly connected: thus $s_G = 1$ and, in particular, the sink component coincides with the whole graph which, again, is aperiodic, thus the dynamic is guaranteed to converge to consensus. The fact that the graph is also strongly connected says that every node is reachable from any other, therefore they update their own opinion in according to the ones of the other nodes. Finally, we can say that, regardless of the initial conditions, the dynamic in question will converge to a consensus.

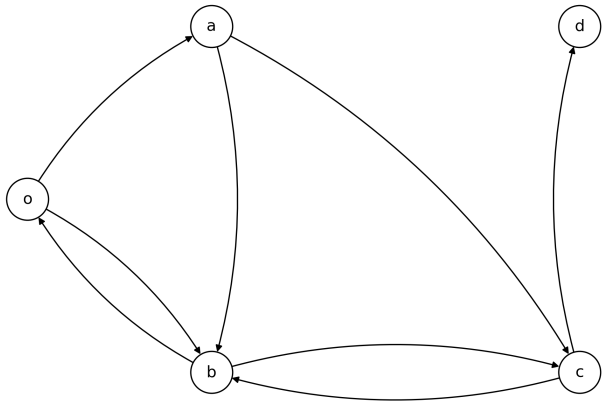

Fig. 2. Network without edges $(d, a), (d, c)$

***Point f)**: variance of the consensus*

In this exercise we assume that the initial state of the dynamics for each node $i \in V$ is given by $x_i(0) = \xi_i$, where the ξ_i are independent and identically distributed zero-mean random variables with variance σ^2.

Since we are assuming that the noises ξ_i are independent with the same variance σ^2, in this case, we can say that the variance of the asymptotic consensus value x satisfies

$$\sigma_{\bar{x}}^2 = \sigma^2 \sum_i \pi_i^2 \tag{17}$$

An important insight that come from the equation above is that the crowd is wiser than any single individual, i.e the crowd is able to reconstruct a more precise estimate of the real state than the single agents of the graph.

To solve this point, we set an arbitrary value for the variance used to generate the initial conditions. We then carry out different simulations to study the behavior of the variance of the consensus the variance of the consensus computed with numerical simulations turns out converge to the variance computed using the equation above for a high number of steps of the simulation.

For example by setting random variables coming from a normal distribution with mean $\mu = 0.0$ and variance $\sigma^2 = 0.5$ as initial condition, the resulting variance is about 0.1 both with numerical simulations and with theoretical results.

***Point g)**: asymptotic behaviour of the dynamics*

It is asked to describe the asymptotic behavior of the dynamics on the graph, but now applying some changes on it. In fact, the edges (d, a) and (d, c) are removed resulting in a graph as the one in figure 2. Due to the changes on the connectivity of the graph, the transition rate matrix Λ has been revised accordingly, in order to take into account the

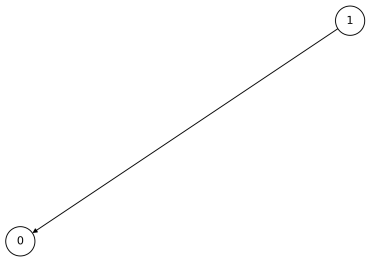

Fig. 3. Condensation graph of network 2

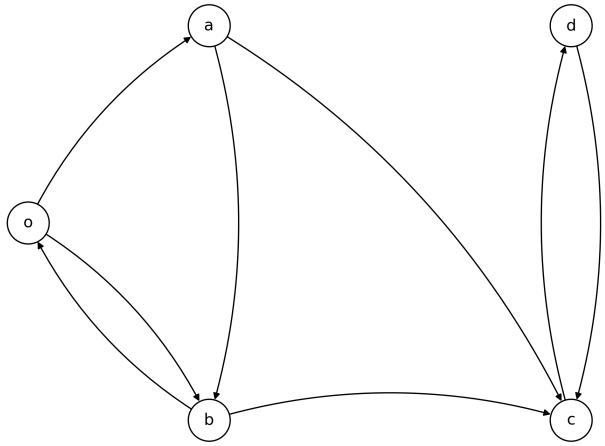

Fig. 4. Network without edges $(c, b), (d, a)$

change of connections between the nodes. In particular, we have set to zero the elements of the row and the column linked to the removed edges. Looking at the condensation graph in figure 3 we see that it has just one sink component, which is represented by the node *d*, and we can easily notice that it is aperiodic. From the theoretical point of view, we know that the presence of one sink component and it's aperiodicity guarantees the consensus. Regarding the initial conditions, we must carefully pay attention to the trapping set. Therefore, in order to reach a consensus is necessary to have a non-zero element corresponding to the initial condition value of the node d. Moreover, we can observe that the invariant distribution centrality π is 0 for all the nodes that do not belong to the sink of the condensation graph. This implies that the initial opinion of the nodes not belonging to the sink component are negligible for the consensus value. Hence, the consensus is reached regardless of the initial conditions, but having at least a non-zero value for the initial conditions corresponding to the nodes in the trapping set.

***Point h)**: French-DeGroot and its asymptotic behaviour*

In the last point of Problem 1, the assignment remarks the concept of asymptotic behavior of the dynamics. As requested, we perform a new change on the initial graph by removing the edges (c, b) and (d, a), resulting as the graph in fig.4. As in the previous point, it is necessary to modify the transition rate matrix Λ accordingly to new changes over the network. The associated condensation graph, reported in fig.5, is made up of two components, one of which is the trapping set formed by nodes *c* and *d*. In particular, it should be noted, that the sink component is not aperiodic this time and so the necessary conditions to achieve the consensus fail to be verified if we do not have both the nodes in the sink component initialized with the same value . In order to challenge the absence of aperiodicity, we define the P_{lazy} matrix by replacing P with $(P+I)/2$. This is equivalent to adding selfloops to each node in the graph with weight equivalent to the degree of the node itself. In addition, as stated before, it is important to set the initial conditions different to 0 for every node belonging to the sink. Complying with these constraints, we let the dynamics to converge to a consensus.

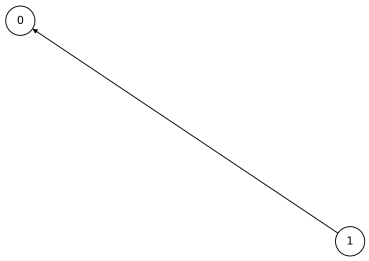

Fig. 5. Condensation graph of network 4

II. Problem 2

In this second exercise we are asked to work with the same network already seen in figure 1, with the same transition rate matrix (equation 4). However, in this case we have to deal with many particles moving around in the network in continuous time. Each of the particles will move around just as the single particle moved in Problem I: it will stay in a node for a time which is exponentially distributed and, on average, it will stay $1/\omega_i$ time-units in a node i before moving to one of its neighbors. Once again, the next node it will visit is based on the probability matrix $P = diag(\omega)^{-1}\Lambda$ where $\omega = \Lambda\mathbb{1}$. We are asked to simulate the system from two different perspective:

1) **particle perspective**, or "follow the particle"; in this case we need to work with single particles and their return time to a specific node. It is almost identical to what we have already seen in Problem 1, but now the simulation is not carried out fixing the number of steps/jumps of a particle in the network, rather we consider the return time of each particle and we compute the average;
2) **node perspective**, or "observe from the node"; in this case we need to monitor the number of particles for each node at the end of the simulation. When doing this you do not have to care about each single particle, but only about how the number of particles in each node changes.

***Point a)**: Particle perspective*

To simulate the particle perspective we decided to deploy a different approach with respect to the one already seen in Problem I (points (a) and (c)). Once again we need to model a continuos time Markov chain, where time flows in a continuum ($t \geq 0$). This time, however, we decided to attach a Poisson clock to each of the particles and move them individually, rather than having a system-wide Poisson clock for all the particles. Each node i is equipped with its own Poisson clock rate $\omega_i = \sum_j \Lambda_{ij}$, and when the particle is in node i and the clock of that node ticks, the particle jump to a neighbor node j with probability $P_{ij} = \Lambda_{ij}/\omega_{ij}$, where P is the conditional probability matrix computed as $P = D^{-1}\Lambda$.

The problem asks us to compute the average return time to node a for a particle, given that we start with 100 particles in node a. Each step of our simulation is summarized by the pseudo-algorithm 1.

Algorithm 1 Particle perspective

Compute cumulative conditional probabilities P_{cum}
for every particle, $1 \rightarrow 100$ **do**
  Set initial position to a
  Pre-compute the time to wait based on initial node a
  while j_{th} particle does not come back to a **do**
    compute next node based P_{cum} of current node
    compute the waiting time to the next transition
    if next node is node a **then**
      save time required
      go to the next particle
    end if
  end while

Running multiple simulations, it is immediate to notice that, differently from what we have already seen in Problem I, this latter result oscillates between **6** and **8 time units**, not properly converging to the theoretical return time computed in point (b) of Problem 1. This may be due to the fact that 100 particles are not enough to get a stable result and, specifically, to converge to the theoretical limit. Notice that in Problem I, we were simulating setting the number of steps/jumps that a single particle in the network was allowed to do. Once we found the random walk of a pre-defined number of steps, we computed the average time between two consecutive passages from node a. We have seen that, in the previous case, considering 10000 steps, the particle travel from node a to node a about 1500 times, which is more than 10 times the number of trips from node a to node a considered in this latter exercise. In this case, we are looking at only 100 particles and, consequently, at 100 trips from node a to node a: it is quite normal that the result obtained is not "stable". This reasoning is supported by the fact that, if I increase the number of particles to 1500 (i.e. similar to what we have seen in point (a) of Problem 1, the average time gets closer and closer to the theoretical value.

TABLE I
AVERAGE RESULTS AT THE END OF THE SIMULATION

	Average # particles
Node o	18.32
Node a	15.19
Node b	21.97
Node c	22.51
Node d	22.01

***Point b)**: Node perspective*

To simulate the node perspective we are asked to consider 100 particles that start in node o and the system is simulated for 60 time units. We could decide to have one Poisson clock attached to each node as for the particle perspective, but since the rate of each node is proportional to the number of particles in the node, this rate would have to change during the simulation. For this reason, we decide to have a system-wide Poisson clock with rate 100. Then, at each tick we randomly, and proportionally to the number of particles in the different nodes, select a node from which we should move a particle. Then a particle from the selected node will move according to the transition probability matrix . Once again, as already seen in Problem I, we exploit as conditional probability matrix Q, where $Q_{ij} = \Lambda_{ij}/\max_i \omega_i,\ i \neq j$ (transition to another node), or $Q_{ii} = 1 - \sum_{i \neq j} Q_{ij}$ (no transition to another node).

We are asked to compute the average number of particles in the different nodes at the end of the simulation, thus we need to run multiple simulations, each one of them runs until the continuous time reaches 60 time units. The simulation is summarized by the pseudo-algorithm 2.

Algorithm 2 Node perspective

compute cumulative conditional probabilities Q_{cum}
for each step, $1 \rightarrow 100$ **do**
  Initialize system putting 100 particles in node o
  Pre-compute the time to wait for next iteration (rate = 100)
  Pre-compute the probabilities of node j of being selected based on number of particles $p_j = \#\text{particles}_j/100$
  while time reaches 60 time units **do**
    Select node **from** which we move a particle(based on p_j)
    Select node **to** which we move particle(based on Q_{cum})
    if node$_{from} \neq$ node$_{to}$ **then**
      update count of particles in both nodes
      re-compute the selection probabilities for each node
    end if
    Store the particles count for each node
  end while
  save number of particles in each node at end of simulation
end for

The simulation has been carried out 100 times, with 100 particles starting each time from node o. Results are reported in table I.

We are then asked to plot the number of particles in each node during the simulation time. To do this, we just consider the results obtained in the last step of the simulation, i.e. from

the 100^{th} run. The behavior of the number of particles for each node during the simulation is reported in figure 6.

As we can clearly see from the graph, after an initial (and obvious) disproportion given by the fact that we start with 100 particles in node o, just after 5 time units the particles are already distributed in a more balanced way. It is fair to notice that, as the results from Table I already anticipated, node a (the orange one in the graph) confirms to have lower number of particles with respect to other nodes.

Finally, we are asked to compare the results obtained with our simulation with the stationary distribution of the continuous-time random walk followed by the single particles. We can compute this distribution solving the linear system:

$$Q\bar{\pi} = \bar{\pi} \tag{18}$$

Solving this system, we find the stationary distribution to be $\pi = [0.19, 0.15, 0.22, 0.22, 0.22]$. Our results are, on the other hand, shown in table I. It is fair to say that the stationary distribution $\bar{\pi}$ represents a sort of probability of a single particle to be in each node. As we can see, our result is comparable with the stationary distribution $\bar{\pi}$ multiplied by 100. This can be easily explained if we consider that we have 100 particles and our result represents the number of particles in each node at the end of the simulation. Thus, it is reasonable to think that the count of particles in each node strictly depends on the stationary distribution $\bar{\pi}$. Thus, we find that the theoretical result supports our simulation results.

III. Problem 3

In the third exercise we are required to analyze the behaviour of the open network when particles are pushed into it at different rates. The objective of this last problem is to understand how different particles affect each other when moving around in a network, always considering the continuous time. The network performances are studied focusing once again on the node perspective (as in the previous exercise), which means that we have to monitor and analyze the number of particles in each node over the time of the simulation. However, now we are required to simulate two different scenarios that differ by what rate the nodes will pass along particles:

1) **proportional rate**, where each node will pass along particles according to a Poisson process with rate equal to the number of particles in the node (i.e. proportional to the number of particles);
2) **fixed rate**, where each node will instead pass along particles with a fixed rate of 1; in this case the rate of the single node does not depend on the number of particles in it.

The network taken into consideration is shown in figure 7 where particles enter the network from node o and exit from node d. To carry out our simulations, we imagined two additional nodes, o' and d', that constitute the source and the sink of our network. In particular, the rate of node o' corresponds to the input rate at which particles enter in the network to node o, while node d' is used to drain the particle out of node d.

For both the scenarios we are asked to find the largest input rate that the system can handle without blowing up, where "blowing up" means that nodes in the network are not able anymore to deal with the increasing number of particles in the network.

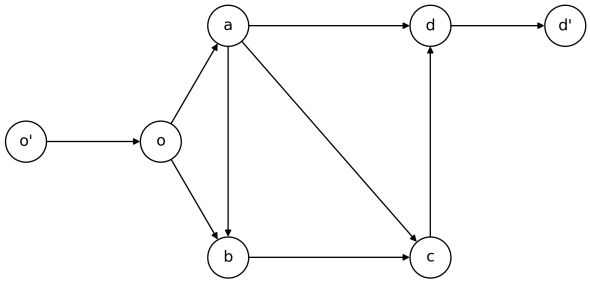

Fig. 7. Open network

To simulate our system we decided to exploit both approaches already seen in previous problem:

- "system-wide" approach, already explained in point (a) of problem I and in point (b) of problem II, exploiting a global Poisson clock whose rate changes with the total number of particles in the system. When this clock ticks we select the node from which we will move a particle based on a probability proportional to the number of particles in each node, while the next node is chosen using matrix Q;
- "node-wide" approach, already explained in point (a) of problem II, exploiting a local clock which each node is equipped with, that depends on the number of particles in that node. When the fastest clock ticks, the node selected move a particle to another node, which is chosen, this time, using matrix P. This second approach is well-suited especially when we have to deal with a bigger network, where number of nodes and link increases, since it does not require the pre-computing of matrix Q.

Let's analyze in details the two different scenarios.

***Point a)**: Proportional rate*

In this first scenario we need to simulate the system for 60 time units, considering nodes that pass particles at a rate which is proportional to their fullness, i.e. the higher the number of particles in the node, the higher the rate of that node. However, the difference with respect to the Problem II is that the number of particles in the network changes during the simulation, due to the openings in the network. Particles are entering the graph from node o, with a rate that is an input parameter (that will be tuned and studied accordingly), and they are exiting from the node d. Carrying out a solution for each one of the approaches previously defined, we simulate the systems using different rates at which particles enter the network, i.e. different input rates:

- input rate = 1
- input rate = 10
- input rate = 100

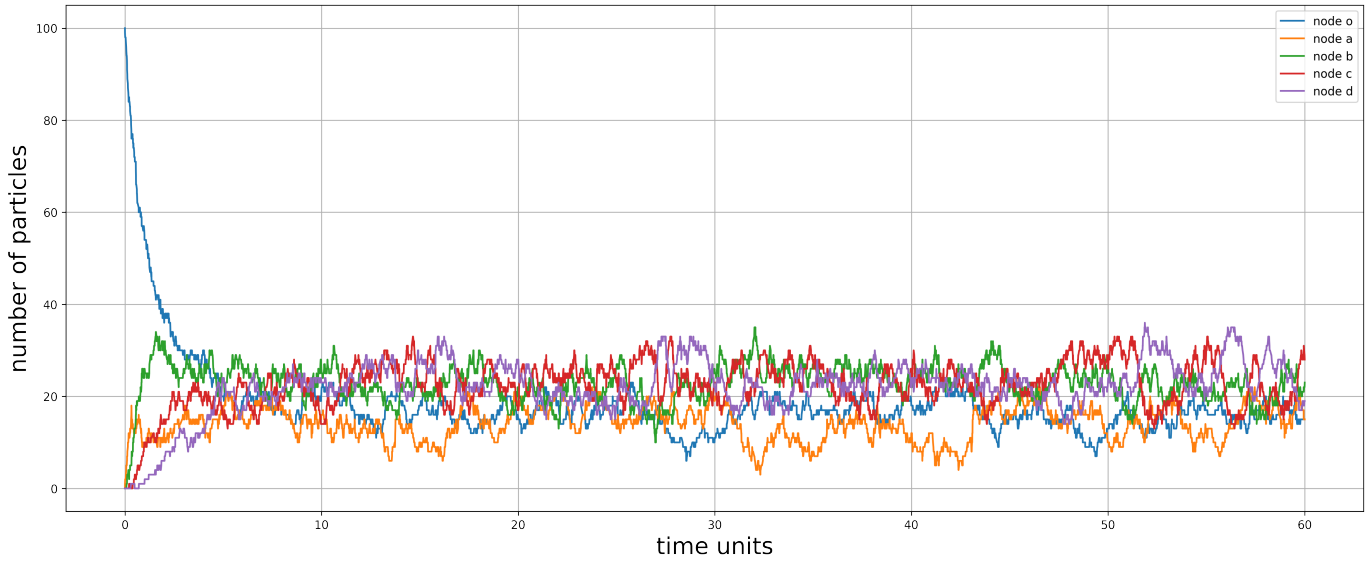

Fig. 6. Evolution of particles in each node over time

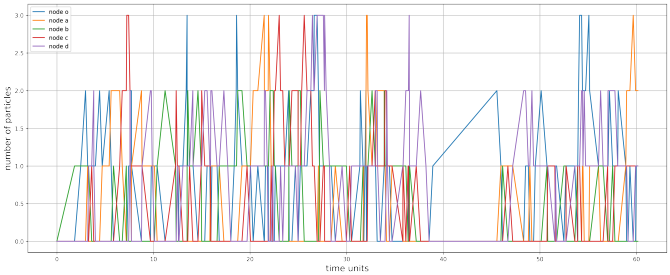

Fig. 8. Evolution of particles in each node (proportional rate - input rate=1)

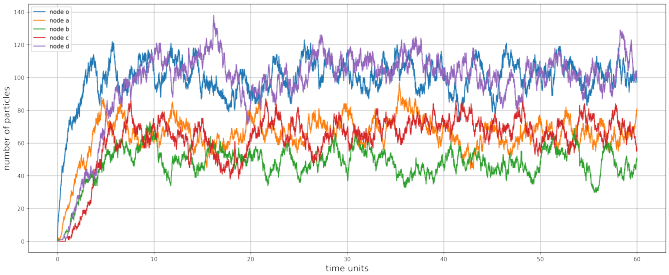

Fig. 10. Evolution of particles in each node (proportional rate - input rate=100)

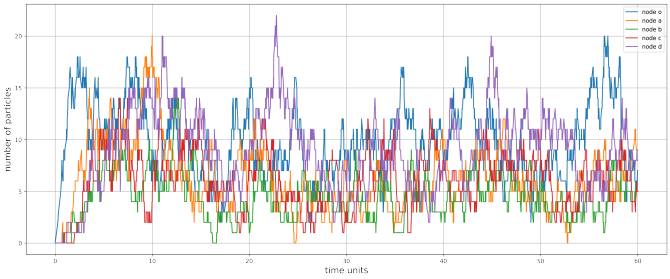

Fig. 9. Evolution of particles in each node (proportional rate - input rate=10)

We study and record the evolution of the number of particles in each node during the simulation and we plot it using a multiple line plot. In figure 8, figure 9 and figure 10 we show the behavior of the network nodes over the time of the simulations, considering different input rates, respectively equal to 1, 10 and 100.

We can see from the plots that, although increasing the rate at which particles enter the network, the fullness of the nodes remains quite stable. In addition, from the third plot, the one where the input rate was set to 100, it is easy to see that each node contain on average a number of particles that is slightly different from the other. In particular, there's no accumulation in any specific node, as we can see also from the table II, where we computed the average number of particles in each node during the simulation. This can be easily explained considering that, at the increase of the input rate, the number of particles inserted in node o (and subsequently in all the other nodes) grows, and with that also the rate at which node o (and equivalently all the other nodes) pass along particles increases accordingly. In this way, nodes are always able to drain the new particles inserted, maintaining a good balance of the network. A last consideration can be carried out looking specifically at figure 10: we can see that, as the table II has already anticipated, there are two nodes that accumulates a little bit more particles than others. We are talking about node o (the blue one) and node d (the purple one): this is reasonable to justify if we think that those are the nodes where particles, respectively, enter and exit from the network.

Note that the results displayed in the table can vary a lot since the average is not computed for different simulations of 60 time units each, but we just run the simulation for 60 time units and we look at the results. This values are reported

TABLE II
RESULTS AT THE END OF THE SIMULATION (RATE=100)

	Average # particles
Node o	99.80
Node a	66.13
Node b	49.88
Node c	62.05
Node d	97.01

here just for the sake of completeness, to show that no evident accumulation is experienced during the simulation.

Point b): Fixed rate

In this second scenario we simulate again the system for 60 time units, considering nodes that pass particles at a rate which is fixed this time. Particles are then pushed in the system through node o at different rates, in particular we have considered:

- input rate = 1
- input rate = 2
- input rate = 5

Once again, we simulate the node perspective with the same strategy used in the previous exercise, carrying out solutions for both approaches (system and node-wide). Results for the different input rate are displayed in figure 11, 12 and 13.

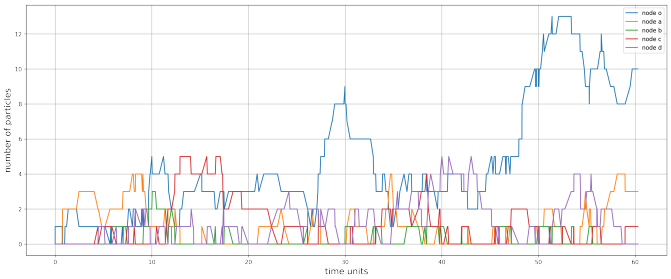

Fig. 11. Evolution of particles in each node (fixed rate, input rate=1)

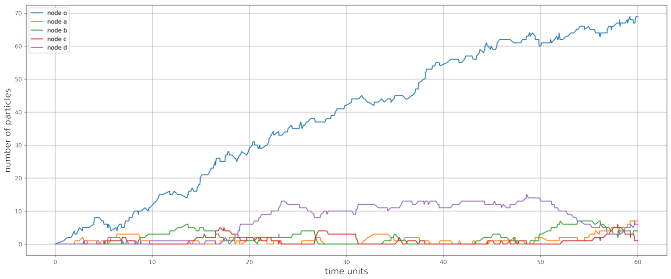

Fig. 12. Evolution of particles in each node (fixed rate, input rate=2)

From the graphs, it is immediate to notice how the system is not able to deal with an increasing input rate. Consider that, at the increase of the input rate, it increases the probability that the "input" clock ticks several times before one of the clock of the nodes. If this happen, a lot of new particles will be inserted in the network through node o, which will not

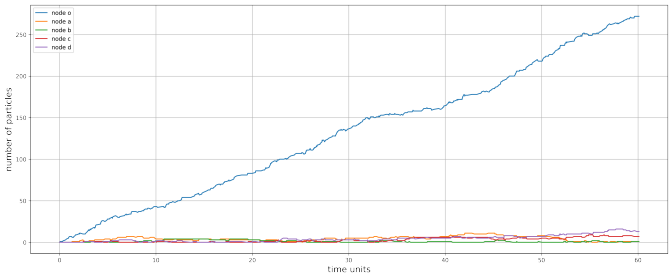

Fig. 13. Evolution of particles in each node (fixed rate, input rate=5)

be able to "drain" them. This behavior can be easily spotted when input rate is greater than the fixed rate of the nodes (figure 12 and 13). The phenomenon is perfectly explainable if we consider that, with input rate = 2 for example, we have a number of particles that enter in node o which is twice as much the number of particles that node o can pass along, since its rate is fixed to 1. The accumulation of particles in node o is clearly explained by this fact: the input clock which determines when a new particle is inserted in the network ticks twice as much as the clock of node o that decide when node o will send out a particle. This statement become even more evident when input rate increases to 5, and so on.

CONCLUSION

We addressed three different problems dealing with random walk and opinion dynamics. For what concern the first topic, we simulated a continuous-time random walk of one particle in the given network, studying the time required to travel from a node to another, confronting it with the theoretical return time. Then, we have also simulated the movement of multiple particles in the same network, understanding once again how much time is required for those particle to travel from a node to another (i.e. the so-called "particle perspective") and, in addition, how those particles redistribute among nodes (i.e. "node perspective"). We then studied how particles affect each other while moving in the network: we simulated how the network deals with increasing input rate of particles, when nodes pass along particles with a rate proportional to the number of particles and when the rate is fixed.

For the second topic, we studied how agents exchange opinions in the network, and how the role of the configuration of the graph plays is crucial in establishing whether the opinion converges or not. We also studied how the asymptotic behavior of the French-DeGroot dynamics, based on different initial conditions and different configurations of the graph.

REFERENCES

[1] G. Como and F. Fagnani, "Lecture on Network Dynamics"
[2] L. Benfenati, "Homework 2 - ND&L, Jupyter Notebooks (.ipynb)"
[3] C. De Cristofaro , "Homework 2 - ND&L, Jupyter Notebooks (.ipynb)"
[4] N. Scarano, "Homework 2 - ND&L, Jupyter Notebooks (.ipynb)"
[5] A. Vacca, "Homework 2 - ND&L, Jupyter Notebooks (.ipynb)"
[6] Networkx manual v. 2.6.2 - https://networkx.org/