Universidade Federal do Rio Grande do Sul
Instituto de Física - Escola de Engenharia

Uso de grafos de visibilidade para tomada de decisões de investimentos na bolsa de valores brasileira

Projeto Final de Curso em Engenharia Física II

Leonardo Silveira Silva
Professor Orientador: Sebastian Gonçalves

Resumo

O presente trabalho consistiu em desenvolver um sistema de tomada de decisões de investimento, explorando a *clusterização* definida a partir da similaridade média dos grafos de visibilidade das séries temporais de preço de fechamento diário dos 100 ativos mais líquidos da bolsa de valores brasileira, entre os anos de 2021 e 2023. O objetivo principal foi encontrar uma combinação de parâmetros que fizesse o sistema ter uma assertividade média superior a 50% e também vencesse em assertividade e rentabilidade a estratégia equivalente, sem o uso da *clusterização*. Depois de 10.221 simulações, foi encontrada uma combinação de parâmetros que atendia os resultados esperados. Por fim, conclui-se que o presente trabalho teve êxito no desenvolvimento e na parametrização do sistema, alcançando vantagem no uso da *clusterização* em relação a estratégia sem *clusterização*.

Abstract

The present work consists of developing an investment decision-making system, exploiting clustering defined based on the average similarity of the visibility graphs of the daily closing price time series of the 100 most liquid assets on the Brazilian stock exchange between the years 2021 and 2023. The main objective was to find a combination of parameters that made the system have an average assertiveness greater than 50% and also beat the equivalent strategy, without the use of clustering in assertiveness and profitability. After 10,221 simulations, a combination of parameters was found that met the expected results. Finally, it is concluded that the present work was successful in the development and parameterization of the system, achieving an advantage in the use of clustering in relation to the strategy without clustering.

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o mundo financeiro tem experimentado um rápido avanço no desenvolvimento e na aplicação de técnicas quantitativas. Diante desse cenário, o mercado de ações brasileiro, um dos mais dinâmicos e desafiadores do mundo, emerge como um campo fértil para a aplicação de métodos inovadores de análise. Este trabalho tem como foco a exploração e aplicação de grafos de visibilidade no estudo de ativos da bolsa brasileira, buscando entender suas correlações e padrões.

A utilização de grafos para analisar séries temporais não é um conceito novo, mas sua aplicação no contexto das finanças e, em particular, no mercado de ações, abre uma perspectiva intrigante. Ao traduzir os movimentos de preços em estruturas gráficas, podemos enxergar novas dimensões de análise e, consequentemente, identificar padrões e correlações anteriormente ocultos.

O presente projeto se estrutura em várias etapas, desde a geração e armazenamento desses grafos até o desenvolvimento de uma função de simulação que permita testar estratégias de investimento com base nas similaridades medidas. Através deste projeto, aspira-se não apenas ampliar o entendimento sobre a dinâmica do mercado de ações brasileiro, mas também contribuir com ferramentas e insights para investidores e profissionais da área.

Em um momento em que a tecnologia e a análise de dados desempenham papéis cada vez mais críticos na tomada de decisões financeiras, a capacidade de entender e prever os movimentos do mercado é mais valiosa do que nunca. Este trabalho busca ser uma contribuição nesse contexto, oferecendo uma nova lente através da qual podemos observar os movimentos do mercado.

2 REFERENCIAL TEÓRICO

Nessa seção serão explanados os conceitos necessários para o entendimento da ideia central do projeto.

2.1 Teoria dos Grafos

Desde os problemas de sete pontes de Königsberg resolvidos por Euler no século XVIII, a teoria dos grafos estabeleceu-se como um pilar fundamental da matemática e das ciências da computação. Em sua essência, a teoria dos grafos envolve o estudo de redes formadas por pontos, denominados vértices, interconectados por linhas chamadas arestas. No entanto, ao longo das últimas décadas, esta teoria tradicional tem encontrado aplicações em domínios vastamente expandidos e, muitas vezes, inesperados, abrangendo desde a biologia à sociologia, da tecnologia da informação à economia. Esta expansão tem sido impulsionada, em grande parte, pelo desenvolvimento e pela compreensão das redes complexas.

As redes complexas representam uma extensão e generalização da teoria dos grafos clássica. Ao invés de considerar redes puramente regulares ou aleatórias, as redes complexas reconhecem e abordam a rica tapeçaria de padrões interconectados e propriedades emergentes observadas em sistemas do mundo real. Estas redes não são nem completamente ordenadas nem completamente aleatórias, mas, ao invés disso, exibem padrões que têm implicações profundas para a dinâmica e a funcionalidade do sistema em questão.

2.1.1 Definições

Um grafo G é genericamente constituido por dois conjuntos, $G = (V, E)$, onde V é um conjunto constituído por uma entidade chamada de vértice e E é um conjunto constituído por uma entidade chamada de aresta. Cada aresta $e \in E$ é constituída pela composição de pelo menos dois elementos distintos v e u tais que $v \in V$ e $u \in V$, sendo assim, $E \subseteq V \times V$. A seguir serão apresentadas definições mais intuitivas sobre as entidades fundamentais dos grafos.

- **Vértice**: também chamado de nó ou nodo, o vértice é a entidade fundamental de um grafo. São graficamente representados por um ponto. No contexto do uso de grafos para a representação de redes, os vértices são as entidades a serem conectadas, podendo ser por exemplo a representação de pessoas em redes sociais ou computadores em redes de computação.
- **Aresta**: é a representação da conexão entre dois vértices. Não existem vértices conectados em si mesmos. São graficamente representadas por retas. Uma aresta pode ter uma direção no caso de grafos direcionados, sendo representada por um par ordenado, ou não no caso de grafos não direcionados, sendo representada por um conjunto. No caso de grafos ponderados, a aresta também tem um peso associado a ela, que pode representar distância, custo, tempo, capacidade ou qualquer outro tipo de medida.
- **Matriz de Adjacência**: é a representação matricial de um grafo. É uma matriz quadrada A de dimensão $n \times n$ onde n é o número de vértices do grafo. Se os nós i e j são conectados, o elemento A_{ij} é igual a 1 no caso de grafos não ponderados e igual a representação númerica do peso no caso de grafos ponderados. Para grafos não direcionados, a matriz de adjacência é simétrica.

2.1.2 Propriedades

A seguir serão revistas as propriedades fundamentais para a descrição dos grafos.

2.1.2.1 Clusterização Local

A clusterização local é um número que quantifica a tendência dos vértices em se agruparem. De maneira geral, a clusterização local C_i do nodo i pode ser calculada como:

$$C_i = \frac{A_v}{\binom{k}{2}}$$

Onde A_v é o número de arestas existentes entre os k vizinhos do nodo i e $\binom{k}{2}$ é o valor da combinação dos k vizinhos do nodo i tomados 2 a 2, que é o número de conexões possíveis entre os k vizinhos vizinhos do nodo i.

2.1.2.2 Clusterização Global

A clusterização global é a clusterização local média da rede. De maneira geral, a clusterização global $\langle C \rangle$ de uma rede com n vértices pode ser definida como:

$$\langle C \rangle = \frac{1}{n} \sum_{i=1}^{n} C_i$$

2.1.2.3 Average Shortest Distance

A métrica *average shortest distance* ($\langle D \rangle$) é um número que representa o menor caminho médio entre dois vértices escolhidos aleatóriamente na rede. De maneira geral, a $\langle D \rangle$ de uma rede complexa com n vértices pode ser calculada como:

$$\langle D \rangle = \frac{1}{n(n-1)} \sum_{i \neq j} d(v_i, v_j)$$

Onde $d(i, j)$ é a função que da a distância entre os vértices i e j que é definida como zero caso não seja possível conectar os vértices em questão. Na implementação do cálculo de $\langle D \rangle$ feita no presente trabalho, será utilizada a função $d(i, j)$ disponibilizada pela biblioteca NetworkX. Um exemplo real de aplicação da *average shortest distance* é o seu uso para quantificar a eficiência de transferência de massa em uma rede metabólica.

2.1.2.4 Centralidade

A centralidade de um nodo é definida como o número de conexões de um nodo dividido pela quantidade de outros vértices que existe na rede. Basicamente, a centralidade expressa a importância do nodo em relação a rede. De maneira geral, a centralidade (Cen_i) de um nodo i pertencente a uma rede com n vértices é calculada como:

$$Cen_i = \frac{g}{n-1}$$

Onde g é o grau do nodo.

2.1.2.5 Assortatividade

A assortatividade, também chamada de homofilia, é um número que quantifica a tendência de vértices semelhantes se conectarem na rede. De maneira geral, a assortatividade r de uma rede complexa é dada por:

$$r = \frac{\sum_{i,j} ij(e_{ij} - q_i q_j)}{\sigma^2}$$

Onde $\sigma^2 = \sum_k k^2 q_k - [\sum_k k^2 q_k]^2$, $q_k = kp_k/\langle k \rangle$ e e_{ij} é a matriz de correlação de graus. Vale ressaltar que $r \in [-1, 1]$ e que $r < 0$ quer dizer que a rede é dissassortativa, se $r = 0$ a rede é não-assortativa e se $r > 0$ a rede é assortativa.

2.1.3 Medidas de Similaridade

Nessa seção serão apresentadas algumas estratégias comuns para a comparação de grafos que objetivam quantificar a sua similaridade.

2.1.3.1 Distância de Edição de Grafos

Dado dois grafos A e B, a distância de edição de grafos (sendo abreviado por *GED* devido a sua nomenclatura em inglês) consiste em verificar a quantidade mínima de operações que seriam necessárias para transformar A em B ou vice-versa. Existem 4 operações fundamentais a serem consideradas: adição e remoção de vértices e de arestas. Existem algumas abordagens mais complexas que consideram também a substituição de vértices e arestas como operações a serem feitas, no entanto, uma substituição pode ser generalizada em uma sequência de remoções e adições.

Nessa técnica, é necessário arbitrar o "custo" de cada operação, para que assim a similaridade possa ser quantificada. Quanto menor for a distância, mais similar os grafos são. Ele é um processo NP-Difícil, o que significa que dependendo do tamanho do grafo e da capacidade de processamento à disposição ele pode ser um processo computacionalmente oneroso.

A seguir será exposto uma implementação simplificada do método realizada pelo o autor deste texto usando a linguagem de programação *Python* com o uso da biblioteca para manipulação de grafos *NetworkX*. Nessa implementação, o custo adotado para cada operação é igual a 1. Essa implementação funciona apenas para grafos que tem nodos com o mesmo *label*.

Código 1 – Implementação do GED em Python

```
1 import networkx as nx
2
3 def GED(g1, g2):
4     g1 = g1.copy()
5     g2 = g2.copy()
```

```
6
7
8       distance = 0
9
10      nodes_g1 = set(g1.nodes())
11      nodes_g2 = set(g2.nodes())
12
13
14      for node in nodes_g1 - nodes_g2:
15          g1.remove_node(node)
16          distance += 1
17
18      for node in nodes_g2 - nodes_g1:
19          g2.remove_node(node)
20          distance += 1
21
22
23      edges_g1 = set(g1.edges())
24      edges_g2 = set(g2.edges())
25
26
27      for edge in edges_g1 - edges_g2:
28          g1.remove_edge(*edge)
29          distance += 1
30
31      for edge in edges_g2 - edges_g1:
32          g2.remove_edge(*edge)
33          distance += 1
34
35      return distance
```

No entanto, a distância de edição quantifica a "diferença" entre dois grafos, sendo seu valor máximo igual a $C_{n,2}$. Para calcular a similaridade usando a *GED*, que neste trabalho referenciada como *sEdicao*, a distância de edição será normalizada de maneira proporcional ao número de vértices dos grafos, tendo em vista que só serão comparados grafos com o mesmo número de nós.

$$sEdicao = \frac{n}{n + GED} \tag{1}$$

Onde n é o número de vértices dos grafos.

2.1.3.2 Distância de Frobenius

A distância de Frobenius é uma métrica comumente usada para quantificar a diferença entre duas matrizes. Para o contexto do cálculo de similaridade entre grafos, pode-se usar a distância de Frobenius das matrizes de adjacência como indicador da similaridade entre os grafos. Dado duas matrizes de adjacência A e B, a distância de Frobenius (D_F) entre elas é dada por:

$$D_F = \sqrt{\sum_{i,j}(A_{ij} - B_{ij})^2} \tag{2}$$

O valor resultante da expressão 2 já poderia ser uma quantificação da similaridade entre os grafos. Porém, algumas abordagens usam como similaridade $sFrobenius$ o resultado da expressão 3:

$$sFrobenius = \frac{n - D_F}{n} \tag{3}$$

Onde n é o número de vértices dos grafos. Usando a abordagem da expressão 3 pode-se concluir que $0 < S \leq 1$, sendo que quanto maior for S maior é a similaridade entre os grafos.

2.1.3.3 Coeficiente de Jaccard para arestas

O coeficiente de Jaccard para arestas consiste basicamente em dividir o número de arestas comuns a dois grafos pelo número de arestas totais que existem considerando os dois conjuntos de arestas. Logo, dados dois grafos G_1 e G_2 e suas respectivas arestas E_1 e E_1, o coeficiente de Jaccard $sJaccard$ é dado por:

$$sJaccard = \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|}$$

Assim, pode-se afirmar que o coeficiente de Jaccard $J(G_1, G_2)$ assume a condição de $0 \leq J(G_1, G_2) \leq 1$, onde 0 significa que os grafos não tem nenhuma aresta em comum e 1 significa que os grafos são idênticos. É um método que funciona bem para grafos com o mesmo número de vértices e com arestas sem pesos.

2.1.4 Grafos de Visibilidade

Dado um conjunto de pontos contidos em um espaço euclidiano, o grafo de visibilidade correspondente a este conjunto tem seus vértices iguais aos pontos e suas conexões são estabelecidas entre pontos que são "visíveis entre si", isto é, só existem arestas para vértices que não tenham nada que sirva de obstáculo para uma conexão em linha reta. Os grafos de visibilidade são amplamente utilizados para converter séries temporais em redes complexas. Plotando uma série temporal de valores como um gráfico de barras, o grafo de visibilidade associado tem seus

vértices associados a cada elemento dessa série e as conexões são dadas entre barras que são "visíveis entre si".

Assim, genericamente, dado uma série temporal $\{y_i | i = 1, 2, 3, ..., N\}$ onde y_i é o valor medido no tempo t_i, os vértices V_a e V_b correspondentes aos elementos da série (t_a, y_a) e (t_b, y_b) serão conectados se qualquer elemento (t_i, y_i), onde $a < i < b$, atender a condição 4:

$$y_i < y_a + (y_b - y_a) \cdot \frac{t_i - t_a}{t_b - t_a} \tag{4}$$

A condição 4 é a formalização de que dois pontos da série são conectados caso não haja uma barra que cruze a reta que os une na representação da série em gráficos de barra. Séries temporais periódicas se transformam em redes regulares e séries temporais randômicas se transformam em redes complexas.

A figura 1 ilustra um exemplo da transformação de uma série temporal aleatória em um grafo de visibilidade.

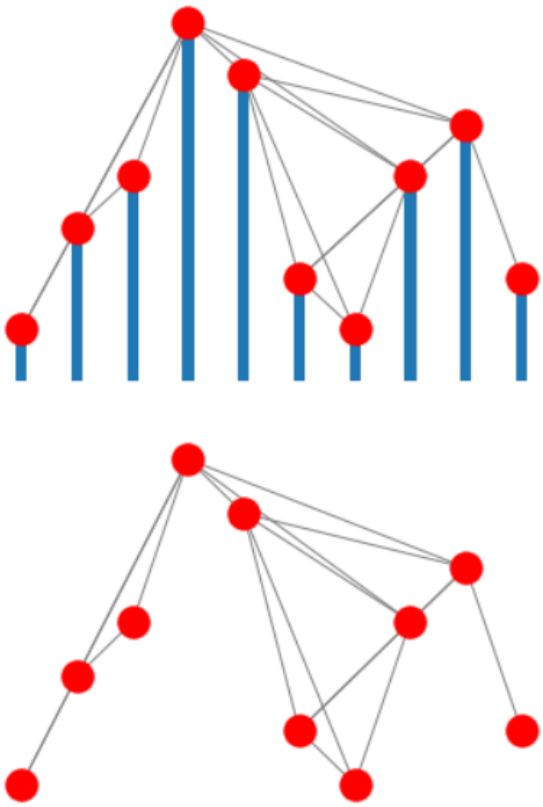

Figura 1 – Série temporal aleatória transformada em um grafo de visibilidade

Dado uma série temporal de valores, segue a implementação da verificação da condição 4 na linguagem de programação Python para a sua transformação em um grafo de visibilidade com o uso da biblioteca *NetworkX*:

Código 2 – Função de transformação de série temporal em grafo de visibilidade

```
import networkx as nx

def visibility_graph(time_series):

    n = len(time_series)
    graph = nx.Graph()

    for i in range(n):
        graph.add_node(i)

    for i in range(n):
        for j in range(i+1, n):
            if all((time_series[j]-time_series[i])/(j-i) > (
               time_series[k]-time_series[i])/(k-i) for k in range(i
               +1, j)):
                graph.add_edge(i, j)

    return graph
```

3 METODOLOGIA

Neste capítulo, será apresentada uma descrição detalhada do projeto com base na teoria descrita no capítulo 2. Será delineada a ideia central e, em seguida, fornecida uma explicação minuciosa das etapas e desafios que já foram mapeados. Em seguida será feita uma análise abrangente do conjunto de dados e das técnicas de manipulação a serem aplicadas. Para encerrar, será feita uma descrição técnica da implementação do projeto.

3.1 Ideia Central

Este projeto visa criar um sistema de decisão de investimento para ativos da bolsa brasileira, utilizando grafos de visibilidade derivados das séries temporais de preços de fechamento diário. A ideia é analisar a similaridade média entre os ativos para criar *clusters* e com isso tomar decisões de investir em cada ativo baseado no comportamento dos ativos do seu *cluster*.

O *cluster* é definido por ativos em que seus grafos tem uma média de similaridades instantâneas passadas maior a igual a um valor mínimo arbitrado. Portanto, o presente trabalho pretende explorar o uso das técnicas de similaridade para clusterização de ativos da bolsa de valores brasileira através da aplicação dos grafos de visibilidade nas séries temporais destes ativos. A imagem 2 mostra uma representação visual do fluxo da estratégia.

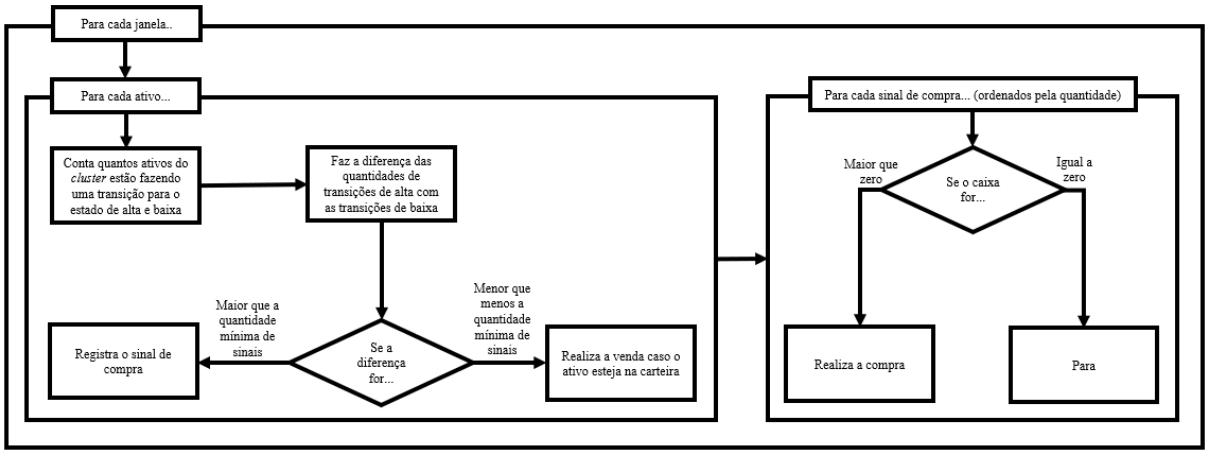

Figura 2 – Fluxo de execução da estratégia.

Uma "janela", como definido aqui, refere-se a uma sub-série temporal, que é essencialmente uma "fatia" da série principal de tamanho reduzido. Os grafos de visibilidade serão construídos a partir de todas as janelas temporais possíveis de tamanho $N \in [5, 50]$, onde N está em dias. A notação para a referência dos grafos se dará no seguinte formato:

$$G_{Ativo,i,i+N-1} \tag{5}$$

Onde $Ativo$ é o código do ativo na bolsa de valores brasileira, i é o tempo do primeiro elemento da série temporal do preço diário do ativo a ser considerado para a construção do grafo e $i + N - 1$ o último. Por exemplo: dado a série temporal $\{P_i | i = 1, 2, 3, 4, 5, 6, 7\}$ de uma ação de código "EXMPL", o grafo denotado por $G_{EXMPL,3,7}$ é o grafo de visibilidade de uma janela de tamanho $N = 5$ da série temporal do ativo "EXMPL" formado a partir de todos os elementos P_i tal que $P_3 \leq P_i \leq P_7$, ou seja, é o grafo de visibilidade montado a partir dos elementos $(P_3, P_4, P_5, P_6, P_7)$.

A partir de cada janela da série temporal de cada ativo, queremos extrair duas informações principais:

1. O estado instantâneo da janela analisada, i.e, se a janela é uma sub-série de tendência de alta, de baixa ou lateral.

2. O nível de correlação com todos os outros ativos a partir da similaridade dos seus respectivos grafos de visibilidade da janela correspondente.

O estado da janela é determinado pela variação entre os valores inicial e final, comparada a uma tolerância preestabelecida ϵ. Dado uma variação R, se $R < -\epsilon$, o estado da janela será considerado de baixa, se $-\epsilon \leq R \leq \epsilon$, o estado da janela será considerado lateral e se $R > \epsilon$

o estado da janela será considerado de alta. O parâmetro ϵ é o limiar a ser definido para evitar considerar variações pequenas como sendo de alta ou de baixa significativa.

Dadas duas séries temporais $A = (3, 1, 5)$ e $B = (5, 1, 3)$, ambas as séries dão origem a grafos de visibilidade idênticos. No entanto a série temporal A tem um comportamento bem distinto de B: enquanto a série A é uma série de "alta" B é uma série de "baixa". Para resolver este problema, vamos considerar essa diferença de comportamento na atribuição do sinal numérico da similaridade. Portanto, quando dois grafos forem comparados, o valor da similaridade calculada terá o sinal numérico negativo (-) se suas janelas forem uma de alta e outra de baixa e positivo (+) caso contrário. Por exemplo, se o módulo da similaridade instantânea entre dois grafos for 0.57, consideraremos ela como -0.57 caso um dos grafos tenha uma janela de alta e outro tenha a janela de baixa e $+0.57$ caso contrário. Para o que segue, vale destacar o que entende-se como sinal de negociação:

- **Sinal de Compra**: ocorre quando, para ativos de um mesmo *cluster*, a quantidade de ativos transicionado simultaneamente para alta supera a quantidade de ativos transicionado simultaneamente para baixa em um valor mínimo arbitrado chamado de "quantidade mínima de transições".
- **Sinal de Venda**: ocorre quando, para ativos de um mesmo *cluster*, a quantidade de ativos transicionado simultaneamente para baixa supera a quantidade de ativos transicionado simultaneamente para alta em um valor mínimo arbitrado chamado de "quantidade mínima de transições".

Uma motivação para o uso dos grafos de visibilidade é que, em comparação com os métodos estatísticos convencionais para a mensuração de correlação de séries temporais, o uso da similaridade entre os grafos pode se mostrar mais eficiente para a análise de séries temporais financeiras porque os grafos de visibilidade capturam dinâmicas não lineares e características locais das séries, além de serem robustos a ruídos. Com isso, o presente trabalho pretende aproveitar a variedade de métricas de teoria dos grafos para medir uma correlação, criando e medindo a eficácia de um sistema que implementa essa técnica.

3.2 Parâmetros

Serão testadas as diferentes combinações de parâmetros:

- **N:** tamanho da janela que irá de 5 a 50 dias, variando-se de 5 em 5 dias. A escolha do tamanho da janela pode afetar significativamente a sensibilidade do modelo a mudanças de tendência no mercado.
- **Método:** Serão considerados a Distância de Edição, a Distância de Frobenius e o Coeficiente de Jaccard.

- **Quantidade mínima de transições:** Este parâmetro, variando entre os valores de 3, 5, 7 e 10 determina o saldo mínimo de ativos do *cluster* realizando transições simultâneas para que exista um sinal de negociação.
- **Similaridade Mínima:** Este parâmetro define o limiar de similaridade acima do qual a correlação entre ativos é considerada suficientemente forte incluí-los em um mesmo *cluster*. Serão testados os valores 0.65, 0.75, 0.85 e 0.9
- **Tamanho Média:** Este parâmetro define quantas medições instantâneas de similaridade imediatamente passadas serão usadas para compor a média móvel de similaridade entre os ativos. Esse parâmetro tem o objetivo de suavizar a volatilidade da composição dos *clusters*. Serão testados os valores de 5, 10 e 15.
- **Concentração máxima:** Estabelece um limite para a concentração do capital investido em um único ativo, variando entre os valores 5%, 7.5% e 10% do capital inicial. Este parâmetro regula a tendência de diversificação do modelo.
- **Intervalo de simulação:** Será adotado um intervalo de 180 dias sequenciais para cada execução de simulação.

3.3 Etapas

A seguir, apresentamos as etapas e os respectivos desafios do projeto.

Etapa 1: Gerar e armazenar grafos de visibilidade para todas as variações de tamanho de janelas e ativos da bolsa brasileira.

Desafio 1.1: Construir um código suficientemente otimizado para que seja possível repetir essa etapa dentro de um tempo razoável caso surja a necessidade de ajustes na escolha de informações extraídas.

Desafio 1.2: Armazenar as informações de constituição dos grafos de modo que, a partir dos parâmetros que identificam o grafo, seja possível acessa-lo e manipula-lo com fluidez.

Etapa 2: Calcular e armazenar a correlação instantânea entre todos os grafos de cada janela usando os métodos de similaridade detalhados no capítulo 2.1.3.

Desafio 2.1: Normalizar todos os valores de similaridades calculados com cada método para um valor entre $[0, 1]$.

Etapa 3: Desenvolver uma função de programação versátil para simular a estratégia, permitindo ajustes nos parâmetros testados. Essa função assumirá a existência de um patrimônio líquido igual a 1 no instante inicial. Desse modo, será mais fácil de entender a rentabilidade alcançada pela estratégia no final da simulação. Será possível determinar

se no instante inicial o que se têm é apenas capital ou se já existe uma carteira de ativos. No caso da segunda opção, deverá ser necessário informar os códigos dos ativos, bem como a proporção entre eles.

Desafio 3.1: Implementar a possibilidade de começar a simulação em uma data qualquer dentro do intervalo de datas disponíveis e também arbitrar a duração da simulação em número de dias a partir dessa data. Por padrão será considerado o primeiro dia da série e toda a série disponível.

Desafio 3.2: Implementar a possibilidade de definir uma concentração máxima de alocação de recursos em um ativo. Assim, quando houver por exemplo um sinal de compra e houver recursos disponíveis para alocação, o sistema não alocará mais que o máximo permitido em comparação com o patrimônio líquido total disponível. Por padrão a concentração máxima será de 10%.

Desafio 3.3: Implementar a possibilidade de escolha do método de similaridade entre os grafos.

Desafio 3.4: Implementar a parametrização da quantidade de correlações instantâneas passadas a serem consideradas para o calculo da correlação atual. A correlação atual será uma média aritmética das correlações passadas consideradas. Isso é interessante porque as correlações instantâneas podem ser voláteis, principalmente em relação ao sinal.

Desafio 3.5: Implementar uma parametrização de sensibilidade do sinal. Essa parametrização envolverá o valor mínimo de correlação para que uma mudança de estado contribua para a composição do sinal bem como a quantidade mínima de ativos correlacionados mudando para o mesmo estado simultaneamente para que o sinal exista.

Etapa 4: Verificar a performance do sistema para variações dos parâmetros da função de simulação criada.

Desafio 4.1: Encontrar uma combinação de parâmetros que atenda os resultados esperados descritos no capítulo 4.

Etapa 5: Desenvolver uma estrutura que permita a visualização dos registros e o teste de variações nos parâmetros.

3.4 Conjunto de dados

Nessa seção será discorrido sobre a massa de dados que será utilizada para a construção do modelo, bem como as estruturas de armazenamento planejadas e o procedimento para gerar visualizações.

3.4.1 Extração

Para a extração dos dados históricos de cotação das empresas da bolsa de valores brasileira foi utilizado a biblioteca gratuita do *YahooFinance* em *Python*. No entanto, ela retorna a série de dados histórica de um ativo em específico e precisa que o código do ativo na bolsa seja passado como parâmetro para a busca. Para conseguir o código de todos os ativos de maneira dinâmica, foi construído um *script* que consulta o site *https://www.dadosdemercado.com.br/bolsa/acoes* e extrai os códigos de uma tabela que existe na página.

Código 3 – Script para extrair os códigos das ações em um site

```
1  import pandas as pd
2  import requests
3  from bs4 import BeautifulSoup
4
5  html_content = requests.get('https://www.dadosdemercado.com.br/bolsa/
       acoes').text
6
7  table = BeautifulSoup(html_content, 'html.parser').find('table')
8
9
10 table_data = []
11 rows = table.find_all('tr')
12 for row in rows:
13     row_data = []
14     cells = row.find_all('td')
15     for cell in filter(lambda x: bool(x), map(lambda x: x.text.strip
           (), cells)) :
16         row_data.append(cell)
17     if row_data:
18         table_data.append(row_data)
19
20 ativos_b3 = pd.DataFrame(table_data)
21 ativos_b3.columns = ['Codigo', 'Nome', 'Volume', 'Ultima', 'Variacao'
       ]
```

A variável *ativos_b3* é um objeto do tipo *DataFrame* da biblioteca *Pandas* com uma coluna com todos os códigos dos ativos. Tendo posse de todos os códigos, a extração das séries temporais foi realizada como o que segue:

Código 4 – Extração das séries

```
1  import yfinance as yf
2  import pandas as pd
3  import requests
```

```
4
5  dfs = []
6
7  for symbol in ativos_b3['Codigo']:
8      try:
9          df = yf.download(symbol+'.SA')
10         df = df[[c for c in df.columns if c != 'Adj Close']]
11         df = df.reset_index()
12
13         df.columns = ['Data', 'Abertura', 'Maximo', 'Minimo', '
               Fechamento', 'Volume']
14
15         df.insert(1, 'Ativo', symbol)
16
17         dfs.append(df)
18
19     except Exception as e:
20         print(f'Simbolo {symbol} deu errado! ', str(e))
21         continue
22
23 df = pd.concat(dfs).reset_index(drop = True)
24 df['Data'] = df['Data'].apply(lambda x: x.strftime('%Y-%m-%d'))
25 df = df.drop_duplicates(subset = ['Data', 'Ativo'], keep = 'last')
26 df.to_csv('b3_historic_data.csv', sep = ';', index = False)
```

Foi criado um *DataFrame* com a série de cada ativo, que por padrão já vem suficientemente tratado e com os tipos de variáveis corretos. As colunas foram renomeadas e todos os *DataFrame's* foram concatenados, totalizando 1.839.112 linhas. Um ajuste no formato das datas foi feito retirando a parte de hora das datas, que estava zerada, foi dado um comando para excluir linhas duplicadas caso tivesse e no fim o *DataFrame* com todos os dados foi salvo em um arquivo do tipo CSV.

3.4.2 Estruturas de armazenamento

Agora serão apresentadas as estruturas de armazenamento planejadas para a execução do projeto, que serão tabelas de um banco de dados do tipo *MariaDB*. A primeira tabela armazenará a série temporal dos preços, bem como algumas propriedades extras dos elementos da série.

A próxima tabela serve para armazenar as informações dos grafos existentes, como o ativo respectivo, a data inicial e final utilizada para a construção do grafo e o número de elementos existentes. O número de elementos existentes pode ser considerado redundante, pois tendo as datas iniciais e finais esse número poderia ser facilmente verificado, no entanto, ele servirá para futuras revisões de consistência dos dados.

Coluna	Tipo	Descrição
data	Date	Data do registro
ativo	Varchar(5)	Código alfanumérico de identificação do ativo
precoAbertura	Float	Preço da primeira transação do dia
precoMaximo	Float	Preço de maximo alcançado dentro do dia
precoMinimo	Float	Preço mínimo alcançado dentro do dia
precoFechamento	Float	Preço da última transação do dia
volume	Float	Volume monetário de negócios do dia

Tabela 1 – Descrição da tabela de armazenamento das séries temporais.

Coluna	Tipo	Descrição
id	Integer(Autoincrement)	Código numérico de identificação do grafo.
ativo	Varchar(5)	Código alfanumérico de identificação do ativo
dataInicial	Date	Data do primeiro elemento considerado para construção do grafo
dataFinal	Date	Data do último elemento considerado para construção do grafo
n	Integer	Tamanho da janela. É a quantidade de elementos da série temporal considerados para a construção do grafo a partir do primeiro.
var	Float	Variação percentual do preço de fechamento do ativo para o grafo em questão

Tabela 2 – Descrição da tabela de armazenamento dos identificadores dos grafos.

Também existirá uma tabela para armazenar os elementos visíveis entre si da série total de cada ativo. Dado a série de cada ativo, para cada elemento, será verificado se ele é visível somente para os elementos posteriores, dessa forma será garantido que a relação de visualização não será duplicada. Esta é a maneira mais otimizada para armazenar essa informação e facilitará a geração das conexões entre os grafos. Nessa tabela só existirão registros que são visíveis entre si.

Coluna	Tipo	Descrição
ativo	Varchar(5)	Código alfanumérico de identificação do ativo. É a chave primaria da tabela
dataA	Date	Data do elemento mais antigo.
dataB	Date	Data do elemento mais recente.

Tabela 3 – Descrição da tabela de armazenamento dos elementos visíveis entre si da série de cada ativo.

Por fim, existirá uma tabela para armazenar as similaridades entre os grafos com cada método.

Coluna	Tipo	Descrição
idGrafoA	Integer	Código numérico de identificação do primeiro grafo.
idGrafoB	Integer	Código numérico de identificação do segundo grafo.
sEdicao	Float	Valor da similaridade de acordo com a distancia de edição
sFrobenius	Float	Valor da similaridade de acordo com a distancia de Frobenius
sJaccard	Float	Valor da similaridade de acordo com o coeficiente de Jaccard

Tabela 4 – Descrição da tabela de armazenamento das similaridades entre os grafos.

3.4.3 Visualização

Todas as visualizações dos dados se darão com o auxilio da biblioteca *MatPlotLib* do *Python* em um ambiente *Jupyter Notebook*.

3.5 Implementação

Por simplicidade e seguindo os códigos já expostos no presente projeto, a implementação das rotinas se dará com o uso da linguagem de programação *Python*. No entanto, assume-se que se no curso do projeto a linguagem escolhida não se mostrar performática o suficiente para o cumprimento do cronograma, alguns *script's* que lidam com um processamento mais custoso poderão ser reescritos na linguagem de programação *Julia*, que apresenta uma sintaxe parecida com a do *Python* mas com uma performance maior por ser compilada.

4 RESULTADOS ESPERADOS

É esperado que se encontre uma combinação de parâmetros que faça o sistema desenvolvido gerar sinais de compra seguidos de sinais de venda em que a venda se deu em um patamar de preço maior que o de compra em uma frequência maior que 50%, sendo essa métrica referenciada como **assertividade**. Além disso, espera-se que a combinação de parâmetros faça a *clusterização* ter uma rentabilidade e assertividade média superiores a estratégia que opera sem clusterização, chamada de estratégia $H0$, com uma combinação equivalente de parâmetros.

5 EXECUÇÃO

Nesta seção será exposta de maneira detalhada toda a execução do projeto. Inicialmente será apresentada uma visão geral da execução, posteriormente será esclarecida e justificada ações de adaptações no conjunto de dados e redução do escopo, passando por um detalhamento das rotinas desenvolvidas e finalizando com a explanação da condução das simulações.

5.1 Visão Geral

A execução do projeto começou com a parametrização nas estruturas de armazenamento dos grafos totais de cada ativo com toda sua série disponível e também com uma parametrização de todos os grafos que poderiam ser usados, variando-se o ativo, a data inicial, a data final e a variação da janela do grafo correspondente. Tendo essas parametrizações armazenadas de maneira estática, foi criada uma função em *Python* que tem como retorno as conexões de um grafo de um determinado ativo entre uma data inicial e uma final, sem precisar recalcular as condições de visibilidade.

A próxima etapa desenvolvida foi a de calcular a similaridade entre todos os grafos equivalentes. Essa etapa se mostrou muito desafiadora no que tange a performance computacional, tendo tempos de execução que inviabilizariam o desenvolvimento do projeto mesmo depois de otimizações no código e reduções de complexidade. Conforme previsto no projeto, foi testado a troca de linguagem de programação de *Python* para *Julia* para essa etapa. Esta troca reduziu de 930 segundos para 40 segundos o tempo de cálculo das similaridades em cada data.

Tendo os grafos parametrizados e as similaridades estaticamente armazenadas, foi desenvolvida a função que, para uma combinação de parâmetros, calculava a rentabilidade da estratégia no período, assertividade de tomada de decisão e tempo de simulação. As simulações foram armazenadas em uma estrutura de dados não prevista previamente, mas que auxiliou na análise dos resultados.

A primeira estratégia adotada para fazer as simulações for realizar uma varredura das combinações de parâmetros possíveis para janelas de 180 dias aleatórias. No entanto o tempo computacional se mostrou ser um desafio novamente então foram realizadas 6500 simulações com combinações de parâmetros aleatórios, inclusive as janelas de 180 dias. Com essas simulações, foi analisado as 30 combinações de parâmetros que tiveram melhor performance em rentabilidade e assertividade de tomada de decisão. As mesmas foram testadas em diferentes janelas de 180 dias para encontrar as combinações de parâmetros que tem estatisticamente a melhor performance e que suprem minimamente os resultados esperados.

5.2 Rotinas

Neste capítulo será explicado em detalhes as particularidades da implementação de cada rotina desenvolvida.

5.2.1 Estratégia H0

A estratégia H0 foi implementada na linguagem de programação *Python* através do seguinte algoritmo:

Com isso, foi criada uma função que recebe 3 parâmetros: uma data inicial, uma data final e o tamanho da janela móvel para se calcular a variação de cada ativo em cada passo

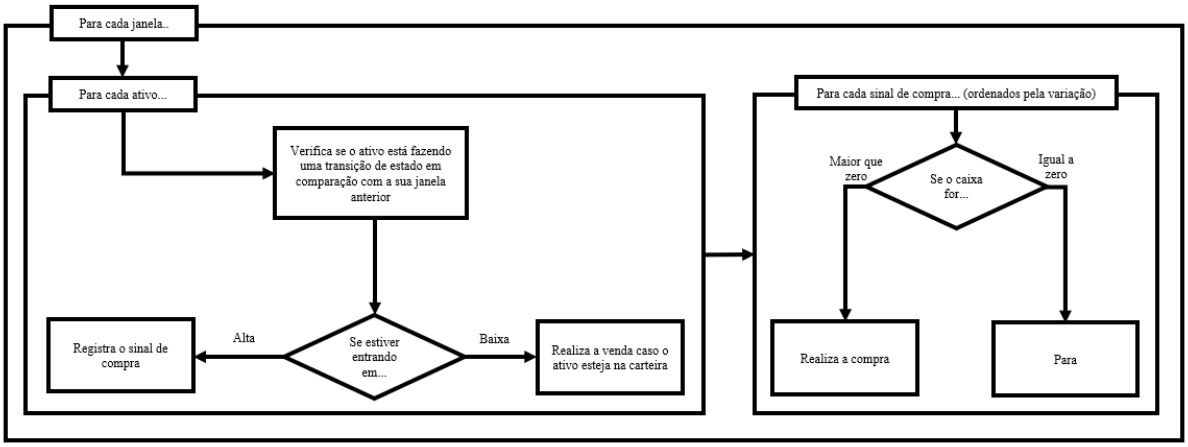

Figura 3 – Algoritmo da estratégia H0

temporal. No fim, a função retorna a quantidade de *trades* realizados pela estratégia, quantos desses foram assertivos, qual a rentabilidade final da estratégia e o tempo de simulação.

5.2.2 Geração dos Grafos

A geração dos grafos foi precedida por um estabelecimento da existência dos mesmos. No presente trabalho, a unicidade de um grafo é dada por um código de ativo, uma data inicial e uma data final, sendo a quantidade de dias entre essas datas o número de vértices do grafo. Sendo assim, foi parametrizado em um banco de dados todas as combinações destes 3 parâmetros que são de interesse utilizar no presente trabalho, i.e, todos os grafos com 5, 10, 15, 20, 25, 30, 35, 40, 45 e 50 vértices de todos os 100 ativos mais líquidos entre as datas de 04/01/2021 até 31/10/2023. Além disso, foi armazenada também a rentabilidade do ativo entre a data inicial e a data final que delimitavam o grafo.

Com isso, foi atribuída a cada combinação um *id* único por uma questão de performance do modelo. Esse *id* posteriormente foi o que referenciou cada grafo no cálculo das similaridades com os outros grafos (também referenciados por um *id*). No total foi estabelecida a existência de 800.750 grafos.

Posteriormente, foram calculadas as conexões do grafo de visibilidade gerado a partir de toda a série temporal de cada ativo. Com isso, é possível gerar individualmente cada subgrafo de interesse, simplesmente olhando para as conexões dos vértices correspondentes entre as datas desejadas (incluindo-as). Para tal foi criada uma função em *Python* que, a partir de um *id*, procura na base o ativo e as datas que delimitam o grafo, procura as conexões existentes entre as datas e retorna uma lista com todas as conexões com os nós normalizados para que o primeiro nó seja o nó “0”.

5.2.3 Calculo das Similaridades

Após a definição dos grafos, o próximo passo foi calcular as similaridades instantâneas entre todos os grafos existentes em cada intervalo de datas. Para tal, foi realizado uma varredura em cada data inicial com todos os tamanhos de janelas propostos: 5, 10, 15, 20, 25, 30, 35, 40, 45 e 50. Em cada passagem, foi definido o conjunto de combinações possíveis de ativos e foi realizado o cálculo da similaridade de cada combinação utilizando os 3 métodos: distância de edição, distância de Frobenius e coeficiente de Jaccard.

Primeiramente, foi realizada a implementação em *Python*, no entanto, cada combinação de data inicial e tamanho de janela levou em média 15 minutos para executar e, tendo 717 datas disponíveis e cerca de 10 janelas em cada data, no total a simulação levaria 74 dias para finalizar. Com isso, o cálculo das similaridades foi implementado usando a linguagem de programação Julia, o que reduziu o tempo de execução para cada combinação de data inicial e tamanho de janela para 40 segundos, totalizando um tempo computacional total de processamento de 3 dias e 6 horas.

5.2.4 Simulação

A função de simulação foi divida em duas partes: a parte responsável pela a extração de dados necessários e a parte que a partir desses dados faz o processamento da estratégia. Essa arquitetura foi adotada porque essa divisão permitiu que a parte responsável pela extração dos dados pode ter seu retorno incluído em cache, otimizando as simulações que utilizavam o mesmo conjunto de dados.

5.2.4.1 Função *get_data*

A função de extração de dados começa executando uma consulta SQL para extrair informações de fechamento de ativos para um intervalo de datas específico. Esses dados são fundamentais para calcular variações de preço dos ativos ao longo do tempo. Além disso, a função realiza uma segunda consulta para obter dados de grafos de visibilidade, que inclui a análise do estado de cada ativo (identificado como 'L' para um estado lateral, 'A' para alta ou 'B' para baixa) com base em uma variação limiar predefinida. Este passo é crucial para posteriormente identificar transições de estado, que são um componente chave na estratégia.

Após a obtenção dos dados de grafos, a função de extração de dados avança para calcular a transição de estados dos ativos. Isto é feito ao comparar o estado atual de um ativo com seu estado anterior, identificando assim uma mudança de estado. Em seguida, a função executa outra consulta SQL para extrair dados de similaridades entre os grafos, os quais são processados para ajustar os valores de similaridade com base nos estados dos ativos. A função finalmente aplica uma média móvel aos dados de similaridade, agrupando-os por pares de ativos. Este processo é vital para suavizar as flutuações de curto prazo e identificar tendências mais consistentes na similaridade entre os ativos. A seguir será exposta a implementação da função*get_data*.

Código 5 – Função de extração dos dados

```
1
2  def get_data(limiar_var, data_inicial, data_final, n, metodo,
       tamanho_media):
3
4      query = f'''SELECT data, ativo, fechamento FROM dados_diarios_b3
           WHERE data BETWEEN '{data_inicial}' AND '{data_final}';'''
5
6      dados_diarios_b3 = pd.read_sql(query, engine)
7
8      query = f'''SELECT *,
9      (IF(var >= -{limiar_var} AND var <= {limiar_var}, 'L', IF(var > {
           limiar_var}, 'A', 'B'))) AS estado
10     FROM grafos
11     WHERE dataFinal BETWEEN '{data_inicial}' AND '{data_final}' AND n
           = {n} AND incluso = 1
12     ORDER BY ativo,  dataFinal ASC;'''
13     grafos = pd.read_sql(query,engine)
14
15     grafos['transicao'] = grafos.groupby('ativo')['estado'].shift(1)
           != grafos['estado']
16     grafos['transicao'] = grafos['transicao'].apply(lambda x: 1 if x
           else 0)
17     grafos['transicao'] = grafos['transicao'].fillna(0)
18
19
20     datas = grafos.sort_values(['dataFinal'], ascending = True)['
           dataFinal'].unique()
21
22     query = f'''SELECT gA.dataFinal AS dataFinal,
23     similaridades.idGrafoA,
24     gA.ativo AS ativoA,
25     (IF(gA.var >= -{limiar_var} AND gA.var <= {limiar_var}, 'L', IF(
           gA.var > {limiar_var}, 'A', 'B'))) AS estadoA,
26     similaridades.idGrafoB,
27     gB.ativo AS ativoB,
28     (IF(gB.var >= -{limiar_var} AND gB.var <= {limiar_var}, 'L', IF(
           gB.var  > {limiar_var}, 'A', 'B'))) AS estadoB,
29     similaridades.{metodo} AS similaridade
30     FROM similaridades
31     LEFT JOIN grafos AS gA ON similaridades.idGrafoA = gA.id
32     LEFT JOIN grafos AS gB ON similaridades.idGrafoB = gB.id
```

```
33      WHERE similaridades.idGrafoA IN {tuple(grafos['id'].unique())}
34      OR similaridades.idGrafoB IN {tuple(grafos['id'].unique())};'''
35      similaridades = pd.read_sql(query, engine)
36
37      similaridades['similaridade'] = similaridades.apply(lambda x: -x[
            'similaridade'] if ((x['estadoA'] == 'B' and x['estadoB'] == '
            A') or (x['estadoA'] == 'A' and x['estadoB'] == 'B')) else x['
            similaridade'], axis = 1)
38      similaridades['similaridadeMedia'] = 0
39
40
41      for (ativoA, ativoB), grupo in similaridades.groupby(['ativoA', '
            ativoB']):
42          similaridades.loc[grupo.index, 'similaridadeMedia'] =
                media_movel_dupla(grupo, tamanho_media)
43
44      return dados_diarios_b3, grafos, datas, similaridades
```

5.2.4.2 Função *processing*

A função *processing*, por outro lado, é dedicada ao processamento e execução da estratégia de investimento com base nos dados preparados pela função de extração de dados. Inicialmente, ela define variáveis iniciais, como o caixa inicial, a carteira de ativos, e um registro de transações. Esta função também agrupa os dados de grafos e similaridades por data, permitindo que a estratégia seja aplicada sequencialmente para cada data no intervalo especificado. Dentro do *loop principal*, a função itera sobre cada data, selecionando ativos com base em sinais de compra e venda derivados da análise de estados e similaridades.

Para cada ativo, a função *processing* calcula o número de sinais de compra e venda baseando-se na quantidade de transições de estado dos ativos do *cluster*. Decisões de compra são tomadas para ativos com um saldo positivo de sinais de compra, enquanto as decisões de venda são baseadas em um saldo positivo de sinais de venda. A função também gerencia a carteira de investimentos, registrando cada transação e ajustando o caixa disponível. Por fim, ela calcula a rentabilidade da estratégia ao final do período, considerando todas as compras e vendas realizadas, e fornece uma análise de desempenho, incluindo o número total de trades, trades assertivos e a rentabilidade percentual total. A seguir será exposta a implementação da função *processing*.

Código 6 – Função de execução da estratégia

```
1
2  def processing(dados_diarios_b3, grafos, datas, qtd_sinais, sim_min,
       similaridades, tamanho_media, limiar_var, concentracao_max):
```

```
    caixa = 1
    carteira = {}
    transacoes = []
    trades = 0
    trades_assertivos = 0


    grafos_agrupados = grafos.groupby('dataFinal')
    similaridades_agrupadas = similaridades.groupby('dataFinal')

    for data in datas[tamanho_media:]:
        if data in grafos_agrupados.groups and data in
           similaridades_agrupadas.groups:
            grafos_data = grafos_agrupados.get_group(data)
            similaridades_data = similaridades_agrupadas.get_group(
               data)
            ativos_data = grafos_data['ativo'].unique()

            sinais_compra_data = []

            for ativo in ativos_data:
                correlacionados_data = similaridades_data[(
                   similaridades_data['similaridadeMedia'] >= sim_min
                   ) & ((similaridades_data['ativoA'] == ativo) | (
                   similaridades_data['ativoB'] == ativo))]
                correlacionados = set(correlacionados_data['ativoA'].
                   unique()).union(set(correlacionados_data['ativoB'
                   ].unique()))

                sinais_compra = grafos_data[grafos_data['ativo'].isin
                   (correlacionados) & (grafos_data['estado'] == 'A')
                   ]['transicao'].sum()
                sinais_venda = grafos_data[grafos_data['ativo'].isin(
                   correlacionados) & (grafos_data['estado'] == 'B')
                   ]['transicao'].sum()

                if (ativo not in carteira) and (sinais_compra -
                   sinais_venda >= qtd_sinais):
                    sinais_compra_data.append({"ativo": ativo, "qtd":
                        sinais_compra - sinais_venda})

                elif (ativo in carteira) and (sinais_venda -
```

```
                    sinais_compra >= qtd_sinais):
32                      var = dados_diarios_b3[(dados_diarios_b3['ativo']
                            == ativo) & (dados_diarios_b3['data'] == data
                        )]['fechamento'].iloc[0] / dados_diarios_b3[(
                        dados_diarios_b3['ativo'] == ativo) & (
                        dados_diarios_b3['data'] == carteira[ativo]['
                        data'])]['fechamento'].iloc[0]
33                      valor_venda = carteira[ativo]['qtd'] * var
34                      caixa += valor_venda
35
36                      transacoes.append({"tipo": "venda", "ativo":
                            ativo, "data": data, "sinais": sinais_venda -
                            sinais_compra, "valor": valor_venda, '
                            rentabilidade': round(var - 1, 4)})
37                      carteira.pop(ativo)
38
39                      trades += 1
40                      if var - 1 > limiar_var:
41                          trades_assertivos += 1
42
43              if sinais_compra_data:
44                  sinais_compra_data_df = pd.DataFrame(
                        sinais_compra_data).sort_values(by='qtd',
                        ascending=False)
45                  sinais_compra_data = sinais_compra_data_df.to_dict(
                        orient='records')
46
47                  for sinal in sinais_compra_data:
48                      if caixa > 0:
49                          valor_investimento = concentracao_max if
                                caixa - concentracao_max >= 0 else caixa
50                          transacoes.append({"tipo": "compra", "ativo":
                                sinal['ativo'], "data": data, "sinais":
                                sinal['qtd'], "valor": valor_investimento,
                                'rentabilidade': '-'})
51                          caixa -= valor_investimento
52                          carteira[sinal['ativo']] = {"data": data, "
                                qtd": valor_investimento}
53                          continue
54                      break
55
56      ultima_data = datas[-1]
```

```
57
58      for ativo in carteira:
59          var = dados_diarios_b3[(dados_diarios_b3['ativo'] == ativo) &
                (dados_diarios_b3['data'] == ultima_data)]['fechamento'].
              iloc[0] / dados_diarios_b3[(dados_diarios_b3['ativo'] ==
              ativo) & (dados_diarios_b3['data'] == carteira[ativo]['
              data'])]['fechamento'].iloc[0]
60          valor_venda = carteira[ativo]['qtd'] * var
61          caixa += valor_venda
62
63          transacoes.append({"tipo": "venda", "ativo": ativo, "data":
              data, "sinais": 0, "valor": valor_venda, 'rentabilidade':
              round(var - 1, 4)})
64
65      return transacoes, trades, trades_assertivos, round((caixa - 1),
          4)
```

5.2.5 Visualização

A visualização dos resultados se deu através de tabelas e renderizações visuais de objetos do tipo *dataframe* da biblioteca *Pandas*. O agrupamento dos ativos foi visualmente inspecionado com o uso de mapas de calor da biblioteca *Seaborn*. Todos dos *scripts* de visualização de dados foram concentrados em um único arquivo de execução na plataforma *Jupyter Notebook*.

5.3 Etapas de Simulação

Neste capítulo será comentado todas as 3 fases de simulações implementadas. No total, foram realizadas 10.221 simulações que alcançaram 274 horas 30 minutos e 6 segundos.

5.3.1 Varredura Inicial

Depois de finalizadas as implementações, foi realizada a testagem inicial de alguns parâmetros e constatação visual da performance da estratégia para certificar o acontecimento do comportamento esperado. A testagem inicial foi fundamental para realizar a correção de alguns erros de implementação e otimização computacional. No fim, foi constatado que cada simulação levaria em média 130 segundos somando a parte de extração de dados e de processamento e 86 segundos em média somente a parte de processamento com a parte de extração em cache.

No fim, foram realizadas 221 simulações e foi constatado que não haveria tempo hábil para testar todas as combinações de parâmetros para todas as janelas de 6 meses possíveis na massa de dados disponível, sendo necessária a aplicação de uma variação randômica de parâmetros para cobrir estatisticamente a maior parte das possibilidades.

5.3.2 Variação Randômica de Parâmetros

A segunda rodada de simulações foi definida pela variação randômica de parâmetros. Foram feitas 6500 combinações diferentes de intervalos de simulação, método de similaridade, tamanho da janela, tamanho da média, concentração máxima, similaridade mínima e quantidade mínima de sinais.

5.3.3 Teste das Melhores Combinações

As combinações de método de similaridade, tamanho da janela, tamanho da média, concentração máxima, similaridade mínima e quantidade mínima de sinais com maior assertividade encontradas na variação randômica da parâmetros foram expostas a pelo menos mais 30 simulações em intervalos de simulação diferentes, para garantir que a combinação não corresse o risco de ser exposta apenas a um intervalo de simulação favorável. No total foram 3000 simulações realizadas nesta etapa.

6 RESULTADOS

Depois das 10.221 simulações realizadas, foram encontradas algumas combinações de parâmetros que já atendiam o resultado esperado. Primeiramente constatou-se que a clusterização dos ativos de fato aconteceu, conforme o exemplo da imagem 4. A similaridade média dos grafos de visibilidade dos ativos gerou agrupamentos que, por mais que fossem dinâmicos ao longo do tempo, tiveram sua composição estabilizada de acordo com o tamanho da média. Na figura 5 temos a mesma figura 4 porém ressaltando a *clusterização* gerada considerando uma similaridade mínima de 0.65 entre os ativos.

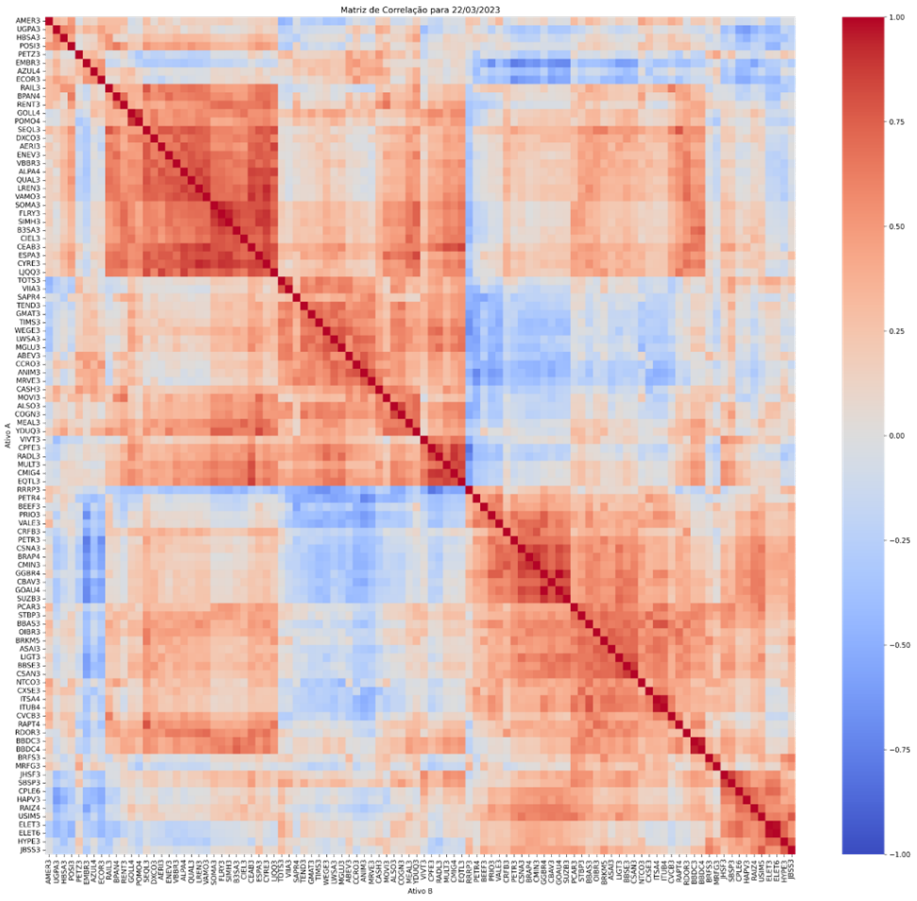

Figura 4 – Representação gráfica da *clusterização* através do mapa de calor das similaridades médias usando distância de Edição para N = 5 e tamanho média = 10 no dia 22/03/2023.

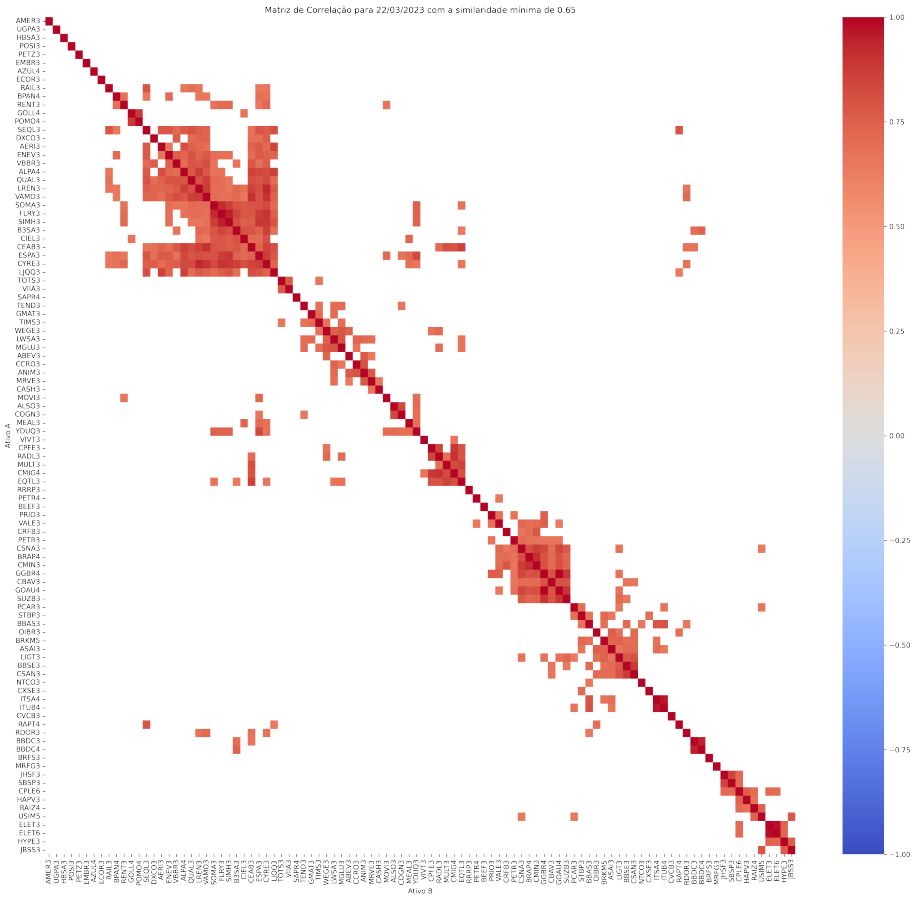

Figura 5 – Representação gráfica da *clusterização* através do mapa de calor das similaridades médias usando distância de Edição para N = 5, tamanho média = 10 e similaridade mínima de 0.65 no dia 22/03/2023.

6.1 Resultados Estratégia H0

Na tabela 5 encontram-se os resultados das simulações da estratégia H0.

n	Assertividade	Rentabilidade
5	0.316326	-0.106784
10	0.310715	-0.086488
15	0.301141	-0.035548
20	0.268809	-0.056912
25	0.256081	-0.064544

Tabela 5 – Resultados da estratégia H0 para todas as janelas móveis testadas

6.2 Combinações mais Assertivas

Na tabela 6 encontram-se os resultados das 10 simulações com maior assertividade média.

N	Método	Quantidade Mínima de Transições	Similaridade Mínima	Tamanho Média	Concentração Máxima	Percentual de Simulações com Assertividade > 0.5	Assertividade Média	Média de Trades por Simulação
5	sEdicao	3	0.90	10	0.050	0.4627	0.5712	11.25
5	sFrobenius	3	0.85	10	0.075	0.5200	0.5352	8.46
5	sEdicao	3	0.90	10	0.100	0.4400	0.5052	8.88
5	sEdicao	3	0.90	10	0.075	0.4500	0.5033	10.39
5	sJaccard	10	0.65	10	0.100	0.3500	0.4524	25.65
50	sFrobenius	5	0.65	10	0.050	0.3077	0.4516	5.56
10	sJaccard	10	0.65	5	0.050	0.4091	0.4502	15.97
5	sFrobenius	7	0.65	10	0.050	0.3827	0.4484	36.78
5	sFrobenius	7	0.65	5	0.075	0.1071	0.4438	71.04
5	sJaccard	10	0.65	15	0.050	0.3000	0.4403	24.34

Tabela 6 – Top 10 soluções mais assertivas

6.3 Combinações mais Rentáveis

Na tabela 7 encontram-se os resultados das 10 simulações com maior rentabilidade média.

N	Método	Quantidade Mínima de Transições	Similaridade Mínima	Tamanho Média	Concentração Máxima	Percentual de Simulações com Rentabilidade > 0	Rentabilidade Média	Média de Trades por Simulação
5	sEdicao	5	0.65	10	0.100	0.7692	0.0891	57.85
35	sFrobenius	3	0.65	5	0.075	0.6786	0.0517	21.14
5	sFrobenius	7	0.65	5	0.075	0.6429	0.0664	71.04
5	sEdicao	7	0.75	5	0.050	0.6207	0.0674	101.34
5	sFrobenius	7	0.65	10	0.100	0.5974	0.0345	20.73
5	sEdicao	5	0.90	5	0.100	0.5926	0.0545	11.56
5	sFrobenius	3	0.85	5	0.100	0.5862	0.0586	33.34
25	sFrobenius	3	0.65	5	0.100	0.5769	0.0579	16.08
5	sJaccard	7	0.65	10	0.075	0.5769	0.0370	53.46
5	sEdicao	7	0.75	10	0.075	0.5769	0.0658	15.04

Tabela 7 – Top 10 soluções com maior rentabilidade média

7 DISCUSSÃO

Na imagem 4 pode-se ter uma constatação visual da clusterização alcançada. A exploração da similaridade dos grafos de visibilidade de fato gerou um agrupamento não desprezível, isto é, não gerou um único *cluster* com todos os ativos ou somente *clusters* com um único ativo.

Em comparação com os resultados obtidos usando a estratégia H0, pode-se concluir que existe vantagem no uso da clusterização por similaridade de grafos de visibilidade. Enquanto para todas as configurações a assertividade da estratégia H0 não fugiu do comportamento probabilístico de $1/3$ de acertar a direção do movimento de um ativo, algumas configurações de clusterização tiveram uma assertividade média maiores que 50%.

A análise das rentabilidades médias de cada configuração foi preterida porque ela é circunstancial. A rentabilidade final dentro de um intervalo de 6 meses pode ser fortemente influenciada pelo momento do mercado, no entanto, a assertividade na geração de sinais é esperada que seja alta independente do tamanho do período testado. Em um momento de mercado não favorável para compra é esperado que o modelo gere menos sinais de compra mas, os que existirem, espera-se que os ativos estejam em um patamar de preço maior quando ocorrer o próximo sinal de venda.

Além disso, a rentabilidade total dentro de um intervalo pode ser impactada por um único trade não assertivo no meio de vários que foram assertivos. No entanto, apesar de ser menos relevante para uma análise, se houve uma combinação de parâmetros que estatisticamente além de ter uma assertividade satisfatória também teve uma rentabilidade média positiva, pode-se concluir que essa combinação além de gerar sinais assertivos os sinais são "de qualidade". Dentre as combinações de parâmetros testadas, a considerada como melhor combinação foi aquela que teve a maior assertividade.

7.1 A Melhor combinação

A melhor combinação de parâmetros encontrada foi: distância de edição, tamanho da janela de 5 dias, pelo menos 3 ativos do *cluster* fazendo uma transição para considerar a existência do sinal de negociação, similaridade média constituída a partir das 10 similaridades instantâneas passadas, similaridade mínima do *cluster* de 0.9 e alocação máxima de 5% do capital inicial em cada ativo.

Essa solução obteve em 130 simulações realizadas em intervalos distintos uma assertividade média de 57.12%, o que alcança o resultado esperado, uma rentabilidade média de 1.22% no período de 6 meses e em 55.22% das simulações a rentabilidade final foi maior que zero, sendo o maior percentual de ocorrência de rentabilidades positivas das simulações mais assertivas, o que acentua a conclusão de que tal combinação é a melhor. Outro fato que acentua a performance de tal combinação é que 3 das 4 combinações mais assertivas são praticamente iguais a menos do parâmetro de concentração máxima, que é apenas um controle de diversificação.

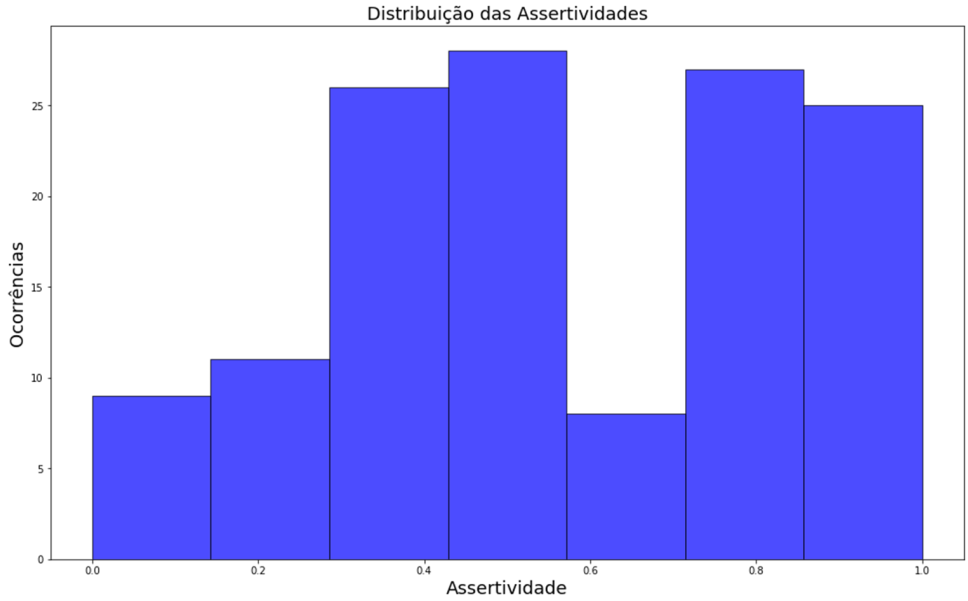

Figura 6 – Distribuição da assertividade das simulações da melhor combinação

Analisando a distribuição da assertividade das simulações da melhor combinação na figura 6 tem-se uma constatação de que existe uma concentração de ocorrências para assertividades maiores que 50%, existindo uma notável concentração para assertividades entre 30% e 60%.

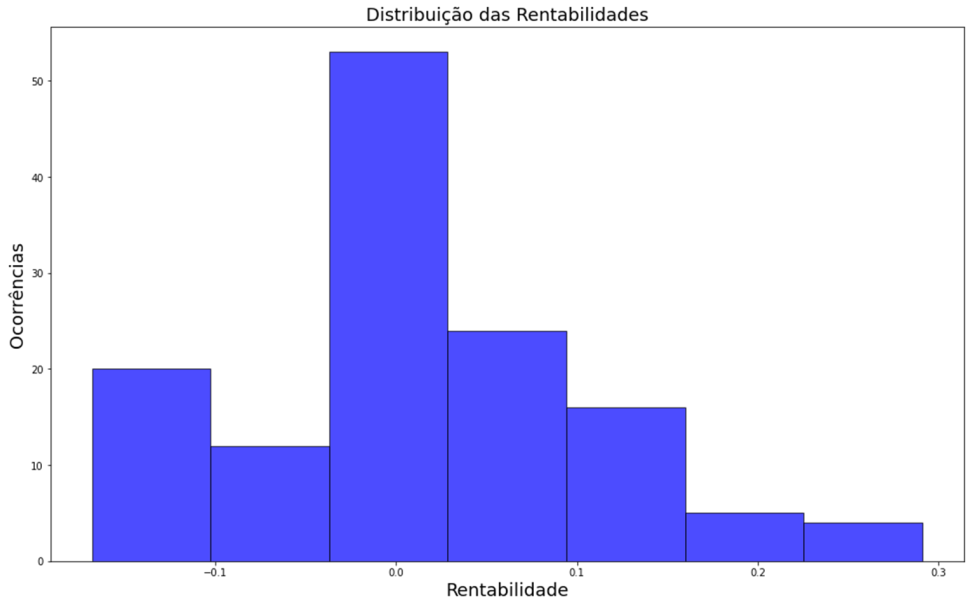

Figura 7 – Distribuição da rentabilidade das simulações da melhor combinação

Analisando a distribuição da rentabilidade das simulações na figura 7, pode-se perceber que há uma grande concentração em torno do zero, no entanto, a amplitude alcança valores positivos maiores em módulo do que os valores negativos. Com isso, pode-se concluir que é

positiva a expectativa da rentabilidade para a execução do modelo com essa combinação de parâmetros.

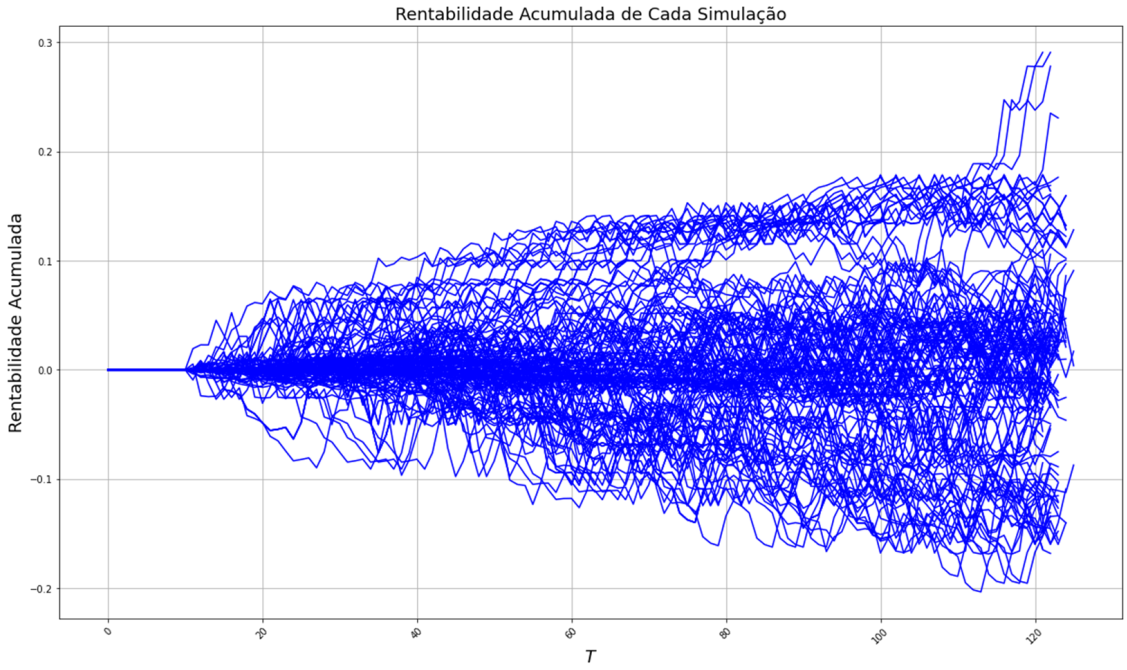

Figura 8 – Evolução das rentabilidades acumuladas das simulações da melhor combinação

A evolução das rentabilidades acumuladas das simulações da melhor combinação se encontra na figura 8. A simulação mais rentável se deu entre as datas 13/12/2022 e 11/06/2023, tendo uma rentabilidade de $+29.08\%$. Já simulação menos rentável se deu entre as datas de 22/09/2022 e 21/03/2023, tendo uma rentabilidade de -16.81%. É interessante notar que existem alguns padrões deslocados, porém, após uma análise mais minuciosa, constatou-se que são performances de intervalos de simulação próximos em que os primeiros sinais aconteceram na mesma data mas em passos diferentes de cada simulação. Nessa imagem também conseguimos ver que a estratégia teve movimentos de rentabilidade acumulada consistentes e não movimentos pontuais que tinham o impacto dissolvido na média.

8 CONCLUSÃO

Por fim, conclui-se que o trabalho em questão cumpriu com o objetivo de desenvolver um sistema que tomasse decisões de investimentos na bolsa de valores brasileira através da clusterização por similaridade média dos grafos de visibilidade da série temporal dos ativos. Foi encontrada uma combinação de parâmetros que atingiram os resultados esperados: vencer a estratégia mais fundamental chamada de H0 e ter uma assertividade de sinais de compra esperada maior que que 50%. Portanto, fica evidente que a metodologia adotada provou sua relevância, pavimentando o caminho para futuras investigações e aplicações dos grafos de visibilidade nas séries temporais financeiras.

9 TRABALHOS FUTUROS

O presente trabalho pode ser encarado com uma parametrização inicial de uma estratégia que pode sofrer sofisticações futuras. Além disso, admite-se a possibilidade de fazer alguns estudos mais aprofundados em alguns comportamentos percebidos.

- Análise do impacto das pequenas variações dos parâmetros
- Comparação da performance da estratégia utilizando outras medidas estatísticas de correlação para fazer a *clusterização*
- Testar a quantidade mínima de transições para a venda menor que a quantidade mínima de transições para compra
- Testar a possibilidade de deixar o sistema fazer vendas a descoberto

REFERÊNCIAS

[1] MANTEGA, Rosário N.; STANLEY, H. Eugene. An Introduction to Econophysics: Correlation and Complexity in Finance. The Pitt Building, Trumpington Street, Cambridge, United Kingdom: Cambridge, 2004. 162 p. ISBN 0-521-62008-2.

[2] Silvio R.A. Salinas. *Introdução à Física Estatística*. EDUSP, São Paulo, 1997.

[3] Tânia Tomé e Mario J. de Oliveira. *Dinâmica estocástica e irreversibilidade*. EDUSP, São Paulo, 2001.

[4] DRGULESCU, Adrian; YAKOVENKO, Victor M. *Statistical mechanics of money*. Eur. Phys. J. B, 4 ago. 2000.

[5] GROSS, J. L.; YELLEN, J.; ZHANG, P. *Graph Theory and Its Applications*. CRC Press, 2006.

[6] GROSS, J. L.; YELLEN, J.; ZHANG, P. (Ed.). *Handbook of Graph Theory, Second Edition*. CRC Press, 2013.

[7] WEST, D. B. *Introduction to Graph Theory*. Prentice Hall, 2001.

[8] ANGLES, R.; GUTIERREZ, C. *A survey of graph database systems*. Computing Research Repository, v. abs/1401.2808, 2014.

[9] CONTE, D.; FOGGIA, P.; SANSONE, C.; VENTO, M. *Graph Matching: Challenges and Future Directions*. Pattern Recognition Letters, v. 18, n. 8, p. 911–916, 1997.

[10] CORDELLA, L. P.; FOGGIA, P.; SANSONE, C.; VENTO, M. *A new algorithm for subgraph isomorphism*. Computer Vision and Image Understanding, v. 93, n. 2, p. 122–137, 2004.

[11] YAGNIK, J. *Graph similarity scoring and matching*. In: Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on, p. 2–7, 2005.

[12] LACASA, L.; LUQUE, B.; BALLESTEROS, F.; LUQUE, J.; NUÑO, J. C. *From time series to complex networks: The visibility graph*. Proceedings of the National Academy of Sciences, v. 105, n. 13, p. 4972-4975, 2008.

[13] YANG, Y.; WANG, J.; YANG, H.; MANG, J. *Multiscale visibility graph of financial time series*. Physica A: Statistical Mechanics and its Applications, v. 490, p. 1271-1279, 2018.

[14] QUEK, C.; SRINIVASAN, D. *Stock market prediction from financial news articles using bi-directional recurrent neural networks*. In: IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr). [s.l.]: IEEE, 2016.

[15] NIEMINEN, J. *On the centrality in a graph*. Scandinavian Journal of Psychology, v. 15, n. 1, p. 332-336, 1974.

[16] ZHANG, Y. C.; ROSSO, O. A. *Complexity-entropy causality plane: A useful approach to quantify the stock market efficiency*. Physica A: Statistical Mechanics and its Applications, v. 364, p. 507-514, 2006.