CLUSTERING STRUCTURAL BRAIN NETWORKS USING PAIRWISE CORRELATIONS OF SPIKE TRAINS

Thiago Henrique Neves Coelho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Rio de Janeiro
Fevereiro de 2025

CLUSTERING STRUCTURAL BRAIN NETWORKS USING PAIRWISE CORRELATIONS OF SPIKE TRAINS

Thiago Henrique Neves Coelho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Aprovada por: Prof. Daniel Ratton Figueiredo
Prof. Giulio Iacobelli
Prof. Daniel Sadoc Menasche
Prof. Luidi Gelabert Simonetti

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2025

Neves Coelho, Thiago Henrique

Clustering structural brain networks using pairwise correlations of spike trains/Thiago Henrique Neves Coelho. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIII, 44 p.: il.; 29, 7cm.

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 42 – 44.

1. Clustering. 2. Simulation. 3. Neuronal model. I. Ratton Figueiredo, Daniel *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À memória do meu avô, cujo apreço pela vida seria capaz de inspirar qualquer ser humano.

Agradecimentos

Gostaria de agradecer aos meus orientadores Daniel e Giulio pelo acompanhamento próximo que me deram. Sem eles, este trabalho não teria chegado aonde chegou. Também ao Guilherme e ao Conrado pela colaboração e por todo esse tempo compartilhado.

Agradeço à minha família por ter estado sempre ao meu lado, sobretudo aos meus pais Silvio e Rosane, que tanto sacrífício fizeram para que eu possa ter chegado até aqui, ao meu irmão Rafael, que foi o primeiro e mais forte exemplo e modelo de pessoa que tive, à minha irmã Raquel, por nunca ter permitido que eu me sentisse sozinho, ao meu primo João, meu maior companheiro durante a infância e ainda hoje um grande amigo, e à minha avó Neide, pelo carinho imenso que sempre teve por mim e por ter sido uma das figuras principais na construção da minha personalidade e caráter, assim como meu avô Mário, que não pôde contemplar a conclusão desta etapa, mas estará para sempre presente na minha memória.

Dedico também estes agradecimentos à Bruna por sempre me incentivar a ser e fazer o meu melhor e deixar mais leves os momentos difíceis, e aos meus amigos, que me proporcionaram tantos momentos divertidos de descontração.

Por fim, gostaria de ressaltar que sou grato à UFRJ e a todos os professores e colegas que fizeram parte da minha trajetória acadêmica e pessoal, sobretudo ao João, Claudson e Rodrigo, cujas conversas foram muito importantes para eu tomar a decisão de iniciar o mestrado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AGRUPAMENTO DE REDES ESTRUTURAIS DO CÉREBRO UTILIZANDO CORRELAÇÕES PAR-A-PAR DE SPIKE TRAINS

Thiago Henrique Neves Coelho

Fevereiro/2025

Orientadores: Daniel Ratton Figueiredo
Giulio Iacobelli

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma análise experimental do problema de recuperação exata de comunidades de redes neuronais a partir de observações de spike trains. Um modelo de atividade neuronal linear foi usado para redes com comunidades e dois tipos de conexões entre neurônios: excitatórias e inibitórias. Uma variação do stochastic block model com arestas direcionadas e pesos também foi adotada. Para gerar dados de spike trains, um simulador parametrizável do processo estocástico de ativação neuronal durante um horizonte de tempo predefinido foi construído. A partir dessas amostras de spike trains, os neurônios foram agrupados em duas comunidades por meio de um algoritmo de spectral clustering que recebe uma matriz simétrica onde as entradas correspondem a uma função da correlação de Pearson par-a-par entre os spike trains. A acurácia da recuperação das comunidades foi avaliada usando diferentes cenários para estudar experimentalmente o threshold para detecção de comunidades a partir dos spike trains. Os resultados sugeriram uma dependência aparentemente linear entre as forças de conexão intra e intercomunidade para uma acurácia fixa. Parâmetros como a probabilidade de ativação espontânea e a fração de arestas excitatórias também parecem estar relacionados entre si. Outro parâmetro fundamentalmente importante para alta acurácia é o tamanho dos spike trains, pois isso determina a qualidade da estimativa das correlações par-a-par. Assim, dados todos os outros parâmetros do modelo, um comprimento mínimo dos spike trains também é necessário para recuperação exata, adicionando uma nova dimensão à detecção de comunidades.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CLUSTERING STRUCTURAL BRAIN NETWORKS USING PAIRWISE CORRELATIONS OF SPIKE TRAINS

Thiago Henrique Neves Coelho

February/2025

Advisors: Daniel Ratton Figueiredo
Giulio Iacobelli

Department: Systems Engineering and Computer Science

This work presents an experimental analysis of the problem of exact recovery of neuronal network communities from spike train observations. A linear neuronal activity model was used for networks with communities and two types of connections between neurons: excitatory and inhibitory. A variation of the stochastic block model with directed edges and edge weights was also adopted. In order to generate spike train data, a parameterizable simulator of the stochastic process of neuronal activation during a predefined time horizon was built. From these spike train samples, neurons were clustered into two communities through a spectral clustering algorithm that receives a symmetric matrix where entries correspond to a function of the pairwise Pearson correlation between the spike trains. The accuracy of recovering the communities was evaluated using different scenarios to experimentally study the threshold for community detection from the spike train data. The results suggested an apparently linear dependence between the intra- and inter-community connection strengths for a fixed accuracy. Parameters such as the probability of spontaneous activation and the fraction of excitatory edges also seem to be related to each other. Another fundamentally important parameter for high accuracy is the length of the spike trains, as this determines the quality of estimation of the pairwise correlations. Thus, given all other model parameters, a minimum spike train length is also required for exact recovery, adding a novel dimension to community detection.

Contents

List of Figures

List of Tables

List of Symbols

C_i	Set of neurons in community i	11
N	Total number of neurons	11
T	Horizon of time of the simulation	13
X	Spike trains matrix	12
α_i	Fraction of neurons in community i	11
β	Probability for a connection to be excitatory	11
η	Matrix of edge types (excitatory/inhibitory)	11
λ	Spontaneous firing probability	11
$\mu_{\nsim}$	Intercommunity connection strenght	11
$\mu_{\sim}$	Intracommunity connection strenght	11
ρ	Pearson correlation matrix	16
θ	Adjacency matrix of the network	11
$i \nsim j$	i and j are in different communities	11
$i \sim j$	i and j are in the same community	11
p	Intracommunity connection probability	11
q	Intercommunity connection probability	11
r	Number of samples for the same parameters set	13

List of Abbreviations

Ber(·)	Bernoulli(·)	11
RMSE	Root mean square error	21
SBM	Stochastic block model	6
w.h.p.	With high probability	5

Chapter 1

Introduction

The structural network of the brain can be modeled as a network with communities, representing each one of its regions, with the nodes being a representation of the neurons, and the connections being a representation of the synapses. Over this network runs a neuronal activity, with neurons firing and interacting with each other through the synapses. Thus, after a neuron fires, it can influence or inhibit the firing of its neighbors. A synapse can be classified as excitatory, when the connected neuron positively influences the firing of its neighbor, or inhibitory, when the influence is negative. The sequence of activations of each neuron at each time is called a spike train.

Inferring information about the structure of a network through its signals is critical for studying structural brain networks, since brain activity scans do not directly reveal its structure. Thus, it is important to use methods that are capable of obtaining relevant information about the network underlying an observed neural activity, e.g., spike trains. There are many works in this direction, with different approaches for dealing with spike trains - see Section 1.1.

A common approach is to obtain information about the relations between neurons from the pairwise correlation of their spike trains. In this work, we used the pairwise Pearson correlation between neurons in synthetic networks to construct a similarity measure that is used for clustering through a spectral clustering algorithm.

Stochastic block model (SBM) is a common choice to model networks with communities and so it is a intuitive choice to model brain networks. The problem of community detection in SBM networks is well studied in the literature and have theoretically established bounds in terms of the model parameters that guarantee exact recovery in networks with 2 communities (see Section 2.2). However, in the context of neuronal networks, another interesting questions arise: how can we achieve exact recovery in a SBM network without looking at its structure, but rather observing a neuronal activity running in this network? As in classic community detection in SBM, is there a threshold in terms of parameters that separates possibility from

impossibility of recovery? How does each parameter influence in the probability of detecting communities? How does the length of the spike trains sequence influence in the community detection?

The objective of this work is to empirically analyse how the ability to detect communities of synthetic graphs from the stochastic block model, observing only the spike trains sequences generated by a linear neuronal activity model over these graphs, depends on the parameters of these models. Analyses of real data are beyond the scope of this work.

To achieve this goal, we performed an experimental study of the problem of exact recovery from signal observations. We assume discrete time and use a linear neuronal activity model over networks following a variation of the stochastic block model with directed and weighted edges. The networks used in this work have two communities, each representing a region of the brain. Thus, we assume that connections between neurons from the same community are stronger and more frequent than between neurons of different communities. At each time t, each neuron has a probability of firing and so has two states: activated or deactivated. There is a spontaneous firing probability to represent a chance of activating independently, but the firing probability of each neuron is influenced by its in-neighbors that fired at the previous time. The model represents the two types of synapses with excitatory and inhibitory edges. Excitatory edges increases the probability of a neuron firing when an in-neighbor fired at the previous time, and inhibitory edges decreases the probability.

A simulator of the neuronal activity stochastic process was built to generate spike train samples and measures of interest, e.g. pairwise Pearson correlation. This correlation is used to detect communities. The objective of this work is to understand how the model parameters influence each other in the exact recovery and to empirically study thresholds for this property, in particular, the influence of spike trains length.

The main contribution of this work is the experimental analysis of the relation between the parameters of the linear neuronal activity model and the ability to detect communities observing the spike train sequences. We expect that this analysis leads to a better understanding of the possibilities and limitations of community detection in neuronal activity networks and facilitate the theoretical study of thresholds based on the experimental results obtained. The simulator is also freely available[1] to researchers who wish to generate similar data.

In Chapter 2, the fundamental concepts will be explained. In Chapter 3, the neuronal activity model, its mathematical formulation, and the simulator developed to generate spike train samples and calculate measures of interest will be described. This chapter will also discuss technical details of the simulator implementation. In

[1] `https://github.com/thiagohnc/neuroscience`

Chapter 4, the clustering framework will be described and there will be a presentation of the experimental results obtained in different parameter scenarios and a detailed discussion of each of them. Finally, Chapter 5 provides a brief conclusion of the work, summarizing the results obtained and highlighting future work.

1.1 Related Work

In HUMPHRIES (2011), similarity measures between spike train series, e.g. Hamming distance and cosine similarity, are used to detect communities using a modularity-based method, which also determines the number of communities from the data. The method was applied both to synthetic spike trains data and to real data obtained from the neuronal activity of the cortex of an anesthetized cat. Another work focused on community detection from signal observation, although not focused on brain networks, is HOFFMANN *et al.* (2020), which uses a hierarchical Bayesian model to detect communities from time series without inferring edges, but propagating uncertainty from the raw data to the community labels. It uses the model to cluster United States cities into climate zones using climate data and to cluster companies using the S&P 100 index, but it was not used to find communities in brain networks using spike trains. The work by PEIXOTO (2019) uses a non-parametric Bayesian method to reconstruct the network and detect communities at the same time from functional observations and shows that these two activities were synergistic. Although intuitively the method seems to be usable for brain networks, they were not contemplated in the experiments.

There are also works by LINDERMAN *et al.* (2016) and KOBAYASHI *et al.* (2019), which use a generalized linear model (GLM), considering neurons of two types: excitatory and inhibitory. Both aim to reconstruct the neural network and discover the type of neurons. The first uses the Markov chain Monte Carlo (MCMC) technique to reconstruct this latent information, applying the framework to both synthetic networks and real neural networks of retinal ganglion cells. The latter fits the GLM to the cross-correlation of the spike data from the neurons. It also works with both artificial networks and real neural networks obtained from data of the hippocampus of a rat.

PAPALEXAKIS *et al.* (2014) uses a linear model for neuronal activity, with a solution inspired by control theory to reproduce the brain response given a stimulus. It assumes a hidden brain activity that is not observed and evolves over time through a linear transformation of its values at the previous time and a linear transformation of an input stimulus. Finally, a linear transformation of this activity to an observed sensor vector is performed. In addition to reproducing brain activity, the model was used to infer functional neuronal connectivity. However, it did not try to detect

communities, since the functional regions were known previously.

Using the same mathematical model presented in this work, COELHO *et al.* (2025) provides theoretical development to statistical measures such as the average firing rate of a neuron and the correlation between a pair of neurons, both for a fixed network and for a random network instance, as the network size increases. CATARCIONE (2025) also takes this same model the basis for the work, adding an analysis regarding the biological aspect of the problem, relating it to its mathematical formulation, and validating theoretical results using the simulator built in this work.

Chapter 2

Preliminary Concepts

This chapter briefly presents important concepts and terminology used throughout this thesis, such as random graph models, community detection in networks and Markov chains.

2.1 Random Graph Models

The concept of random graphs are similar to random variables, but extended to graphs. Given a set of graphs with n vertices (sample space), a random graph model defines a probability distribution among all elements of this sample space. In this work, we will refer to a random graph (or random network) as a sample generated by a random graph model. Moreover, the words network and graph will be used interchangeably.

One of the simplest random network models is $\mathbb{G}(n, p)$. The parameter n defines a set of numbered vertices in the network. For each pair of vertices, there is a probability p that an edge between them exists and $1 - p$ that it does not exist. Thus, each pair follows a Bernoulli distribution with parameter p, independently. This model was introduced by GILBERT (1959) and is quite similar to the definition of $\mathbb{G}(n, m)$, used by Erdős and Rényi in their papers on random graphs (see ERDŐS and RÉNYI (1959)). In $\mathbb{G}(n, m)$, the number of edges for any graph is always equal to m, and they are randomly distributed between the pairs of vertices.

Note that each possible graph within the sample space of $\mathbb{G}(n, m)$ has an equal probability of being sampled, since every pair of vertices is unique and has the same probability as any other of having an edge between them.

2.1.1 Threshold Function

In random graph theory, it is very common to study conditions for the occurrence of a given property in graphs asymptotically in the number of vertices. For ex-

ample, one can study under which conditions the graphs generated by the $\mathbb{G}(n,p)$ are connected, or under which conditions they are guaranteed to have at least one edge. The existence conditions for many properties are defined as a function of the model parameters. For example, for $\mathbb{G}(n,p)$, the existence conditions are given by a function $p(n)$ for the probability of edge between each pair of vertices, where $p(n)$ must be sufficiently large to guarantee the property will exist with high probability (w.h.p.) for $n \to \infty$.

As mentioned in FRIEZE and KAROŃSKI (2015), an interesting fact is that the existence of properties usually has a well-defined phase transition, with an abrupt change in behavior when $p(n)$ is asymptotically below a certain function compared to when it is asymptotically above (see also BOLLOBÁS (2001)). In order to characterize this nature, there is the concept of *threshold function*, which is the function that delimits when graph samples will have the property w.h.p. and when, with high probability, the graphs will not have the property. In other words, the function $t(n)$ is a threshold function for the monotonically increasing[1] property $\mathcal{P}$ in $G(n,p)$ if

$$\lim_{n\to\infty} \mathbb{P}(G_{n,p} \in \mathcal{P}) = \begin{cases} 0 & \text{if } p(n)/t(n) \to 0 \\ 1 & \text{if } p(n)/t(n) \to \infty \end{cases} .$$

FRIEZE and KAROŃSKI (2015) and BOLLOBÁS (2001) show threshold functions for some properties. For example,

$$t_{\text{connected}}(n) = \frac{\log(n)}{n}$$

is a threshold for the property of the graph being connected in the $\mathbb{G}(n,p)$, so if $p(n) \gg log(n)/n$, then the graphs are almost always connected as $n \to \infty$, and if $p(n) \ll log(n)/n$, they are almost always disconnected as $n \to \infty$. Other example is the threshold for the property of containing at least one edge

$$t_{\text{non-empty}}(n) = n^{-2} .$$

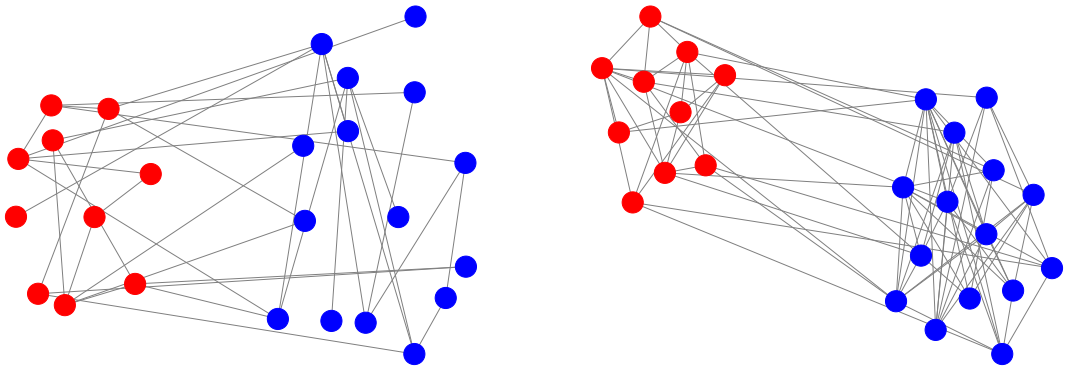
2.1.2 Stochastic Block Model (SBM)

SBM is a random network model useful for generating graphs with communities of vertices. It was initially proposed by HOLLAND *et al.* (1983) to model social networks[2] in which actors are partitioned into communities called blocks. The relationship between vertices and blocks are defined a priori, i.e., before generating a sample of the random graph. Given the set of blocks B, the probability of connec-

[1] If a property is monotonically increasing and random graphs with n vertices have it w.h.p., then random graphs with $n' > n$ vertices will also have it w.h.p.

[2] Networks that model relationships between people or groups.

tion between any pair of vertices that are in blocks B_i and B_j (i can be equal to j) is equal to $p_{B_iB_j}$.



(a) (b)

Figure 2.1: Samples of SBM with two communities. In (a), $p = q = 0.1$. In (b), $p = 0.4$ and $q = 0.1$. Both networks were generated using NetworkX (HAGBERG *et al.* (2008))

In a simpler version, we can consider the probability of connection between vertices in the same community as p, and of vertices from different communities as q. Thus, if $p > q$, graphs with a higher edge density within the blocks will be obtained. Similarly, with $p < q$, the graphs will have higher edge density between different blocks. The case $p = q$ is equivalent to $G(n, p)$.

2.2 Community Detection

A widely explored problem in the literature detecting communities in graphs. In general, given a graph G, with set of vertices $V(G)$ and set of edges $E(G)$, the goal is to determine to which community each vertex belongs. Several algorithms have been proposed to detect communities in graphs, such as the Girvan-Newman method (GIRVAN and NEWMAN (2002)), the Louvain algorithm (BLONDEL *et al.* (2008)), among others.

Although the problem is relevant to any type of network, including real networks, an interesting particular case is that of random networks generated by the SBM. Within this topic, there is the concept of *exact recovery*, which involves being able to recover the partition (of the communities) correctly with high probability as the number of vertices increases. Considering the simplest case of SBM - two communities and connection probabilities p and q -, intuitively, the difference between p and q must be large enough to be possible to recover the partition of the networks generated by the model. If $p = q$, we have a network identical to $G(n, p)$, making any type of identification of the communities impossible. As the distance between

the probabilities increases, it is intuitive that recovery becomes easier. The question that arises is to determine what would be the threshold for the values of p and q above which the exact recovery is guaranteed. In ABBE *et al.* (2014) it was proved that if $\alpha = pn/\log(n)$ and $\beta = qn/\log(n)$ are constants, with $\alpha > \beta$, then exact recovery (with high probability) is possible if $\frac{\alpha+\beta}{2} - \sqrt{\alpha\beta} > 1$ and impossible if $\frac{\alpha+\beta}{2} - \sqrt{\alpha\beta} < 1$.

2.2.1 Clustering

In this work, we detect communities using a clustering algorithm, called spectral clustering, that is applied to a similarity function of the correlations of spike trains. A clustering algorithm clusters data based on some characteristic that make them similar, such as the distance of points, cosine similarity, Pearson correlation, etc.

The spectral clustering (see LUXBURG (2007)), in particular, requires the number k of clusters to be given in advance. There are many variations of the algorithm, but the general idea is to compute the Laplacian L of the given similarity matrix and the first k eigenvectors $u_1, u_2, .., u_k$ of L. Then, create a matrix U whose columns are each one of these vectors. Define y_i as the i-th row of U, for $1 \leq i \leq N$, where N is the dimension of the square matrix of similarities. Finally, we cluster each y_i into k clusters using the $k - means$ algorithm.

Since we are considering two communities, the algorithm uses $k = 2$ in this work.

2.3 Markov Chains

A Markov chain is a stochastic process defined by a state space $\mathcal{X}$ and a matrix P of transition probabilities between them (see, e.g. LEVIN and PERES (2017)). If the state of the Markov chain at time t is equal to $i \in \mathcal{X}$, then at time $t+1$, the state will be $j \in \mathcal{X}$ with probability P_{ij}. Specifically, a time-homogeneous Markov chain is a set of random variables $(X_0, X_1, ..)$, such that

$$\mathbb{P}(X_{t+1} = j | X_t = i) = P_{ij} \,. \tag{2.1}$$

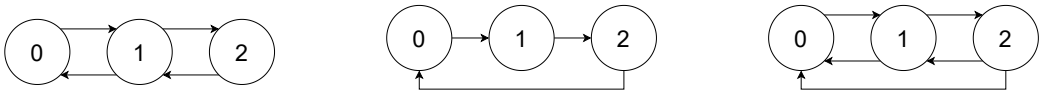
A Markov chain has the property of being memoryless, i.e., given the present, the future state does not depend on the past states. Furthermore, since P_{ij} is a probability matrix, we have that

$$\sum_j P_{ij} = 1, \forall i \,.$$

A Markov chain is said to be irreducible if it is possible to reach, from any state,

all of the other states in a finite number of steps using the non-zero transitions defined by the matrix P. In other words, it is irreducible if, $\forall(i,j) \in \mathcal{X}^2$, there exists an integer t such that $P_{ij}^t > 0$.

Considering $\mathcal{T}(i) := \{t > 0 : P_{ii}^t > 0\}$ the set of all time units after which it is possible to reach state i from itself, we can define the period of i as $\gcd \mathcal{T}(i)$. We say that the Markov chain is aperiodic if the period of every state i within the state space $\mathcal{X}$ is equal to 1.



(a) (b) (c)

Figure 2.2: Markov chains with state space of size 3. In the chain (a), $\mathcal{T}(0) = \mathcal{T}(1) = \mathcal{T}(2) = \{2, 4, 6, ...\}$, so the period of all states is 2 and the chain is periodic. In (b), we have $\mathcal{T}(i) = \{3\}$ for all states, since it is only possible to return after traveling the entire chain, so the period is 3 for all states. The chain (c) is aperiodic, since $\mathcal{T}(i) = \{2, 3, ...\}$ for every state, so every state have period 1.[3]

Given an initial probability distribution μ_0 over all elements of the state space of the chain, we can define the evolution of the distribution over time from the transition matrix, with $\mu_t = \mu_{t-1}P = \mu_0 P^t$. If there exists a probability distribution π such that $\pi = \pi P$, then we say that π is the *stationary distribution*, which is independent of time t. For a stationary distribution to exist and be unique, the chain must be irreducible and aperiodic.

Before presenting a theorem that guarantees the convergence to the stationary distribution in Markov chains, it is useful to define another concept. Given two probability distributions μ and ν, the total variation distance between them is

$$d_{\mathrm{TV}}(\mu, \nu) := \max_{A \subseteq \mathcal{X}} |\mu(A) - \nu(A)| .$$

According to the Convergence Theorem (see Theorem 5.2 in HÄGGSTRÖM (2002)), if P is irreducible and aperiodic, with stationary distribution π, then

$$\lim_{t \to \infty} d_{\mathrm{TV}}(\mu_t, \pi) = 0 .$$

In other words, the probability distribution converges to the stationary distribution as $t \to \infty$ regardless the initial condition. The minimum amount of time units for the distribution to be as close to π as an ϵ, starting from the worst possible initial distribution, is the *mixing time* $t_{\mathrm{mix}}(\epsilon)$, i.e.

[3]In this case, since the chains are irreducible, all states have the same period. This is not necessarily the case for non-irreducible chains (see LEVIN and PERES (2017).)

$$t_{\text{mix}}(\epsilon) := \min \left\{ t : \sup_{\mu_0 \in \mathcal{P}} d_{\text{TV}}(\mu_0 P^t, \pi) \leq \epsilon \right\},$$

where $\mathcal{P}$ is the set of all probability distributions on $\mathcal{X}$. When the dynamic is simulated in a Markov chain, it is said to be fast mixing if the mixing time is small, i.e., in a relatively small number of simulation steps compared to the size of the state space, the chain has reached its stationary distribution.

Chapter 3

Neuronal Activity Model and Simulator

In order to represent the neuronal network and activity, we consider a discrete-time neuronal activity model over a random directed network based on SBM with two communities to represent the functional regions of the brain.

The neuronal activity model considers a linear influence from an active neuron onto its neighbors in the next time, with neurons from same community having a stronger interference and more connections. The model assumes two types of edges: excitatory, where there is an increase of the firing probability of a neuron after the activation of one of its in-neighbors in the previous time, and inhibitory, where there is a decrease in this probability.

In addition to the dynamics described above, each neuron also has a spontaneous firing probability, independent of the stimuli from its neighbors.

3.1 Model Formulation

Let $[N] = \{1, 2, ..., N\}$ be the neurons and C_1 and C_2 a partition of the neurons in two communities, such that $[N] = C_1 \cup C_2$ and $C_1 \cap C_2 = \emptyset$. The sizes of the communities are $|C_i| = \alpha_i N$, with $\alpha_1 + \alpha_2 = 1$. Given two neurons $i, j \in [N]$, we have $i \sim j$ if and only if i and j are in the same community, i.e. $i \sim j \Leftrightarrow (i, j) \in (C_1 \times C_1) \cup (C_2 \times C_2)$. Let p and q denote, respectively, the connection probabilities between neurons from same communities and neurons from different communities, and $\theta = (\theta_{ij})_{1 \leq i,j \leq N}$ the network adjacency matrix, we have

$$\theta_{ij} \sim \begin{cases} \text{Ber}(p), \text{ if } i \sim j \\ \text{Ber}(q), \text{ if } i \not\sim j \,, \end{cases}$$

where $\text{Ber}(p)$ is the Bernoulli random variable with parameter p.

The division of excitatory and inhibitory edges is controlled by parameter β, that represents the fraction of excitatory connections. So, being $\eta = (\eta_{ij})_{1 \leq i,j \leq N}$ the random variable that determines the type of the edge between each pair of neurons, its value will be

$$\eta_{ij} = \begin{cases} 1 \text{ with probability } \beta \\ -1 \text{ with probability } 1 - \beta \,. \end{cases}$$

We consider that θ and η are independent.

The model assumes discrete time, such that at each time period t, a neuron can be activated (i.e. firing) or deactivated. To represent this binary state of the neuron, consider the random variable X, such that $X_{i,t} = 1$ if neuron i fires at time t and $X_{i,t} = 0$ if it does not. The dynamic of neuron activation follows a stochastic process in which there is a probability of its state to be equal to each of these two values at each period of time based on the state of network previously. This probability is defined by the sum over all of the contributions from the neurons that fired in the previous time, considering those of its in-neighbors with weight $\mu_\sim$ and from its out-neighbors with weight $\mu_\nsim$, normalized by the total number N of neurons in the network. Finally, it is added a spontaneous activation probability λ, independent from the neuron.

In that way, being $X_t = \{X_{1,t}, X_{2,t}, ..., X_{N,t}\}$ and given a graph G, the firing probability of a neuron at time t will be governed by Equation 3.2. It is easy to notice that this probability is independent from t.

Let

$$p_i(x) = \lambda + \frac{\mu_\sim}{N} \sum_{j \sim i} \theta_{ji} \eta_{ji} \, x_j + \frac{\mu_\nsim}{N} \sum_{j \nsim i} \theta_{ji} \eta_{ji} \, x_j \,. \tag{3.1}$$

So, the firing probability of a neuron i is

$$P(X_{i,t} = 1 | X_{t-1} = x) = \begin{cases} 0, & \text{if } p_i(x) < 0 \\ p_i(x), & \text{if } 0 \leq p_i(x) \leq 1 \\ 1, & \text{if } p_i(x) > 1 \,. \end{cases} \tag{3.2}$$

To understand this probability, first we have to look at $p_i(x)$. It is divided in three parts:

1. λ: The spontaneous activation probability is the baseline value for $p_i(x)$, independent from neighbors of the neuron. If no in-neighbor of i fires, the firing probability of i will be λ.

2. $\frac{\mu_\sim}{N} \sum_{j \sim i} \theta_{ji} \eta_{ji} \, x_j$: The influence from neurons in the same community as i.

The sum is selecting only neurons j such that $j \sim i$, i.e., that are in the same community as i. θ_{ji} is equal to 1 if there is an edge from j to i, so the influence of j will be counted only if it is an in-neighbor of i. η_{ji} can be either $+1$ or -1, depending on the connection type, so excitatory connections will have a positive influence, while inhibitory connections will have a negative influence. As we use $X_{t-1} = x$ in Equation 3.2, x_j will be 1 if neuron j fired at time $t-1$. Finally, we multiply the sum by $\mu_\sim$, which is the strength of intracommunity connections and normalize by N to prevent $p_i(x)$ from exploding as N increases. Summarizing, each in-neighbor of i that belongs to its community and fired at time $t-1$ will increase the firing probability of i at time t by $\mu_\sim/N$ if its edge to i is excitatory, or decrease it by the same amount if it is inhibitory.

3. $\frac{\mu_\nsim}{N} \sum_{j \nsim i} \theta_{ji}\eta_{ji}\, x_j$: The influence from neurons that are not in the same community as i. The analysis is analogous to the previous item, but for neurons of the other community, contributing with $\mu_\nsim/N$ instead of $\mu_\sim/N$.

Note that depending on the parameter choices, it is possible to have $p_i(x) \notin [0,1]$, thus not defining a probability function. Since the firing probability must be between 0 and 1, we apply the minimum and maximum functions to enforce that it is in the range, so the firing probability is defined as $\max(\min(p_i(x), 1), 0)$, as shown in Equation 3.2.

The evolution of X_t over time can be characterized by a Markov chain with state space $\{0,1\}^N$, with the initial state $\{0\}^N$ and transition probabilities defined as

$$P(X_t = y | X_{t-1} = x) = \prod_i p_i(x)^y \left(1 - p_i(x)\right)^{1-y} .$$

Although there are works proposing models with firing probabilities depending on neighborhood activity, the Equation 3.1 follows an original approach of using $\mu_\sim$ and $\mu_\nsim$ to describe the firing probability of our model.

3.2 Simulator

We built a simulator for the neuronal model in C++ to generate samples of spike trains over a horizon T of time given the initial parameters. In the beginning of the execution of the program, a random directed graph is generated following the SBM, with parameters p and q to define the connection probability between each pair of neurons. The communities sizes are also governed by parameters and defined before the generation of the graph. For each edge, a pseudorandom number is calculated

to define its type, being excitatory with probability β and inhibitory with $1-\beta$. In that way, each edge in the graph has 4 possible weights: $\mu_{\sim}$, $-\mu_{\sim}$, $\mu_{\nsim}$ and $-\mu_{\nsim}$.

The calculation of spikes at time 0, that represents the initial state, in the simulator is done using only the spontaneous firing probability, begin $X_{i,0} = 1$ with probability λ and $X_{i,0} = 0$ with probability $1-\lambda$ for all $i \in [N]$, i.e. $X_{i,0} \sim Ber(\lambda)$. For the other periods of time $t > 0$, $X_{i,t}$ is calculated using the previous state $X_{i,t-1}$, as described in Equation 3.2. The simulation follows as this until it has generated spike trains during T units of time for the N neurons.

The ultimate goal of sampling spike trains through the simulator is to use them to calculate measures of interest, like the correlation between neurons and firing averages, which will be critical to perform experimental analysis. Also, the correlation will be provided as input for the framework that will be presented in Chapter 4. However, as described in Section 3.1, the random variable X_t can be defined as the state of a Markov chain. To avoid bias in spike trains samplings, it would be better to start the chain from its stationary state. To achieve that, the simulator considers an additional parameter $BURN_T$ that defines the number of rounds of simulation that will be performed at the beginning and will be discarded. With this parameter, the number of simulation rounds will be $T + BURN_T$. In general, we will assume $BURN_T = 0$ to emphasize that the initial state did not influence the results of the experiments, as discussed in Section 4.3.

It is also possible to configure the number of graph and simulations performed for the same group of parameters through the r parameter to ensure greater statistical reliability in the obtained results. For each one of the r samples, a new graph will be generated and used in the simulation with the same parameters. The simulator outputs listed in Table 3.2 are generated individually for each sample.

3.2.1 Pearson Correlation

A measures of interest calculated by the simulator is the Pearson correlation between the spike train time series of each pair of neurons in the network. The intuition behind this choice is that neurons from the same community will have stronger influence on each other's firing compared with neurons from different communities. This influence can be either excitatory or inhibitory. Since the network dynamics depend on the time elapsed, a lag (delay) parameter d was added to calculate the correlation. If it is non-zero, the correlation between two time series will be obtained by shifting one of them by d units to the right. This parameter aims to capture influences from connected neurons. Thus, the correlation ρ_{ij} between two time series X_i and X_j with lag d will be

$$\rho_{ij} = \frac{\sum_{t=1}^{T-d}(X_{i,t} - \bar{X}_i)(X_{j,t+d} - \bar{X}_j)}{\sqrt{\sum_{t=1}^{T-d}(X_{i,t} - \bar{X}_i)^2 \sum_{t=1}^{T-d}(X_{j,t+d} - \bar{X}_j)^2}}, \tag{3.3}$$

where $\bar{X}_i = \frac{1}{N}\sum_{t=1}^{T-d} X_{i,t}$.

Table 3.1: Simulator Inputs

Parameter	**Description**
$group_N$	List with the number of neurons in each community
T	Horizon of time of the simulation
p	Intracommunity connection probability
q	Intercommunity connection probability
β	Probability of an edge to be excitatory
$\mu_\sim$	Absolute value of intracommunity edges weight
$\mu_\nsim$	Absolute value of intercommunity edges weight
λ	Spontaneous activation probability
d	Lag for Pearson correlation calculation
r	Number of samples for the same parameters set
seed	Seed used for random number generation[1]

Table 3.2: Simulator outputs

Output	**Description**
Spike trains	$N \times T$ spike trains Matrix
Spike average	Average of spikes for each neuron
Adjacencies	$N \times N$ weighted adjacency matrix
Pearson	ρ matrix (Equation 3.3)

3.2.2 Efficiency

As discussed in Chapter 4, the parameter T is critical for community detection, so it is necessary for the algorithms to be optimized in order to achieve an execution speed that allows experiments with longer horizons of time. Depending on the implementation, RAM consumption can also become prohibitive, so another goal of this work is to provide a simulation and calculation method with low use of this resource.

[1]Using the same seed with all identical parameters will result in exactly the same simulation, however it is possible to start with a random seed if its value is set to *auto*. This last setting was used for all experiments performed in this work. The seed is actually a root to generate r seeds, one for each sample. This way, it is guaranteed that different samples of the same simulation have different dynamics.

Pearson correlation

Increasing the network size and the number r of samples has a direct impact on the simulation time and calculation of the pairwise Pearson correlation. In order to better explore these parameters in a reasonable time, some optimizations were made to the code, especially regarding the calculation of the Pearson correlation.

The naive algorithm consists of computing Equation 3.3 for all pairs of neurons. The first visible opportunity for improvement is by pre-computing the two terms in the denominator. Considering $Var(X_i) = \sum_{t=1}^{T-d}(X_{i,t} - \bar{X}_i)^2$ and $Var_d(X_j) = \sum_{t=1}^{T-d}(X_{j,t+d} - \bar{X}_j)^2$, we can store the values of Var and Var_d for all neurons, so that we do not need to recalculate them for each pair whose correlation is computed.

For the numerator, let us consider two matrices A and B, where $A_{it} = X_{i,t} - \bar{X}_i$ and $B_{it} = X_{i,t+d} - \bar{X}_i$, for $i \leq N$ and $t \leq T$. We can also precompute only once and store the average firing rate $\bar{X}_i$ of each neuron i. Note that

$$\rho_{ij} = \frac{(AB^T)_{ij}}{\sqrt{Var(X_i)Var_d(X_j)}}\,. \tag{3.4}$$

From this, it is possible to use a matrix multiplication algorithm, which will be able to calculate more efficiently than the naive algorithm of summing the products of the elements of each spike trains pair. To obtain a better performance than the naive implementation of the multiplication, the C++ Eigen library (GUENNEBAUD *et al.* (2010)) was used for these operations.

Memory consumption

The main factors responsible for large memory consumption are the spike train matrix and the matrices for calculating Pearson correlation. For each one, a strategy was defined to reduce consumption.

The spike trains matrix has size $N \times T$. Since the entire matrix is needed to calculate the Pearson coefficient, we used the `vector<vector<bool>>` structure, which is optimized to use approximately 1 bit per entry, resulting in approximately NT bits of storage, i.e., $\frac{NT}{8}$ bytes.

Matrices A and B of Equation 3.4 have dimensions $N \times T$, but their elements are of type *double*, which represents a consumption of $64NT$ bits if they are fully stored, i.e., 64 times the amount to store the spike trains. To reduce this value, we can divide the matrix into blocks of size $S \times T$, multiply them and add the results in a matrix $N \times N$, which, at the end of the operation, will have the value of AB^T.

Let $A[x_0 : x_1][y_0 : y_1]$ be the submatrix of A formed by the rows in range $[x_0, x_1]$ and columns in range $[y_0, y_1]$. It is simple to prove that the sum of the multiplications of each block corresponds to the multiplication of the two matrices. Assume, for

simplicity, that T is divisible by S. If it is not, we can use a value $S' \equiv T \mod S$ for the last block.

$$
\begin{aligned}
(AB^T)_{ij} &= \sum_{t=1}^{T} A_{it} B_{tj} \\
&= \sum_{k=1}^{T/S} \left(\sum_{t=1+(k-1)S}^{kS} A_{it} B_{tj} \right) \\
&= \sum_{k=1}^{T/S} \left(A[1:N][1+(k-1)S : kS] \cdot B[1:N][1+(k-1)S : kS]^T \right)_{ij} .
\end{aligned}
$$

□

For this work, it was used the value $S = T/(20 \cdot 64)$, so that the two matrices together only need 10% of the space occupied by the spike trains matrix.

Chapter 4

Framework and Results

4.1 Framework

We used a simple model-agnostic framework for community detection by observing only spike trains. The main idea is to use Pearson correlations to calculate similarity between sequences and clustering through the spectral clustering algorithm implemented in scikit-learn (PEDREGOSA *et al.* (2011)).

The first step of the framework is to get the Pearson correlation calculated after the simulation for each pair of spike trains, considering the lag d when applicable. Those correlations are stored in a matrix ρ, which will be used to construct the similarity matrix as

$$Sim_{ij} = \frac{|\rho_{ij}| + |\rho_{ji}|}{2} \,. \tag{4.1}$$

We used the absolute values of correlations because it is expected that both positive and negative values will be stronger in neurons of the same community, since the parameters $\mu_{\sim}$ and $\mu_{\nsim}$ determine the absolute value of the connection strength, regardless of it being positive or negative.

Furthermore, since the similarity matrix must be symmetric to be used by the spectral clustering, we calculate the average between the two absolute values. This also makes sense because $i \sim j$ if and only if $j \sim i$, therefore a higher value for both $|\rho_{ij}|$ and $|\rho_{ji}|$ is a greater indicative of $i \sim j$ than only one of them.

Finally, we provide this matrix as input for the spectral clustering algorithm, specifying that there are 2 groups, as it must be determined priorly for the algorithm to be used.

Figure 4.1 shows a summary of the process from parameters input to cluster assignment for each neuron.

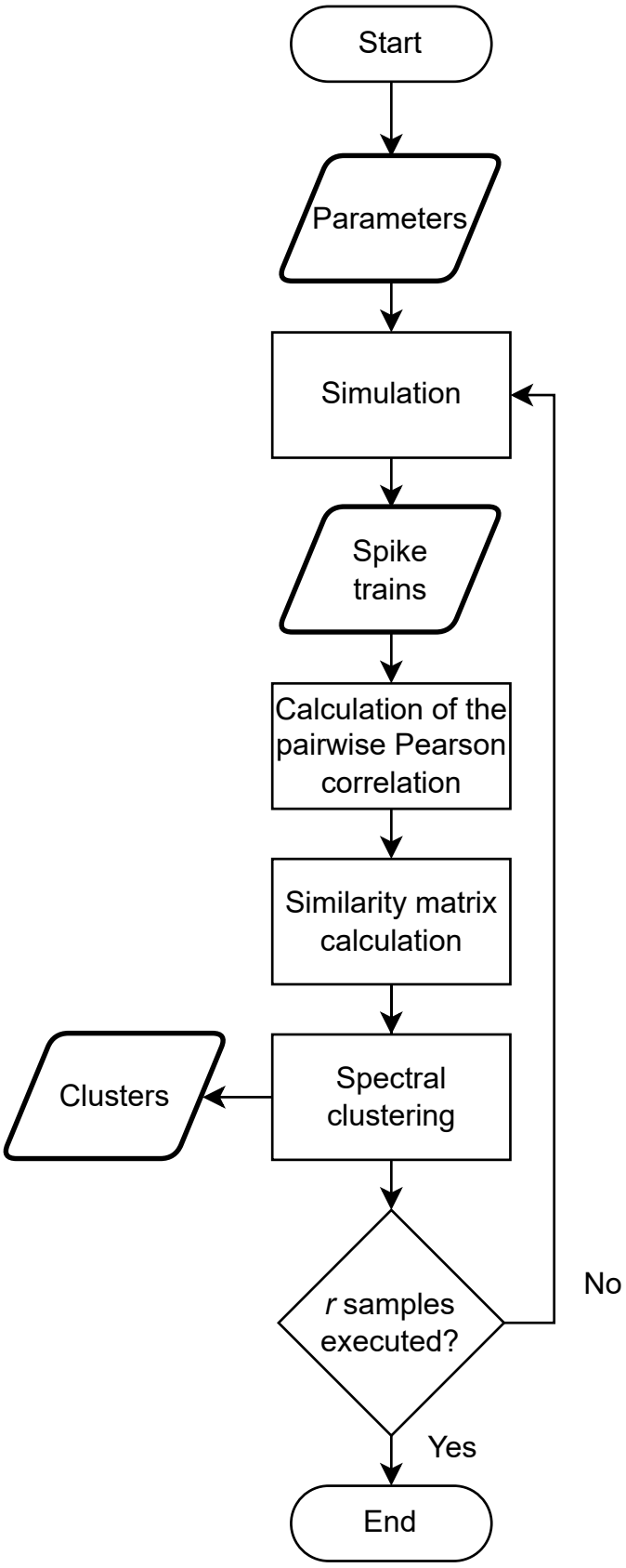


Figure 4.1: Flowchart of the framework

It is possible to notice that some stages of the process were chosen among several possibilities.

Correlation measure

Other measures of correlation between spike trains could have been chosen, such as cosine similarity or the Euclidean distance between vectors.

Pearson correlation was chosen for its simplicity and its ability to detect negative correlations, which are important due to inhibitory connections. This is not directly captured by measures such as cosine similarity, for example.

Clustering algorithm

Other clustering methods were possible alternatives to spectral clustering in this work. One of them is the construction of a distance matrix between neurons based on the correlation of spike trains, where $dist_{ij} = 1 - (|\rho_{ij}| + |\rho_{ji}|)/2$, followed by the application of a multidimensional scaling algorithm (see BORG and GROENEN (2005)), assigning to each neuron a coordinate in the Cartesian plane and, finally, clustering these points with K-means.

Another possibility would be considering the constructed similarity matrix as a complete graph and applying a community detection, such as the Leiden algorithm (see TRAAG *et al.* (2019)), in this graph, limiting the number of communities that can be detected.

Spectral clustering was a fairly straightforward choice for a framework that works with pairwise Pearson correlation, since the similarity matrix constructed with the Equation 4.1 serves directly as input for the algorithm.

4.1.1 Accuracy calculation

Let g be the ground truth of neurons communities, such that g_i is the community of neuron i, and g^k the community assignments by the clustering algorithm for the k-th sample, for $1 \leq k \leq r$. To calculate the accuracy of the detection, the fraction of the number of correctly predicted neurons to the total is calculated, taking the maximum when considering all community permutations. Since there are only two communities, the community detection accuracy of a sample k is

$$SampleAccuracy_{g^k} = \max\left(\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(g_i = g_i^k\right), 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(g_i = g_i^k\right)\right),$$

where $\mathbb{I}(\cdot)$ is the indicator function, defined as

$$\mathbb{I}(A) = \begin{cases} 1, & \text{if } A \text{ is true,} \\ 0, & \text{if } A \text{ is false.} \end{cases}$$

Hence, the accuracy will always be greater than or equal to 0.5.

Finally, we calculate the accuracy for the whole experiment, consisting in r samples. The experiment accuracy is the average of all sample accuracies, so

$$Accuracy = \frac{1}{r} \sum_{k=1}^{r} SampleAccuracy_{g^k} .$$

4.2 Pairwise correlations

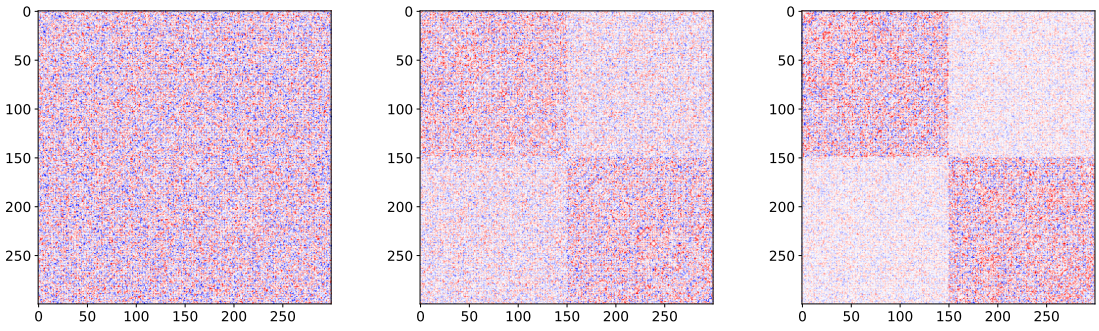
In Figure 4.2, there are 3 heatmaps of the Pearson lag-0 correlation matrices generated by one run ($r = 1$) of the simulation. All parameters were fixed for the figures, varying only the value of $\mu_\sim$. In Figure 4.2a, with $\mu_\sim = 1$, the values of $\mu_\sim$ and $\mu_\nsim$ were not far enough apart to ensure a clear demarcation of the communities given the other parameters. In Figure 4.2b, with $\mu_\sim = 3$ it is possible to visually notice the demarcation of the communities, since the strength of the connections within them is greater and, therefore, ensures more correlated spike train sequences. Finally, in Figure 4.2c, with $\mu_\sim = 4$, the communities are much more visible. The relation between the increase in $\mu_\sim$ and the ability to detect communities from the spike train sequences is clear here.

In Figure 4.3, where the correlations are lag-1, the results are more expressive than in the lag-0 case, since Figure 4.3a ($\mu_\sim = 2$) has much more demarcated communities than 4.2c, even with a lower value of $\mu_\sim$ and T.

Furthermore, as the value of T increases, the correlation heatmap (Figure 4.3a, with $T = 10^6$, and Figure 4.3b, with $T = 10^8$) approaches the weighted adjacency heatmap of the graph (Figure 4.3c). Normalizing the correlations by the maximum of their absolute values and doing the same with the weighted adjacency matrix (i.e., normalizing the adjacency matrix by the maximum of its absolute values), the root mean square error (RMSE) between the correlations with $T = 10^6$ and the weights is 3.10417, while that between $T = 10^8$ and the weights is 0.62679.

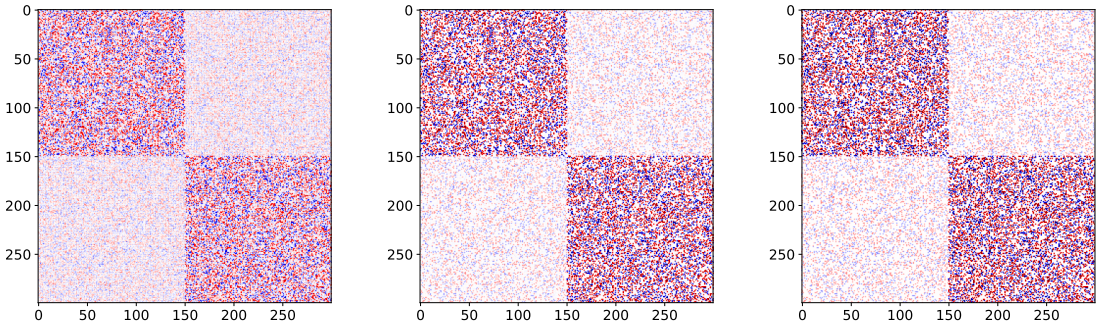
Figure 4.4 shows the heatmap for networks with $\alpha_1 \neq \alpha_2$ for lag-0. The correlation within the smaller community apparently weakens as the proportion α_1/α_2 grows, where $\alpha_1 > \alpha_2$ without loss of generality. This result leads to the intuition

[1]The correlations of neurons with themselves were artificially set to 0, since the diagonal with value 1 polluted the color scale of the graphs. The color scale of all heatmaps in this chapter is defined between [-*max_ abs*,+*max_ abs*], where *max_ abs* is the maximum among the absolute values of the matrix.



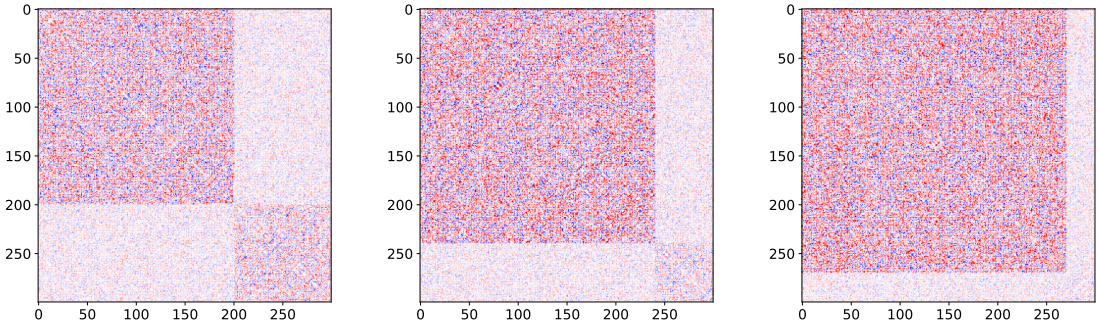
(a) $\mu_\sim = 1$ (b) $\mu_\sim = 3$ (c) $\mu_\sim = 4$

Figure 4.2: Pairwise Pearson correlation heatmaps for lag-0. Other parameters: $N = 300$, $\alpha_1 = 0.5$, $T = 10^7$, $p = 0.3$, $q = 0.2$, $\mu_\nsim = 0.5$, $\beta = 0.6$, $\lambda = 0.25$.[1]



(a) $T = 10^6$ (b) $T = 10^8$ (c) weighted adjacencies

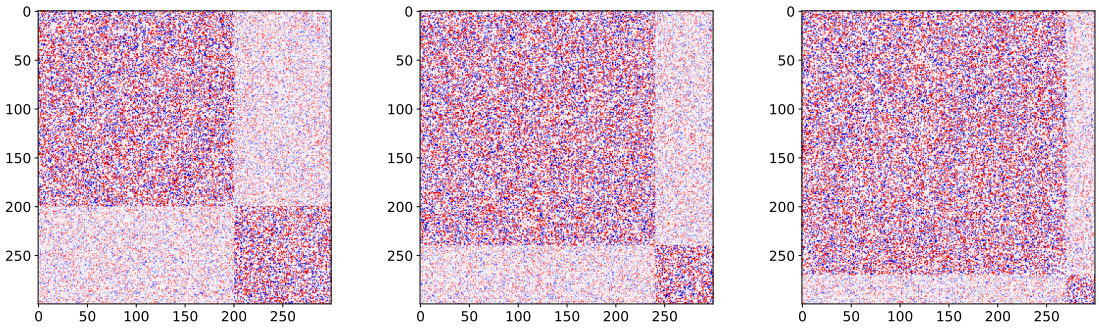
Figure 4.3: Pairwise Pearson correlation heatmaps for lag-1 in (a) and (b) and graph weighted adjacency matrix in (c). The same graph was used for all figures. $\mu_\sim = 2$ and other parameters are the same as in Figure 4.2.



(a) $\alpha_1 = 66.7\%$ (b) $\alpha_1 = 80\%$ (c) $\alpha_1 = 90\%$

Figure 4.4: Pairwise Pearson correlation heatmaps for lag-0 and uneven communities with $\mu_\sim = 4$. Other parameters are the same as in Figure 4.2.

that detecting communities in an uneven network may be more difficult than in a balanced one. No noticeable difference is present for lag-1 in Figure 4.5



(a) $\alpha_1 = 66.7\%$ (b) $\alpha_1 = 80\%$ (c) $\alpha_1 = 90\%$

Figure 4.5: Pairwise Pearson correlation heatmaps for lag-1 and uneven communities with $\mu_\sim = 1$. Other parameters are the same as in Figure 4.2.

4.3 Markov chain mixing time

To ensure that the samples are not biased, we want to start sampling from the stationary state of the chain. To study this, we observe the impact that $BURN\_T$ has on the community detection accuracy (Figure 4.6) as well as the similarity of the spike train mean with the asymptotic result for the theoretical values (Figure 4.7). According to COELHO *et al.* (2025), the theoretical firing mean m is given by

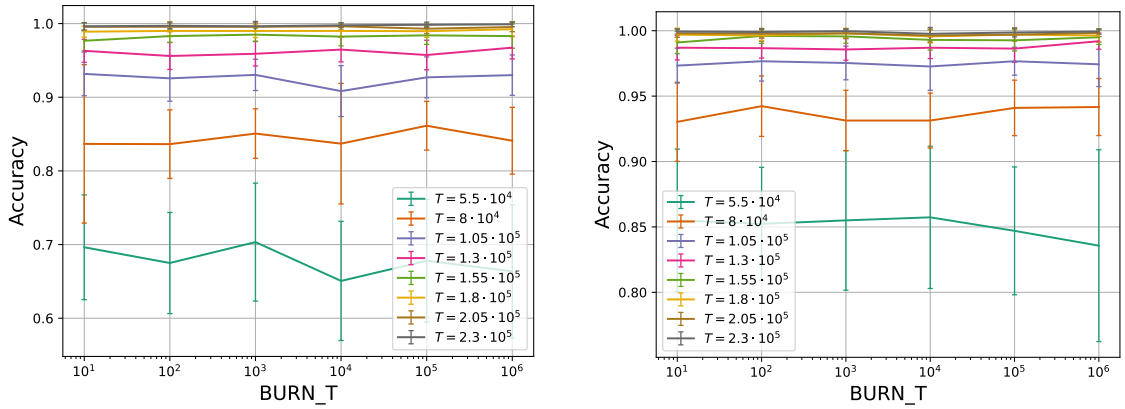
$$m = \left(I_N - A^T\right)^{-1} \lambda 1_N \,, \tag{4.2}$$

where 1_N denotes a column vector with all entries equal to 1, I_N the N-dimensional identity matrix and A is the interaction matrix, defined as

$$A_{ij} := \begin{cases} \frac{\mu_\sim}{N}\theta_{ij}\eta_{ij} \,, \text{ if } i \sim j \\ \frac{\mu_\nsim}{N}\theta_{ij}\eta_{ij} \,, \text{ if } i \nsim j \,. \end{cases}$$

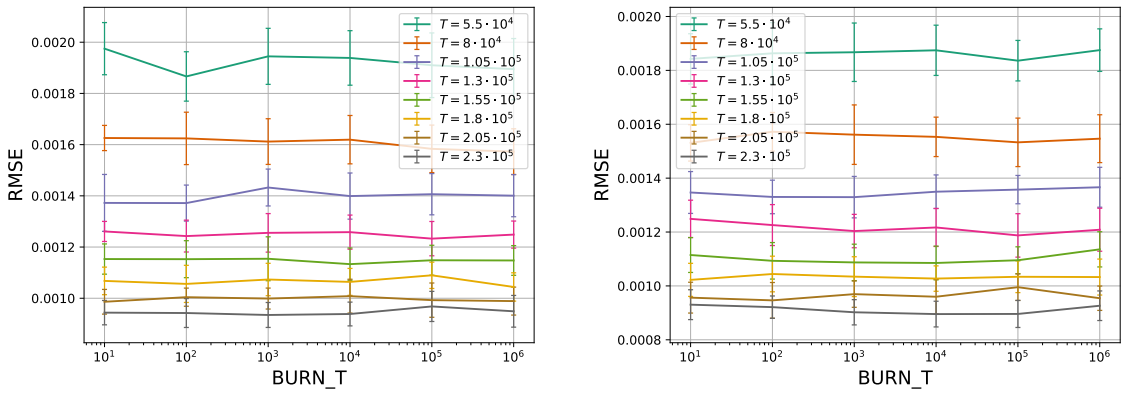
In both figures, it is possible to notice that there was no pattern of growth or decrease in accuracy or similarity with theoretical results for different values of $BURN\_T$. In Figure 4.6, there is a considerable variation, especially for smaller values of T, but it is worth noting that this can be easily explained by the fact that the data have a high standard deviation. This, in turn, may be due to the fact that these values of T and the other parameters used are in the transition between non-detection and exact recovery of communities, therefore being more sensitive to the graph generated for the simulation.

Because of this, we consider $BURN\_T = 0$ for the experiments in this work. On the other hand, although $BURN\_T$ appears to be irrelevant to the experiments, T is a very important parameter, as it strongly interferes with the similarity of the experimental results with the theoretical ones and with the exact recovery capacity. It is possible to notice that, the higher the value of T, the closer the simulation



(a) lag-0 (b) lag-1

Figure 4.6: Community detection accuracy by $BURN\_T$ for different values of T. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $p = 0.3$, $q = 0.2$, $\mu_\sim = 4.5$, $\mu_\nsim = 0.5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 20$.



(a) lag-0 (b) lag-1

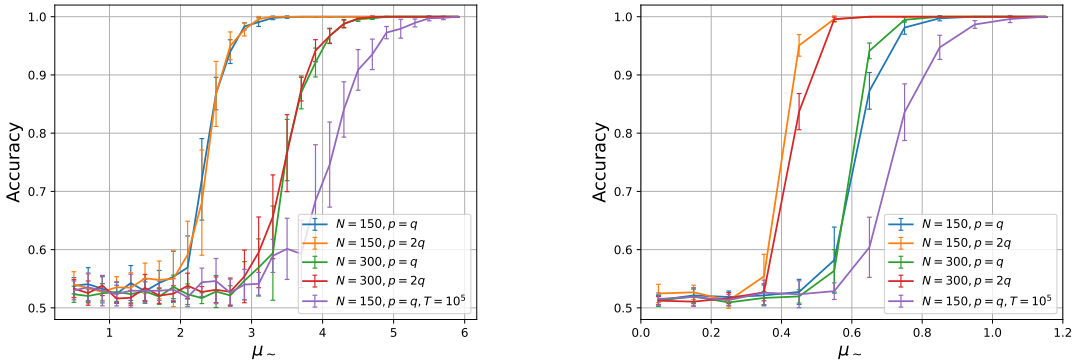
Figure 4.7: Mean RMSE of the firing mean of each neuron with its theoretical mean (Equation 4.2).

becomes to the theoretical values, which is expected, since the latter are asymptotic. Furthermore, larger time horizons guarantee higher accuracies for the same parameters.

4.4 Accuracy by $\mu_\sim$

The parameters $\mu_\sim$ and $\mu_\nsim$ define the connection strengths between neurons, while p and q define the number of connections. Thus, these parameters must be critical for the accuracy of community detection. In Figure 4.8, the network parameters were fixed, varying $\mu_\sim$ and calculating the average of the accuracies obtained in each point. As expected, the higher the value of $\mu_\sim$, the higher the accuracy obtained.

The simulations suggest that there is a phase transition in the curves, with the accuracy very close to 0.5 for values of $\mu_\sim$ less than a certain limit and are equal to



(a) lag-0 (b) lag-1

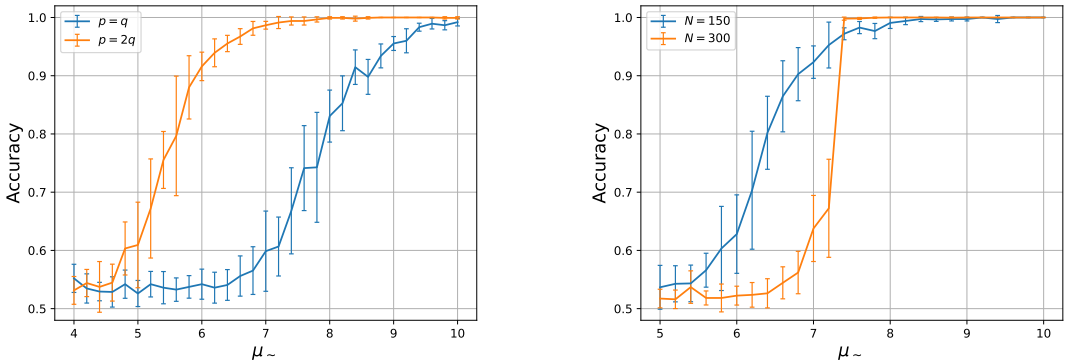
Figure 4.8: Community detection accuracy by $\mu_\sim$ for different scenarios. Other parameters: $\alpha_1 = 0.5$, $T = 10^6$ (except otherwise indicated), $p = 0.3$, $\mu_\nsim = 0.5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 10$.

1 for $\mu_\sim$ greater than another limit. Between these two points, the phase transition occurs and the curve increases abruptly, with a higher slope near the midpoint on the y-axis (accuracy 0.75). Furthermore, the standard deviation of the accuracies is considerably higher in this intermediate interval, showing stability in the non-detection (accuracy 0.5) and exact recovery (accuracy 1) phases. The value of T also seems to influence this behavior, since the curves $T = 10^5$ are smoother than $T = 10^6$, which suggests that the transition is more demarcated with longer time horizons. On top of that, for lower values of T it was harder to detect communities, being necessary a higher $\mu_\sim$. It is also noticeable that the curves from lag-1 are considerably steeper than the ones from lag-0, suggesting that the phase transition occurs quicker for lag-1.

In addition, Figure 4.8 shows an interesting result regarding the parameters N, p and q. At lag-0, it is clear that the more neurons there are in the network, the more difficult it is to detect communities, but the change in the ratio p/q does not seem to have made any substantial difference for this scenario. On the other hand, at lag-1 the observed effect is the opposite. While the number of vertices did not produce any effect on the accuracy obtained, it is clear that the curves $p = 2q$ achieved a higher accuracy for the same value of $\mu_\sim$ than the curves $p = q$.

It is also interesting to notice that the graphs from the $p = q$ scenario would be structurally identical to graphs generated by $\mathbb{G}(n, p)$, so if we were considering unweighted edges, it would be impossible to detect communities. In fact, if we observe the edges from such graphs, there would not be any structure resembling communities. However, since parameters $\mu_\sim$ and $\mu_\nsim$ are being used, it is possible to detect communities in this scenario when $\mu_\sim$ and $\mu_\nsim$ are sufficiently distant from each other. In other hand, with $p = 2q$, the edge density inside communities is

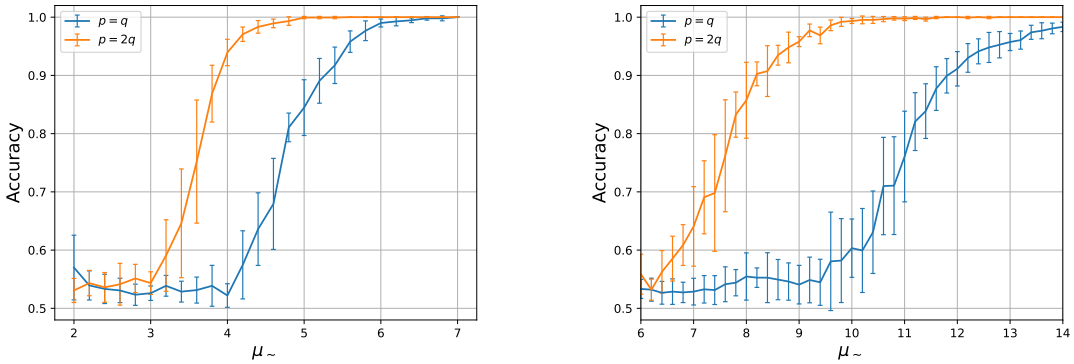
considerably higher than between communities. In this scenario, we would be very likely to find structure of communities in the graph if we were able to observe its edges. Thus, it is not surprising that a lower value of $\mu_{\sim}$ is necessary to detect communities with $p = 2q$ than with $p = q$.



(a) lag-0 (b) lag-1

Figure 4.9: Community detection accuracy by $\mu_{\sim}$ for different q (lag-0) and N (lag-1) with $\mu_{\nsim} = 5$. Other parameters: $N = 150$ (for figure in left), $\alpha_1 = 0.5$, $T = 10^6$, $p = 0.3$, $q = 0.15$ (for figure in right), $\mu_{\nsim} = 5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 10$.

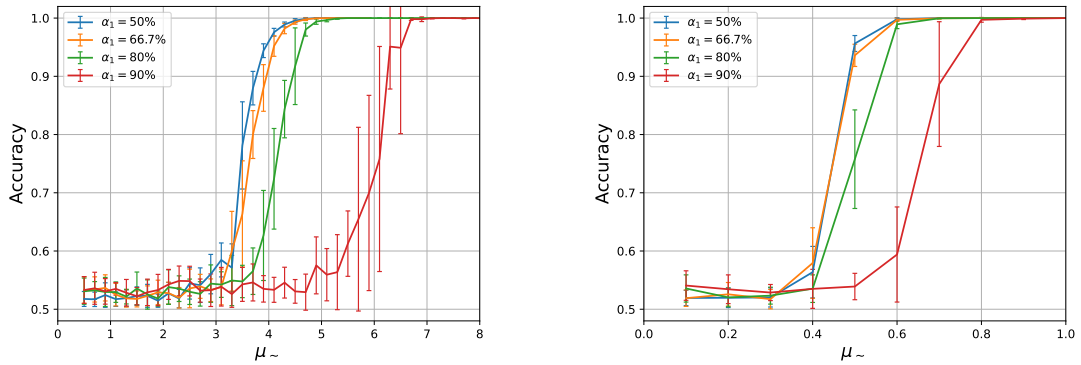
Figure 4.9 shows that, for larger values of $\mu_{\sim}$ and $\mu_{\nsim}$, it is possible to notice a considerable difference in curves $p = q$ and $p = 2q$ for lag-0, the latter being a much easier scenario to detect communities, since it reaches an accuracy of 100% at $\mu_{\sim} \approx 7$, while the other curve only reaches it at $\mu_{\sim} \approx 9.5$. At lag-1 there is also a change in behavior, since the curves grow differently. The non-detection point is different for both, being at $\mu_{\sim} \approx 5$ for $N = 150$ and at $\mu_{\sim} \approx 6.5$ for $N = 300$, but the curious fact observed is that for $N = 300$ the curve seems to reach 100% accuracy a little before the curve for $N = 150$, having an abrupt increase at $\mu_{\sim} \approx 7.4$.



(a) $\mu_{\nsim} = 3$ (b) $\mu_{\nsim} = 7$

Figure 4.10: Community detection accuracy by $\mu_{\sim}$ for different q for lag-0. $N = 150$, $\mu_{\nsim}$ is specified and other parameters are the same as in Figure 4.9.

In Figure 4.10 it is possible to see how the curves $p = q$ and $p = 2q$ differ for different values of $\mu_{\nsim}$ in lag-0. Combining these results with those of Figures 4.9a ($\mu_{\nsim} = 5$) and 4.8a ($\mu_{\nsim} = 0.5$), we can see that the higher the value of $\mu_{\nsim}$, the further apart the two curves will be. Thus, for higher values of $\mu_{\sim}$ and $\mu_{\nsim}$, there are more advantages in using a higher ratio of p/q.



(a) lag-0 (b) lag-1

Figure 4.11: Community detection accuracy by $\mu_{\sim}$ for different community sizes. Other parameters: $N = 300$, $T = 10^6$, $p = 0.3$, $q = 0.15$, $\mu_{\nsim} = 0.5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 10$.

As suggested by the heatmap in Figure 4.4, the more unequal the communities sizes, the harder it seems to be to detect them using Pearson correlation with lag-0. The results in Figure 4.11 confirm that this increase in difficulty actually happens for both lag-0 and lag-1. The curve $\alpha_1 = 50\%$ represents the case with equal size communities. As the value of α_1 increases, an increasingly larger value of $\mu_{\sim}$ is required to achieve the same average accuracy.

Note that the aforementioned behavior occurs for both lag-0 and lag-1, however the difference in the absolute value of $\mu_{\sim}$ to reach an accuracy of 100% between the curves was considerably smaller in lag-1, reaffirming the fact that with lag-1 it is easier to detect communities.

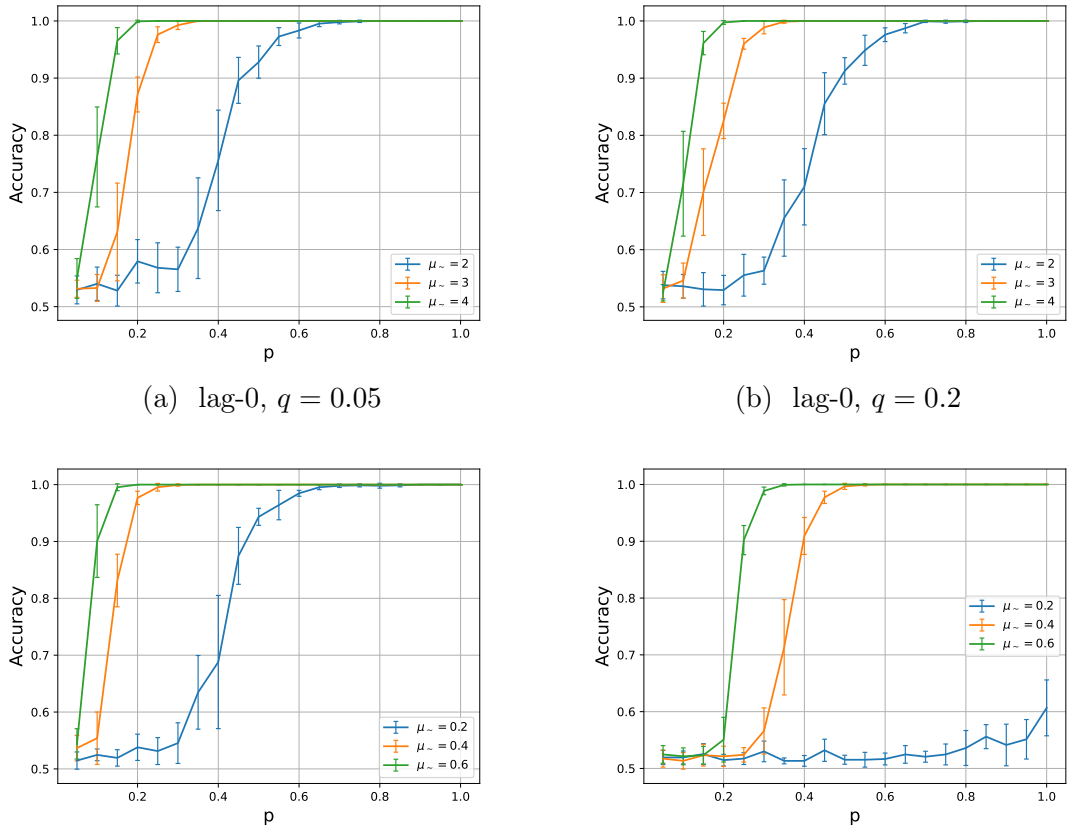
4.5 Accuracy by p

Figure 4.12 shows the relation between community detection accuracy and p. As intuitively expected, the results show that accuracy increases as p increases, with higher accuracies for higher values of $\mu_{\sim}$. This is observed for both lag-0 and lag-1.

When comparing Figures 4.12a and 4.12b, we can notice that the curves are very similar, with detection in the $q = 0.2$ scenario being only slightly more difficult than in $q = 0.05$ for lag-0. This complements the discussion conducted in Section 4.4 about the results of Figure 4.8a ($\mu_{\nsim} = 0.5$, as in the experiments conducted in

this section), since the value of p was the same for the $p = q$ and $p = 2q$ scenarios, changing only q. Combining the analyses of these results, it is possible to notice that changes in p caused a large difference in the scenarios with $\mu_{\nsim} = 0.5$, while changes in q did not have the same impact. One explanation for this result is the fact that, since the weight of intercommunity edges is small, changing the number of these edges will have a reduced impact in the accuracy (which does not occur in the scenario $\mu_{\nsim} = 5$, in Figure 4.9a).

In Figure 4.12d, it can be observed that although the accuracy slightly increased for higher values of p, exact recovery was not achieved for $\mu_{\sim} = 0.2$, not even when $p = 1$, i.e., when all pairs of neurons within the same community were interconnected. Thus, just as the values of $\mu_{\sim}$ and $\mu_{\nsim}$ can make community detection feasible in a graph without community structures (as discussed in Section 4.4), these parameters can also render it unfeasible in graphs with a strong community structure.



(a) lag-0, $q = 0.05$

(b) lag-0, $q = 0.2$

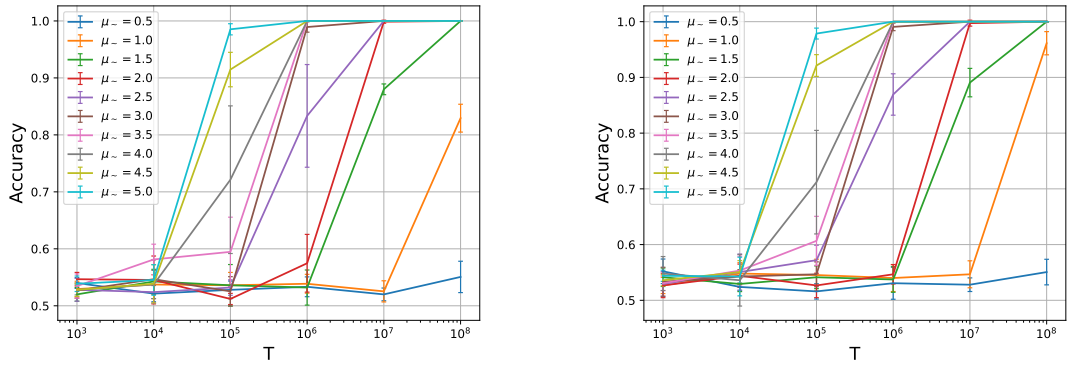
(c) lag-1, $q = 0.05$

(d) lag-1, $q = 0.2$

Figure 4.12: Community detection accuracy by p for different $\mu_{\sim}$. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^6$, $\mu_{\nsim} = 0.5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 10$.

4.6 Accuracy by T

The horizon of time T used in the simulations is important to produce sufficient number of spikes to identify the correlations between the spike trains. In the context of synthetic data, T is directly associated with the execution time of the neuronal activity simulation and the Pearson coefficient calculation, the former being considerably more time consuming. When dealing with real data, the maximum value of T will be determined by the dataset.



(a) $p = q$ (b) $p = 1.5q$

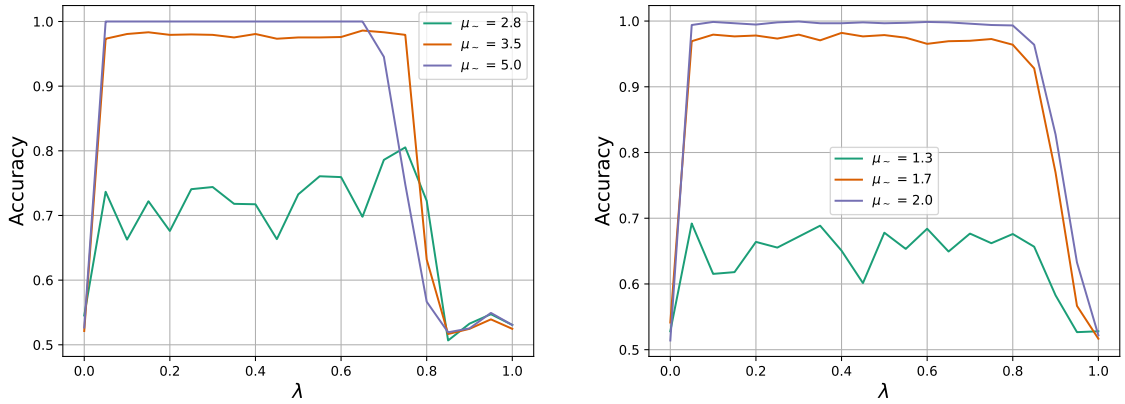
Figure 4.13: Accuracy by T for different values of $\mu_\sim$ for lag-0. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $p = 0.3$, $\mu_\nsim = 0.5$, $\beta = 0.6$, $\lambda = 0.25$, $r = 5$.

As already observed in the previous sections, T is an important parameter for the ability to detect communities. In fact, the results presented in Figure 4.13 reinforce this. We can see that, with the other parameters fixed, the lower the value of $\mu_\sim$, the longer the horizon of time required to obtain high accuracy. For values of $\mu_\sim$ far enough from $\mu_\nsim$, such as $\mu_\sim = 4.5$ and $\mu_\sim = 5$, an accuracy average above 90% was obtained with $T = 10^5$. For values closer to $\mu_\nsim$, such as $\mu_\sim = 1$, a horizon $T = 10^8$ was required to obtain an accuracy higher than 60%.

4.7 Accuracy by λ

The parameter λ defines the probability of spontaneous activation of a neuron, independent of the influence of its in-neighbors. It is critical that $\lambda > 0$, as this ensures that the network has some activity. At time 0, the firing of each neuron is according to Ber(λ), so if $\lambda = 0$, no neuron starts firing and the network remains silent throughout the dynamics.

Results in Figure 4.14 suggest that there is no strong relation between the parameter λ and the accuracy in detecting communities within a considerable range of values of this parameter.



(a) lag-0 (b) lag-1

Figure 4.14: Community detection accuracy by λ for different values of $\mu_{\sim}$. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^6$, $p = 0.3$, $q = 0.15$, $\mu_{\sim} = 2$, $\beta = 0.6$, $r = 10$.

However, for values of λ closer to 1, the accuracy abruptly decreases. This is observed for $\lambda \approx 0.7$ in Figure 4.14a and $\lambda \approx 0.8$ in Figure 4.14b, with visibly faster growth in the former. This behavior is possibly explained by the fact that, with an excess of spontaneous activations in the network, it becomes extremely active, to the point that some neurons are firing in all time units, making it almost impossible to calculate the correlation.

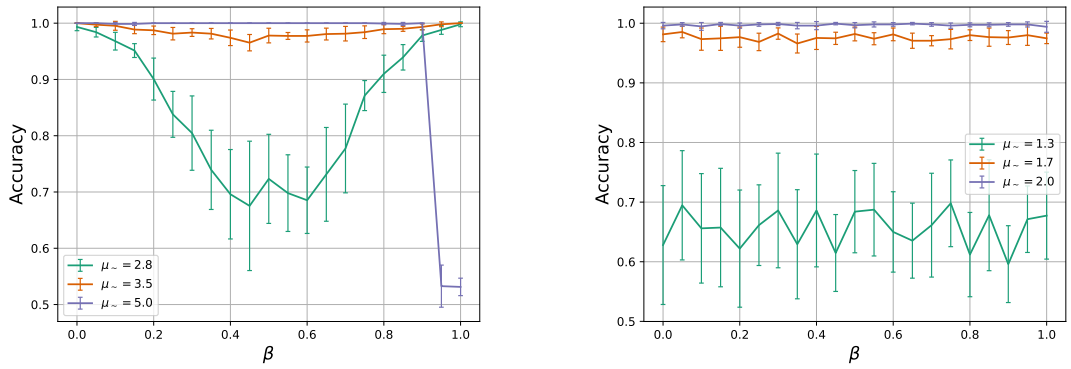
Thus, the results suggest that λ interferes with network activity in a uniform way, without collaborating with or hindering the ability to detect communities, except in extreme cases that will be more discussed in Section 4.9.

It is important to note that $\lambda = 1$ does not guarantee that all neurons will fire at all instants of time, since, as can be seen in Equation 3.2, the probability of firing will be reduced if a neuron has some in-neighbor with an inhibitory connection firing at the previous time.

4.8 Accuracy by β

The parameter β defines the probability of a connection being excitatory. A value of $\beta > 0.5$ favors the creation of networks with a greater tendency towards excitatory connections and, similarly, towards inhibitory connections when $\beta < 0.5$. With $\beta = 0.5$, more balanced networks tend to be generated, with a similar number of excitatory and inhibitory connections.

For lag-0 (Figure 4.15a), it is possible to note that β influence in the accuracy. The steepest curve is that on $\mu_{\sim} = 2.8$ and, considering the high values of standard deviation, we will not consider the actual minimum value, but the shape of the curve. A tendency for reduced accuracy for values of β closer to 0.5 is observed



(a) lag-0 (b) lag-1

Figure 4.15: Community detection accuracy by β for different values of $\mu_\sim$. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^6$, $p = 0.3$, $q = 0.15$, $\mu_\nsim = 2$, $\lambda = 0.25$, $r = 10$.

at $\mu_\sim = 2.8$ and $\mu_\sim = 3.5$.[2] Thus, it can be concluded that very close amounts of excitatory and inhibitory edges make community detection more difficult. The intuition for this is that neurons from the same community in excitatory networks will tend to have positive correlations, while those in inhibitory networks will tend to have negative correlations, but these two results possibly cancel each other out in mixed networks.

Thus, the best accuracies are obtained in fully excitatory or inhibitory networks. Furthermore, we observed symmetry in the curves, which suggests that predominantly excitatory networks do not have great advantages over predominantly inhibitory networks. This result corroborates the importance of using a metric that also captures negative correlations, such as Pearson's correlation.

Exceptions to the above pattern are values of $\beta \geq 0.95$ for $\mu_\sim = 5$ (Figure 4.15a), where a rapid decrease in accuracy is observed. A probable reason for this is that there is an overload of activity in the network. A high value of $\mu_\sim$ combined with a high value of β makes the network very active, which can lead to difficulties in identifying correlations, similar to what occurred with high values of λ as analyzed in Section 4.7.

Interestingly, the results for lag-1 (Figure 4.15b) showed no difference in accuracy for different values of β. This is possibly due to the strong edge detection between neurons in lag-1 correlation, approximating the adjacency matrix regardless of the edge type.

[2]Higher $\mu_\sim$ values imply higher accuracies, which is why the curve in $\mu_\sim = 2.8$ did not occur in $\mu_\sim = 5$, since the latter has an accuracy of 100% even for β values close to 0.5.

4.9 Activity Overload/Underload

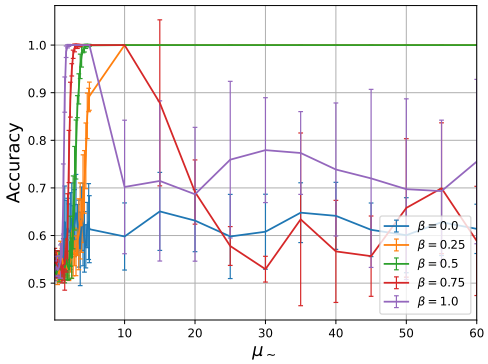
As previously discussed in Sections 4.7 and 4.8, high values of λ and β associated with high values of $\mu_\sim$ can cause an overload of activity in the network, which makes community detection difficult. Intuitively, the same should happen for underload, since the network would be so silent that it would not be possible to detect communities. Similarly, this underload is expected to occur with low values of λ, β and $\mu_\sim$.

In Figure 4.16b it is possible to see the activity underload scenario, since networks with smaller β values require higher $\mu_\sim$ to reach accuracy 1. While the curve $\beta = 1$ reached accuracy 1 with $\mu_\sim < 2$, the curve $\beta = 0.5$ only reached it with $\mu_\sim \approx 4.5$, $\beta = 0.25$ with $\mu_\sim > 5$ and $\beta = 0$ never goes beyond accuracy 0.7 (as can be seen in Figure 4.16a), possibly because the network was too inactive for any correlation to be observed. This pattern does not occur in Figures 4.16d and 4.16f, since the value of λ is high enough to avoid it. We can observe that these two results reveal an easier detection with more extreme values of β, in agreement with what was discussed in Section 4.8. Figure 4.16d seems to show an intermediate result to the other two, since the values of $\beta = 0$ and $\beta = 1$ had the fastest growth, but at the same time, the "worst" curve among the intermediate ones was the one with the lowest β.

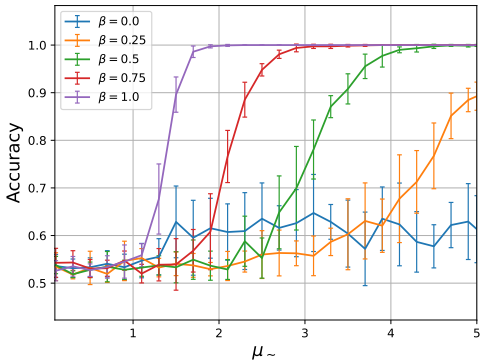
Figures 4.16a, 4.16c and 4.16e show the activity overload scenario, since all values of $\beta > 0.5$ decreased as $\mu_\sim$ increased. In all results, the first curve to decrease is $\beta = 1$, followed by $\beta = 0.75$, which is in line with the expectations, since the higher the values of β, the faster the overload will occur. However, an interesting result is that the highest values of β are not always the lowest in accuracy. In Figures 4.16a and 4.16c, for example, the curve $\beta = 1$ is above $\beta = 0.75$ for almost all values of $\mu_\sim > 20$, even though the former started to decrease at a lower value of $\mu_\sim$.

For lag-1, both overload and underload seem to be more difficult to occur than for lag-0. To begin with, while $\lambda = 0.01$ was already a small enough value to see faster growth at higher values of β for lag-0, this did not occur for lag-1, as can be seen in Figure 4.17d. The difference in growth of the curves at a value of $\lambda = 0.001$ can be observed in Figure 4.17b, but they all reached accuracy 1 at a considerably smaller value of $\mu_\sim$ compared to those for lag-0.

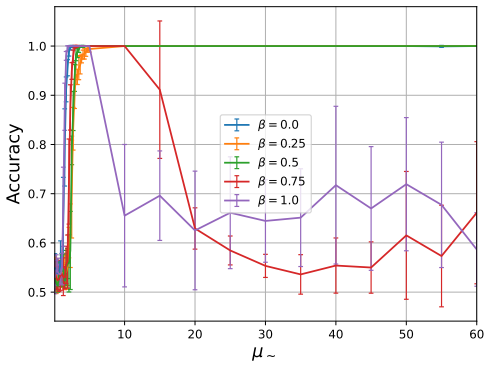
There are two behaviors observed in lag-0 that are also seen in lag-1 (Figures 4.16a, 4.16c and 4.16e). First, the $\beta = 1$ curve decreases earlier and is above the $\beta = 0.75$ curve for larger values of $\mu_\sim$. Second, the larger the value of λ, the smaller the "advantage" of the $\beta = 1$ curve over $\beta = 0.75$.



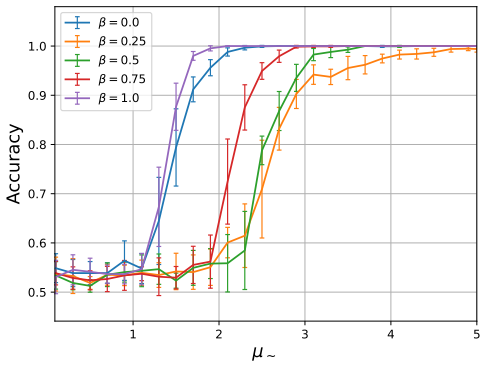
(a) $\lambda = 0.01$



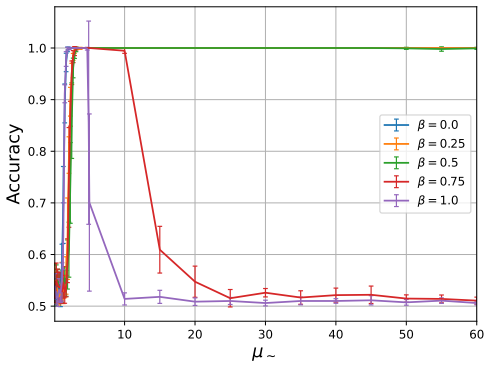
(b) Zoom of (a)



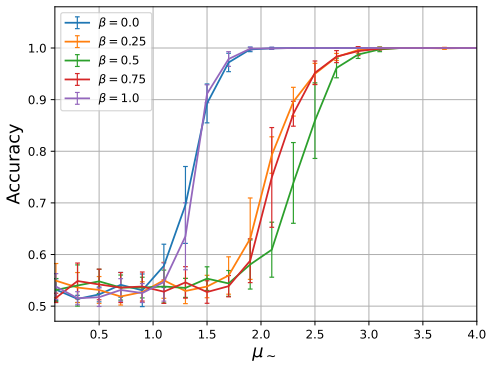
(c) $\lambda = 0.02$



(d) Zoom of (c)

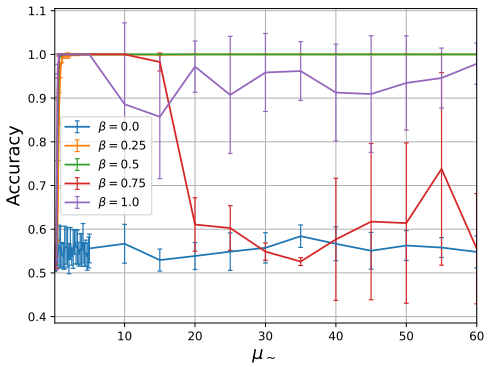


(e) $\lambda = 0.25$

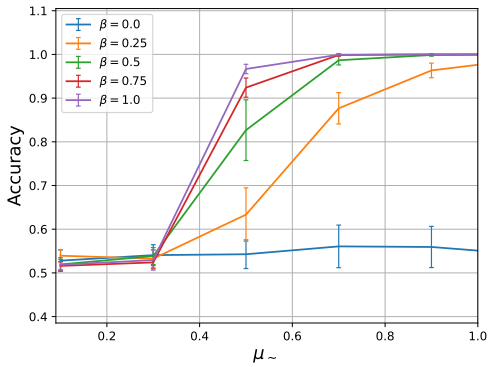


(f) Zoom of (e)

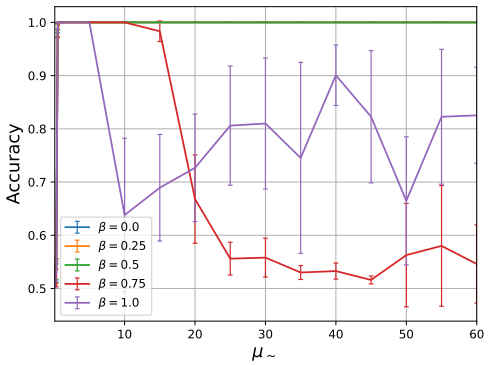
Figure 4.16: Community detection accuracy by $\mu_\sim$ for different values of β for lag-0. There is noise in the left figures because it was used a smaller step for $\mu_\sim \leq 5$. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^6$, $p = 0.3$, $q = 0.15$, $\mu_\sim = 0.5$, $r = 10$.



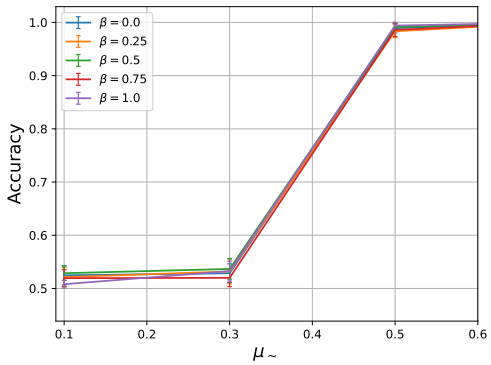
(a) $\lambda = 0.001$



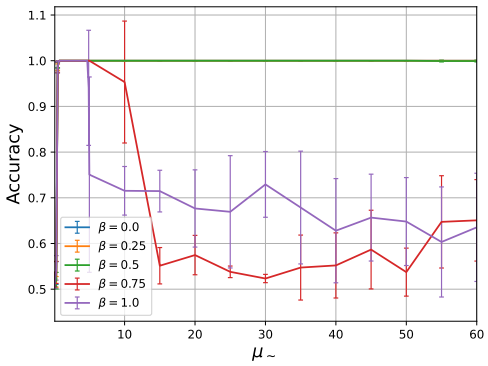
(b) Zoom of (a)



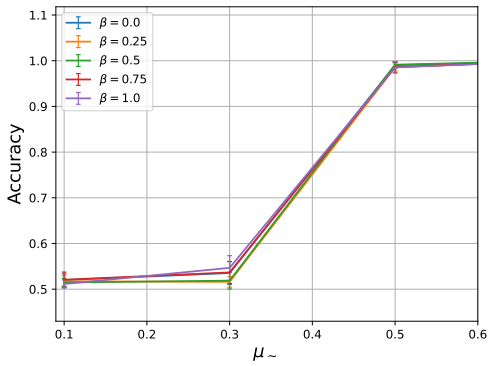
(c) $\lambda = 0.01$



(d) Zoom of (c)



(e) $\lambda = 0.25$

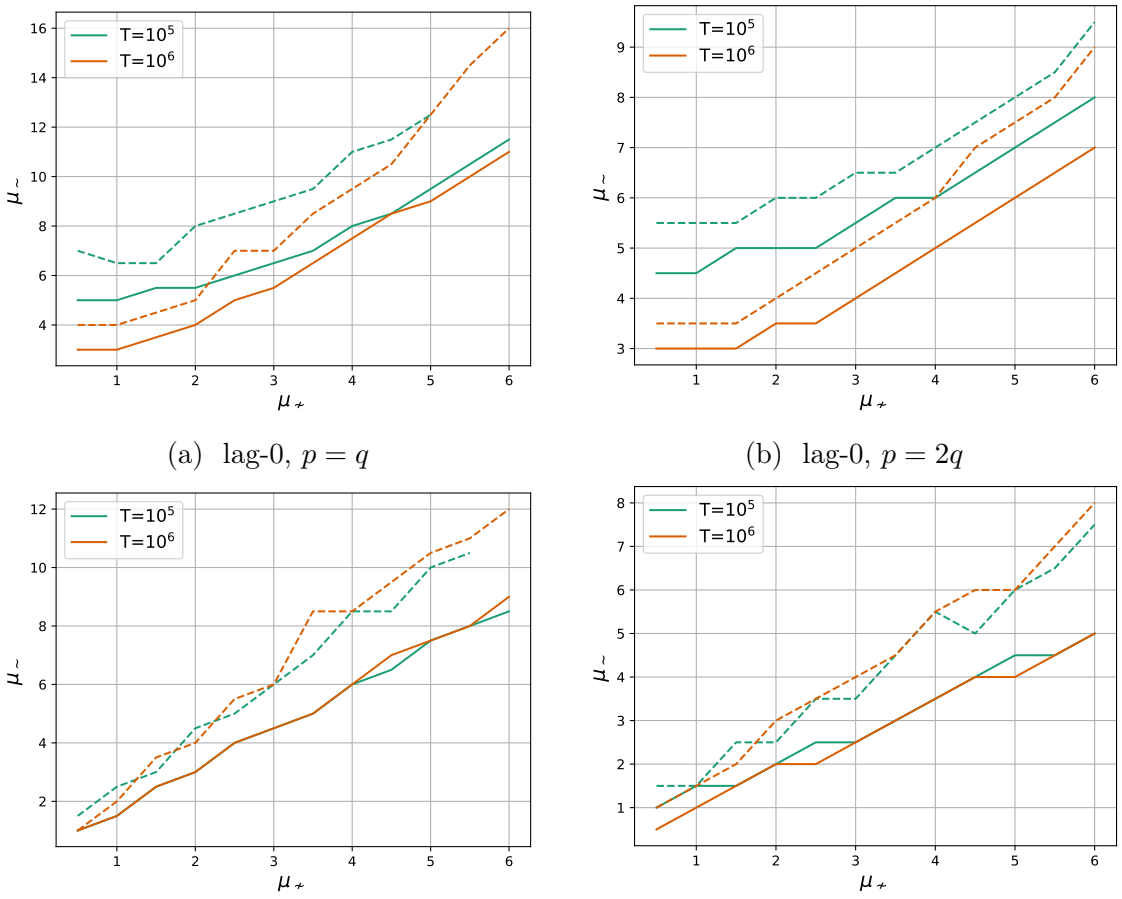


(f) Zoom of (e)

Figure 4.17: Community detection accuracy by $\mu_\sim$ for different values of β for lag-1. There is noise in the left figures because it was used a smaller step for $\mu_\sim \leq 5$. Other parameters are the same as in 4.16.

4.10 Thresholds for Experimental Community Detection

The relation between parameters $\mu_\sim$ and $\mu_\nsim$ and how changes in their values should interfere with the network dynamics is clear, but a question that arises is: as $\mu_\nsim$ increases, at what rate should $\mu_\sim$ be increased to achieve exact recovery?



(a) lag-0, $p = q$

(b) lag-0, $p = 2q$

(c) lag-1, $p = q$

(d) lag-1, $p = 2q$

Figure 4.18: Minimum value of $\mu_\sim$ by $\mu_\nsim$ to achieve 95% of accuracy (solid line) and 100% accuracy (dashed line) with $\mu_\sim \geq \mu_\nsim$ for lag-0[3]. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $p = 0.3$, $\beta = 0.6$, $\lambda = 0.25$, $r = 20$.

Figure 4.18 shows the results for exact recovery (100%) and almost exact recovery (95%). Although we are interested in the exact recovery case, this metric is very sensitive to small perturbations in data. For example, if out of the 20 samples, 19

[3]This figure is intended to show how $\mu_\sim$ must increase as $\mu_\nsim$ increases to keep good accuracy in detecting communities. Using $\mu_\nsim > \mu_\sim$ for lag-0 would lead to an abrupt decrease in the curve for high values of $\mu_\nsim$, since it is possible to detect communities with high accuracy in these scenarios with low values of $\mu_\sim$ (see Section 4.11 for more details). However, the same does not occur for lag-1, as can be seen in Figure 4.21, so this restriction is not needed. Considering that lag-1 correlations are so informative that we can have almost exact recovery with $p = 2q$ and $\mu_\sim < \mu_\nsim$, the restriction was only applied to lag-0.

have accuracy 1, but one has accuracy 0.99, then the result will not be considered as exact recovery. On the other hand, almost exact recovery is less sensitive to such variations.

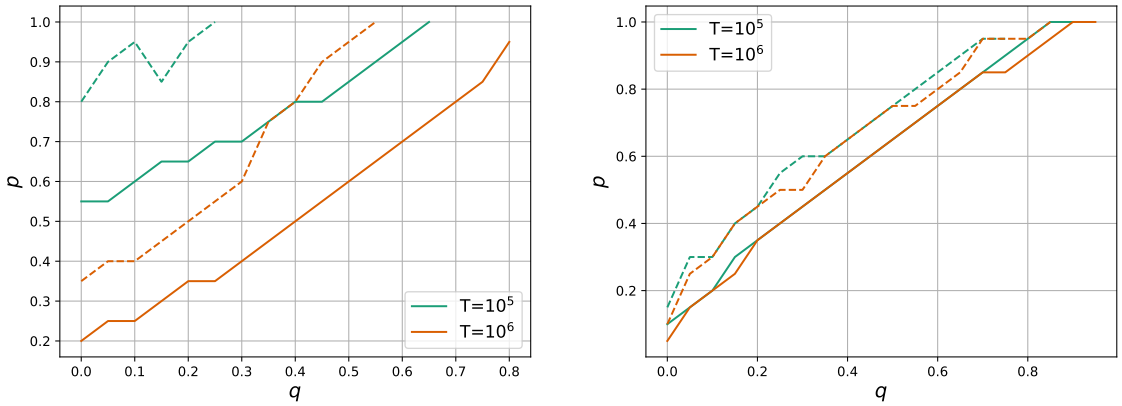
In addition to confirming, as expected, the need for higher values of $\mu_\sim$ given higher values of $\mu_\nsim$, the results suggest a linear dependence between $\mu_\sim$ and $\mu_\nsim$, especially for almost exact recovery. In Figures 4.18a and 4.18b, this relation is more noticeable for $T = 10^6$, although $T = 10^5$ also resembles a linear dependence. It is possible to see that there is a difference in the curves for different values of T, since higher values of $\mu_\sim$ are needed when T is lower, as expected due to the results of the previous sections.

Assuming a linear dependence, it is clear that the lines have different slopes depending on the relation between p and q, with the line $p = 2q$ being steeper. This visual result confirms the results of Figure 4.10, which shows that the difference in the minimum values of $\mu_\sim$ to achieve exact and almost exact recovery between $p = q$ and $p = 2q$ is more evident the higher the value of $\mu_\nsim$ for lag-0.

For lag-1 with $p = 2q$, it was sufficient to use $\mu_\sim \leq \mu_\nsim$ to achieve almost exact recovery. This shows that the structure of the network and the fact that lag-1 correlations are more informative are already sufficient to guarantee community detection, regardless of the connection strength.

Using $\mu_\nsim > \mu_\sim$ for lag-0 would lead to an abrupt decrease in the curve for high values of $\mu_\nsim$, since when $\mu_\nsim$ is

Moreover, the curves of $T = 10^5$ and $T = 10^6$ are very close for lag-1, suggesting that $T = 10^5$ ensures good convergence of the results in this scenario.



(a) lag-0, $\mu_\sim = 3.5$ (b) lag-1, $\mu_\sim = 2$

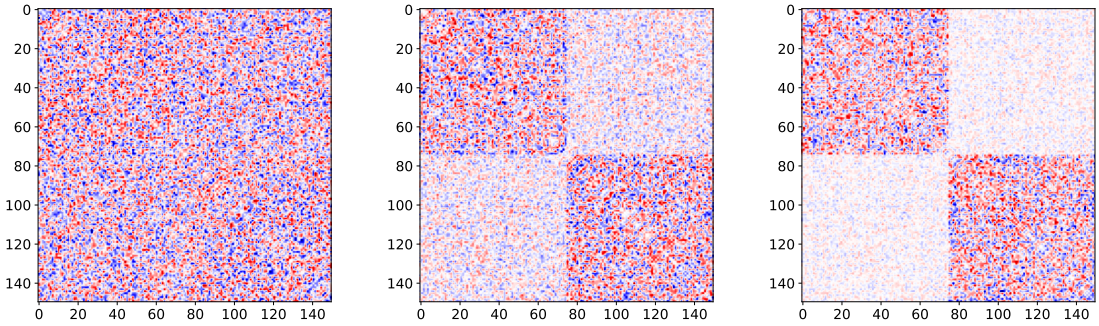
Figure 4.19: Minimum value of p by q (using only $p \geq q$)[4] to achieve 95% of accuracy (solid line) and 100% accuracy (dashed line). $\mu_\nsim = 2$ and other parameters are the same as in Figure 4.18.

[4]Same reason $\mu_\sim \geq \mu_\nsim$ for lag-0 was used in Figure 4.18.

In a similar analysis, Figure 4.19 shows the experimental threshold plots of p by q. As expected, as q increases, higher values of p must be used to achieve the same accuracy when fixing all the other parameters. The results showed in this figure also suggest a linear dependence between the two edge probability parameters.

4.11 Strongly Interconnected Communities

Although the focus of this work is on networks with $p > q$, it is also possible to detect communities with $q > p$ and $\mu_{\nsim} > \mu_{\sim}$. This is in line with threshold results obtained for community detection in SBM (see Section 2.2), which establishes the transition point based on the distance from p to q, regardless of which is greater.



(a) $\mu_{\nsim} = 1$ (b) $\mu_{\nsim} = 2$ (c) $\mu_{\nsim} = 3$

Figure 4.20: Pairwise Pearson correlation heatmaps for lag-0. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^7$, $p = 0.1$, $q = 0.4$, $\mu_{\sim} = 0.5$, $\beta = 0.6$, $\lambda = 0.25$.

Figure 4.20 shows this community structure at lag-0. Analogous to the results for $p > q$, the structure is more noticeable for larger values of $\mu_{\nsim}$ when all other parameters are fixed.

To get an intuition of why communities are well detected in this scenario at lag-0, consider a bipartite graph, i.e., $p = 0$. In this case, with neurons $u \in C_1$ and $v, w \in C_2$, if u fires at time t, there is an increase in the probability of both v and w, which are in the same community, firing simultaneously at time $t + 1$, increasing their lag-0 correlation.

For lag-1, however, the framework used in this work cannot detect communities with $q > p$, since, as shown in Figure 4.21, the correlations with the highest absolute values occur between pairs of neurons from different communities. Thus, spectral clustering is not able to group neurons from the same community, since their correlations are very low. As discussed in Section 4.2, this figure is an expected result, since the correlations in lag-1 are similar to the weighted adjacency matrix, which, in this scenario, has more intercommunity connections.

Although this community detection framework is not suitable for the strongly

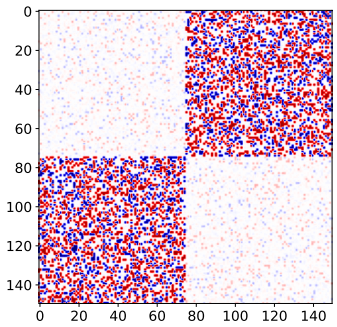


Figure 4.21: Pairwise Pearson correlation heatmap for lag-1. $\mu_{\nsim} = 3$ and other parameters are the same as in Figure 4.20

interconnected scenario, it is possible to make some modifications in the method to handle it better. However, this is out of the scope of this work.

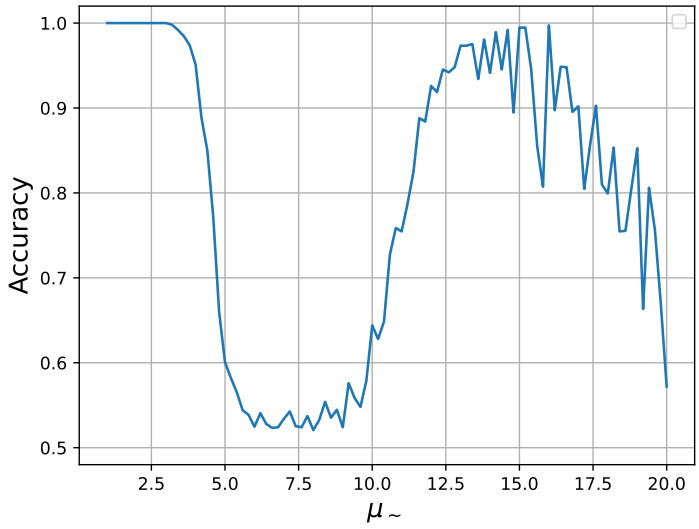


Figure 4.22: Community detection accuracy by $\mu_{\sim}$ for lag-1. Other parameters: $N = 150$, $\alpha_1 = 0.5$, $T = 10^6$, $p = 0.3$, $q = 0.3$, $\mu_{\nsim} = 7$, $\beta = 0.6$, $\lambda = 0.25$, $r = 10$.

Figure 4.22 demonstrates behaviors that summarize much of the discussions held throughout this chapter. To capture the interaction between $\mu_{\sim}$ and $\mu_{\nsim}$, it was used $p = q$. Since $\mu_{\nsim} = 7$, the community detection accuracy is 100% for smaller values of $\mu_{\sim}$, up to approximately $\mu_{\sim} \approx 3$ due to the behavior of strongly interconnected communities, as discussed in this section. After this point, an abrupt decrease in the curve is identified, indicating a phase transition, as can be seen in figures of Section 4.4. For $\mu_{\sim} \in [5, 10]$, the difference between $\mu_{\sim}$ and $\mu_{\nsim}$ is too small for the framework to be able to detect communities, with an average accuracy between 50% and 60% being obtained. After this, there is another phase transition, as the accuracy increases as $\mu_{\sim}$ moves away from $\mu_{\nsim}$, but since the connection strength values are too high at this point, the accuracy starts to drop before even reaching

100% again, as the networks become extremely active, preventing the identification of neurons that have similar stimuli, as discussed in Section 4.9.

4.12 Computation time analysis

Table 4.1 shows the execution time (in seconds) of the simulation and pairwise Pearson correlation for several values of N and T and equally sized communities. The values of the other parameters remained the same among all the experiments. The values shown in the table are the mean of the runtime of 10 samples. The time to calculate the similarity matrix and run the spectral clustering algorithm was negligible, so it is not considered in this analysis.

It is possible to notice that the execution time depends linearly on T, both for simulation and Pearson correlation calculation, such that multiplying the value of the parameter T by 10 would result in a runtime approximately 10 times greater.

In other hand, the results show a superlinear dependence of the runtime and the number of neurons, in a way that doubling the value of N leaded to a runtime approximately three times greater.

	Simulation			**Pearson**			**Total**		
$T \backslash N$	74	150	300	74	150	300	74	150	300
10^4	0.03	0.09	0.3	0.02	0.04	0.15	0.05	0.14	0.46
10^5	0.28	0.85	2.91	0.13	0.36	1.15	0.41	1.23	4.06
10^6	2.68	8.53	29.1	1.27	3.76	11.5	3.95	12.3	40.7
10^7	26.5	84.5	293	13.9	38.4	116	40.4	123	409

Table 4.1: Mean time (in seconds) for simulation, pairwise Pearson correlation calculation and total runtime for 10 samples and different values of N and T. Other parameters: $\alpha_1 = 0.5$, $p = 0.5$, $q = 0.3$, $\mu_\sim = 3$, $\mu_\nsim = 0.5$, $\beta = 0.6$, $\lambda = 0.25$.

The experiments in this section were conducted using a single thread on a machine with the following specifications:

- **CPU:** AMD Ryzen 5 5600G, 3.90GHz
- **RAM:** 32 GB DDR4, 3200MHz
- **Storage:** Kingston NV2 SSD, M.2 2280, NVMe 4.0, 3.5GB/s read speed
- **Operating System:** Ubuntu 22.04.3 LTS running on Windows Subsystem for Linux (WSL) within Windows 10 Home Single Language (64-bit)

Chapter 5

Conclusion

In this work, we simulated a linear neuronal activity model in SBM networks and conducted several experiments to analyse how the parameter models relate to each other and influence in the ability to detect communities with a simple clustering framework.

The results showed a strong interdependence between different parameters, especially p and $\mu_{\sim}$ with q and $\mu_{\nsim}$. In general, the further away p and $\mu_{\sim}$ are from q and $\mu_{\nsim}$, the easier it becomes to detect communities. Also, the results showed a phase transition in values of $\mu_{\sim}$ for the accuracy of the detected communities.

Communities with different sizes were also analysed, showing that the higher the difference in these sizes, the harder it is to detect communities from lag-0.

The horizon of time T also is very important, since higher values of this parameter always guaranteed higher accuracies, unless convergence was already achieved.

In addition to these results, this work analysed extreme scenarios, with very high or very low values of $\mu_{\sim}$, λ and β. These scenarios caused an overload or underload of activity in the simulation, leading to impossibility of detecting communities.

It was also seen that both excitatory and inhibitory connections bring a large amount of information to the spike train sequences, contributing to the ability to detect communities. Indeed, both full excitatory and full inhibitory networks could have their communities fully recovered depending on other parameters of the network.

Furthermore, in all cases with $p \geq q$, lag-1 correlations were much more informative than lag-0, since the influences of neurons on their neighbors were being directly captured. With T large enough, lag-1 correlation matrices are very similar to the weighted adjacency matrix.

Finally, we conducted an analysis of the experimental thresholds for $\mu_{\sim}$ by $\mu_{\nsim}$, where the results suggested a linear dependence between these two parameters. The same was done for p by q, also having the results suggesting a linear dependence between thee two them.

5.1 Future Work

As future work, three main points stand out:

- **Establish theoretical conditions to achieve exact recovery**. As showed in this work, parameters have a strong influence in the capacity to achieve exact recovery. Interesting advances would be finding theoretical conditions in terms of the parameters for this property.

- **Characterize thresholds for community detection in terms of the model parameters**. Experimental results for threshold for $\mu_{\sim}$ by $\mu_{\nsim}$ and p by q were analysed in this work for some scenarios. Still, finding theoretical thresholds in terms of the parameters remains an open problem.

- **Extend the analysis to more complex community graphs**. We have conducted the analysis for graphs with two communities and probabilities p and q. However, the model and analysis can be extent to graphs with more communities and matrices of edge probabilities and weights.

- **Detect communities using lag-0 and lag-1 simultaneously**. In most experiments, lag-1 correlations were considerably more informative than lag-0 ones. Could we get even more informative results for community detection if both were combined in some way to construct similarities?

References

HUMPHRIES, M. D. "Spike-train communities: finding groups of similar spike trains", *Journal of Neuroscience*, v. 31, n. 6, pp. 2321–2336, 2011.

HOFFMANN, T., PEEL, L., LAMBIOTTE, R., et al. "Community detection in networks without observing edges", *Science Advances*, v. 6, n. 4, Jan. 2020. ISSN: 2375-2548. doi: 10.1126/sciadv.aav1478.

PEIXOTO, T. P. "Network Reconstruction and Community Detection from Dynamics", *Physical Review Letters*, v. 123, n. 12, Sep. 2019. ISSN: 1079-7114. doi: 10.1103/physrevlett.123.128301.

LINDERMAN, S., ADAMS, R. P., PILLOW, J. W. "Bayesian latent structure discovery from multi-neuron recordings". In: Lee, D., Sugiyama, M., Luxburg, U., et al. (Eds.), *Advances in Neural Information Processing Systems*, v. 29. Curran Associates, Inc., 2016.

KOBAYASHI, R., KURITA, S., KURTH, A., et al. "Reconstructing neuronal circuitry from parallel spike trains", *Nature Communications*, v. 10, n. 1, pp. 4468, 2019.

PAPALEXAKIS, E. E., FYSHE, A., SIDIROPOULOS, N. D., et al. "Good-enough brain model: challenges, algorithms and discoveries in multi-subject experiments". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, p. 95–104, New York, NY, USA, 2014. Association for Computing Machinery. ISBN: 9781450329569. doi: 10.1145/2623330.2623639.

COELHO, T., CATARCIONE, C., OST, G., et al. "Predicting Spike Train Correlations and Neuron Clusters in Structural Brain Networks", *Under review*, 2025.

CATARCIONE, C. *Analysis of firing rate and correlation of spike trains from a brain network model with clusters.* Master's thesis, Federal University of Rio de Janeiro, 2025.

GILBERT, E. N. "Random Graphs", *The Annals of Mathematical Statistics*, v. 30, n. 4, pp. 1141–1144, 1959. ISSN: 00034851, 21688990.

ERDŐS, P., RÉNYI, A. "On Random Graphs I", *Publicationes Mathematicae Debrecen*, v. 6, pp. 290–297, 1959.

FRIEZE, A., KAROŃSKI, M. *Introduction to Random Graphs*. Cambridge University Press, 2015.

BOLLOBÁS, B. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

HOLLAND, P. W., LASKEY, K. B., LEINHARDT, S. "Stochastic blockmodels: First steps", *Social Networks*, v. 5, n. 2, pp. 109–137, 1983. ISSN: 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90021-7.

HAGBERG, A., SWART, P., CHULT, D. "Exploring Network Structure, Dynamics, and Function Using NetworkX". 06 2008. doi: 10.25080/TCWV9851.

GIRVAN, M., NEWMAN, M. E. J. "Community structure in social and biological networks", *Proceedings of the National Academy of Sciences*, v. 99, n. 12, pp. 7821–7826, 2002. doi: 10.1073/pnas.122653799.

BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R., et al. "Fast unfolding of communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, v. 2008, n. 10, pp. P10008, Oct. 2008. ISSN: 1742-5468. doi: 10.1088/1742-5468/2008/10/p10008.

ABBE, E., BANDEIRA, A. S., HALL, G. "Exact Recovery in the Stochastic Block Model", *CoRR*, v. abs/1405.3267, 2014.

LUXBURG, U. "A tutorial on spectral clustering", *Statistics and Computing*, v. 17, n. 4, pp. 395–416, Dec. 2007. ISSN: 0960-3174. doi: 10.1007/s11222-007-9033-z.

LEVIN, D., PERES, Y. *Markov Chains and Mixing Times*. MBK. American Mathematical Society, 2017. ISBN: 9781470429621.

HÄGGSTRÖM, O. *Finite Markov Chains and Algorithmic Applications*. London Mathematical Society Student Texts. Cambridge University Press, 2002. ISBN: 9780521890014.

GUENNEBAUD, G., JACOB, B., OTHERS. "Eigen v3". http://eigen.tuxfamily.org, 2010.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.

BORG, I., GROENEN, P. *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. Springer, 08 2005. ISBN: 9781475727135. doi: 10.1007/978-1-4757-2711-1.

TRAAG, V. A., WALTMAN, L., VAN ECK, N. J. "From Louvain to Leiden: guaranteeing well-connected communities", *Scientific Reports*, v. 9, n. 1, Mar. 2019. doi: 10.1038/s41598-019-41695-z.