Analytically:
Assumption: All initial states xi(0)=Ei are i.i.d. random variables with variance σ

Network Dynamics and Learning

Homework 2

Ali Ghorbanpour
Student ID: s309992

Niusha Parsa
Student ID: s312923

Sina Hamedani
Student ID: s315495

Markov Chains are mathematical models used to describe systems that undergo transitions from one state to another, where the probability of moving to the next state depends only on the current state and not on the sequence of previous states (this is known as the memoryless property).

Introduction to Continuous Time Markov Chains (CTMC):

Continuous Time Markov Chains (CTMCs) represent an advanced area of study within stochastic processes, uniquely characterized by their evolution in a continuous time framework. This report delves into the intricacies of CTMCs, where time flows without discrete breaks, and transitions between states are governed by the Poisson clock mechanism.

At the core of our exploration is the graph $G = (\mathcal{X}, \Lambda)$, where $\mathcal{X}$ is a set of discrete states and Λ denotes the weight matrix that outlines the potential state transitions. In CTMCs, these transitions occur at random times, determined by the ticks of the Poisson clock, which are distributed according to an exponential distribution. This is captured by the equation:

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

where u is a uniformly distributed random variable and r is the rate of the Poisson clock.

Our exploration divides into two methodological approaches for modeling CTMCs:

1. **Global Poisson Clock Approach**: In this method, we introduce a single global Poisson clock with a rate ω^*, resulting in the discretization of continuous time. This approach utilizes a matrix Q, known as the jump chain of the CTMC, to describe transitions. The jump probabilities are given by $Q_{ij} = \frac{\Lambda_{ij}}{\omega^*}$ for $i \neq j$, and $Q_{ii} = 1 - \sum_{i \neq j} Q_{ij}$.

2. **Individual Poisson Clock Approach**: In this method, each node is equipped with its own Poisson clock with a rate $\omega_i = \sum_j \Lambda_{ij}$. The transition probabilities are defined as $P_{ij} = \frac{\Lambda_{ij}}{\omega_i}$.

Both approaches provide unique perspectives on modeling the dynamics of CTMCs, and we will thoroughly analyze them through mathematical formulations and simulations.

This report will delve deeply into the mathematical foundations of CTMCs, exploring properties such as the memoryless nature of the exponential distribution and the formulation of invariant probability vectors. Our goal is to elucidate the complexities of these Markov chains, highlighting their applications and significance in modeling dynamic systems in continuous time.

Problem1.

The first part of this assignment involves studying a single particle performing a continuous-time random walk on the network depicted by the graph in Fig. 1, using the following transition rate matrix:

Transition Rate Matrix

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

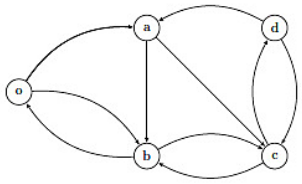


Figure 1: Closed network in which particles move according to the transition rate matrix (1).

Your task is to simulate the continuous-time movement of a particle within the network based on the transition rate matrix (1).

Solution:

In this exercise, we aim to simulate a continuous random walk within the provided network graph. Our task is to simulate the particle's movements in this network using the given transition rate matrix. To do this, we will display the graph according to the inputs, with Λ as our transition matrix, and apply the second approach mentioned earlier.

For our simulation, we need to define certain metrics and vectors. Based on our transition matrix, we will calculate the transition rate matrix as follows for each node ω_i:

$$\forall i, j \in \mathcal{X}, \omega_i = \sum_j \Lambda_{ij} \tag{1}$$

In the next step, the conditional probabilities of moving between nodes i and j, denoted as Q, and staying at the current node, which is $1 - Q$, need to be calculated. We will use a single-position clock for each node with a rate equivalent to ω_i, which represents our transition rate.

The matrix P is defined as a conditional probability matrix that represents the likelihood of moving between nodes i and j, given that the position clock ticks at node i. Specifically, P_{ij} is calculated as follows:

$$P_{ij} = \frac{\Lambda_{ij}}{\omega_i} \tag{2}$$

We will compute the invariant probability vector $\bar{\pi}$ of the CTMC by determining the leading eigenvector of the matrix Q'. This vector represents the transition from element i to node i. Overall, our objective in this simulation is to move according to the position clock with rate ω_i for each node i in the network.

• **Part(A)**: What is, according to the simulations, the average time it takes a particle that starts in node b to leave the node and then return to it?

In Section A, the objective is to simulate the average time it takes for a particle to return to a specific node in a network. This simulation is based on a continuous-time random walk, defined by the transition rate matrix Λ. The process involves calculating the cumulative sum of the modified transition rates matrix Q, which is derived from Λ and normalized by the maximum transition rate w^*. The simulation employs a stochastic approach to determine the time intervals between transitions and the subsequent nodes visited by the particle. The average return time is then computed over a large number of simulations. The output provides the average time for a particle to return from a specified starting node to the same node, offering insights into the network's dynamics and the efficiency of information dissemination or particle movement within the network.

In this part we rely on random walk and large number of simulation for getting the return time

But in part 'b' we rely on the formulation, specially P(hat) and the w(hat). This functions(P(hat) and w(hat)) exclude the starting node from the previous parametrs.

Average travel time for a round trip starting and ending at node b: 6.707553801345337

• **Part(B)**: How does the result in a) compare to the theoretical return-time $E_b[T_b^{(+)}]$? (Include a description of how this is computed.)

Section B compares the average return time obtained from the simulations in Section A with the theoretical return time. The theoretical return time is calculated by solving a linear system that involves the restricted normalized transition matrix $\hat{P}$ and the vector $\hat{w}$. The expected return time is given by the sum of the inverse of the leaving rate from the start node and the dot product of the transition probabilities with the hitting times to the set S.

$$E_b[T_b^{(+)}] = \frac{1}{\omega_b \bar{\pi}_b}$$

$$\hat{x} = \mathbf{1} + \hat{P}\hat{x}$$

Return time (Theoretical): 6.708333333333333
Simulation and theory Difference: 0.0007795319879964069

• **Part(C)**: What is, according to the simulations, the average time it takes to move from node o to node d? Exactly same procedure like the part A, with different parameters (for starting node and the destination node.).

In Section C, the focus is on determining the average time for a particle to travel from one specific node (node 'o') to another (node 'd') in the network. This is done by modifying the simulation technique used in Section A to account for different starting and ending nodes. The goal is to gain insights into the connectivity and efficiency of paths between various nodes in the network.

The output provides the average time taken for a particle to move from node 'o' to node 'd', emphasizing the network's structure and the impact of different paths on the movement of information or particles.

Estimated average time for travelling from node 'o' to 'd' : 10.678613306870794

• **Part(D)**: How does the result in c) compare to the theoretical hitting-time $E_o[T_d]$? (Describe also how this is computed.)

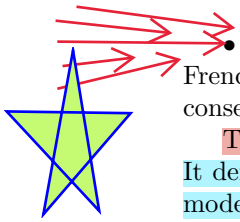
Section D focuses on calculating the theoretical hitting time from node 'o' to node 'd', using a method similar to that in Section B. The expected hitting times are determined by solving a linear system involving $\hat{P}$ and $\hat{w}$. This theoretical result is then compared with the simulation results from Section C to validate the accuracy of the simulated model.

The difference of the part D and the part B is that, in part B we only exclude the starting node, but here we only insert the rows and columns of the reachable_nodes

$$\mathbb{E}_o[T_d] = \left(\mathbf{I} - \hat{P}\right)^{-1} \frac{\mathbf{1}}{\hat{w}} \tag{3}$$

The output presents the calculated hitting time and the simulation error, offering insights into the effectiveness of the paths between nodes.

Calculated theoretical hitting time (o to d): 10.766666666666666
Difference between simulation and theory: 0.0880533597958717



• **Part(E)**: Interpret the matrix Λ as the weight matrix of a graph $G = (\nu, E, \Lambda)$, and simulate the French-DeGroot dynamics on G with an arbitrary initial condition x(0). Do the dynamics converge to a consensus state for every initial condition x(0)? Motivate your answer.

This section is crucial for understanding the linear averaging model in the context of network dynamics. It demonstrates that, under certain connectivity conditions, consensus will be the eventual outcome of the model. The consensus value, which is the weighted average of the agents' initial opinions $x_i(0)$, corresponds to the invariant distribution π_i of the graph. This correspondence highlights the significance of the invariant distribution centrality π in a dynamic context. When the graph is strongly connected, it has a single sink that is reached aperiodically, resulting in convergence to the consensus state $x(t)$. This state represents the sole sink of the graph, emphasizing the model's tendency to achieve uniformity across the network.

We implement the French-DeGroot dynamics to determine if the states of all nodes converge to a consensus value or not.

The dynamics are likely to converge to a consensus state for other initial conditions as well, given the properties of the transition matrix and the graph structure, though this would need to be verified with additional simulations or a formal proof.

For convergence to a valuewe need to consider the properties of the transition matrix PMatrix:

Stochastic Matrix: If PMatrix is a stochastic matrix (each row sums to 1), it represents a probability distribution nd ensures that the state values remain within a bounded range.

Type text here

Consensus state: [0.315 0.315 0.315 0.315 0.315]
Consensus value: 0.3153846153846155

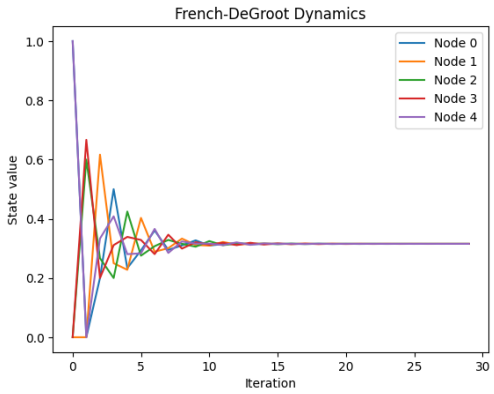


Figure 2: French-DeGroot Dynamics

The plot illustrates the French-DeGroot dynamics over 30 iterations, showing the evolution of state values for five nodes. Initially, the nodes exhibit diverse state values, but they converge towards a consensus state over time, demonstrating the dynamics of opinion formation within the network.

• **Part(F)**: Assume that the initial state of the dynamics for each node i $\in \nu$ is given by $x_i(0) = \xi_i$, where $\xi_{i i\in\nu}$ are independent random variables with variance

$$\sigma_a^2 = \sigma_b^2 = \sigma_1^2 = 1, \quad \sigma_d^2 = \sigma_2^2 = 2$$

For this part we use French-DeGroot Dynamics for ensuring convergence.

Compute the variance of the consensus value, and compare your results with numerical simulations.

In this section of the exercise, the emphasis is on calculating the variance of the consensus value in the French-DeGroot dynamics given random initial conditions. This analysis is essential for understanding the stability and robustness of the consensus state under varying initial conditions. The equation used to compute variance is $\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$, where x_i are the final states from simulations and $\bar{x}$ is their mean. This section integrates both analytical and numerical methods to examine the dynamics' sensitivity to initial conditions, offering a thorough view of the model's behavior.

Also for attaining the variance in our code we use updating the state of the nodes by multiplying the current state by the transition matrix 'PMatrix'.

Calculated Simulated Consensus Variance: 0.019925820512378654
Theoretical Expected Consensus Variance: 0.017465483234714013

• **Part(G)**: Remove the edges (d, a) and (d, c). Describe and motivate the asymptotic behavior of the dynamics. If the dynamics converges to a consensus state, how is the consensus value related to the initial condition x(0)? Assume that the initial state of the dynamics for each node i $\in \nu$ is given by $x_i(0) = \xi_i$, where $\{\xi_i\}_{i\in\nu}$ are i.i.d random variables with variance σ^2. Compute analytically the variance of the consensus value.

If the dynamics converge to a consensus state, the consensus value is a weighted average of the initial conditions of all the nodes. we can understand this by this formula: Consensus value = Sum(i) (consensus vector(i). xi(0))

This section explores the impact of structural changes to the network, specifically the removal of certain edges, on the French-DeGroot dynamics. The analysis focuses on how these alterations affect the convergence to a consensus state and its relationship with the initial conditions. By re-evaluating the dynamics under these new structural conditions, we can observe how changes in the network's topology influence the overall behavior and stability of the consensus process. The variance of the consensus state under these modified conditions is also calculated, providing insights into the model's adaptability to network modifications.

This is the randomly generated initial state for the 5 nodes in the network before running the dynamics. Each node starts with a value between 0 and 1, which are the initial conditions for the French-DeGroot dynamics simulation.

Initial Random Condition: [0.3381 0.083 0.7417 0.4245 0.0732]
Asymptotic State: [0.0732 0.0732 0.0732 0.0732 0.0732]

indicating that the system has reached a consensus state(after 70 iterations of the dynamics.).

The dynamics will still converge to a consensus state for the remaining connected nodes because the remaining subgraph (excluding node 'd') forms a strongly connected component. Node 'd' will not significantly affect the consensus value due to the minimal self-loop weight.

The consensus value is the weighted average of the initial conditions of the connected nodes. Since node 'd' has minimal influence, its initial condition will have a negligible effect on the consensus value.

Analytically:
Assumption: All initial states xi(0)=Ei are i.i.d. random variables with variance.
Calculation: The variance of the consensus value can be computed using the consensus vector v:

Var(consensus value) = Sum(i=1)(n-1) v(i)^2 σ^2

Here, v is the consensus vector obtained from the modified transition matrix P(modified), and n-1 represents the number of connected nodes excluding node 'd'.

Modified Consensus State Variance: 0.08599982629394877

• **Part(H)**: Consider the graph (ν, E, Λ), and remove the edges (c, b) and (d, a). Analyze the French-DeGroot dynamics on the new graph. In particular, describe how the asymptotic behaviour of the dynamics varies in terms of the initial condition x(0), and motivate your answer.

In this final section, the exercise further modifies the network's structure by removing additional edges. The objective is to analyze the resulting changes in the French-DeGroot dynamics, particularly how different initial conditions affect the convergence pattern and the asymptotic behavior. This analysis provides a deeper understanding of the dynamics' sensitivity to network topology and the conditions under which the system stabilizes or diverges. The exercise demonstrates the complex interplay between network structure and dynamic processes, highlighting the importance of understanding both in the context of network analysis.

This final part of the exercise involves analyzing the French-DeGroot dynamics on a network that has been modified by additional edge removals. The aim is to understand how varying initial conditions $(x(0))$ influence the dynamics' convergence pattern and the asymptotic behavior. This analysis sheds light on the dynamics' sensitivity to both network topology and initial conditions, revealing the complex interplay between these factors.

Methodology: The `evolveStates` function was used to simulate the dynamics with diverse initial states, allowing observation of the resulting convergence patterns. The simulation tracks the evolution of the system over 30 iterations, providing insights into the long-term behavior of the network.

Observations:

- The initial state varied significantly across nodes, with some starting at higher values (e.g., 0.8683 and 0.7973) and others much lower (e.g., 0.0181).
- The final state showed a more balanced distribution of values across the nodes, though not entirely uniform, indicating a trend toward equilibrium.
- Nodes with higher initial values maintained a relative advantage in the final distribution, suggesting the lasting influence of initial conditions on the network's dynamics.

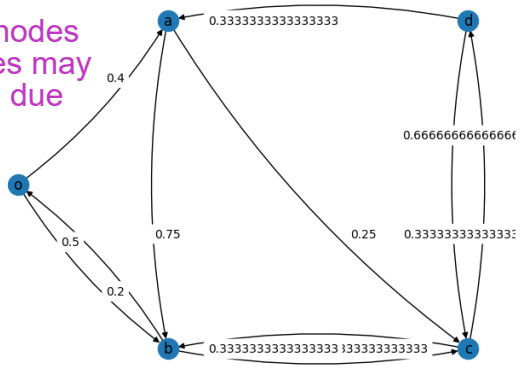
Example Simulation Results:

The initial state values x(0) will still start as random values.

Initial Random Condition $x(0)$: [0.8683, 0.7973, 0.5393, 0.197, 0.0181]
Final State after Evolution: [0.1007, 0.0938, 0.1557, 0.0181, 0.197]

Nodes with fewer connections will have less influence from other nodes.

The final state values indicate how the nodes have influenced each other. Some nodes may not fully converge to a consensus value due to the modified connectivity.



The dynamics might still converge to a consensus state, but the consensus value will be more dependent on the initial conditions of the nodes that remain well-connected.

Figure 3: Reconstructed network graph with modified edge weights.

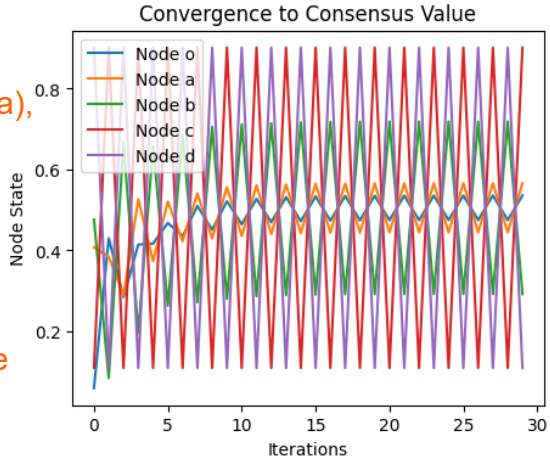
Motivation:
The edges represent the transition probabilities between nodes, adjusted based on the given transition rate matrix.

Less Information Exchange: By removing edges, the information (state values) exchange between nodes is reduced, leading to a slower convergence rate or a different consensus value.

Dependence on Initial Conditions: Nodes that are less connected or isolated will retain their initial state values more strongly, influencing the overall consensus value.

Answer to the Question: With these edges removed, the convergence behavior changes significantly:
1. Initial Condition Influence: The consensus value becomes more dependent on the initial conditions of the remaining well-connected nodes.
2. Connectivity: Nodes with reduced connectivity will have a diminished influence on the overall consensus, and their final states may remain closer to their initial values.
3. Slower Convergence: The dynamics might converge more slowly due to fewer pathways for state values to propagate throughout the network.

*** By removing edges (c, b) and (d, a), the French-DeGroot dynamics show that the network's convergence to a consensus state is slower and more dependent on the initial conditions of well-connected nodes. This is evidenced by the altered final state values, where certain nodes retain closer values to their initial states due to reduced influence from others.



Markov Chains are mathematical models used to describe systems that undergo transitions from one state to another, where the probability of moving to the next state depends only on the current state and not on the sequence of previous states (this is known as the memoryless property).

Figure 1: Convergence to Consensus value over 30 iterations.

The plot shows the evolution of node states from initial random conditions, demonstrating the dynamic behavior and eventual stabilization of node states within the network.

Problem2.

Section(A): Particle perspective

Similar to problem 1

- *If 100 particles all start in node a, what is the average time for a particle to return to node a?*
- *How does this compare to the answer in Problem 1, why?*

In this section, we simulate the trajectories of 100 particles, each executing a continuous-time random walk through the network. The primary goal is to determine the average time it takes for these particles to return to their initial starting node. For each particle, the time spent at each node is modeled using an exponential distribution, governed by the network's transition rate matrix. The transition time is calculated using the equation $T = -\frac{\ln(U)}{w_i}$, where U is a uniformly distributed random variable and w_i is the departure rate from node i.

Average return time: 6.716753657328

Simulation Error: 0.008420323994666745

The difference in the outputs is due to the inherent variability and randomness in the simulation process. The theoretical calculation provides an exact expected return time based on the transition probabilities and network structure, while the simulation approximates this value through repeated random sampling of particle movements. The small error (0.0014943236205544608) indicates that the simulation is very close to the theoretical value but not exactly the same due to the stochastic nature of the random walk process used in the simulation.

Section(B): Node perspective

- *If 100 particles start in node o, and the system is simulated for 60-time units, what is the average number of particles in the different nodes at the end of the simulation?*
- *Illustrate the simulation above with a plot showing the number of particles in each node during the simulation time.*
- *Compare the simulation result in the first point above with the stationary distribution of the continuous-time random walk followed by the single particles*

This part examines the system from the perspective of individual nodes. We track the dispersion of 100 particles throughout the network, starting from a specific node, over a simulated period. The departure of particles from each node is modeled as a Poisson process, with rates proportional to the

These values represent the number of particles at each node at the end of the simulation. Starting with 100 particles at node o, the particles move through the network according to the transition probabilities defined by PMatrix.

These values are the expected number of particles at each node in the long run, derived from the stationary distribution of the Markov chain represented by PMatrix. The stationary distribution gives the proportion of time a particle spends at each node in the long term.

number of particles at each node. This method offers insights into the dynamics of particle distribution over time. The simulation aims to evaluate the average number of particles at each node at various time intervals and compare these results to the network's stationary distribution.

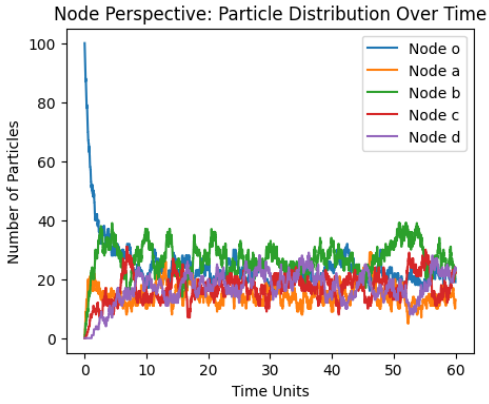
Final Particles Per Node: Node 'o': 19.00, Node 'a': 13.00, Node 'b': 23.00, Node 'c': 22.00, Node 'd': 23.00

Expected Particles from Stationary Distribution: [16.15384615 18.46153846 26.92307692 23.07692308 15.38461538]

Dynamic Behavior: The plot shows the dynamic nature of the particle movement through the network.

Convergence: The system is trending towards a steady state where particle distribution remains relatively stable.

Long-term Expectation: The stationary distribution provides a benchmark for the expected long-term behavior, which the simulation approximates over time.



After an initial period of significant fluctuation, the number of particles at each node starts to stabilize, though some variation remains. This suggests that the system is approaching a steady state but hasn't fully converged within the simulation duration.

Figure 5: Time Evolution of Particle Distribution Across Nodes
This analysis demonstrates the system's behavior over time and the tendency towards a steady-state distribution of particles across the nodes.

Problem3.

In this section, we examine the interactions between particles as they move around a network in continuous time. We focus on the open network depicted in Figure 2, utilizing the transition rate matrix Λ_{open} as defined below.

Stationary Distribution: The stationary distribution of a Markov chain is a probability distribution over the states of the chain that remains unchanged as the chain evolves over time. In other words, if the Markov chain reaches its stationary distribution, then the probabilities of being in each state are stable and do not change with further transitions.

$$\Lambda_{\text{open}} = \begin{array}{c} \begin{matrix} o & a & b & c & d \end{matrix} \\ \begin{pmatrix} 0 & 3/4 & 3/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 2/4 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} o \\ a \\ b \\ c \\ d \end{matrix} \end{array}$$

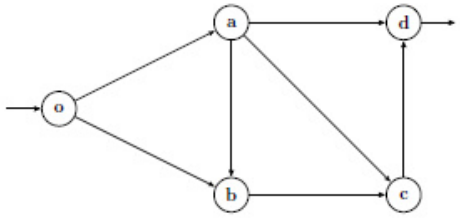


Figure 6: Open network.

Analytically:
For convergence to a valuewe need to consider the properties of the transition matrix PMatrix:
Assumption: All initial states xi(0)=Ei are i.i.d. random variables with variance

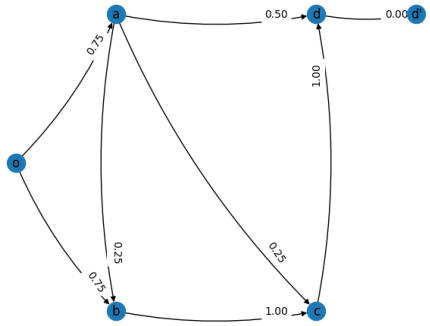


Figure 7: Open network with edge weights.

The simulation aims to test how different input rates of particles affect the stability of the network. Specifically, it checks whether the system can handle increasing input rates without "blowing up," which means reaching an unsustainable state where the number of particles at any node grows uncontrollably.

Section(A): Proportional rate

- *Simulate the system for 60 time units and plot the evolution of the number of particles in each node over time with input rate = 100.*
- *What is the largest input rate that the system can handle without blowing up?*

In Section A, we simulate a network system where the rate at which each node transfers particles is proportional to the number of particles currently present. This scenario, known as proportional rate, necessitates a dynamic approach to accurately simulate particle movements. The simulation spans 60 time units, with the primary objective being to observe the evolution of particle counts in each node and to determine the maximum input rate the system can handle without becoming unstable.

We implement a dynamic network simulation where nodes transfer particles based on their current load over a period of 60 time units.

For the simulation, the core function `simulate_system_a` is used. This function iteratively selects a starting node based on particle distribution, calculates the next transition time, and updates particle counts. The simulation runs until the specified time limit is reached, tracking the evolution of the system.

We test various input rates (ranging from 200 to 800) to identify the threshold before the system reaches an unstable state, defined by the BLOWINGUP_LIMIT_A of 1000 particles. The simulation results, visualized through plots, demonstrate that our system remains stable across the tested rates and does not exceed the blowup limit, leading to the conclusion that the system "does not blow up for any input rate." Additionally, tests for an input rate of 5000 indicate that the network does not reach a blowup state.

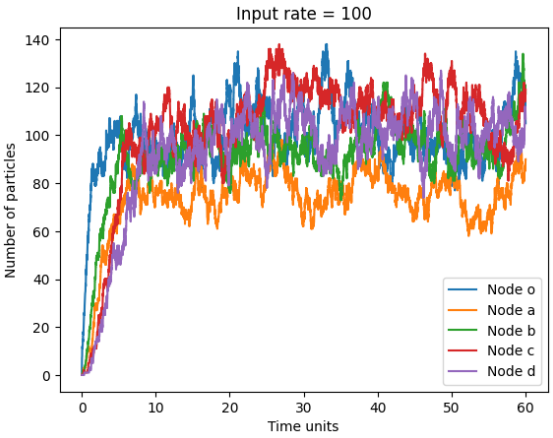
Analyzing the results from the plots, we observe that the system's behavior stabilizes as it manages the influx of particles without reaching critical capacity, highlighting the system's robustness within the specified input rate range.

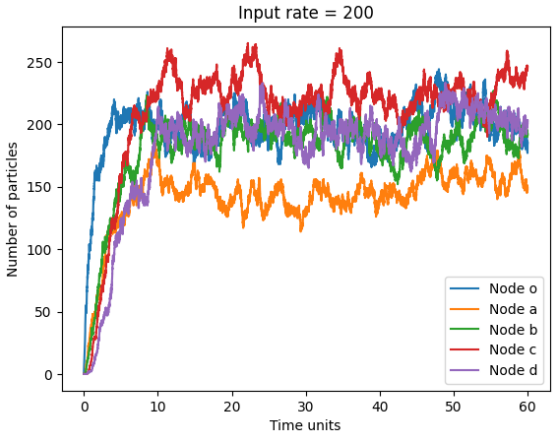
Initialization: The function initializes particle counts, transition events, and history.
Particle Movement: For each time step, particles move between nodes based on the cumulative transition matrix ('CumulativeQ').
Termination: The simulation runs until the specified time units are exceeded.

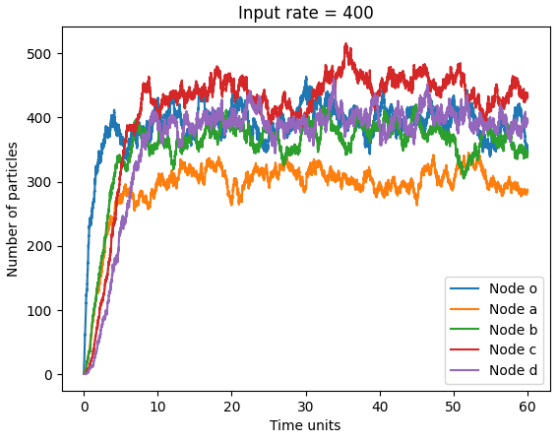
Stable Input Rates:
The results indicate that the system does not blow up for any tested input rate up to 800. This means the network can handle these rates without any node accumulating an unsustainable number of particles.

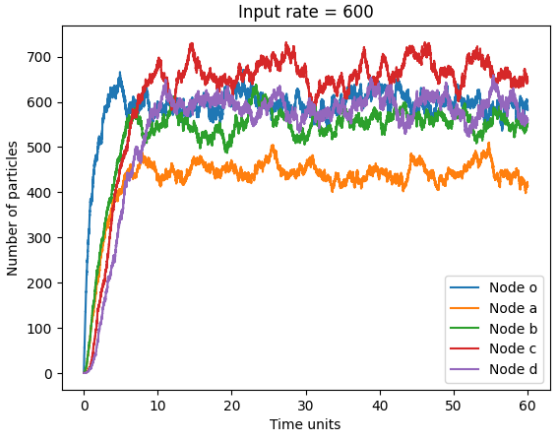
Reasoning:
Network Capacity: The network's structure and transition probabilities are well-balanced, allowing particles to distribute evenly without overloading any specific node.
Convergence: The network reaches a stable state where the number of particles fluctuates around a mean value but does not grow indefinitely.

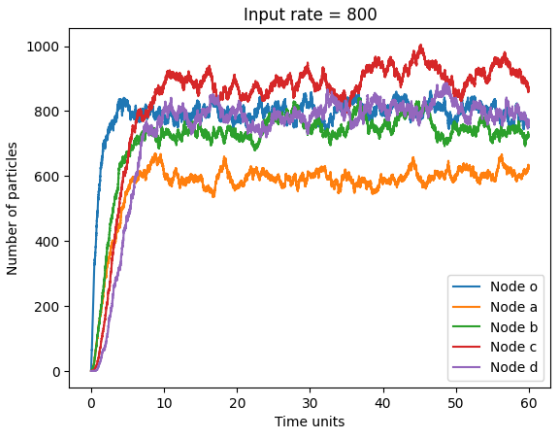
The network demonstrates robustness by handling increasing input rates without any node reaching the blow-up limit. This stability suggests the network's transition probabilities effectively distribute the load, preventing any node from becoming overwhelmed. The simulation helps understand the network's capacity and resilience under varying conditions, ensuring it can operate efficiently even as the input rate increases.









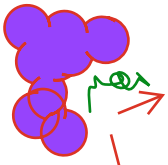


Section(B): Fixed rate

- *Simulate the system for 60-time units and plot the evolution of number of particles in each node over time.*
- *What is the largest input rate that the system can handle without blowing up? Why is this different from the other case?*

This simulation is designed to test the stability of the network under various fixed input rates of particles. It examines whether the system can handle these rates without reaching a state where the number of particles at any node exceeds a set limit (in this case, 30 particles).

Initialization: Initializes particle counts, transition events, and history.
Particle Movement: For each time step, particles move between nodes based on the cumulative transition matrix ('CumulativeQ').
Termination: The simulation continues until the specified time units are exceeded.



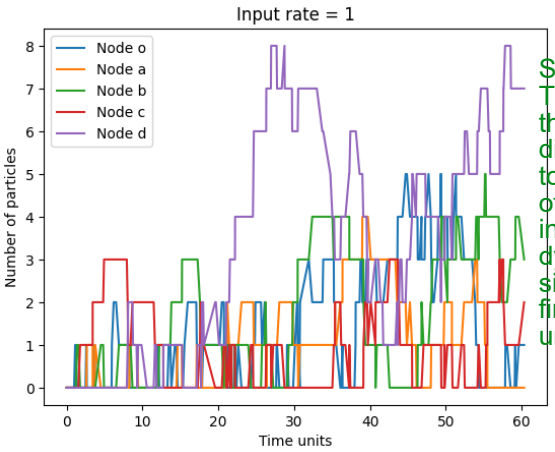
For Section B of Exercise 3, the objective was to simulate a network system under a fixed rate regime. Unlike the proportional model used in Section A, each node in Section B transmits particles at a constant rate of 1 particle per time unit. This setup ensures a predictable and steady flow of particles through the network, simulated over 60 time units. The primary function, `simulate_system_b`, managed the simulation by selecting nodes based on this fixed rate and monitoring the transitions and distribution of particles.

The test input rates ranged from 0.1 to 2.1 in increments of 0.2. The simulations indicated that the system could handle rates up to 1.3 without instability, which was defined by a blowup limit of 30 particles at any node. Rates exceeding 1.3 resulted in exponential growth in particle numbers at node 'o', signifying an unstable system. The plots from the simulations effectively visualized the network's capacity, providing clear evidence of the threshold at which the system transitions from stable to unstable.

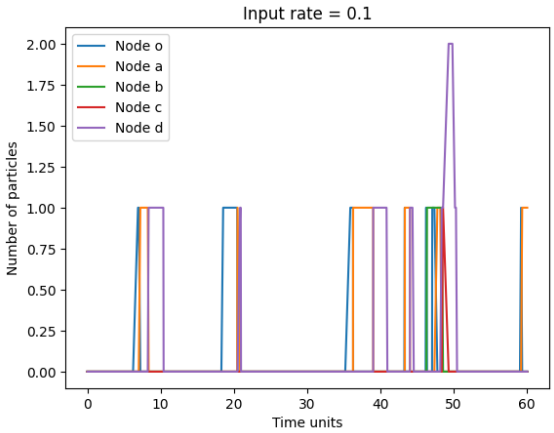
Stable Input Rates:
The results indicate that the system can handle input rates up to 1.3 without blowing up. This means the network can manage these rates without any node accumulating an unsustainable number of particles.

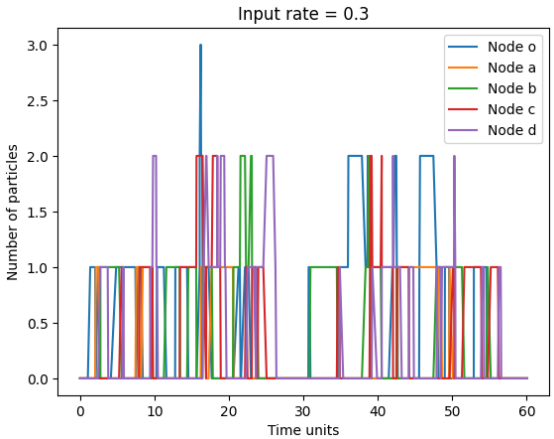
Reasoning:
Network Capacity: The network's structure and transition probabilities distribute the particles effectively, preventing any single node from becoming overloaded.

Convergence: The system reaches a stable state where the number of particles fluctuates but does not grow indefinitely.



Summary:
The largest input rate without blowing up in the second simulation is 1.3. The discrepancy from the first simulation is due to different blowing-up limits, the granularity of tested rates, and the inherent differences in how the simulations handle particle dynamics and node transitions. The second simulation provides a more detailed and fine-grained analysis of the system's stability under smaller input rates.





The network demonstrates robustness by handling input rates up to 1.3 without any node reaching the blow-up limit. This stability suggests that the network's transition probabilities effectively distribute the particles, preventing overload. However, at an input rate of 1.5, the system becomes unstable, exceeding the particle limit at some nodes. This analysis helps understand the network's capacity and resilience under varying conditions.

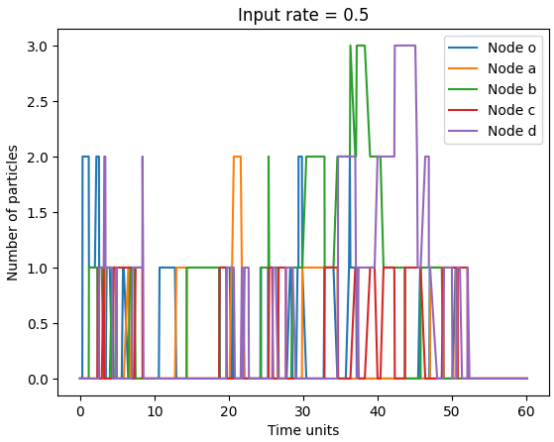
Differences Between the Two Cases:

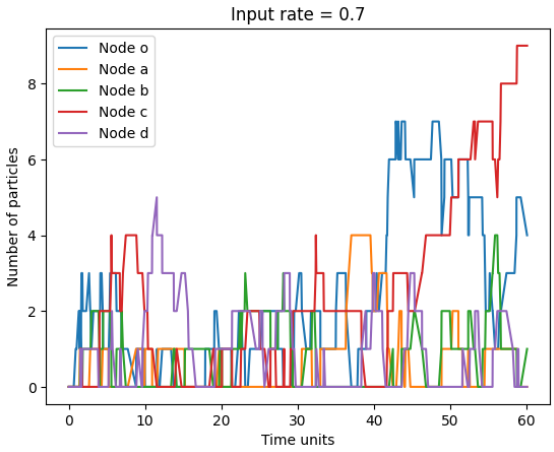
1- Simulation Dynamics:

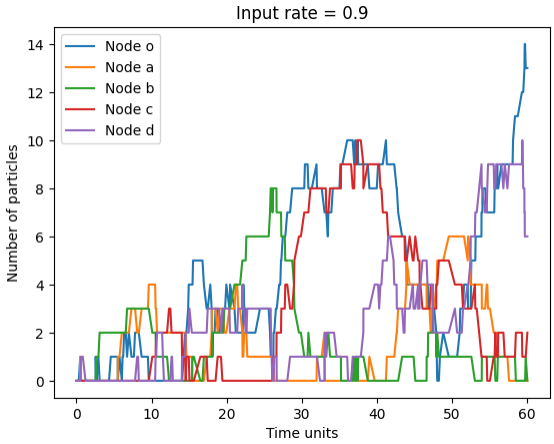
In the first simulation (rate A), the input rates tested ranged from 100 to 800, and the system did not blow up at any of these rates. This indicates a higher capacity to handle particle inflow.
In the second simulation (rate B), the input rates were tested from 0.1 to 2.1 in smaller increments. The system blew up at an input rate of 1.5, showing a lower capacity compared to the first case.

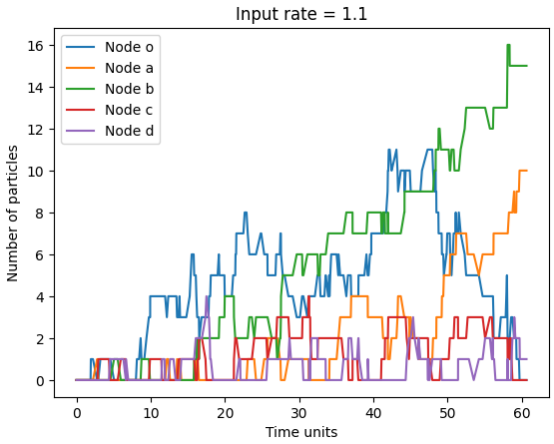
3- Granularity of the Testing Rates:

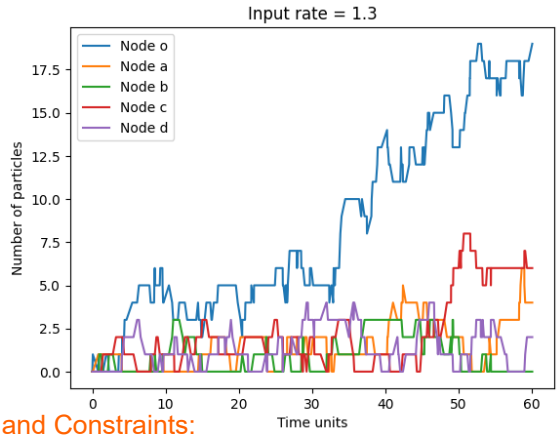
The first simulation tested broader ranges with larger increments, potentially missing smaller-scale instabilities that the second simulation caught by using finer increments.
The second simulation's detailed rate testing reveals the exact point of instability more accurately, explaining the discrepancy in observed maximum input rates.

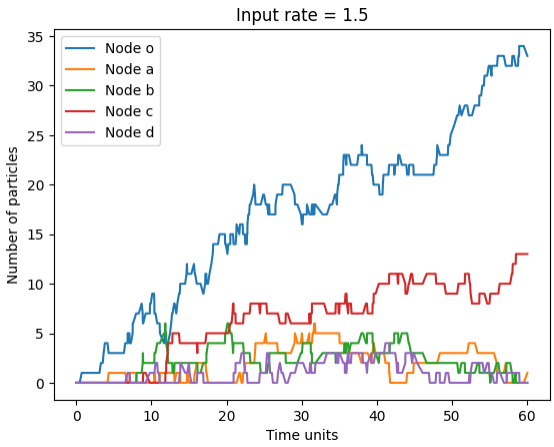












2- Initial Setup and Constraints:

The first case tested higher absolute input rates with a blowing-up limit set at 1000 particles. The network showed high resilience and stability under these conditions.
The second case used much smaller input rates, with a lower blowing-up limit of 30 particles. The smaller increments in input rates allowed for a more granular analysis of the system's stability.

3- Transition Matrix and Node Dynamics:

The dynamics of how particles move between nodes and how nodes handle incoming and outgoing particles can significantly impact stability. The cumulative transition matrix CumulativeQ and the specific behavior of nodes under different rates influence the system's capacity to avoid blowing up.
In the first case, the network may have a structure and transition dynamics that are better suited to handling higher inflows of particles without causing instability.