Gleyce Alves Pereira da Silva

Algoritmo de Discretização e Solvência da Dinâmica de Multiestados de um Sistema Contínuo

Recife

28/02/2023

Universidade Federal Rural de Pernambuco

Pró-Reitoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em Biometria e Estatística Aplicada

Algoritmo de Discretização e Solvência da Dinâmica de Multiestados de um Sistema Contínuo

Dissertação submetida como exigência parcial adequada para obtenção do título de Mestre em Biometria e Estatística Aplicada.

Área de concentração: Biometria e Estatística Aplicada

Orientador: Dr. Cláudio Tadeu Cristino

Recife

28/02/2023

Universidade Federal Rural de Pernambuco
Pró-Reitoria de Pesquisa e Pós-Graduação
Programa de Pós-Graduação em Biometria e Estatística Aplicada

Algoritmo de Discretização e Solvência da Dinâmica de Multiestados de um Sistema Contínuo

Gleyce Alves Pereira da Silva

Dissertação submetida como exigência parcial adequada para obtenção do título de Mestre em Biometria e Estatística Aplicada.

Orientador:

Dr. Cláudio Tadeu Cristino
Orientador

Banca examinadora:

Dr. Antonio Samuel Alves da Silva
Universidade Federal Rural de Pernambuco.

Dr.a Edneide F. Ramos Ramalho
Universidade Federal de Pernambuco.

Este trabalho é dedicado aos meus pais, Ana e Timoteo,
e minhas irmãs, que foram meu alicerce
durante essa jornada.

Agradecimentos

Agradeço aos meus pais, Ana e Timoteo, por me ensinarem a importância dos estudos e me proporcionarem uma criação com amor e incentivo para que eu continuasse nesse caminho. Às minhas irmãs, Geovanna e Gleyciane, agradeço por serem meu alicerce durante toda essa jornada. Aos amigos, meus mais profundos agradecimentos.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES por me disponibilzarem uma bolsa de estudos a qual tornou possível a relização do mestrado. À Universidade Federal Rural de Pernambuco - UFRPE, onde passei 7 anos, incluindo graduação e mestrado, minhas mais sinceras gratidões por essa instituição.

Ao meu orientador, o professor Dr. Cláudio Tadeu Cristino, agradeço por toda sua prestabilidade e dedicação durante os anos, cuja orientação foi parte primordial para a finalização desse trabalho e cujo o incentivo durante a graduação me levou ao ingresso no mestrado e continuidade no meio acadêmico.

"A ciência é mais que um corpo de conhecimento,
é uma forma de pensar,
uma forma cética de interrogar o universo
com pleno conhecimento da falibilidade humana."
(Carl Sagan)

Resumo

Dada a incerteza em que operam os sistemas reais, principalmente quando este envolve, por sua natureza, ações humanas imprevisıveis ou avarias de máquinas. Faz-se necessário a busca por modelos determinısticos, que contribuam para a compreensão do comportamento dinâmico de um sistema, a nıvel básico. Tais sistemas podem ser descritos por modelos probabilısticos, aproveitando certas caracterısticas de regularidade que eles apresentam. Assim, pode-se recorrer a Processos Estocásticos como uma forma de tratar quantitativamente estes fenômenos, a depender de certas caracterısticas pode recorrer a Processos Markovianos. Dado um sistema dinâmico, cuja dinâmica pode ser dada no tempo contınuo ou discreto, faz-se necessário um estudo de sua evolução de estados ao longo do tempo. Em algumas aplicações, a distinção entre sistemas contínuos e discretos não é crítica, e a escolha se dá por conveniência. Esse trabalho busca a descrição de um algoritmo que seja capaz de discretizar o sistema estudado, de maneira natural, formando assim um grafo conexo, para poder estudar a dinâmica de estados deste mesmo grafo ao longo do tempo. Podendo assim, dado o input, gerar o output adequado, respondendo assim as questões pertinentes ao sistema.

Palavras-chaves: Grafos. Espaço de Estados. Processos de Difusão. Modelagem. Modelos SIR.

Abstract

Given the uncertainty in which real systems operate, especially when it involves, by its nature, unpredictable human actions or machine malfunctions. It becomes necessary to search for determinıstic models, which contribute to the understanding of the dynamic behavior of a system, at the basic level. Such systems can be described by probabilıstic models, taking advantage of certain features of regularity that they exhibit. Thus, one can resort to Stochastic Processes as a way to treat these phenomena quantitatively, depending on certain characteristics one can resort to Markov Processes. Given a dynamical system, whose dynamics can be given in continuous or discrete time, a study of its evolution of states over time is necessary. In some applications, the distinction between continuous and discrete systems is not critical, and the choice is made for convenience. This work seeks the description of an algorithm that is able to discretize the system studied, in a natural way, thus forming a connected graph, in order to study the dynamics of states of this same graph over time. Thus, given the input, it can generate the appropriate output, thus answering the questions pertinent to the system.

Keywords: Graphs. State Spaces. Diffusion process. Modelling. SIR Model.

Lista de Figuras

Sumário

1 Introdução

Os sistemas podem ser descritos por modelos probabilísticos, aproveitando certas características de regularidade que eles apresentam, podendo-se assim recorrer a Processos Estocásticos como uma forma de tratar quantitativamente estes fenômenos, a depender de certas características pode-se recorrer a Processos Markovianos. Os sistemas podem ser tratados através de equações diferenciais parciais, cálculo variacional, equações diferenciais ordinárias, álgebra, entre outros, o modelo de tratamento vai variar de acordo com a característica do sistema em questão. Zuben (2003) afirma ainda que encontrar a descrição matemática de um sistema equivale a determinar um conjunto de relações matemáticas entre os atributos dos objetos que compõem o sistema, a partir do conhecimento das relações entre estes objetos e entre os atributos de um mesmo objeto. Podendo-se assim associar a um sistema dinâmico uma entidade matemática precisa, a dinâmica pode ser dada no tempo contínuo ou discreto, e sua evolução no tempo é não antecipativa, ou seja a cada instante de tempo $t \in T$ (onde $T \in \mathbb{R}^+_{\{0\}}$ ou $\mathbb{Z}^+_{\{0\}}$ se o sistema operar no tempo contínuo ou discreto respectivamente) o sistema dinâmico recebe alguma entrada $u(t) \in U$ e emite alguma saída $y(t) \in Y$, (onde $U, Y \in \mathbb{R}^n$, em que n não é necessariamente o mesmo para U e Y).

Todo sistema opera com um conjunto de valores das grandezas físicas, denominado como *estado*, que tem é necessário e suficiente para caracterizar univocamente a situação física deste sistema. Zuben (2003) afirma que em uma abordagem de entrada-saída, pode-se dizer que o estado é uma parametrização dos pares de entrada-saída. A evolução do estado de um sistema dinâmico ao longo do tempo é denominada trajetória (ou fluxo, ou órbita) no espaço de estados, sendo determinada pela função de transição de estado. Um sistema dinâmico é dito de tempo contínuo se e somente se a variável temporal t assume valores contínuos, e é dito de tempo discreto se e somente se a variável temporal t assume valores discretos. Da mesma maneira um sistema dinâmico é dito de estado contínuo se e somente se o espaço de estados X é contínuo, e dito de estado discreto se e somente se o espaço de estados X é discreto.

Por vezes se faz necessário discretizar o sistema para tornar os cálculos dos valores mais fáceis de serem feitos, por vezes essa discretização ocorre de maneira natural, sistemas de tempo contínuo correspondem, por exemplo, aos modelos da física clássica, enquanto

que sistemas de tempo discreto surgem, por exemplo, sempre que computadores digitais fazem parte do sistema. Em algumas aplicações, a distinção entre sistemas contínuos e discretos não é crítica, e a escolha se dá por conveniência. Esse trabalho busca a descrição de um algoritmo que seja capaz de discretizar o sistema estudado, de maneira natural, formando assim um grafo conexo, para poder estudar a dinâmica de estados nesse grafo ao longo do tempo. Podendo assim, dado o *input*, gerar o *output* adequado, respondendo assim as questões pertinentes ao sistema.

Em dezembro 2019, na cidade de Wuhan na China, foi constatado um espalhamento em escala global da doença, denominada de COVID-19, causada pelo novo Coronavírus (SARS-CoV-2). Desde então diversos trabalhos foram produzidos na área, gerando assim um volume deveras denso de publicações, impossibilitando a revisão bibliogŕafica. Uma simples procura no google scholar, dada as palavras-chaves *covid19, dynamic* ,retorna quase 3 milhões de trabalhos publicados.

2 Objetivos

2.1 Objetivo Geral

Escrever um algoritmo que discretize um sistema, composto por três estados puros ou combinações desses estados, e descreva a dinâmica nesse grafo ao longo do tempo relativamente à mudança de estados representada pelos vértices do grafo. Este algoritmo dever ser capaz de responder com precisão fixada, determinada pelo usuário, às questões tais como em qual estado se encontra o sistema após determinado tempo, ou quantas mudanças de estados são necessárias para que o sistema adquira determinada configuração

2.2 Objetivos Específicos

- Descrever um sistema que pode ter três estados puros ou combinações de estados. Tal sistema, ou conjunto de unidades, será denotado por $\mathcal{U}$, onde o estado do sistema é dado por $S_t(\mathcal{U}) \subset \mathbb{R}^{n-1}$, em particular nesse trabalho será estudado o $S_t(\mathcal{U}) \subset \mathbb{R}^2$;
- Gerar um sistema de coordenadas que localiza o estado atual do sistema relativamente aos três estados puros;
- Descrever a dinâmica estocástica dos estados dada por uma função $\pi(\tau, \mathbf{y} \,|\, \sigma, \mathbf{x})$ que é a probabilidade de que a configuração de estados seja $\mathbf{y} \in S_t(U)$ no instate τ, dado que no instante σ, $\sigma < \tau$, a configuração seja $\mathbf{x} \in S_t(U)$. Estas equação deverão obedecer ao Teorema de Kolmogorov (Øksendal, 2010).
- Escrever o algoritmo que dado um valor $p = k^{-1}$, $k = 2, 3, \ldots$, discretize o espaço de estados, gerando de maneira sistemática um grafo e o modelo de percolação sobre o mesmo;
- Determinar o intervalo de confiança e fazer testes de hipóteses necessários para se determinar a precisão da solução apresentada.

3 Metodologia

Como descrito na seção anterior, esse trabalho visa buscar um método de solução de dinâmicas de estados utilizando estruturas discretas. As estruturas discretas utilizadas serão os grafos, que surgirão naturalmente após aplicação do algoritmo a ser construído para esse fim.

3.1 Processos Estocásticos

Segundo Alves, Menezes e Zimmermann (2006), o processo estocástico é definido como um coleção indexada de variáveis aleatórias $X(t); t \in T$, onde T normalmente é composto por números não-negativos, e X_t é a característica mensurável de interesse no tempo t, os valores assumidos por X_t são chamados de estados, e o conjunto de todos os possíveis valores forma o espaço de estado do processo. Alves e Delgado (1997) afirmam que pode-se classificar os processos estocásticos analisando-se:

1. O espaço de estados $\rightarrow$ se X for um conjunto de estados finitos ou contável ($X = 0, 1, 2, ...$, ou seja, o conjunto de inteiros não negativos), $X(t)$ é um "processo de estados discretos" ou, como é usualmente referido, uma "cadeia", caso contrário o processo é designado por "processo em estado contínuo";

2. A variável temporal $\rightarrow$ se X for finito ou contável, $X(t)$ é um "processo de tempo discretos", caso contrário $X(t)$ é designado por "processo em tempo contínuo";

3. Características estatísticas das variáveis aleatórias $\rightarrow$ o processo diz-se estacionário se o seu comportamento for independente do tempo, Markoviano se for estacionário e gozar da propriedade de Markov ou da "perda de memória", em que seu comportamento futuro depender apenas pelo estado presente, ou ainda, Semi-Markov em que acontecimentos sucessivos deixam de estar "restritos" à distribuição exponencial, podendo seguir qualquer distribuição de probabilidade

3.2 Processos Markovianos

Processos de Markov constituem um tipo especial de processo estocástico, que segundo Alves, Menezes e Zimmermann (2006) possui a propriedade de que as probabilidades associadas com o processo num dado instante do futuro dependem somente do estado presente, sendo, portanto, independentes dos eventos no passado. Desse modo, os processos markovianos são caracterizados pelo que se designa como "falta de memória", ou mais formalmente, o processo estocástico $X(t); t \in T$ é dito markoviano, quando para qualquer tempo, $t_0 < t_1 < ... < t_n < t$, a distribuição condicional de X_t para os valores dados de $X_{t_0}, X_{t_1}, ..., X_{t_n}$ dependem somente de X_{t_n}.

Ainda segundo Alves, Menezes e Zimmermann (2006) é fundamental no estudo de processo de Markov a noção de estado. Propriedade comum entre indivíduos (ou objetos) caracterizam o que designamos por estados. Os processos de Markov sempre envolvem a a variável "tempo", seja considerada de forma discreta onde o tempo varia em saltos, ou de formas contínuas podendo assumir valores reais.

3.3 Diagrama de fases

O diagrama de fases é uma representação gráfica dos estados físicos de um sistema sob diferentes condições de temperatura e pressão. Um diagrama de fase típico tem pressão no eixo y e temperatura no eixo x.

Gráficos ternários ou diagramas ternários são representações triangulares das composições de objetos que podem ser expressas por misturas de três componentes. Os gráficos ternários são encontrados em diversas áreas, como a química e geologia.Na geologia comumente são utilizados para demonstrar a composição de rochas e minerais, dependendo do conteúdo mineral, proporção de extremidades ou mesmo composição química. No presente trabalho, utilizaremos os diagramas ternários, veja abaixo um exemplo:

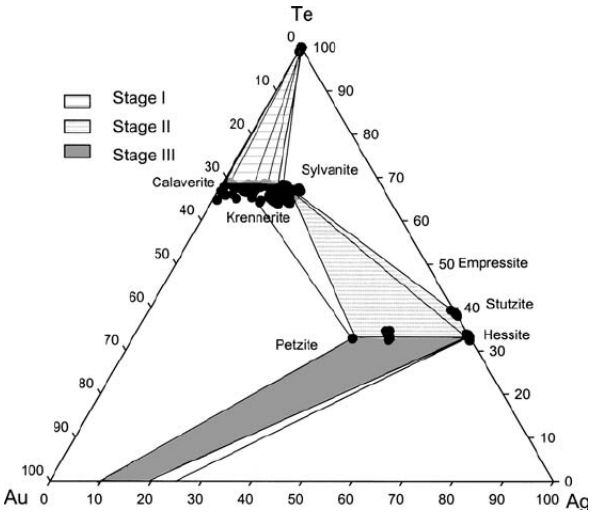

Figura 1 – Exemplo de um sistema Au, Ag e Te representado por um digrama ternário.

O digrama 1 constituido por Au, Ag e Te em proporções atômicas para os minerais analisados no presente estudo do depósito do Imperador. Todos os dados estão listados em Pals (2002). As linhas sólidas indicam composições representativas de fases coexistentes. As assembleias de estágio II são ricas em Te e Au, enquanto as assembleias de estágio III são ricas em Ag. Os conjuntos de metais preciosos do estágio IV são pobres em Te e contêm teluretos ricos em Ag mais ouro nativo = electrum. Observe o continuum de composições entre calaverita, krennerita e silvanita.

3.4 Grafos

3.4.1 Definições

Definição 3.4.1 (Grafo)**.** Denominamos de *grafo* um par $G = (V, E)$ que satisfaz $E \subseteq [V]^2$, onde $[V]^2$ representa o conjunto de todos os subconjuntos de 2 elementos de V.

Para evitar ambiguidades de notação, assume-se tacitamente que $V \cap E = \varnothing$. Os elementos de V são os vértices do grafo G e os elementos de E são as suas arestas. O conjunto de vértices de um grafo G é referido como $V(G)$ e a suas arestas definidas como $E(G)$. Uma aresta $\{x, y\}$ é geralmente escrita como xy.

Definição 3.4.2 (Tamanho de um Grafo)**.** Seja $G = (V, E)$ um grafo. A *ordem* de é o número de vértices em G, isto é, $|V|$. O *tamanho* de G é o número de arestas, isto é, $|E(G)|$.

Definição 3.4.3 (Distância de dois vértices)**.** A *distância* $d_G(u, v)$ em G de dois vértices u, v é o comprimento do caminho mais curto uv em G, se não existe tal caminho, definimos $d(u, v) := \infty$. A maior distância entre quaisquer dois vértices no grafo G é o *diâmetro* de G, indicados por $diam(G)$.

Definição 3.4.4 (Adjacência)**.** Dado um *grafo* G, dois vértices x, y de G são *adjacentes*, ou *vizinhos*, se xy é uma aresta de G.

Na definição utilizada não será permitida a existência de laços (arestas cujos vértices iniciais e finais são iguais), nem multi-arestas (duas ou mais arestas com os mesmos vértices iniciais e finais).

Definição 3.4.5 (Grau de um vértice)**.** Seja um grafo $G = (V, E)$ com u e $v \in V$. O grau de v é o número de arestas com as quais v é incidente. O grau de v se denota por $\Gamma_G(v)$ ou, se não houver risco de confusão, simplesmente por $\Gamma(v)$. Em outras palavras $\Gamma(v) = |N(v)|$.

Na aplicação de grafos para modelagem, o grau de um vértice representa as relações entre indivíduos. Nesse caso essas ligações estão relacionadas com quais relações estamos querendo desenvolver para esses individuos.

Definição 3.4.6 (Vizinhança do vértice v_ℓ)**.** Considerando o grafo $G = (V, E)$ e supondo que u e v sejam vértices de G. Se u e v são adjacentes, dizemos que u e v são vizinhos. O conjunto de todos os vizinhos de um vértice v é chamado vizinhança de v e se denota por $N(v)$. Isto é, $N(v) = u \in V : u \sim v$. O número de vizinhos de um vértice é chamado grau do vértice.

Para esse trabalho é importante salientar que os sistemas trabalhados são sistemas dinâmicos complexos com relações locais, ou seja, um vértice muito distante não irá afetar a dinâmica pois a probabilidade de que um vértice passe para outro em um período muito curto é de zero, consequentemente a dinâmica se dá entre vizinhanças.

3.4.2 Representação de um Grafo

Um grafo pode ser representado através da *lista de adjacência* onde é armazenado o relacionamento entre os vértices de uma estrutura de listas e também pode ser repsentado através de matrizes. Uma representação importante é através da matriz de adjacência.

Definição 3.4.7 (Matriz de Adjacência)**.** Dado um *grafo* G, sua *matriz de adjacência* $M_{m\times m}$ (onde m é o total de vértices do grafo) é denominada por $A = (a_{ij})_{nxn}$ e definida da seguinte forma:

$$a_{ij} = \begin{cases} 1, & \text{se} \quad v_i v_j \in E \\ 0, & \text{caso contrário.} \end{cases}$$

Definição 3.4.8 (Matriz de Incidência)**.** Um grafo $\mathcal{G}_{\mathcal{N}}$, com n vértices e m arestras, pode ser representado por uma matriz bidimensional M_{mxn} denominada de matriz de incidência, onde suas dimensões são dadas pelos vértices e arestas.

3.4.3 O Espaço de Estados

Seja $\mathcal{T}$ um sistema constituído por três estados puros distintos e únicos, denotados por S_0, S_1 e S_2, ou como uma combinação convexa destes três estados:

$$x_0 S_0 + x_1 S_1 + x_2 S_2 \tag{3.1}$$

em que $x_j \geq 0$, $j = 0, 1, 2$ e $x_0 + x_1 + x_2 = 1$. Deve-se notar que se for denotado por $S_{\mathcal{T}}$ o estado atual do sistema o mesmo pode ser descrito pelas "coordenadas" (x_0, x_1, x_2) relativamente à base de estados puros $S_0 = (1, 0, 0)$, $S_1 = (0, 1, 0)$ e $S_2 = (0, 0, 1)$. O *espaço de estados*, o conjunto que representa todos os estado possíveis do sistema $\mathcal{T}$ pode ser representado como um triângulo equilátero (Figura 2) e as coordenadas do estado atual ser dada via coordenadas baricêntricas relativas aos estados puros. Denotemos o espaço de estados por $\mathbb{T}$ e, obviamente é uma variedade de dimensão 2 com bordo, representada em $\mathbb{R}^3$ pelo conjunto:

$$\{(x, y, z) \in \mathbb{R}^3 \, : \, x + y + z = 1, x \geq 0, y \geq 0, z \geq 0\}.$$

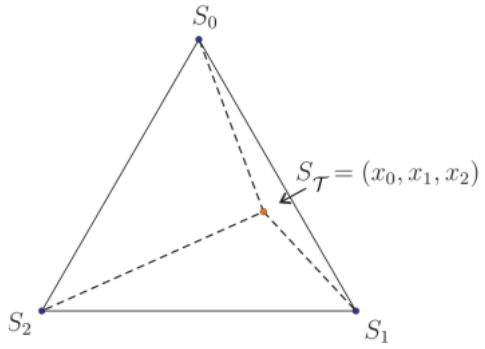

Figura 2 – Espaço de estados $\mathcal{T}$.

O estado atual é considerado como uma função do tempo, $S_{\mathcal{T}} = S_{\mathcal{T}}(t)$ e a dinâmica de tal estado é dada pela função de probabilidade condicional a seguir:

$$\pi(\tau, E \,|\, \sigma, \mathbf{x}) \tag{3.2}$$

é a probabilidade de se ter o estado $S_{\mathcal{T}}(\tau) \in E \subset \mathbb{T}$ (considerando E como um conjunto arbitrário, aberto e mensurável), no instante τ, tal que no instante σ o estado era $(\sigma) = \mathbf{x} = (x_0, x_1, x_2)$. Tal movimento é chamado *processo de Markov*. A completa caracterização do movimento aleatório é dada pela densidade de probabilidade condicional $\pi(\tau, E \,|\, \sigma, \mathbf{x})$ correspondente à distribuição condicional $\Pi(\tau, E \,|\, \sigma, \mathbf{x})$, isto é,

$$\Pi(\tau, E \,|\, \sigma, \mathbf{x}) = \int_{E \subseteq \mathbb{T}} \pi(\tau, \mathbf{y} \,|\, \sigma, \mathbf{x}) d\mathbf{y} \tag{3.3}$$

com

$$\int_{\mathbb{T}} \pi(\tau, \mathbf{y} \,|\, \sigma, \mathbf{x}) d\mathbf{y} = 1 \tag{3.4}$$

e sendo válida a *igualdade de Markov*

$$\pi(\tau, \mathbf{y} \,|\, \sigma, \mathbf{x}) = \int_{\mathbb{T}} \pi(s, \mathbf{z} \,|\, \sigma, \mathbf{x}) \cdot \pi(\tau, \mathbf{y} \,|\, s, \mathbf{z}) d\mathbf{z} \tag{3.5}$$

Consideramos que seja válida a condição de *continuidade forte* para o processo de Markov, isto é,

$$\lim_{\Delta\sigma \to 0} \frac{1}{\Delta\sigma} \int_{|\mathbf{y}-\mathbf{x}|>\delta} \pi(\sigma, \mathbf{y} \,|\, \sigma - \Delta\sigma, \mathbf{x}) d\mathbf{y} = 0 \tag{3.6}$$

para todo $\delta > 0$. Assim a probabilidade de que um ponto aleatório tomará grandes incrementos em um pequeno intervalo de tempo tem medida nula.

Também serão consideradas as seguintes condições:

1. As funções
$$\frac{\partial\,\pi(\tau,\mathbf{y}\,|\,\sigma,\mathbf{x})}{\partial\,x^i} \quad \text{e} \quad \frac{\partial^2\,\pi(\tau,\mathbf{y}\,|\,\sigma,\mathbf{x})}{\partial\,x^i\,x^j},$$
existem e são funções contínuas para todo $i,j \in \{0,1,2\}$, para todo $\mathbf{x},\mathbf{y} \in \mathbb{T}$ e para todo $\sigma,\tau \in^+$, $\sigma < \tau$.

2. Para todo $\delta > 0$, os limites abaixo existem:
$$\lim_{\Delta\sigma\to 0}\frac{1}{\Delta\sigma}\int_{|\mathbf{x}-\mathbf{y}|<\delta}(y^i-x^i)\pi(\tau,\mathbf{y}\,|\,\sigma-\Delta\sigma,\mathbf{x})d\mathbf{y} = b^i(\sigma,\mathbf{x}), \qquad i=0,1,2. \tag{3.7}$$
$$\lim_{\Delta\sigma\to 0}\frac{1}{\Delta\sigma}\int_{|\mathbf{x}-\mathbf{y}|<\delta}(y^i-x^i)(y^j-x^j)\pi(\tau,\mathbf{y}\,|\,\sigma-\Delta\sigma,\mathbf{x})d\mathbf{y} = 2a^{ij}(\sigma,\mathbf{x}), \quad i,j=0,1,2. \tag{3.8}$$

Estes dois limites são costumeiramente considerados uniformes no sentido da definição segundo ε's e δ's. Também pelo Teorema de Heine (KRANTZ, 2005), quando o processo está restrito a um conjunto compacto do ^{n} a uniformidade é automática.

Teorema 3.4.1 (Kolmogorov)**.** *Sobre as condições acima,* $\pi(\tau,\mathbf{y}\,|\,\sigma,\mathbf{x})$ *como uma função de* σ *e* $\mathbf{x}$ *satisfaz:*
$$a^{ij}(\sigma,\mathbf{x})\frac{\partial^{\,2}\pi}{\partial\,x^i\partial\,x^j} + b^i(\sigma,\mathbf{x})\frac{\partial\,\pi}{\partial\,x^i} = -\frac{\partial\,\pi}{\partial\,\sigma} \tag{3.9}$$

Em(ANTONELLI; STROBECK, 1977) pode-se encontrar a demonstração desse teorema.

Quando a^{ij} e b^i não dependem explicitamente do tempo, dizemos que a difusão markoviana é *homogênea no tempo*. Neste caso, com $\tau > \sigma$:
$$\pi(\tau,\mathbf{y}\,|\,\sigma,\mathbf{x}) = \pi(\tau-\sigma,\mathbf{y}\,|\,0,\mathbf{x}), \tag{3.10}$$

3.4.4 Construção do grafo de estados puros

Seja S um sistema contendo três estados puros, no qual denominaremos por S_0, S_1, S_2 onde o espaço desse sistema é representado por um triângulo equilátero com cada vértice representando um desses estados. O vértice S_0 é oposto a base do triângulo queé determinada pelos vértices S_1 e S_2, os vértices são nomeados a partir do vértice S_0 seguindo-se em sentido horário. A Figura 3 representa esse sistema inicial.

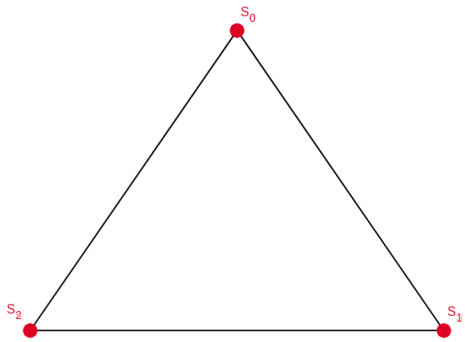

Figura 3 – Representação do sistema S e seus três estados puros

3.4.5 Construção do grafo de estados mistos

Na construção do grafo de estados mistos, denominado de $\mathcal{G}_{\mathcal{N}}$, é necessário a estruturação de linhas, as quais chamamos de *isoestados* que são relativas aos estados puros, essas linhas são sempre opostas ao estado puro referenciado e dividem as arestas adjacentes a este estado em N pedaços iguais. Nas Figuras 4, 5 e 6 estão representadas as isolinhas relativas aos estados puros S_0, S_1 e S_2 respectivamente, com o $N = 4$.

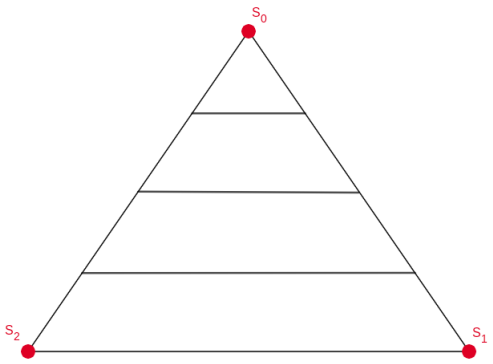

Figura 4 – Isolinhas relativas ao estado puro S_0

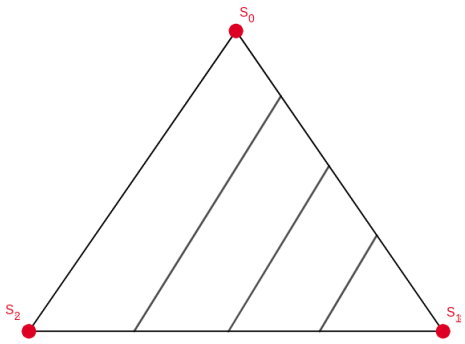

Figura 5 – Isolinhas relativas ao estado puro S_1

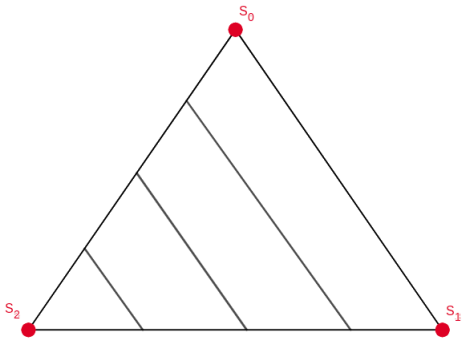

Figura 6 – Isolinhas relativas ao estado puro S_2

A quantidade de isolinhas, N, será definida pelo usuário e será a mesma para todos os estados puros. Quando traçadas as isolinhas dos três estados puros na mesma representação é gerada uma malha triangular sobre o sistema S, como mostra a Figura 7.

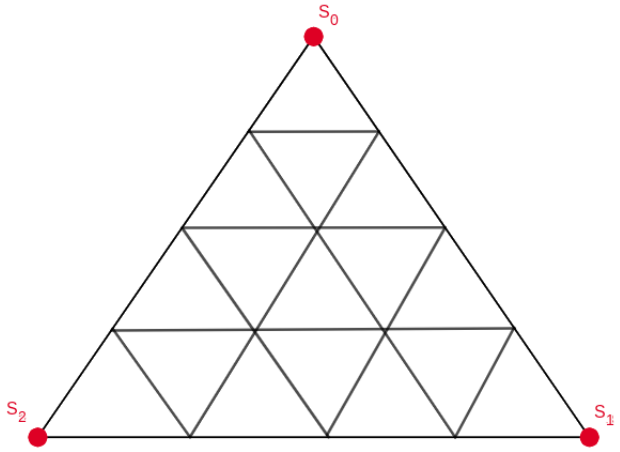

Figura 7 – Representação do sistema S e seus três estados puros

Dada essa malha triangular, tomamos o baricentro de cada um dos triângulos gerados onde esses baricentros serão os vértices do grafo G e cada vértice v_ℓ estará ligado ao vértice v_i se os triângulos possuem, ao menos, uma aresta em comum.

Lembrando que pode-se encontrar esses baricentros por alguns métodos:

- Pode-se determinar o baricentro pelo encontro dos segmentos de reta que unem cada vértice do triângulo ao ponto médio do lado oposto, ou seja, o encontro das

medianas.

- Pode-se determinar o baricentro pelo encontro dos segmentos de reta perpendiculares a um lado do triângulo ou ao seu prolongamento, traçado pelo vértice oposto, ou seja, o encontro das alturas.
- Pode-se determinar o baricentro pelo encontro das retas perpendiculares a um segmento de reta e que passa pelo ponto médio deste segmento, ou seja, o encontro das mediatrizes.

No caso de um triângulo equilátero, ambos os métodos geram o mesmo ponto. Na Figura 8 pode-se verificar a representação dos baricentros gerados.

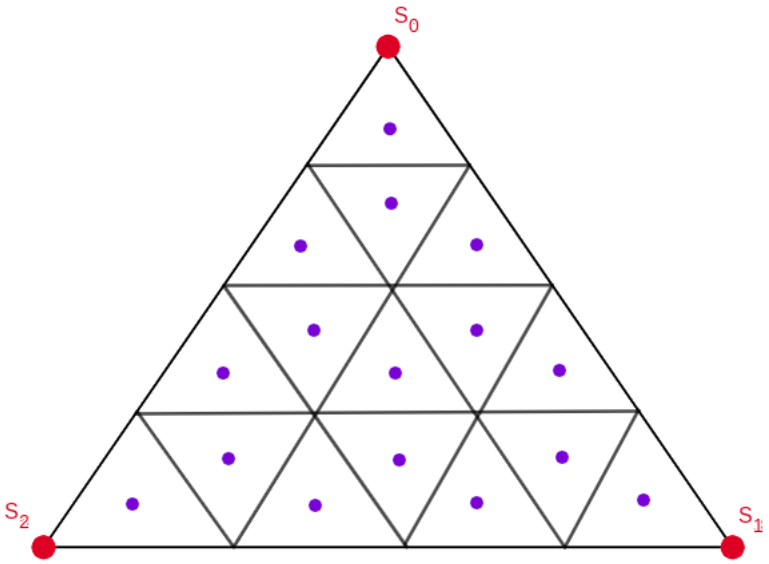

Figura 8 – Baricentros dos triângulos formados pelas *isolinhas*.

Dado o grafo gerado, pode-se identificar seus vértices, neste caso, eles serão denominados por v_ℓ onde $l = 1, 2, ..., N^2$ e $l \in \mathbb{N}$. O vértice v_1 será o primeiro vértice localizado logo abaixo do vértice S_0 e os vértices seguintes serão numerados da esquerda pra direta e em ordem crescente. Os vértices colineares são os pontos que pertencem a mesma reta (linha), neste caso, paralela a aresta S_1S_2), essas linhas denominamos como *níveis* e serão identificados por $L_1(S_0), L_2(S_0), L_3(S_0)$ e assim sucessivamente. Na Figura 9 podemos verificar os vértices e seus respectivos níveis.

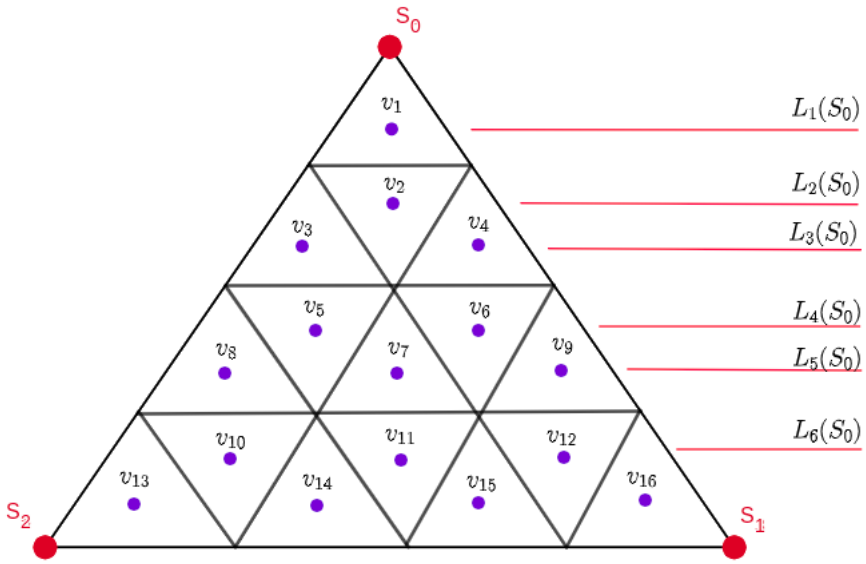

Figura 9 – Representação do sistema S com o grafo pronto sobre ele.

Alguns resultados para os grafos de estados são apresentados a seguir.

Proposição 3.4.2 (Quantidade de níveis)**.** *Dado um grafo G que representa o espaço de um sistema S contendo três estados puros, tem a quantidade de níveis dado por $(2N-1)$, onde o N é a quantidade de partes em que as arestas do triângulo serão divididas, posteriormente definiremos como* quantil.

Prova. Pela lei de formação dos grafos, vemos que a cada nova sub-divisão, ou acréscimo de faixa, acrescentamos dois novos níveis ao grafo, pois excetuando-se a primeira faixa (que contém apenas um nível), todas as faixas contém dois níveis, como temos tantas faixas quanto seja o n, temos que o número de níveis é dado por $N = (2n-1)$. □

Proposição 3.4.3 (Diâmetro do Grafo)**.** *Todo grafo G que representa o espaço de estados de um sistema S que contém três estados puros, tem o seu diâmetro dado por $N = (2n-2)$, onde n é a quantidade de partes em que as arestas do triângulo são divididas.*

Prova. Pela geometria euclidiana, temos que em um triângulo equilátero a maior distância entre dois pontos que pode existir é a distância entre dois de seus vértices. Assim dada a definição de $diam(G)$, temos que ele será dado pela distância entre os vértices do grafo, mais próximos dos vértices do triângulo, como o triângulo é equilátero podemos tomar a última faixa como parâmetro para definir o diâmetro do grafo. Como na última faixa

temos $2n-1$ vértices, em linha, e precisamos de 2 vértices para obter uma aresta, então teremos $2n-1-1=2n-2$ arestas que é o $diam(G)$. □

As proposições só são válidas a partir de $n=2$, pois para $n=1$, existe apenas um vértice cujo grau é $|\Gamma(0)|=1$.

Proposição 3.4.4 (Quantidade de vértices com grau 1)**.** *Dado um grafo G que representa o espaço de um sistema S contendo três estados puros, a quantidade de vértices com grau 1 é definido por $|\Gamma(v_1)|=3$, em que N é a quantidade de partes na qual as arestas do triângulo $\mathbb{T}$ serão divididas.*

Proposição 3.4.5 (Quantidade de vértices com grau 2)**.** *Dado um grafo G que representa o espaço de um sistema S contendo três estados puros, a quantidade de vértices com grau 2 é definido por $|\Gamma(v_2)|=3(N-2)$, onde o N é a quantidade de partes em que as arestas do triângulo serão divididas.*

Proposição 3.4.6 (Quantidade de vértices com grau 3)**.** *Dado um grafo G que representa o espaço de um sistema S contendo três estados puros, a quantidade de vértices com grau 2 é definido por $|\Gamma(v_3)|=N^2-3N+3$, onde o N é a quantidade de partes em que as arestas do triângulo serão divididas.*

3.4.6 Coordenadas dos vértices

Para cada subtriângulo $\mathbb{T}_i$, $i=1,\ldots,N^2$ gerado, existe um vértice no baricentro de cada um desses substriângulos. As coordenadas baricêntricas de um vértice de $\mathcal{G}_{\mathcal{N}}$ é dada pela igual ponderação das coordenadas dos vértices do subtriângulo onde esta localizado. Já as coordenadas de cada vértice de um subtriângulo podem ser definidas pelas linhas de isoestados que formam cada um destes subtriângulos, a partir disso, tem-se as coordenadas desses vértices relativas aos estados puros.

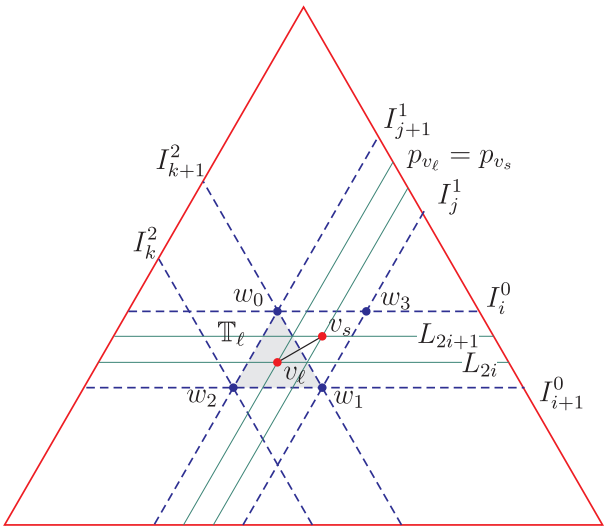

Figura 10 – Relação entre os vértices e as linhas de isoestados.

Seja I^r_j a j-ésima linha de isoestado relativa a S_r, $r = 0, 1, 2$. Assim se N é o valor definido para o inverso do quantil inicial, temos

$$I^0_i = \left\{ \left(\frac{N-i}{N}, x_1, x_2 \right) \,:\, x_1 + x_2 = 1 - (N-i)/N \right\}, \quad i = 1, 2, \ldots, N, \tag{3.11}$$

$$I^1_j = \left\{ \left(x_0, \frac{N-j}{N}, x_2 \right) \,:\, x_0 + x_2 = 1 - (N-j)/N \right\}, \quad j = 1, 2, \ldots, N, \tag{3.12}$$

e

$$I^2_k = \left\{ \left(x_0, x_1, \frac{N-k}{N} \right) \,:\, x_0 + x_1 = 1 - (N-k)/N \right\}, \quad k = 1, 2, \ldots, N. \tag{3.13}$$

Seja v_ℓ um vértice de $\mathcal{G}_\mathcal{N}$, no nível $L_{v_\ell} = 2i + 1$ (ímpar!), $i = 0, 1, \ldots, 2N$, na posição $p_{v_\ell} = m$. O subtriângulo $\mathbb{T}_\ell$ é delimitado por I^0_i e I^0_{i+1}, ou seja, pelas linhas de isoestados:

$$I^0_i = \left(\frac{N-i}{N}, x_1, x_2 \right), \qquad e \qquad I^0_{i+1} = \left(\frac{N-i-1}{N}, x_1, x_2 \right)$$

Dado um vértice v_ℓ, num nível $L_{v_\ell} = 2i$ $i = 1, \ldots, 2N$ com $i = \frac{L_{v_\ell}}{2}$, as mesmas conclusões devem ser feitas.

Dado o estado puro S_1, temos que para $\phi(v_\ell) = (2i + 1, p)$, ou $\phi(v_\ell) = (2i, p)$, existe a seguinte sequência de linhas e isoestados e posições nos níveis: $\ldots, I^1_{N-p+1}, \phi(v_\ell) =$

$(2i+1,p), \phi(v_s) = (2i,p), I^1_{N-p}, \ldots$, em que $i = 0, 1, \ldots, N$ e v_ℓ e v_s são vizinhos com $s = \phi^{-1}(2i, j)$. Logo o subtriângulo $\mathbb{T}_\ell$ é delimitado pelas isolinhas I^1_{N-p+1} e I^1_{N-p} e por

$$I^1_{N-p+1} = \left(x_0, \frac{N-(N-p+1)}{N}, x_2\right) = \left(x_0, \frac{p-1}{N}, x_2\right), \qquad \text{e}$$

$$I^1_{N-p} = \left(x_0, \frac{N-(N-p)}{N}, x_2\right) = \left(x_0, \frac{p}{N}, x_2\right)$$

De posse dessas definições, pode-se calcular as coordenadas dos vértices ω_1, ω_2 e ω_3 do subtriângulo $\mathbb{T}_\ell$. Como ω_1 pertence à linha de isoestado I^0_i, I^1_{j+1} e I^2_{k+1}, em que $i = \frac{L_{v_\ell}}{2}$ (ou $i = \frac{L_{v_\ell}-1}{2}$) e $j = N - p_{v_\ell}$, tal vértice tem coordenadas:

$$\omega_0 = \left(\frac{N-i}{N}, \frac{N-j-1}{N}, \frac{N-k-1}{N}\right),$$

No caso em que $L_{v_\ell} = 2i+1$, ou seja, $i = (L_{v_\ell}-1)/2$ e $j = N-p$, temos

$$\omega_0 = \left(1 - \frac{L_{v_\ell}-1}{2N}, \frac{p-1}{N}, S_2(w_0)\right),$$

em que

$$S_2(\omega_0) = 1 - \left(1 - \frac{L_{v_\ell}-1}{2N} + \frac{p-1}{N}\right) = \frac{L_{v_\ell} - 2p_{v_\ell} + 1}{2N},$$

Então,

$$\omega_0 = \left(1 - \frac{L_{v_\ell}-1}{2N}, \frac{p-1}{N}, \frac{L_{v_\ell} - 2p_{v_\ell}+1}{2N}\right) \tag{3.14}$$

$\ell = 0 \ldots, N-1$ e $j = 0, \ldots, N-1$ (onde v_ℓ está em um nível ímpar).

Dado que ω_1 pertence à linha de isoestado I^0_{i+1} e I^1_j e está sobre a mesma linha de isoestado I^2_k, logo ω_1 tem a terceira coordenada igual a de ω_0.

Então,

$$\omega_1 = \left(1 - \frac{L_{v_\ell}+1}{2N}, p_{v_\ell}, \frac{L_{v_\ell}-2p_{v_\ell}+1}{2N}\right) \tag{3.15}$$

Agora, ω_2 está sobre a mesma linha de isoestado I^1_j que ω_1 e mesma linha de isoestado I^0_i de ω_2, e sobre I^2_k. Portanto

$$\omega_2 = \left(1 - \frac{L_{v_\ell}+1}{2N}, \frac{p-1}{N}, \frac{L_{v_\ell}-2p_{v_\ell}}{2N}\right). \tag{3.16}$$

A última coordenada de ω_2 foi obtida fazendo $1 - S_0(\omega_2) - S_1(\omega_2)$.

O vértice v_ℓ é uma ponderação de ω_1, ω_2 e ω_3:

$$v_\ell = \frac{1}{3}\omega_0 + \frac{1}{3}\omega_1 + \frac{1}{3}\omega_2 = \left(1 - \frac{3L_{v_\ell}+1}{6N}, \frac{3p_{v_\ell}-2}{3N}, \frac{3L_{v_\ell}-6p_{v_\ell}+5}{6N}\right). \tag{3.17}$$

Para $L_{v_s} = 2i$, $i = 1, \ldots, N-1$, e $j = N - p_{v_s}$, teremos

$$\omega_0 = \left(1 - \frac{L_{v_s}}{2N}, \frac{p_{v_s}-1}{N}, \frac{L_{v_s}-2p_{v_s}+2}{2N}\right) \tag{3.18}$$

$$\omega_1 = \left(1 - \frac{L_{v_s}+2}{2N}, \frac{p_{v_s}}{N}, \frac{L_{v_s}-2p_{v_s}+2}{2N}\right) \tag{3.19}$$

$$\omega_2 = \left(1 - \frac{L_{v_s}}{2N}, \frac{p_{v_s}}{N}, \frac{L_{v_s}-2p_{v_s}}{2N}\right). \tag{3.20}$$

Das três equações acima (veja Algoritmo 2), temos

$$v_s = \frac{1}{3}w_0 + \frac{1}{3}w_1 + \frac{1}{3}w_2 = \left(1 - \frac{3L_{v_s}+2}{6N}, \frac{3p_{v_s}-1}{3N}, \frac{3L_{v_s}-6p_{v_s}+4}{6N}\right). \tag{3.21}$$

3.4.7 Fluxo discretizado sobre o Grafo $\mathcal{G}_{\mathcal{N}}$

Para obter-se a função de probabilidade condicional que determina a dinâmica do espaço de estados $\mathbb{T}$ utiliza-se a (3.2) que satisfaz (3.5) e suas hipóteses.

$$p_{ij}^{nk} = p(n+k, v_j \,|\, n, v_i), \quad n, k = 0, 1, 2, \ldots; \; i, j = 1, \ldots, N^2 \tag{3.22}$$

onde p_{ij}^{nk} representa a probabilidade de se estar no instante n em v_i e se atingir em k passos o vértice v_j, tal que seja válida.

$$p(m, v_j \,|\, n, v_i) = \sum_{v_k \in V(n)} \sum_{\ell=n}^{m} p(\ell, v_k \,|\, n, v_i) \cdot p(m, v_j \,|\, \ell, v_k). \tag{3.23}$$

Também sabe-se que:

$$b^r(m, v_i) = \sum_{v_j \in Nb(v_i)} [S^r(v_i) - S^r(v_j)] \cdot p_{ij}^{m-1,1}, \; r = 0, 1, 2, \tag{3.24}$$

$$2a^{rs}(m, v_i) = \sum_{v_j \in Nb(v_i)} [S^r(v_i) - S^r(v_j)][S^s(v_i) - S^s(v_j)] \cdot p_{ij}^{m-1,1}, \; r, s = 0, 1, 2. \tag{3.25}$$

O vetor $(b^r(m, v_l))_{r=0,1,2}$ relaciona-se com a média dos estados vizinhos de v_l dados pelas probabilidades de transição em um passo, no instante m.

Esta média é dado por (3.7). De forma análoga, a matriz $\left(a^{rs}(m, v_l)\right)_{rs}$ relaciona-se à covariância entre as coordenadas dos estados puros.

A equação de Kolmogorov para o caso dos grafos de estados mistos podem ser escrita como

$$a^{rs}(m-1, v_\ell) \frac{\Delta^2 p_{\ell i}^{m-1,1}}{\Delta_{\ell i} S^r \Delta_{\ell i} S^s} + b^r(m-1, v_\ell) \frac{\Delta p_{\ell i}^{m-1,1}}{\Delta_{\ell i} S^r} = -\Delta p_{\ell i}^{m-1,1}, \quad v_\ell \in V(N), \tag{3.26}$$

em que $\Delta_{\ell i}S^r = S^r(v_\ell) - S^r(v_i)$, $\Delta p_{\ell i}^{m-1,1} = p_{\ell i}^{m-1,1} - p_{i\ell}^{m,-1}$ e $\Delta p_{\ell i}^{m-1,1} = -\Delta p_{i\ell}^{m,-1}$.

Utilizando a expressão acima e a reescrevendo, tem-se:

$$\frac{\Delta^2 p_{\ell i}^{m-1,1}}{\dfrac{\Delta_{\ell i}S^r \Delta_{\ell i}S^s}{a^{rs}(m-1,v_\ell)}} + \frac{\Delta p_{\ell i}^{m-1,1}}{\dfrac{\Delta_{\ell i}S^r}{b^r(m,v_\ell)}} = -\Delta p_{\ell i}^{m-1,1}, \tag{3.27}$$

Logo, para todo vértice $v_\ell \in V(N)$, tem-se:

$$\sum_{r,s=0}^{2}\left[\sum_{v_i\in Nb(v_\ell)} \frac{(p_{\ell i}^{m-1,1} - p_{i\ell}^{m,-1})^2}{\dfrac{[S_r(v_\ell)-S_r(v_i)][S_s(v_\ell)-S_s(v_i)]}{\sum_{v_j\in Nb(v_\ell)}[S_r(v_\ell)-S_r(v_j)][S_s(v_\ell)-S_s(v_j)]}} \cdot p_{\ell i}^{m-1,1}\right]$$

$$+\sum_{r=0}^{2}\left[\sum_{v_i\in Nb(v_\ell)} \frac{(p_{\ell i}^{m-1,1} - p_{i\ell}^{m,-1})}{\dfrac{[S_r(v_\ell)-S_r(v_i)]}{\sum_{v_j\in Nb(v_\ell)}[S_r(v_\ell)-S_r(v_j)]}} \cdot p_{\ell i}^{m-1,1}\right] = \sum_{v_i\in Nb(v_\ell)} p_{i\ell}^{m,-1}$$

3.4.8 Rotulação de vértices e posicionamento em níveis

Dado um grafo $\mathcal{G}_N$, L_{v_ℓ} é o nível no qual o vértice se encontra, ou seja, qual a isolinha relativa ao estado S_0 o vértice se encontra no grafo. Já a posição p_{v_ℓ} do vértice v_ℓ no nível L_{v_ℓ} é contada da esquerda para direita na ordem dos vértices de $\mathcal{G}_N$.

Pela Proposição 3.4.2 sabe-se que o grafo $\mathcal{G}_N$ possui $2N-1$ níveis. Pode-se verificar também a quantidade de posições por níveis:

- Dado um nível k par do grafo $\mathcal{G}_N$, temos que a quantidade de posições é dada por $k/2$;
- Caso o nível k seja ímpar, a quantidade de posições por nível é dada por $(k+1)/2$.

Dessa forma, os níveis dados por $2m-1$ e $2m$ têm m vértices.

Seja m o índice do nível L_m, $m = 1, 2, \ldots, 2N-1$, onde N é o inverso do quantil escolhido inicialmente. Se m é par, os vizinhos (i, j, k) do vértice v_ℓ que está na posição i do nível m, que é representado por (m, i), é dado por:

$$Nb(m,i) = \Big\{(m-1,i),(m+1,i),(m+1,i+1)\Big\}, \quad i = 1,\ldots,\frac{m}{2}. \tag{3.28}$$

Se m é ímpar, a vizinhança do vértice que está na posição i do nível m é:

$$Nb(m,i) = \begin{cases} \{(m-1,1),(m+1,1)\}, & \text{se} \quad i = 1, \\ \{(m-1,i-1),(m-1,i),(m+1,i)\}, & \text{se} \quad i = 2,\ldots,\frac{m+1}{2}-1, \\ \left\{\left(m-1,\frac{m+1}{2}\right),\left(m+1,\frac{m+1}{2}\right)\right\}, & \text{se} \quad i = \frac{m+1}{2}. \end{cases} \tag{3.29}$$

No caso do último nível, o $L_{v_\ell} = 2N-1$ a vizinhança de v_ℓ é dada por:

$$Nb(2N-1,i) = \begin{cases} \{(2(N-1),j)\}, & \text{se} \quad i = 1, \\ \{(2(N-1),j-1),(2(N-1),j)\}, & \text{se} \quad i = 2,\ldots,N, \\ \{(2(N-1),N)\}, & \text{se} \quad i = N. \end{cases} \tag{3.30}$$

3.4.8.1 Relação do vértice v_ℓ, sua posição $p_{v\ell}$ e nível L_k

É factível que se faça uma relação entre o vértice v_ℓ, sua posição $p_{v\ell}$ e nível L_k, ou seja, representar a sua imagem por $\phi(k, p_{v_\ell})$.

Para k par, tem-se:

$$\ell = p_{v_\ell} + \sum_{n=1}^{\frac{k}{2}-1} 2n + \frac{k}{2} \quad \therefore \quad 4\ell = 4p_{v_\ell} + k^2 \tag{3.31}$$

Para k ímpar e $\ell \geq 3$:

$$\ell = p_{v_\ell} + \sum_{n=1}^{\frac{k-1}{2}} 2n \quad \therefore \quad 4\ell + 1 = 4p_{v_\ell} + k^2. \tag{3.32}$$

Dado o nível L_k, as posições dos vértices respeitam as seguintes desigualdades:

$$1 \leq p \leq \left\lceil \frac{k}{2} \right\rceil,$$

em que $\lceil x \rceil$ é a função teto de x, ou seja, é o menor inteiro maior ou igual a x.

Logo,

$$4 \leq 4p_{v_\ell} \leq 4 \left\lceil \frac{k}{2} \right\rceil. \tag{3.33}$$

Fazendo a substituição de 3.33 em 3.31, obtemos:

$$4 + k^2 \leq 4\ell \leq 4 \left\lceil \frac{k}{2} \right\rceil + k^2$$

agora, de posse da primeira desigualdade e fazendo as devidas simplificações, tem-se:

$$4 + k^2 \leq 4\ell \therefore k^2 \leq 4(\ell - 1) \therefore k \leq \sqrt{4\ell - 4}. \tag{3.34}$$

Dado um k inteiro e par e utilizando a segunda desigualdade de 3.34, tem-se:

$$4\ell \leq 4 \left\lceil \frac{k}{2} \right\rceil + k^2 = 2k + k^2 \;\; \therefore \;\; k^2 + 2k - 4\ell \geq 0 \;\; \therefore \;\; k \geq \sqrt{4\ell + 1} - 1. \tag{3.35}$$

Fazendo a substituição da primeira desigualdade de 3.33 em 3.32, obtêm-se:

$$4 + k^2 \leq 4\ell + 1 \leq 4 \left\lceil \frac{k}{2} \right\rceil + k^2, \tag{3.36}$$

e, usando a primeira desigualdade acima, temos;

$$4 + k^2 \leq 4\ell + 1 \quad \therefore \quad k^2 \leq 4\ell - 3 \quad \therefore \quad k \leq \sqrt{4\ell - 3}. \tag{3.37}$$

Agora, usando a segunda desigualdade, obtemos:

$$4\ell \leq 4\left\lceil \frac{k}{2} \right\rceil + k^2 = 4\left(\frac{k+1}{2}\right) + k^2 \;\; \therefore \;\; k \geq 2\sqrt{\ell} - 1 \tag{3.38}$$

Assim, para k qualquer (par ou ímpar) usando 3.34 e 3.37, temos:

$$k \leq \max\left\{\lfloor\sqrt{4\ell - 4}\rfloor, \lfloor\sqrt{4\ell - 3}\rfloor\right\}, \tag{3.39}$$

em que $\lfloor x \rfloor$ é a função teto de x,ou seja, é o maior inteiro, menor ou igual a x, e usando 3.35 e 3.38, temos:

$$k \geq \min\left\{\lceil\sqrt{4\ell + 1} - 1\rceil, \lceil 2\sqrt{\ell} - 1\rceil\right\} \tag{3.40}$$

Proposição 3.4.7. *Dado um $\ell \in \mathbb{Z}^+$, tem-se:*

$$\max\left\{\lfloor\sqrt{4\ell - 4}\rfloor, \lfloor\sqrt{4\ell - 3}\rfloor\right\} = \min\left\{\lceil\sqrt{4\ell + 1} - 1\rceil, \lceil 2\sqrt{\ell} - 1\rceil\right\}. \tag{3.41}$$

De fato, pela Proposição 3.4.7 e as equações 3.39 e 3.40 obtem-se que k é o nível do vértice v_ℓ. Para obter a posição p_{v_ℓ}, utilizamos a Equação 3.31 e 3.32 e os valores de ℓ e k.

3.5 Algoritmo

Corman, Leiserson, e Rivest (1999) afirmam que, informalmente, um *algoritmo* é qualquer procedimento computacional que toma algum valor ou conjunto de valores, como entrada e produz algum valor ou conjunto de valores, como saída. Um algoritmo é, portanto, uma sequência de passos computacionais que transformam a entrada em saída.

Também pode-se visualizar um algoritmo como uma ferramenta para resolver problemas computacionais bem especificado. O enunciado do problema especifica em termos gerais o relacionamento de entrada/saída desejada e o algoritmo descreve um procedimento computacional específico para alcançar essa relação entrada/saída.

Um algoritmo é os passos necessários para realizar uma tarefa. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser

humano. A maneira mais simples de se pensar um algoritmo é por uma lista de procedimentos bem definida, na qual as instruções são executadas passo a passo a partir do começo da lista, uma ideia que pode ser facilmente visualizada através de um fluxograma. Cada campo da ciência possui seus próprios problemas e respectivos algoritmos adequados para resolvê-los. Exemplos clássicos são algoritmos de busca, de ordenação, de análise numérica, de teoria de grafos, de manipulação de cadeias de texto, de geometria computacional, de análise combinatória, de aprendizagem de máquina, de criptografia, de compressão de dados e de interpretação de texto.

Para ser possível construir um algoritmo que resolva a dinâmica de um grafo, serão necessário algumas etapas que abrangem:

- Para descrever um sistema físico que possui três estados puros, denotados por S_0, S_1 e S_2, ou combinações convexas de estados. Também será necessário, definir o espaço de estados, fazendo a descrição do conjunto de equação que o descreveria;
- Para gerar um sistema de coordenadas que localiza o estado atual do sistema relativamente aos três estados puros, será especificado uma *n-upla* de escalares para cada ponto do espaço do espaço $(n-1)$-*dimensional*. O sistema de coordenadas utilizados será o sistema de coordenadas baricêntricas de um triângulo equilátero cujo os vértices são os estados puros, denotados por S_0, S_1 e S_2.

3.5.1 Algoritmo de definição do grafo $\mathcal{G}_{\mathcal{N}}$ de estados mistos

Dado um espaço de estados $\mathbb{T}$, o usuário escolhe, de forma arbitrária, o número de quantis, denominado de N onde $N \in \mathbb{N}$. Deste modo é definido o *quantil* de discretização, $p = N^{-1}$ e posteriormente é definido o grafo de estados mistos $\mathcal{G}_{\mathcal{N}}$ seguindo o algoritmo abaixo.

▷ Leia o número de quantis N.

▷ Divida cada lado do triângulo $\mathbb{T}$ em N partes iguais (cada ponto representa um p-quantil das linhas $(0, x_1, x_2)$, $(x_0, 0, x_2)$ e $(x_0, x_1, 0)$).
▷ Use cada p-quantil para determinar linhas de *isoestados* relativamente a cada estado puro. Estas linhas são paralelas aos lados dos triângulos e chamaremos de isolinhas.
▷ Após isso serão formados N^2 triângulos $\mathbb{T}_i$. Em cada baricentro de tais $\mathbb{T}_i$'s considere um vértice e una dois vértices por uma aresta se, e somente se, dois dos dois triângulos correspondentes tiverem um lado em comum.

O grafo proveniente do resultado do algoritmo 3.5.1, para $N = 4$, é representado pelas Figuras 12 e 11.

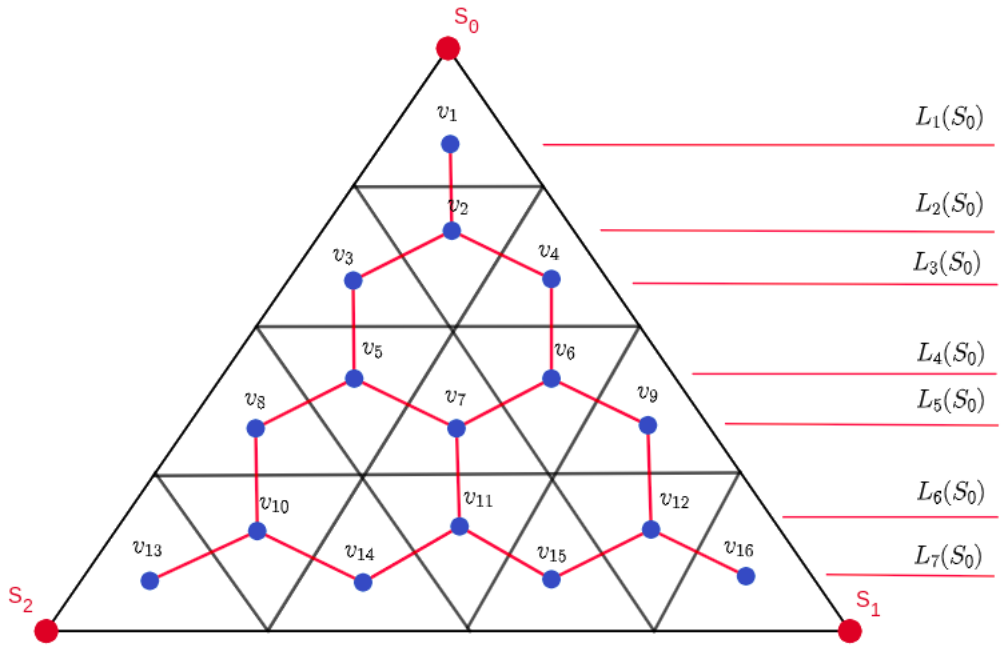

Figura 11 – Representação do grafo de estados mistos $\mathcal{G}_4$ do sistema S

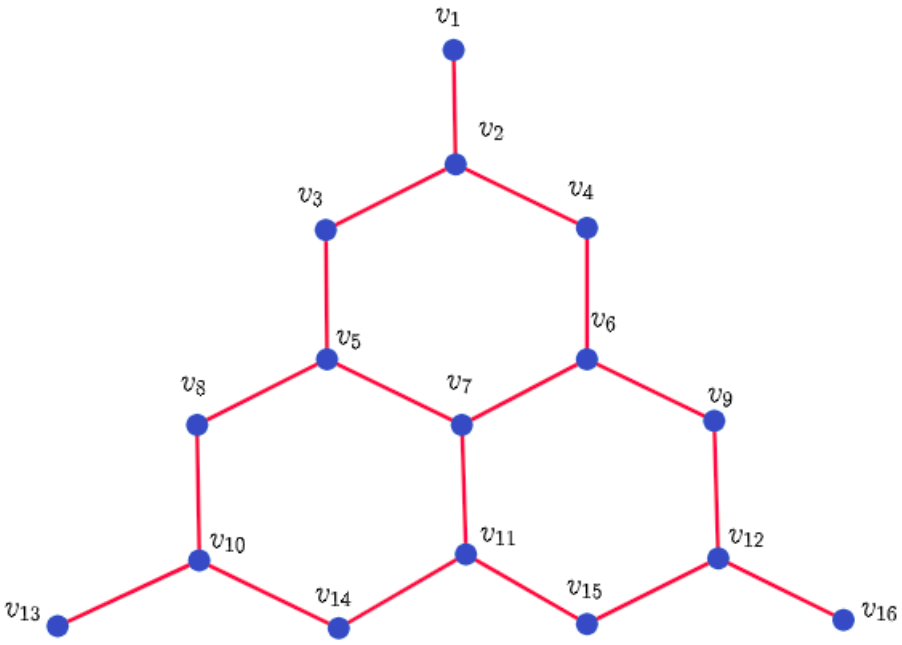

Figura 12 – Grafo de estados mistos $\mathcal{G}_4$

A discretização é feita considerando apenas as coordenadas baricêntricas dos vértices dos triângulos, ou seja, $(v_1, v_2, v_3, ...v_{16})$ as quais representa o estado atual do sistema, o v_ℓ com $\ell = (1, 2, ...N^2) \in \mathbb{N}$. A dinâmica é feita considerando a transição do estado atual, v_1, no instante σ para o instante $\sigma + 1$. O estado pode permanecer em v_1 ou mudar para v_2, para essa transição denominamos de transição suave nesse espaço discreto e as coordenadas do vetor de estados v_ℓ é denotada por $v_\ell = (S_0(v_\ell), S_1(v_\ell), S_2(v_\ell))$. A quantidade de níveis é dada pela Proposição 3.4.2, para $N = 4$, temos sete níveis representados respectivamente por $L_1(S_0), L_2(S_0), ..., L_7(S_0)$. Utilizando a Definição 3.4.7 para $N = 4$ pode-se gerar a matriz de adjacência desse grafo.

3.5.2 Algoritmo da construção da matriz de adjacência do grafo $\mathcal{G}_\mathcal{N}$

O Algoritmo 1 a seguir é utilizado para construir a matriz de adjacência do grafo $\mathcal{G}_\mathcal{N}$, atribuindo a vizninhança de cada vértice, representado pela figura 12.

Algoritmo 1: Matriz de adjacência do grafo $\mathcal{G}_{\mathcal{N}}$

Entrada: N = valor do quantil dado.
Saída: A matriz de adjacência do grafo
início
  $L_{v_1} = 1$; $p_{v_1} = 1$; $Matriz(1,2) = 1$;
  repita
    $L_{v_\ell} = \max\{\lfloor\sqrt{4\ell-4}\rfloor, \lfloor\sqrt{4\ell-3}\rfloor\}$ `% Calcula o nível do vértice` v_ℓ;
    se L_{v_ℓ} *é par* **então**
      $p_{v_\ell} = [4\ell - (L_{v_\ell})^2]/4$;
      $i = [4p_{v_\ell} + (L_{v_\ell} - 1)^2 - 1]/4$;
      $j = [4p_{v_\ell} + (L_{v_\ell} + 1)^2 - 1]/4$;
      $k = [4p_{v_\ell} + (L_{v_\ell} + 1)^2 + 3]/4$;
      } `% Vizinhos de` v_ℓ;
      $Matriz(\ell, i) = 1$; $Matriz(\ell, j) = 1$; $Matriz(\ell, k) = 1$;
    senão
      $p_{v_\ell} = [4\ell - (L_{v_\ell})^2 + 1]/4$;
      se $L_{v_\ell} \neq 2N - 1$ **então**
        se $p_{v_\ell} = 1$ **então**
          $i = [4 + (L_{v_\ell} - 1)^2]/4$; $j = [4 + (L_{v_\ell} + 1)^2]/4$;
          $Matriz(\ell, i) = 1$; $Matriz(\ell, j) = 1$;
        senão
          se $p_{v_\ell} = (L_{v_\ell} + 1)/2$ **então**
            $i = [2(L_{v_\ell} - 1) + (L_{v_\ell} - 1)^2]/4$; $j = [2(L_{v_\ell} + 1) + (L_{v_\ell} + 1)^2]/4$;
            $Matriz(\ell, i) = 1$; $Matriz(\ell, j) = 1$;
          senão
            $i = [4(p_{v_\ell} - 1) + (L_{v_\ell} - 1)^2]/4$; $j = [4p_{v_\ell} + (L_{v_\ell} - 1)^2]/4$;
            $k = [4p_{v_\ell} + (L_{v_\ell} + 1)^2]/4$;
            $Matriz(\ell, i) = 1$; $Matriz(\ell, j) = 1$; $Matriz(\ell, k) = 1$;
          fim
        fim
      senão
        se $p_{v_\ell} = 1$ **então**
          $i = [4 + (L_{v_\ell} - 1)^2]/4$; $Matriz(\ell, i) = 1$;
        senão
          se $p_{v_\ell} = (L_{v_\ell} + 1)/2$ **então**
            $i = [2(L_{v_\ell} - 1) + (L_{v_\ell} - 1)^2]/4$; $Matriz(\ell, i) = 1$;
          senão
            $i = [4(p_{v_\ell} - 1) + (L_{v_\ell} - 1)^2]/4$; $j = [4p_{v_\ell} + (L_{v_\ell} - 1)^2]/4$;
            $Matriz(\ell, i) = 1$; $Matriz(\ell, j) = 1$;
          fim
        fim
      fim
    fim
  até $\ell = N^2$;
fim

3.5.3 Algoritmo das coordenadas dos vértices do grafo $\mathcal{G}_{\mathcal{N}}$

O Algoritmo 2 abaixo será utilizado para atribuir as coordendas de cada vértice relativas aos estados puros (S_0, S_1, S_2). Relembrando qeu cada vértice foi colocado dentro do baricentro de um dos subtriângulos $\mathbb{T}_i, i = 1, \cdots N^2$ representados pela figura 11. O vértice v_ℓ pode ser representado pelas coordenadas $v_\ell = (S_0, S_1, S_2)$.

Algoritmo 2: Coordenadas dos vértices do grafo G_N relativas aos estados puros S_0, S_1 e S_2.

Entrada: N = valor do quantil ;
L_{v_ℓ}, p_{v_ℓ}, nível e posição do vértice $v_\ell \in V(\mathcal{G}_{\mathcal{N}})$
Saída: As coordenadas baricêntricas de cada vértice v_ℓ de $\mathcal{G}_{\mathcal{N}}$
início
  repita
    se L_{v_ℓ} *é ímpar* **então**
      $v_\ell = \left(1 - \frac{3L_{v_\ell}+1}{6N}, \frac{3p_{v_\ell}-2}{3N}, \frac{3L_{v_\ell}-6p_{v_\ell}+5}{6N}\right)$;
    senão
      $v_\ell = \left(1 - \frac{3L_{v_\ell}+2}{6N}, \frac{3p_{v_\ell}-1}{3N}, \frac{3L_{v_\ell}-6p_{v_\ell}+4}{6N}\right)$;
    fim
  até $\ell = N^2$;
fim

3.5.4 Algoritmo da dinâmica

O Algoritmo 3 gera a caminhada no grafo durante o tempo estabelecido. Gera uma dinâmica entre os vértices utilizando os resultados dos Algoritmos 1 e 2 e com probabilidades geradas de forma aleatória ou arbitrária.

Algoritmo 3: Algoritmo de simulação de trajetórias no grado de estados.

início
  repita
    $\ell = 1$ **repita**
      $u = random(0, 1)$;
      se $0 < u <= p_{\ell\ell}$ **então**
        $trajetoria[dia, m] = dados[\ell, 1]$
      senão
        se $p_{\ell\ell} < u \leq (p_{\mathrm{H}} + p$ **então**
          $trajetoria[dia, m] = dados[\ell, 1]$
      fim
      senão
        se $(p_{\ell\ell} + p_{\ell i} < u \leq p_{\ell\ell} + p_{\ell i} + p_{\ell,j})$ **então**
          $trajetoria[dia, m] = dados[\ell, 2]$
      fim
      senão
        se $(p_{\ell\ell} + p_{\ell i} + p\ell, j < u \leq 1$ **então**
          $trajetoria[dia, m] = dados[\ell, 3]$;
      fim
      $\ell = trajetoria[dia, m]$
    até $dia = tempo$;
  até $m = 1000$;
fim

Para o Algoritmo 3, as variáveis são definidas da seguinte forma:

1. **u** é uma variável aleatória uniforme entre 0 e 1, sorteada a cada iteração do *loop*.
2. $\mathbf{p_{\ell\ell}}$ é a probabilidade do vértice v_ℓ permanecer nele mesmo.
3. $\mathbf{p_{\ell i}}$ é a probabilidade do vértice v_ℓ mudar para o vértice v_i.
4. $\mathbf{p_{\ell j}}$ é a probabilidade do vértice v_ℓ mudar para o vértice v_j.
5. $\mathbf{p_{\ell k}}$ é a probabilidade do vértice v_ℓ mudar para o vértice v_k.
6. **dados**$[\ell, \mathbf{coluna}]$ é um vetor bidimensional que recebe como entrada o vértice atual e uma coluna fixa. Esse valor de coluna é referente ao local no qual é armazenado um dos vizinhos do vértice ℓ. Por exemplo, $dados[\ell, 1]$ é o vizinho i do vértice ℓ. $dados[\ell, 2]$ é o vizinho j do vértice ℓ. $dados[\ell, 3]$ é o vizinho k do vértice ℓ.

7. **trajetoria**[**dia**, **m**] é um vetor bidimensional que recebe como entrada o valor de m (o número da simulação) e *dia* o dia ao qual está ocorrendo a dinâmica. Lembrando que esses valores são dados pois dois loops distintos e aninhados.

A dinâmica se dá da seguinte forma:

1. Inicia-se um loop com a quantidade de simulações a ser gerada;
2. Inicia-se um loop com a quantidade de dias ao qual se quer verificar a trajetória;
3. Para cada iteração do loop, é gerado de forma aleatória um número u entre 0 e 1;
4. É analisada as condições estabelecidas: Se u estiver no intervalo $]0, pll]$ o vértice permanece no mesmo local, ou seja, vertice atual = v_ℓ.
5. Se u estiver no intervalo $]p_{\ell\ell}, (p_{\ell\ell} + p_{\ell i})]$ o vértice atual muda para o seu vizinho **i**, ou seja, vertice atual é igual a v_i.
6. Se u estiver no intervalo $](p_{\ell\ell} + p_{\ell i}), (p_{\ell\ell} + p_{\ell i} + p_{\ell j})]$ o vértice atual muda para o seu vizinho v_j, ou seja, vertice atual é v_j.
7. Se u estiver no intervalo $](p_{\ell\ell} + p_{\ell i} + p_{\ell j}), 1]$ o vértice atual muda para o seu vizinho v_k, ou seja, vertice atual é o v_k.
8. Após o fim de cada iteração do loop, o valor de **l** é atualizado pelo valor dado por $trajetoria[dia, m]$.

Cada trajetória é calculada m vezes, no Algoritmo 3 o $m = 1000$.

3.6 Probabilidade e Probabilidade Condicional

Morgado *et al.* (2006) define *Teoria das Probabilidade* como o ramo da Matemática que cria, desenvolve e em geral pesquisa *modelos* que podem ser utilizados para estudar experimentos ou fenômenos aleatórios. O modelo matemático utilizado para estudar um fenômeno aleatório particular varia em sua complexidade matemática, dependendo do fenômeno estudado.

Magalhães (2007), define probabilidade, mais formalmente, como sendo uma função $P(\cdot)$ que atribui valores numéricos aos eventos do espaço amostral (Ω), se satisfaz as seguintes condições:

(i) $0 \leq P(A) \leq 1, \forall A \subset \Omega$;

(ii) $P(\Omega) = 1$;

(iii) $P\left(\bigcup_{j=1}^{n} A_j\right) = \sum_{j=1}^{n}(A_j)$, com os $A_{j's}$ disjuntos.

Em muitas situações práticas, o fenômeno aleatório com o qual é trabalhado pode ser separado em etapas. A informação do que ocorreu em uma determinada etapa pode influenciar nas probabilidades de ocorrências das etapas sucessivas. Nestes casos, houve ganho de informação e deve-se "recalcular" as informações de interesse, essas probabilidades "recalculadas" são chamadas de probabilidades condicionais, e são definidas da seguinte maneira:

Dados dois eventos A e B, a probabilidade condicional de A dado que ocorreu B é representada por $P(A|B)$ e dada por:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0.$$

Caso $P(B) = 0$, $P(A|B)$ pode ser definido arbitrariamente; neste contexto será utilizado $P(A|B) = P(A)$

3.7 Percolação

A teoria da percolação foi proposta pelos matemáticos Broadbent e Hammersly em 1957 (Navin, 2012) para estudar a propagação de fluidos em meios desordenados. É bastante relevante em física para explicar eventos como recuperação de petróleo a partir de meios porosos, fogos florestais, modelos de epidemia, redes e terremotos. O modelo de percolação, permite explicar este tipo de eventos e introduzir alguns conceitos fundamentais dos sistema complexos (grafos), a Teoria da Percolação é um ramo da teoria das probabilidades que trata das propriedades dos meios aleatórios agregadas a conceitos geométricos e probabilísticos. Existem dois tipos de percolação, Crotti (2009): percolação por ligações e percolação por pontos. Considerando um *cluster* qualquer, a percolação por ligações é o estudo de fenômenos de percolação nas arestas, o modelo da infiltração de óleo em solos permeáveis, enquanto que a percolação por pontos estuda os fenômenos de percolação nos vértices (nós) sendo o caso da propagação de fogos florestais. Estes dois tipos de percolação, embora diferentes, apresentam propriedades semelhantes que são estudadas sobre um espaço L^d, sendo d a dimensão.

Um processo de percolação consiste na propagação do estado de uma célula ativa às células vizinhas que, depois de ativadas, continuam o processo de propagação. O processo termina quando não há mais células do agregado que possam ser ativadas. A

duração do processo de percolação depende de dois fatores, sendo eles o tamanho do agregado e a forma como estão ligados. É evidente que quanto maior for o agregado maior será a duração do processo de percolação e um agregado muito ligado percola muito mais rapidamente do que um agregado com poucas ligações.

3.8 Código em Python

3.8.1 Código em Python da matriz de adjacência do grafo $\mathcal{G}_{\mathcal{N}}$

O código referente ao Anexo A é a implementação do Algoritmo 1, o qual é responsável pela construção da matriz de adjacência do grafo $\mathcal{G}_{\mathcal{N}}$, cálculo da vizinhança do vértice v_ℓ, seus respectivos níveis $L_{v\ell}$ e posições $p_{v\ell}$.

Este código foi totalmente construído na Linguagem Python utilizando as bibliotecas Numpy, Pandas, CSV e math

- A variável **N** corresponde ao quantil N definido pela Proposição 3.4.2;
- A variável **nivel_Lvl** corresponde ao nível L_{v_ℓ} definido pela Proposição 3.4.2;
- A variável **pos_vl** corresponde a posição, $p_{v\ell}$, do vértice v_ℓ definida pela Equação 3.33;
- A variável **matriz** corresponde a matriz de adjacência do grafo $\mathcal{G}_{\mathcal{N}}$, $A = (a_{ij})_{nxn}$, e é definida pela Equação 3.4.7;
- A variável **l** corresponde ao vértice v_l;
- A variável **i**, caso exista, corresponde ao vértice vizinho v_i do vértice v_l e é dado pela Definição 3.4.6;
- A variável **j**, caso exista, corresponde ao vértice vizinho v_j do vértice v_l e é dado pela Definição 3.4.6;
- A variável **k**, caso exista, corresponde ao vértice vizinho v_k do vértice v_l e é dado pela Definição 3.4.6;
- A variável **vertices_vizinhos** corresponde a um arquivo .csv onde é armazenado os vértices v_ℓ e sua vizinhança para cada valor N de quantil definido.

3.8.2 Código em Python das coordenadas dos vértices do grafo $\mathcal{G}_{\mathcal{N}}$

O código referente ao Anexo B é a implementação do Algoritmo 2, o qual é responsável pela obtenção das coordenadas baricêntricas do grafo $\mathcal{G}_{\mathcal{N}}$, representado pela Figura 8.

Este código foi totalmente construído na Linguagem *Python*. Considere como válidas as definições dada por 3.8.1.

Tome $v_\ell = (S_0, S_1, S_2)$ representado pela 8,

- A variável **si0** corresponde a primeira coordenada baricêntrica do vértice v_ℓ, o S_0.
- A variável **si1** corresponde a segunda coordenada baricêntrica do vértice v_ℓ, o S_1.
- A variável **si2** corresponde a terceira coordenada baricêntrica do vértice v_ℓ, o S_2.

3.8.3 Código em Python da dinâmica da simulação do grafo $\mathcal{G}_\mathcal{N}$

O código referente ao Anexo C descreve a dinâmica estocástica dos estados do sistema e é a implementação do Algoritmo 1, Algoritmo 2 em conjunto com a dinâmica estocástica dos estados e construção do desenho do grafo $\mathcal{G}_\mathcal{N}$.

Este foi totalmente construído na Linguagem Python utilizando as bibliotecas Numpy, Pandas, CSV e math. Considere como válidas as definições dada por 3.8.1. No código **??** definimos 3 novas variáveis, **o,r,s** responsavéis pela dinâmica com o banco de dados real.

Tome $v_\ell = (S_0, S_1, S_2)$ representado pela 8,

- A variável **o** corresponde a função teto da primeira coordenada baricêntrica do vértice v_ℓ, o S_0, multiplicada pelo quantil **N**, ou seja, $o = \lceil N_0 S_0 \rceil$.
- A variável **r** corresponde a função teto da segunda coordenada baricêntrica do vértice v_ℓ, o S_1, multiplicada pelo quantil **N**, ou seja, $r = \lceil N_0 S_1 \rceil$.
- A variável **s** corresponde a função teto da terceira coordenada baricêntrica do vértice v_ℓ, o S_2, multiplicada pelo quantil **N**, ou seja, $s = \lceil N_0 S_2 \rceil$.

3.9 Banco de dados

3.9.1 Sistemas Gerenciadores de Banco de dados

Os Gerenciadores de Banco de dados, popularmente conhecidos como *SGBD* são softwares especializados para gerenciamento de bases de dados.

As principais funções de um SGBD são:

- Alteração da estrutura de campos;
- Cópia e eliminação de ficheiros;
- Inserção, removeção e criação de relações entre tabelas;
- Importação e exportação de dados entre bases de dados;
- Criação de chaves externas e primárias;
- Consultas nas tabelas;
- Criação de usuários com permissões de acesso.

O SGBD utilizado será o PostgreSQL em conjunto com a plataforma pgAdmin4.

3.9.2 Linguagem de Consulta Estruturada

A linguagem de consulta estruturada, popularmente conhecida como SQL (Structured Query Language), é a linguagem é uma das linguagens mais importantes para manipulação de registros em banco de dados relacionais.

A linguagem SQL possui uma organização estrutural baseadas em subdivisões dos seus comandos, cada qual divididos em 5 subconjuntos diferentes.

DML - Data Manipulation Language: Também conhecida como linguagem de manipulação de dados, esse subcobjunto do SQL define os comandos usados para manipular os dados armazenados em um banco. Esse é um dos conjuntos mais utilizados, pois ele fornece operadores que nos permitem inserir, excluir e alterar os registros de uma tabela, por exemplo. Os comandos mais importantes desse subconjunto são: INSERT, DELETE e UPDATE.

DQL - Data Query Language: Também conhecida como linguagem de consulta de dados, esse subconjunto do SQL define o SELECT, comando essencial para consulta de dados armazenados no banco.

DDL - Data Definition Language: Também conhecida como linguagem de definição de dados, esse subconjunto do SQL define os comandos usados para gerenciar as estruturas do banco de dados. É o responsável por criar, atualizar e remover objetos da base, como tabelas e índices. Os comandos definidos pelo DDL são: CREATE, DROP e ALTER.

DCL - Data Control Language: Também conhecida como linguagem de controle de dados, esse subconjunto do SQL define os comandos para controle do acesso aos dados da base de dados. É o responsável por estabeler restrições e permissões para o acesso ao banco. Os comandos definidos pelo DCL são: GRANT e REVOKE.

DTL - Data Transaction Language ou TCL (Transaction Control Language): Também conhecida como linguagem de transação de dados, esse subconjunto do SQL define os comandos que utilizamos quando é necessário gerenciar transações feitas no banco. É o responsãvel por iniciar, confirmar e desfazer alterações. s comandos definidos pelo DTL|TCL são: COMMIT, BEGIN e ROLLBACK.

Para esse caso, apenas serão utilizado os subconjuntos DQL e DDL.

3.9.3 PostgreSQL e pgAdmin4

PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional, open source, baseado no POSTGRES e desenvolvido na Universidade da Califórnia no Departamento de Ciências da Computação em Berkeley. No PostgreSQL tudo que é criado é tratada como um objeto, por exemplo banco de dados, tabelas, triggers, views, etc. Já o pgAdmin4 é um software gráfico para administração do SGBD PostgreSQL.

3.9.4 Banco de dados

O banco de dados utilizado para análise foi o criado pelo Wesley Cota e se encontra disponível em Banco de dados COVID BR. O arquivo foi disponibilizado na extensão .csv e importado para o pgAdmin4 pelo script abaixo.

3.9.4.1 DDL: Importando os dados para o pgAdmin4

Através do comando COPY é possível importar dados de um arquivo .csv para uma tabela, criada pelo comando CREATE TABLE. Verifique o Anexo **??**.

Dada as devidas modificações, o mesmo processo se repete para inserção dos dados relacionados ao Algoritmo C. As tabelas provenientes desse Algoritmo foram denominadas de *verticesn10_x* sendo *10_x* o valor do quantil escolhido pelo usuário.

A saída do script do Anexo D deverá ser parecida com a Figura 13:

Data Output | Messages | Notifications

	id [PK] integer	epi_week numeric (10)	date date	country character varying (50)	state character varying (50)	city character varying (50)	newdeaths numeric (100)	deaths numeric (100)
1	1	9	2020-02-25	Brazil	SP	TOTAL	0	0
2	2	9	2020-02-25	Brazil	TOTAL	TOTAL	0	0
3	3	9	2020-02-26	Brazil	SP	TOTAL	0	0
4	4	9	2020-02-26	Brazil	TOTAL	TOTAL	0	0
5	5	9	2020-02-27	Brazil	SP	TOTAL	0	0
6	6	9	2020-02-27	Brazil	TOTAL	TOTAL	0	0
7	7	9	2020-02-28	Brazil	SP	TOTAL	0	0

Total rows: 1000 of 29106 | Query complete 00:00:00.129 | Ln 1, Col 24

Figura 13 – Saída do script D

3.9.4.2 DQL: Consultas no banco

A query abaixo é responsável por extrair as porcentagens de suscetíveis, infectados e recuperados da população total do Brasial, baseado nos dados disponíveis em Banco de dados COVID BR sendo a saída da query uma tabela com as respectivas porcentagens dia a dia.

A saída do script Anexo E deverá ser parecida com a Figura 14:

Data Output | Messages | Notifications

	data_registro date	n_suscetiveis numeric	n_infectados numeric	n_recuperados numeric
65	2020-04-29	0.9995997373358348968 1	0.0013012382739212007 5	0.0000760131332082551594 75
66	2020-04-30	0.9995633161350844277 7	0.0014386444652908067 5	0.0000855581613508442776 74
67	2020-05-01	0.9995351594746716697 9	0.0015056285178236397 7	0.0000917307692307692307 69
68	2020-05-02	0.9995103752345215759 8	0.0015399249530956848 0	0.0000969981238273921200 75
69	2020-05-03	0.9994882739212007504 7	0.0015883536585365853 7	0.0001014024390243902439 02
70	2020-05-04	0.9994540994371482176 4	0.0016124906191369606 0	0.0001059990619136960600 38
71	2020-05-05	0.9994165853658536585 4	0.0016522326454033771 1	0.0001159521575984990619 14
72	2020-05-06	0.9993642213883677298 3	0.0017747091932457786 1	0.0001368433395872420262 66

Total rows: 1000 of 1056 | Query complete 00:00:00.168

Figura 14 – Saída do Script C

4
Resultados

4.1 Aplicação do modelo com probabilidades arbitrárias

Nesta Seção será abordada a implementação do modelo de discretização para cenários fictícios com probabilidades arbitrárias geradas para cada vértice do grafo. De ínicio, será utilizado o modelo com probabilidades arbitrárias, escolhidas de forma lógica em aderência à uma sistema ternário que considera um espaço de estados com 3 estados puros S_0, S_1, S_2 respectivamente: suscetíveis, infectados e removidos.

Para a primeira análise e como forma de exemplificar de facilitar o entendimento do leitor, considera-se um sistema S, com 3 estados puros S_0, S_1, S_2 e com valor de $N = 4$. O grafo referente a esse sistema pode ser visualizado na Figura 12 e sua construção se dá pelos Algoritmos 1, 2 acima descritos.

De posse do grafo e de todas as informações que o carregam, foram feitas simulações de possíveis trajetórias (caminhada do vértice). Note que para o caso específico tratado, a dinâmica da epidemia de uma doença, o COVID-19, o vértice do primeiro dia necessariamente será o vértice v_1, ou seja, $v_\ell = v_1$

Para esse caso, foi gerada 1000, $m = 1000$, simulações de possíveis dinâmicas para o Grafo com $N = 4$.

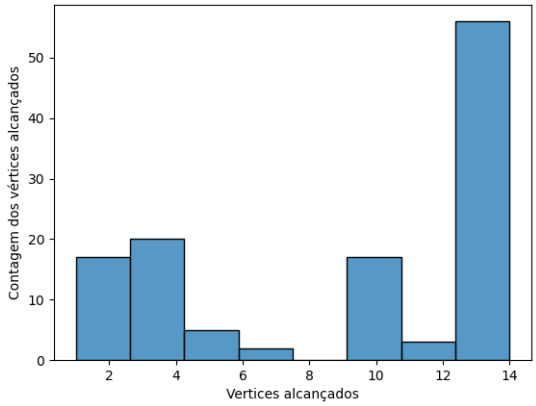

Figura 15 – Histograma contendo a simulação da dinâmica para 730 dias com N =4 e m=2.

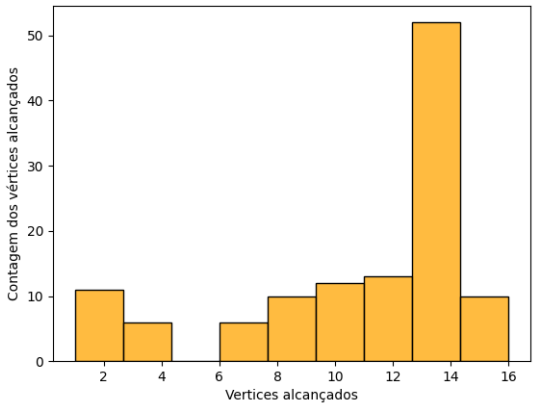

Figura 16 – Histograma contendo a simulação da dinâmica para 730 dias com N =4 e m =180.

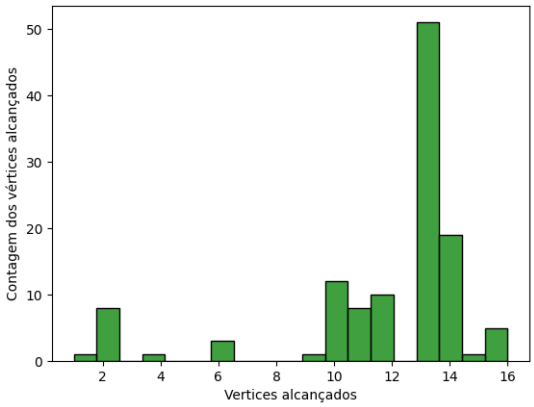

Figura 17 – Histograma contendo a simulação da dinâmica para 730 dias com N =4 e m =295.

Na Figura 15 foi plotado um histograma da simulação da trajetória com $m = 2$ onde o eixo y se refere a quantidade de vértices alcançados durante a trajetória e o eixo x os respectivos vértices. Analogamente, a Figura 16 e a Figura 17 são os histogramas para $m = 180$ e $m = 295$, respectivamente.

Nota-se até de forma visual, que para o caso com 16 vértices, o último vértice v_16 é atingido majoritariamente em ambas as simulações.

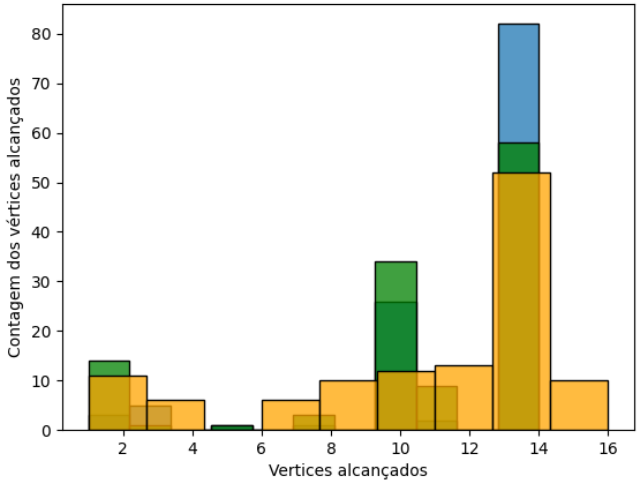

Figura 18 – Histogramas sobrepostos referente as simulações das trajetórias de 730 dias para um grafo com $N = 4$

A Figura 18 demonstra de forma clara a variabilidade de trajetórias para cada instância da simulação.

Ainda para $N = 4$, utiliza-se o gráfico box plot para demonstrar graficamente a variabilidade dos dados relativos a dinâmica. Para gerar esse gráfico, é feita uma relação entre os vértices alcançados na trajetória da dinâmica para os 730 dias e suas respectivas coordenadas baricêntricas. Vale salientar que nesse caso, as coordenadas baricêntricas do grafo representam respectivamente a porcentagem pela população total de suscetíveis, infectados e rcuperados. Pela Figura 19 se verifica que a dinâmica percorre todos os vértice do grafo em apenas um mês.

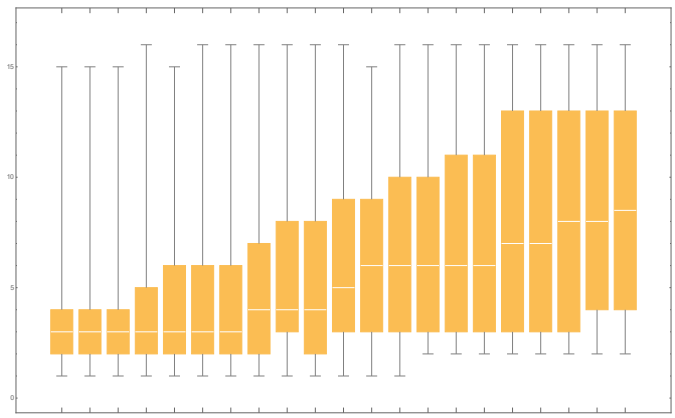

Figura 19 – Gŕafico boxplot para $N = 4$ dos dias 10 ao 30.

No Gráfico da Figura 20 é possível visualizar como se comporta as curvas relativas as porcentagens de suscetíveis, infectados e recuperados durante as trajetórias simuladas. O que, de fato, se assemelha bastante ao esperado numa epidemia. Os números de suscetíveis diminui durante o decorrer do tempo, o de infectados aumenta e o de recuperados varia de forma comitente aos outros dois.

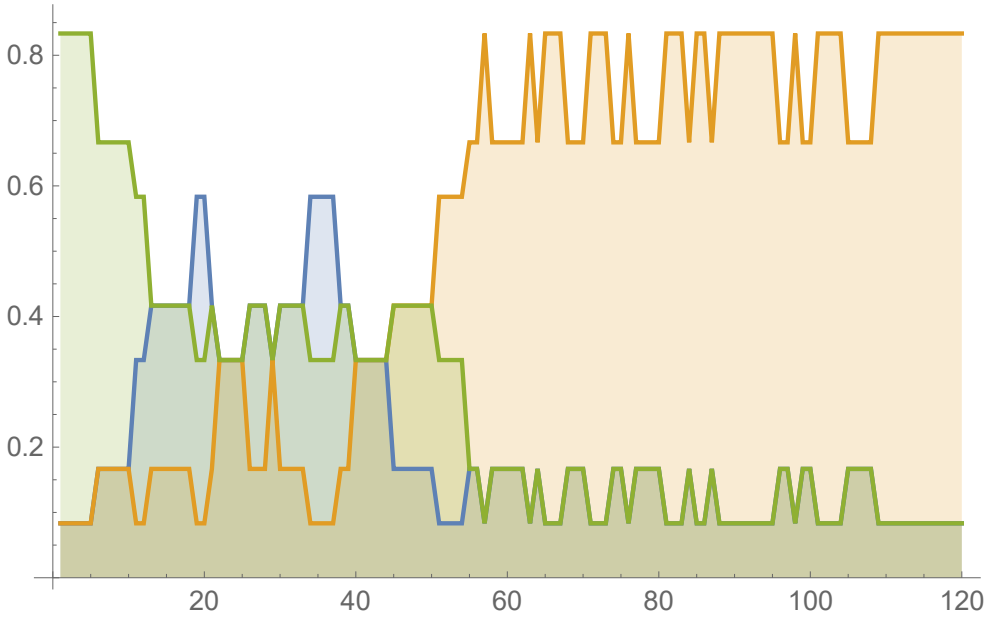

Figura 20 – Curvas das dinâmicas dos suscetíveis, infectados e recuperados para um grafo com $N = 4$. Verde: suscetíveis; Azul: infectados; Laranja: recuperados.

Na Figura 20 a curva verde representa a porcentagem dos suscetíveis, a azul a porcentagem dos recuperados e a laranja a porcentagem dos infectados.

Um outro exemplo de aderência do modelo aos dados reais, utiliza-se a simulação para um grafo com $N = 100$, 360 dias e $m = 1000$ simulações. As probabilidades foram construídas de forma que os triângulos em que as arestas coincidem de forma invertida, vide Figura 21, a probabilidade do número de suscetíveis aumentar é zero, ou seja, $pji = 0$, pois não se considera um aumento do número de suscetíveis neste caso.

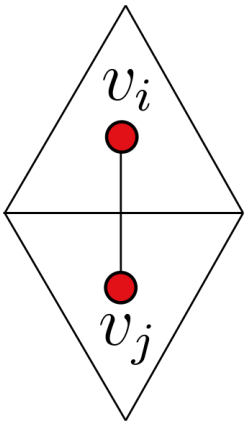

Figura 21 – Subtriângulos em posições invertidas do grafo. Nestes casos, a probabilidade de se passar do vértice v_j para v_i é zero.

Na sequência de gráficos box plot apresentada na Figura 22 visualiza-se graficamente que a dinâmica se dá de forma deveras semlhante ao real para todas as simulações geradas.

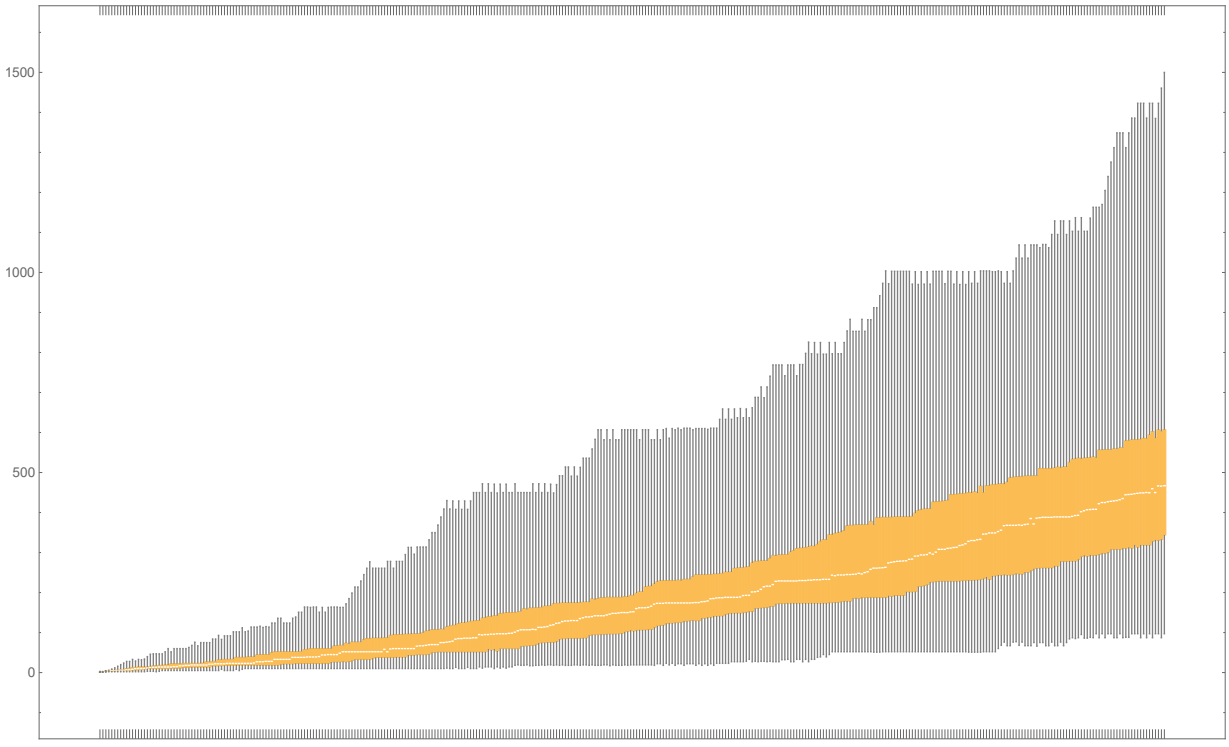

Figura 22 – Gŕafico boxplot para $N = 100$ referente ao 1° ano (360 dias). No eixo horizontal tem-se dos dias da simulação e na vertical o índice dos vértices atingidos no conjunto das simulações para o dia correspondente.

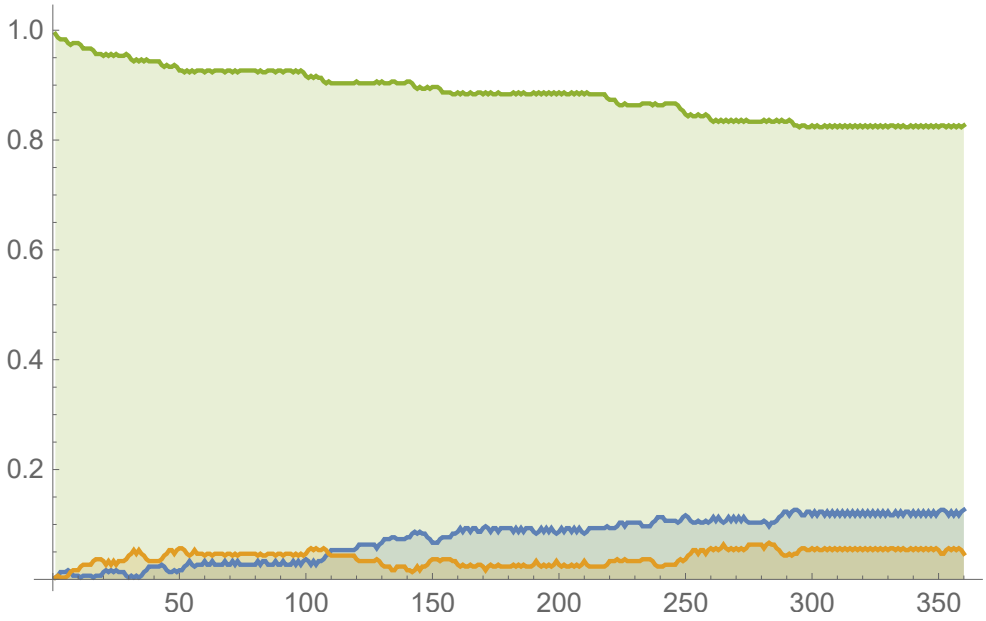

Figura 23 – Curvas das dinâmicas dos suscetíveis, infectados e recuperados para um grafo com $N = 100$. Curva verde: suscetíveis. Azul: infectados. Laranja: recuperados.

Para essa simulação é observado uma aderência maior ao modelo, utilizando as simulações geradas verifica-se que para o dia 360, a curva de suscetíveis, infectados e recuperados é de $78,3\%$, 4.3%, $17,4\%$, respectivamente.

Pela tabela resultante da query referente ao Anexo E para o dia 360, tem-se que:

```
1 pySELECT
2 py  *
3 pyFROM "coord_covid"
4 pyWHERE EXTRACT(year from data_registro) = 2021 AND EXTRACT(month
      from data_registro) = 03
5 pyORDER BY data_registro ASC
```

Listing 4.1 – Query para o dia 360

Note que a consulta acima usa como condição que retorne apenas os resultados para o ano de 2021 e mês de março, basicamente desde os 360 dias do ínicio da extração de dados que existe no banco. A query se dá da seguinte forma: dada essas condições, o PgAdmin procura todas os dados cujas relações são satisfeitas e retorna a data (ano/mês/dia) e as respectivas coordenadas de suscetiveis, infectados e recuperados do banco real. A saída se dá parecida com a Figura 24.

data_registro date	n_suscetiveis numeric	n_infectados numeric	n_recuperados numeric
2021-03-01	0.9493564909006066 2622	0.0720585720951936 5376	0.0440392487167522 1652
2021-03-02	0.9490699533364442 3705	0.0723370788614092 3938	0.0443596033597760 1493
2021-03-03	0.9487211945870275 3150	0.0726772235184321 0453	0.0445558329444703 6864
2021-03-04	0.9483581427904806 3462	0.0730318478768082 1279	0.0448249323378441 4372
2021-03-05	0.9480118712085860 9426	0.0733701259916005 5996	0.0450595193653756 4162
2021-03-06	0.9476927391507232 8511	0.0736823098460102 6598	0.0451984321045263 6491
2021-03-07	0.9473064489034064 3957	0.0751296780214652 3565	0.0454478021465235 6510
2021-03-08	0.9471305926271581 8945	0.0753003173121791 8805	0.0458579328044797 0135

Figura 24 – Resultado da consulta que retorna os valores das porcentagens relativas aos suscetiveis, infectados e recuperados do Banco de dados real, para o mês de março de 20221.

É possível verificar, que para simulação da dinâmica da doença pelo modelo proposto, mesmo com probabilidades aleatórias a semlhança de resultados é nítida. Demosntrandom, mais uma vez, a aderência do modelo aos dados reais.

4.2 Aplicação do modelo com probabilidades aleatórias

Análogo a seção anterior, nessa será abordada a implementação do modelo de discretização para cenários fictícios com probabilidades aleatórias geradas para cada vértice do grafo. Será utilizado o modelo com probabilidades aletórias, cada vértice possui um conjunto de probabilidades aleatórias distintas de passagem ou não para outro vértice vizinho.

De posse do grafo e de todas as informações que o carregam, foram feitas simulações de possíveis trajetórias (caminhada do vértice). Note que para o caso específico tratado, a dinâmica da epidemia de uma doença, o COVID-19, o vértice do primeiro dia necessariamente será o vértice v_1, ou seja, $v_\ell = v_1$

Para esse caso, foi gerada 1000, $m = 1000$, simulações de possíveis dinâmicas para o Grafo com $N = 10^3$.

A análise dos histogramas se dá de forma semelhante ao exemplo anterior, do grafo $N = 4$,

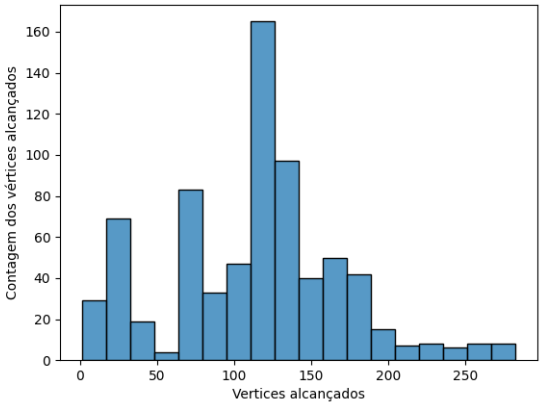

Figura 25 – Histograma contendo a simulação da dinâmica para 120 dias com $N = 1000$ e $m = 1$.

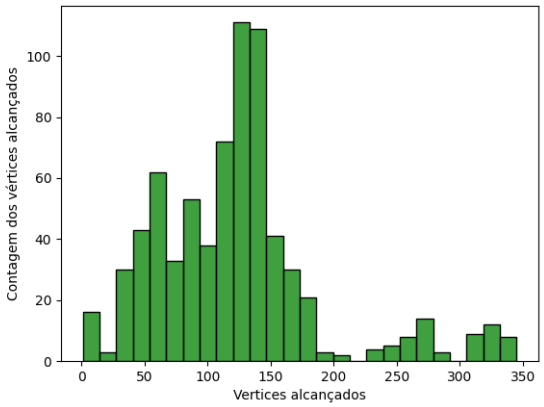

Figura 26 – Histograma contendo a simulação da dinâmica para 120 dias com $N = 1000$ e $m = 2$.

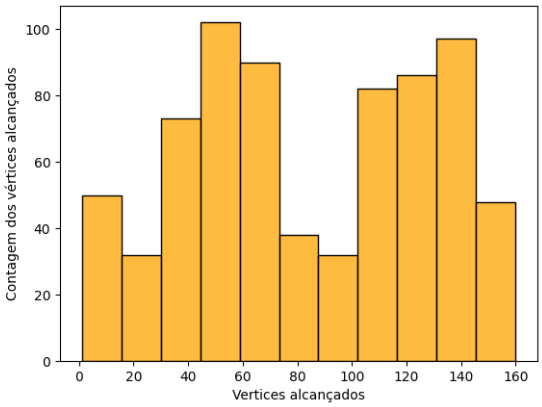

Figura 27 – Histograma contendo a simulação da dinâmica para 120 dias com $N = 1000$ e $m = 100$.

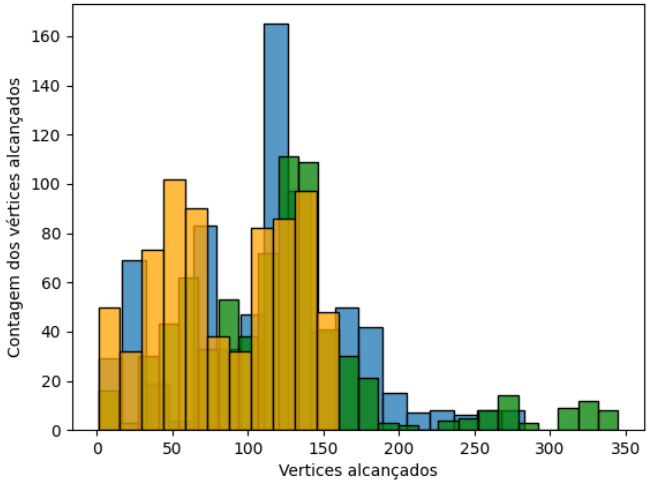

Figura 28 – Histogramas sobrepostos referente as simulações das trajetórias de 120 dias para um grafo com $N = 1000$

Nesse caso é visualizado uma variabilidade mais discrepante para cada simulação, isso se dá, também, pelas probabilidades adotadas de forma aleatória. Na Figura 28 é fácil notar a discrepância de variações para cada instância da trajetória.

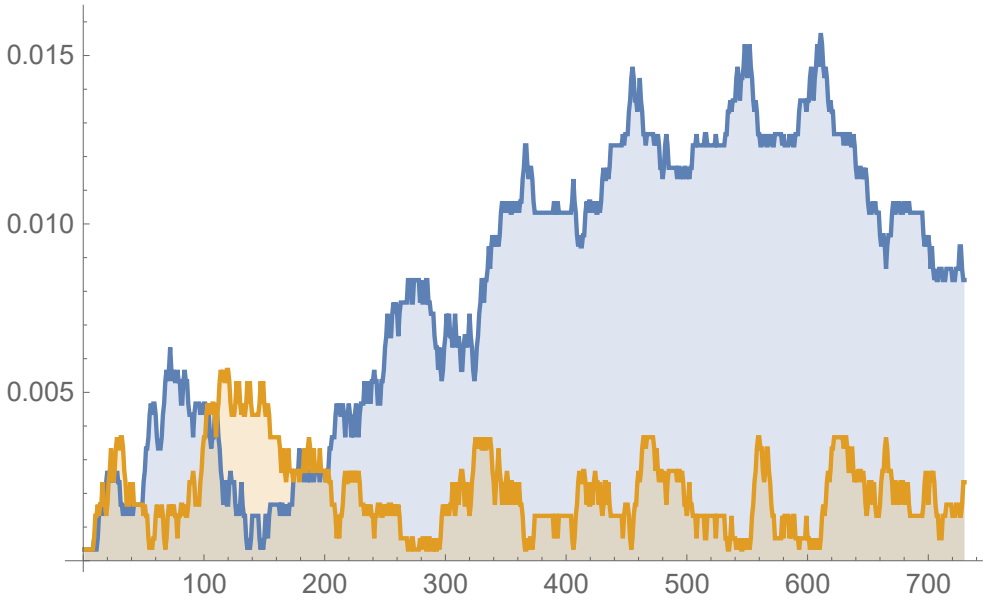

Figura 29 – Curvas das dinâmicas dos infectados (azul) e recuperados (laranja) para um grafo com $N = 1000$.

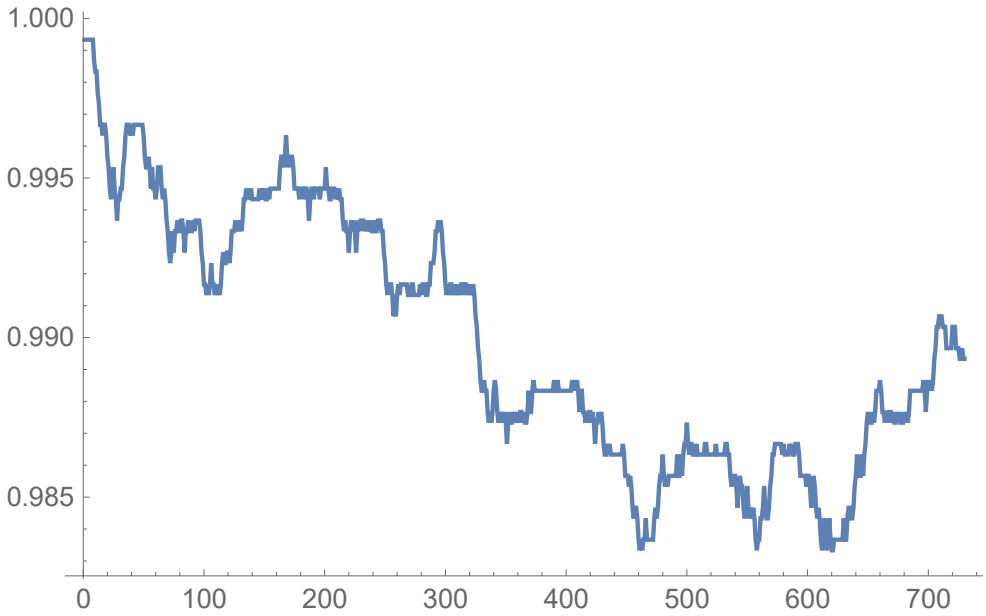

Figura 30 – Curvas das dinâmicas dos suscetíveis para um grafo com $N = 1000$.

Nas Figuras 29 e 30 é visto que o comportamento das curvas relativas as porcentagens de suscetíveis, infectados e recuperados durante as trajetórias simuladas, também se assemelha com o esperado para a dinâmica do COVID-19. Por questão de escala, a curva dos suscetíveis foi plotada em um gráfico separado.

Para analisar e a variabilidade dos dados relativos a dinâmica para um grafo com $N = 1000$ foi plotado um gráfico boxplot com a relação entre entre os vértices alcançados na trajetória da dinâmica para o 1° e 2° e suas respectivas poercentagens pela população total de suscetíveis, infectados e recuperados.

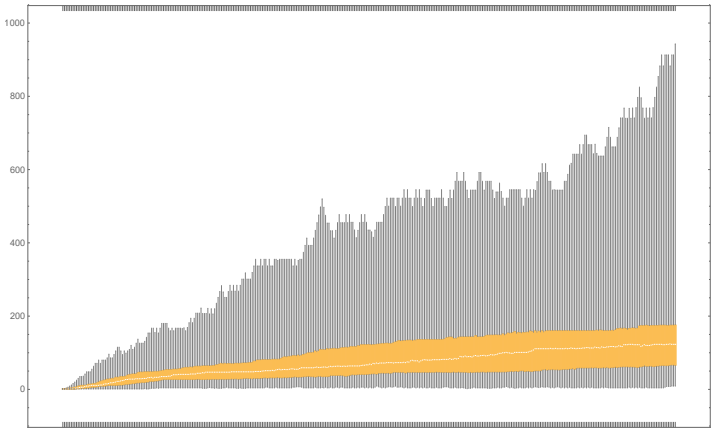

Figura 31 – Gŕafico boxplot para $N = 1000$ referente ao primeiro ano.

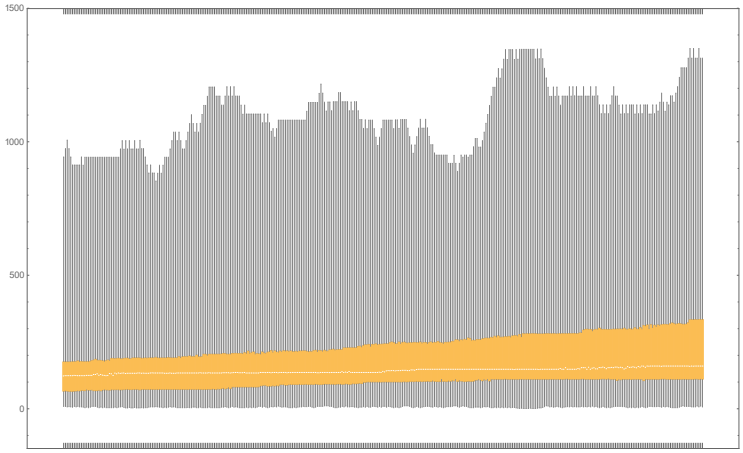

Figura 32 – Gŕafico boxplot para $N = 1000$ referente ao segundo ano.

Nas Figuras 31 e 32 é visto que durante as diversas trajetórias simuladas o modelo se ajusta, comprovando assim a aderência do modelo à cenário reais.

5
Conclusão

5.1 Implementação do algoritmo para linguagem computacional e Dificuldades

A discretização de um sistema ternário em forma de um grafo, sugerido pelo modelo proposto foi testado e observado para diversos cenários. A primeira variação foi se a construção do Grafo $\mathcal{G}_{\mathcal{N}}$ estava correta. Para isso foi verificado essa construção para parâmetros distintos.

A primeira variação foi dada pela quantidade de isolinhas, N, no qual é divido o grafo, essas isolinhas são as que ditam a quantidade de substriângulos que aparecem no grafo. Para esse primeiro parâmetro, inicialmente, foram utilizados valores variando de 5 em 5, $N = 5$, $N = 10, \cdots$ para que seja feito a verificação da veracidade dos resultados e contrução manual do Grafo (comparando a saída do programa com o que seria esperado do algoritmo). Ainda na construção do Grafo, foi verificado se os vértices e seus respectivos vizinhos eram, de fato, os esperados do algoritmo. Após a escolha do banco de dados utilizado nesse trabalho, a próxima etapa seria de fato discretizar esse sistema, sendo feita uma transformação do banco de dados real para o grafo G_N de forma a se tornar possível a implementação do modelo. O banco utilizado para esse trabalho foi o do Wesley Cota, disponibilizado no GitHub disponibilizado no link: <https://github.com/wcota/covid19br>. Utilizando a linguagem SQL em conjunto com o software open source PgAdmin, foi filtrado apenas os dados de interesse (total de casos, recuperados e data de registro). Como a informação de suscetíveis não era disposta de forma clara no bando de dados, foi feita uma rotina para estimar a população total de cada ano. Basicamente, dado os sensos do IBGE da quantidade de indivíduos no Brasil, para o ano de 2020, 2021, 2022 e 2023 e retirado os infectados e recuperados para cada dia, foi atribuído os valores para a população diárias de suscetíveis. Essa rotina é melhor exemplificada no Anexo E. De posse do grafo e do banco de dados tratado, foi de fato, feita a

discretização do sistema real. Em suma, foi gerado uma lista contendo todos os dados disponíveis do Grafo, incluindo as coordenadas dos subtriângulos gerados pelas isolinhas do grafo nos quais os vértices são os baricentros desses triângulos. Dadas essas coordenadas e de posse do banco de dados real e tratado, o que é feito é uma comparação de coordenadas. Caso as porcentagens da população do banco real esteja dentro do subtriângulo do vértice, é retornado o baricentro daquele subtriângulo que é efetivamente o vértice v_ℓ do Grafo e assim se torna completa a discretização do banco de dados real. Logo têm-se o todas as informações do sistema discretizado, vertices, coordenadas baricêntricas, em qual região do grafo aquele dia se encontra e etc. O resultado final disso é a dinâmica real da doença dado o modelo proposto.

5.2 Aderência e potencialidade do modelo

Fundamentado pelas simulaçoes computacionais geradas, o modelo sugerido para discretização de um sistema ternário é considerado adequeado para modelagem de uma dinâmica. O modelo demonstra exatidão no que propõe pois mesmo dado probabilidades aleatórias e sem relação direta com um sistema específico, a dinâmica gerada é concordante com o esperado no mundo real, essa discretização de um sistema para um Grafo traz uma facilidade no tratamento de problemas antes tão complexos. Também é possível verificar que a contrução do grafo se dá de forma eficiente pelos algoritmos listados, onde toda e qualquer informação do grafo pode ser gerada computacionalmente de forma simples. Ademais, não é aqui que sucumbe as possibilidades de aplicação desse modelo, caso as probabilidades de passagem de um vértice para outro fosse gerado pelas equações de Kolmogorov, seria visto uma dinâmica ainda mais semelhante com ao mundo real. Nesse trabalho restirngimos as aplicações do modelo para o caso de uma dinâmica de uma doença, mas vale salientar que a mesma modelagem pode ser aplicada para diversos sistemas ternários, dadas suas ressalvas.

Referências Bibliográficas

ANTONELLI, P.; STROBECK, C. The geometry and random drift i. stochastic distance and diffusion. **Advances in Applied Probability**, v. 9, n. 2, p. 238–249, 1977.

BOLOBÁS, B. **Modern Graph Theory**. New York: Springer, 1998. v. 184. (Graduate Texts in Mathematics, v. 184). ISBN 0387984887.

BONDY, A.; MURTY, U. **Graph Theory**. [S.l.]: Springer Science & Business Media, 2008. v. 244. (Graduate Texts in Mathematics, v. 244).

COTA, W. **Bando de dados COVID - BR**. Disponível em: <https://github.com/wcota/covid19br>.

KRANTZ, S. G. **Real Analysis and Foundations**. 2^{nd} ed.. ed. Boca Raton: Chapman & Hall, 2005.

LOVÁSZ, L. Random walks on graphs: A survey. **Combinatorics, Paul erdos is eighty**, v. 2, n. 1, p. 1–46, 1993.

MOLCHANOV, S. A. Diffusion process and riemannian geometry. **Russian Math. Surveys**, v. 30, n. 1, p. 1–63, 1975.

NUMPY, P. **Documentation Numpy**. Disponível em: <https://numpy.org/doc/>.

PANDAS, P. **Documentation Pandas**. Disponível em: <https://pandas.pydata.org/docs/user_guide/index.html>.

PYTHON. **Documentation Python**. Disponível em: <{https://docs.python.org/pt-br/3}.>

ALIZON, S; HURFORD, A; MIDEO, N; BAALENS, VAN M., Journal compilation: European society for evolutionary biology, **Virulence evolution and the trade-off hypothesis: history, current state of affairs and the future**. Suiça. Laudanne, 2008.

ALVES, Rui; DELGADO, Catarina, Universidade de Economia.**Processos estocásticos.**, Portugal, 1997.

BRAUMANN, Carlos A. **Introdução às Equações Diferenciais Estocásticas e Aplicações**. XIII Congresso Anual: Sociedade portuguesa de estatística. Ericeira, 2005.

CÂMARA, Fernando Portela. **Dinâmica das epidemias virais e efeito caótico**, Brasil. Rio de Janeiro, [data desconhecida]. IMPPG, UFRJ.

DAY, Troy; PROULX, Stephen R. **A General Theory for the Evolutionary Dynamics of Virulence**. Chicago. V. 163, n. 4, April 2004.

ETHERIDGE, Alison. **A Course in Financial Calculus**.[S.l.]: Cambridge University Press, 2002.

Anexos

ANEXO A – Código em Python da matriz de adjacência do grafo $\mathcal{G}_N$

```python
from math import floor, sqrt
import numpy as np
import csv
import random
from numpy.random import default_rng
''' [Algoritmo 01] Matriz de adjacência de Gn '''
# Inicia as variavéis
# N: número de níveis a serem considerados na partição do espaço de estados
N = 4
nivel_Lv1 = pos_v1 = 1
l = nivel_Lvl = pos_vl = 0
# Define a matriz indentidade N x N
matriz = np.eye(N**2, dtype=int)
matriz[1-1][2-1] = 1  # o -1 é para evitar o inicio de contagem do python com 0
#Inicia arquivo pra guardar os vertices e seus respectivos vizinhos : i,j,k,l
vertices_vizinhos = open('vizinhos_N' + str(N) + '.csv','w', newline='', encoding='utf-8')
w = csv.writer(vertices_vizinhos)
for l in range(N**2):
    l = l + 1
    # Calcula o nível do vértice vl
    nivel_Lvl = max(floor(sqrt(4*l-4)), floor(sqrt(4*l-3)))
    if (nivel_Lvl % 2 == 0):
        # Calculando posição
        pos_vl = int(np.round((4*l - (nivel_Lvl)**2)/4))
        # Calculando vizinhos
        i = int(np.round((4*pos_vl + (nivel_Lvl - 1)**2 - 1)/4))
        j = int(np.round((4*pos_vl + (nivel_Lvl + 1)**2 - 1)/4))
        k = int(np.round((4*pos_vl + (nivel_Lvl + 1)**2 + 3)/4))
        # Adicionando elemento na matriz
        matriz[l-1, i-1] = 1
        matriz[l-1, j-1] = 1
        matriz[l-1, k-1] = 1
```

```
33 py     else:
34 py         # Calculando posição
35 py         pos_vl = int((4*l - (nivel_Lvl)**2 + 1)/4)
36 py         if nivel_Lvl != (2*N - 1):
37 py             if pos_vl == 1:
38 py                 # Calculando vizinhos
39 py                 i = int(np.round((4 + (nivel_Lvl - 1)**2)/4))
40 py                 j = int(np.round((4 + (nivel_Lvl + 1)**2)/4))
41 py                 k = 'NULL'
42 py                 # Adicionando elemento na matriz
43 py                 matriz[l-1, i-1] = 1
44 py                 matriz[l-1, j-1] = 1
45 py             else:
46 py                 if int(pos_vl == ((nivel_Lvl + 1)/2)):
47 py                     # Calculando vizinhos
48 py                     i = int(np.round((2*(nivel_Lvl - 1) + (
                             nivel_Lvl - 1)**2)/4))
49 py                     j = int(np.round((2*(nivel_Lvl + 1) + (
                             nivel_Lvl + 1)**2)/4))
50 py                     k = 'NULL'
51 py                     # Adicionando elementos na matriz
52 py                     matriz[l-1, i-1] = 1
53 py                     matriz[l-1, j-1] = 1
54 py                 else:
55 py                     # Calculando vizinhos
56 py                     i = int(np.round((4*(pos_vl - 1) + (nivel_Lvl -
                             1)**2)/4))
57 py                     j = int(np.round((4*pos_vl + (nivel_Lvl - 1)
                            **2)/4))
58 py                     k = int(np.round((4*pos_vl + (nivel_Lvl + 1)
                            **2)/4))
59 py                     # Adicionando elementos na matriz
60 py                     matriz[l-1, i-1] = 1
61 py                     matriz[l-1, j-1] = 1
62 py                     matriz[l-1, k-1] = 1
63 py         else:
64 py             if pos_vl == 1:
65 py                 # Calculando vizinhos
66 py                 i = int(np.round((4 + (nivel_Lvl - 1)**2)/4))
67 py                 j = 'NULL'
68 py                 k = 'NULL'
69 py                 # Adicionando elementos na matriz
70 py                 matriz[l-1, i-1] = 1
71 py             else:
72 py                 if int(pos_vl == (nivel_Lvl + 1)/2):
```

```
73 py                    # Calculando vizinhos
74 py                    i = int(np.round((2*(nivel_Lvl - 1) + (
                             nivel_Lvl - 1)**2)/4))
75 py                    j = 'NULL'
76 py                    k = 'NULL'
77 py                    # Adicionando elementos na matriz
78 py                    matriz[l-1, i-1] = 1
79 py                else:
80 py                    # Calculando vizinhos
81 py                    i = int(np.round((4*(pos_vl - 1) + (nivel_Lvl -
                             1)**2)/4))
82 py                    j = int(np.round((4*pos_vl + (nivel_Lvl - 1)
                            **2)/4))
83 py                    k = 'NULL'
84 py                    # Adicionando elementos na matriz
85 py                    matriz[l, i-1] = 1
86 py                    matriz[l, j-1] = 1
87 py    w.writerow([i,j,k,l])
88 pyvertices_vizinhos.close()
```

Listing A.1 – Matriz de adjacência do Grafo Gn na linguagem Python

ANEXO B – Código em Python das coordenadas dos vértices do grafo $\mathcal{G}_N$

```
''' [Algoritmo 02] Coordenadas baricêntricas de cada vértice vl de Gn
    Usei o int(np.round()) pra evitar que os índices i,j,k retorne valores float
    Utiliza os dados do algoritmo 01 [Matriz de adjacência]'''

if nivel_Lvl % 2 == 0:
    si0 = round(1 - (3*nivel_Lvl + 1)/(6*N), 8)
    si1 = round((3*pos_vl - 2)/(3*N), 8)
    si2 = round((3*nivel_Lvl - 6*pos_vl + 5)/(6*N), 8)
else:
    si0 = round(1 - (3*nivel_Lvl - 2)/(6*N), 8)
    si1 = round((3*pos_vl - 1)/(3*N), 8)
    si2 = round((3*nivel_Lvl - 6*pos_vl + 4)/(6*N), 8)
```

Listing B.1 – Coordenadas baricêntricas do vértice v_ℓ

ANEXO C – Código em Python da dinâmica da simulação do grafo $\mathcal{G}_N$

```
from math import floor, sqrt
import numpy as np
import csv
import random
from numpy.random import default_rng
import pandas as pd

df = pd.read_csv('vertices_N' + str(N) '.csv')
dia = 1
N = 100

dados_grafo = open('simulacao' + str(tempo) + '.csv',
                   'w', newline='', encoding='utf-8')
w = csv.writer(dados_grafo)
v_atual = 1
trajetoria = v_atual
# trajetoria = []
# trajetoria = [[0 for x in range(N)] for x in range(tempo)]
trajetoria = np.array([[], []])
simulacoes = 1000

for m in range(1, simulacoes):
    v_atual = 1
    while dia <= tempo:
        dia += 1
        u = random.random()
        if u <= df.iat[v_atual, 7]:  # pll
            sn = df.iat[v_atual, 0]  # l
            trajetoria[(dia), (simulacoes)].append(sn)

        elif (df.iat[v_atual, 7] < u <= (df.iat[v_atual, 7] + df.
            iat[v_atual, 9])):
            sn = df.iat[v_atual, 1]  # i
            trajetoria[(dia), (simulacoes)].append(sn)

        elif ((df.iat[v_atual, 7] + df.iat[v_atual, 9]) < u <= (df.
```

```
            iat[v_atual, 7] + df.iat[v_atual, 9] + df.iat[v_atual,
            10])):
36 py            sn = df.iat[v_atual, 2]  # j
37 py            trajetoria[(dia), (simulacoes)].append(sn)
38 py
39 py        elif ((df.iat[v_atual, 7] + df.iat[v_atual, 9] + df.iat[
            v_atual, 10]) < u <= 1):
40 py            sn = df.iat[v_atual, 3]  # k
41 py            trajetoria[(dia), (simulacoes-5)].append(sn)
42 py        v_atual = trajetoria[(dia), (simulacoes)]
43 py    w.writerow([trajetoria[m, dia]])
44 pydados_grafo.close()
```

Listing C.1 – Simulação da dinâmica da caminhada dos vértices v_ℓ

ANEXO D – DDL: Importando os dados para o pgAdmin4

```
CREATE TABLE covid_br (
  ID serial PRIMARY KEY,
  epi_week NUMERIC(10),
  date DATE,
  country VARCHAR(50),
  state VARCHAR(50),
  city VARCHAR(50),
  newDeaths NUMERIC(100),
  deaths  NUMERIC(100),
  newCases NUMERIC(100),
  totalCases  NUMERIC(100),
  deathsMS NUMERIC(100),
  totalCasesMS NUMERIC(100),
  deaths_per_100k_inhabitants NUMERIC(100),
  totalCases_per_100k_inhabitants NUMERIC(100),
  deaths_by_totalCases NUMERIC(100),
  recovered NUMERIC(100),
  suspects NUMERIC(100),
  tests NUMERIC(100),
  tests_per_100k_inhabitants NUMERIC(100),
  vaccinated  NUMERIC(100),
  vaccinated_per_100_inhabitants NUMERIC(100),
  vaccinated_second NUMERIC(100),
  vaccinated_second_per_100_inhabitants NUMERIC(100),
  vaccinated_single NUMERIC(100),
  vaccinated_single_per_100_inhabitants NUMERIC(100),
  vaccinated_third NUMERIC(100),
  vaccinated_third_per_100_inhabitants NUMERIC(100)
);

COPY covid_br() FROM '[endereço onde está seu arquivo .csv]'
   DELIMITER ',' CSV HEADER;
```

Listing D.1 – PostgreSQL: Script de inserção do banco de dados no SQL

ANEXO E – DQL: Consultas no banco

```
1 pyWITH coord_covid AS(
2 py  SELECT
3 py    date AS "data_registro",
4 py    CASE
5 py      WHEN EXTRACT(year from date) = 2020 THEN ((213200000 -
            totalcases - deaths)/213200000)
6 py      WHEN EXTRACT(year from date) = 2021 THEN ((214300000 -
            totalcases  - deaths)/214300000)
7 py      WHEN EXTRACT(year from date) = 2022 THEN ((207750291  -
            totalcases  - deaths)/207750291)
8 py      END "n_suscetiveis",
9 py    CASE
10 py     WHEN EXTRACT(year from date) = 2020 THEN ((totalcases +
            suspects)/213200000)
11 py     WHEN EXTRACT(year from date) = 2021 THEN ((totalcases +
            suspects)/214300000)
12 py     WHEN EXTRACT(year from date) = 2022 THEN ((totalcases +
            suspects)/207750291)
13 py     END "n_infectados",
14 py   CASE
15 py     WHEN EXTRACT(year from date) = 2020 THEN ((recovered)
            /213200000)
16 py     WHEN EXTRACT(year from date) = 2021 THEN ((recovered)
            /214300000)
17 py     WHEN EXTRACT(year from date) = 2022 THEN ((recovered)
            /207750291)
18 py     END "n_recuperados"
19 py  FROM covid_br
20 py    WHERE state = 'TOTAL'
21 py)
22 pySELECT * FROM coord_covid
```

Listing E.1 – PostgreSQL: Tratamento de dados do banco do COVID-19 no Brasil

```
1 pyWITH coord_covid AS(
2 py  SELECT
3 py    date AS "data_registro",
4 py    CASE
5 py      WHEN EXTRACT(year from date) = 2020 THEN ((213200000 -
            totalcases - deaths)/213200000)
```

```
      WHEN EXTRACT(year from date) = 2021 THEN ((214300000 -
          totalcases  - deaths)/214300000)
      WHEN EXTRACT(year from date) = 2022 THEN ((207750291  -
          totalcases  - deaths)/207750291)
      END "n_suscetiveis",
    CASE
      WHEN EXTRACT(year from date) = 2020 THEN ((totalcases +
          suspects)/213200000)
      WHEN EXTRACT(year from date) = 2021 THEN ((totalcases +
          suspects)/214300000)
      WHEN EXTRACT(year from date) = 2022 THEN ((totalcases +
          suspects)/207750291)
      END "n_infectados",
    CASE
      WHEN EXTRACT(year from date) = 2020 THEN ((recovered)
          /213200000)
      WHEN EXTRACT(year from date) = 2021 THEN ((recovered)
          /214300000)
      WHEN EXTRACT(year from date) = 2022 THEN ((recovered)
          /207750291)
      END "n_recuperados"
  FROM covid_br
    WHERE state = 'TOTAL'
)
SELECT
  *
FROM(
  SELECT
    CASE
      WHEN (nivel_Lvl % 2 <> 0) THEN 1
        WHEN (c."n_suscetiveis" <= g.w0_x) THEN 1
            WHEN (c."n_infectados" >= g.w0_y) AND (c."n_infectados"
                <= g.w1_y) THEN 1
              WHEN (c."n_recuperados" >= g.w1_z) AND (c."
                  n_recuperados" <= g.w2_z) THEN 1
    END AS vertice
  FROM coord_covid c
    JOIN grafon10_4 g ON c."n_suscetiveis" >= g.w2_x
  UNION
  SELECT
    CASE
      WHEN (nivel_Lvl % 2 = 0) THEN 1
        WHEN (c."n_suscetiveis" <= g.w0_x) THEN 1
            WHEN (c."n_infectados" >= g.w0_y) AND (c."n_infectados"
                <= g.w1_y) THEN 1
```

```
40 py              WHEN (c."n_recuperados" >= g.w1_z) AND (c."
                      n_recuperados" <= g.w2_z) THEN 1
41 py    END AS vertice
42 py  FROM coord_covid c
43 py    JOIN grafon10_4 g ON c."n_suscetiveis" >= g.w1_x
44 py  )x
45 pyORDER BY x.vertice ASC
```

Listing E.2 – PostgreSQL: Dinâmica da simulação