UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

MATHEUS NETO GURGEL
Orientador: Jadson Castro Gertrudes

FOSC: UMA FERRAMENTA EM PYTHON PARA VISUALIZAÇÃO DE AGRUPAMENTOS BASEADOS EM HIERARQUIAS

Ouro Preto, MG
2025

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

MATHEUS NETO GURGEL

FOSC: UMA FERRAMENTA EM PYTHON PARA VISUALIZAÇÃO DE AGRUPAMENTOS BASEADOS EM HIERARQUIAS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Jadson Castro Gertrudes

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Matheus Neto Gurgel

FOSC: Uma Ferramenta em Python para Visualização de Agrupamentos Baseados em Hierarquias

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 3 de Abril de 2025.

Membros da banca

Jadson Castro Gertrudes (Orientador) - Doutor - Universidade Federal de Ouro Preto
Rodrigo César Pedrosa Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto
Fernando Henrique Oliveira Duarte (Examinador) - Mestre - Programa de Pós Graduação em Ciência da Computação (UFOP)

Jadson Castro Gertrudes, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 3/04/2025.

Documento assinado eletronicamente por **Jadson Castro Gertrudes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 04/04/2025, às 09:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0886308** e o código CRC **7657924B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Dedico este trabalho aos meus pais: Kélcio e Simone, e homenageio meu falecido avô Honis, que deve estar recebendo essa boa notícia de algum lugar do universo...

Agradecimentos

Gostaria de agradecer a Deus, pela oportunidade e por ter iluminado meu caminho. Aos meus pais, por terem me criado e dado todo o apoio possível. À minha tia Ivone, por ter apoiado minha jornada acadêmica desde o início e a toda a minha família, por estar sempre presente e fazer parte da minha vida. A todos meus amigos, pela companhia, risadas, apoio e bons momentos, em especial TUTUZERO e VAROS. Galera do 20.2, meu período, pessoal do 21.1, e brothers do busão de Itabirito: foram várias matérias difíceis e vocês fizeram tudo se tornar mais leve, muito obrigado, de coração, todos feras demais. Ao meu orientador, Jadson, por ter me apoiado desde a IC e sempre com um sorrisão no rosto. Ao professor Tiago, por ter me ensinado e treinado em Engenharia de Software, o que contribuiu ao meu primeiro emprego e aos professores do DECOM, pelos aprendizados. Aos meus superiores de estágio, Milton, Arilton e Luiggi, por terem sido tão flexíveis e solícitos. Por fim, a Masashi Kishimoto, por ter criado a obra da minha vida e ter acendido a chama de nunca desistir, que carrego comigo até hoje.

"Se você não gosta do seu destino, não o aceite. Em vez disso, tenha a coragem de mudá-lo do jeito que você quer que ele seja."

Naruto Uzumaki

Resumo

Os algoritmos não supervisionados e semissupervisionados transferem a tarefa de rotulação para as máquinas, inibindo a necessidade de grandes conjuntos de dados rotulados. Dentro desses algoritmos, estão os algoritmos hierárquicos, que produzem uma hierarquia de grupos, que pode ser visualizada por meio de uma árvore chamada dendrograma. A extração desses grupos é feita de maneira convencional a partir de um corte horizontal no dendrograma. No entanto, esse método não representa todas as possibilidades de formação desses grupos. O trabalho de Campello et al. (2013) – FOSC – contorna isso por meio de uma extração ótima baseada em uma métrica de estabilidade. Para visualizar esses tipos de dados, em grandes conjuntos, o dendrograma não é suficiente, pois fica ilegível. Dessa forma, este trabalho tem como objetivo desenvolver uma ferramenta de visualização para exploração desses tipos de dados, em cima da extração ótima proporcionada pelo FOSC, que contenha gráficos adequados a grandes conjuntos de dados - Silhouette-like e Alcançabilidade. Para isso, é gerado um pacote em Python, por meio de técnicas de Programação Orientada a Objetos e Engenharia de Software, e disponibilizado à comunidade, juntamente com uma documentação para consulta. Em conclusão, o *framework* conseguiu gerar gráficos em um grande conjunto de dados: Handl e Knowles (2007).

Palavras-chave: Aprendizado de máquina. Aprendizado não supervisionado. Aprendizado semissupervisionado. Agrupamento de dados. Visualização de dados.

Abstract

Unsupervised and semi-supervised algorithms transfer the labeling task to machines, eliminating the need for large labeled data sets. These algorithms include hierarchical algorithms, which produce a hierarchy of groups that can be visualized using a tree called a dendrogram. These groups are extracted in a conventional manner by cutting horizontally in the dendrogram. However, this method does not represent all the possibilities for forming these groups. The work of Campello et al. (2013) – FOSC – overcomes this by using optimal extraction based on a stability metric. Visualizing these types of data in large sets with a dendrogram is not adequate, as it becomes illegible. Thus, this work aims to develop a visualization tool for exploring these types of data, leveraging the optimal extraction provided by FOSC. The tool includes graphs designed for large datasets, such as the Silhouette-like Plot and the Reachability Plot. To achieve this, a Python package is developed using Object-Oriented Programming and Software Engineering principles and made available to the community, along with comprehensive documentation for reference. In conclusion, the framework was able to generate graphs in a large data set: Handl e Knowles (2007).

Keywords: Machine learning. Unsupervised learning. Semi-supervised learning. Data clustering. Data visualization.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução e Justificativa

Atualmente, a Inteligência Artificial (IA) vem se tornando uma ferramenta extremamente útil para a sociedade. Ela é empregada em diversas áreas, desde a detecção facial em escolas até o desenvolvimento de veículos autônomos (YADAV; YADAV; AGRAWAL, 2022). Além disso, em setores como fabricação, logística, finanças e saúde, a IA facilita a automação de operações e o aprimoramento na tomada de decisões (JHA et al., 2023). Dentro do campo da Inteligência Artificial, encontra-se o Aprendizado de Máquina, que permite aos humanos treinarem computadores para aprenderem conhecimentos específicos e, com base nesses aprendizados, realizarem tarefas em contextos não previamente vistos.

Os métodos de aprendizado de máquina são categorizados em três tipos principais: Aprendizado Supervisionado, Semissupervisionado e Não Supervisionado. O primeiro tipo depende de uma ampla quantidade de dados rotulados para gerar resultados eficazes. Nesse caso, os dados são divididos em conjuntos de treinamento e teste, que servem como entrada para o algoritmo que, por meio de cálculos matemáticos, produz um modelo preditivo. Entretanto, a obtenção de dados rotulados é um processo custoso (ABDEL-HAKIM; DEABES, 2020), que exige anotações manuais por especialistas ou o uso de dispositivos avançados para sua geração (ZHU; GOLDBERG, 2009). Portanto, devido ao alto custo associado à rotulação de dados, o uso do aprendizado supervisionado para tarefas específicas frequentemente se torna inviável.

Em contrapartida, os algoritmos não supervisionados apresentam a vantagem de eliminar completamente a necessidade de dados rotulados, transferindo a tarefa de rotulação para as máquinas. Nesse tipo de aprendizado, alguns algoritmos inferem padrões na estrutura dos dados sem rótulos prévios. Consequentemente, dados semelhantes são agrupados em *clusters* (ENGELEN; HOOS, 2020), permitindo a descoberta de informações sobre os padrões de dados que seriam difíceis de observar manualmente (PARASHAR; GULATI, 2012). Os algoritmos de agrupamento podem ser hierárquicos ou particionais. Nos hierárquicos, os *clusters* são extraídos de uma hierarquia, geralmente representada por uma árvore denominada dendrograma (CAMPELLO et al., 2013).

A extração de grupos nos algoritmos hierárquicos geralmente ocorre por meio de um corte horizontal no dendrograma. No entanto, essa abordagem é limitada, pois não permite que dados em diferentes níveis da árvore sejam agrupados no mesmo *cluster*. Assim, o *framework* proposto por Campello et al. (2013), denominado FOSC (do inglês: *framework* para extração ótima de *clusters*), possibilita a extração ótima de *clusters*, permitindo agrupar objetos de diferentes níveis hierárquicos. O FOSC também é compatível com algoritmos semissupervisionados, que combinam características dos métodos supervisionados e não supervisionados. Esses algoritmos emergem em duas formas principais: uma baseada no aprendizado supervisionado e outra que

se assemelha a uma extensão do aprendizado não supervisionado, recebendo um conjunto de restrições que definem como os objetos devem ser agrupados (ZHU; GOLDBERG, 2009), sendo essa última abordagem coberta pelo FOSC.

Contudo, os resultados obtidos por métodos não supervisionados e semissupervisionados devem ser interpretados por humanos para extrair significados úteis dos agrupamentos gerados pelos algoritmos. Portanto, surge a necessidade de desenvolver ferramentas que auxiliem na interpretabilidade e exploração desses dados, o que motiva a criação de novas técnicas de visualização.

Para conjuntos de dados menores, os dendrogramas são eficazes; no entanto, para conjuntos maiores, a interpretabilidade fica extremamente comprometida (SANDER et al., 2003). Assim, foram propostas outras formas de visualização, como os gráficos de silhueta e de alcançabilidade (CAMPELLO et al., 2013). Portanto, desenvolver um *framework* que integre esses métodos de visualização seria extremamente valioso para a comunidade científica. Além disso, estender o FOSC para incorporar essas visualizações poderia representar um avanço no estado da arte.

1.1 Objetivos

O objetivo deste trabalho é desenvolver uma ferramenta ou *framework* para visualização de agrupamentos (*clusters*), com o intuito de facilitar a interpretação e exploração dos resultados gerados pelo FOSC. Para isso, serão integrados métodos de visualização, como gráficos de silhueta e gráficos de alcançabilidade. A fim de atingir o objetivo geral, os seguintes objetivos específicos serão estabelecidos:

- **Integrar técnicas de visualização:** Incorporar os métodos de visualização, como o gráfico de silhueta e o gráfico de alcançabilidade, em um único *framework*.
- **Extender do FOSC**: Adaptar o FOSC para suportar visualizações que facilitem a interpretação de resultados em algoritmos semissupervisionados.
- **Validar e aplicação prática:** Validar a ferramenta em conjuntos de dados simulados.
- **Disponibilizar a ferramenta** como um recurso de código aberto ou acessível à comunidade científica.

1.2 Estrutura da Monografia

O restante do trabalho é estruturado da seguinte forma: o Capítulo 2 contextualiza o tema, trazendo conceitos importantes para o entendimento do trabalho. O Capítulo 3 traz os trabalhos relacionados. No Capítulo 4, são apresentados a metodologia adotada para solucionar o problema, e a formulação dos experimentos a serem realizados. O Capítulo 5 mostra os resultados obtidos

por meio da ferramenta e, por fim, o Capítulo 6 apresenta as discussões, considerações e trabalhos futuros.

2 Fundamentos

2.1 Agrupamento de dados

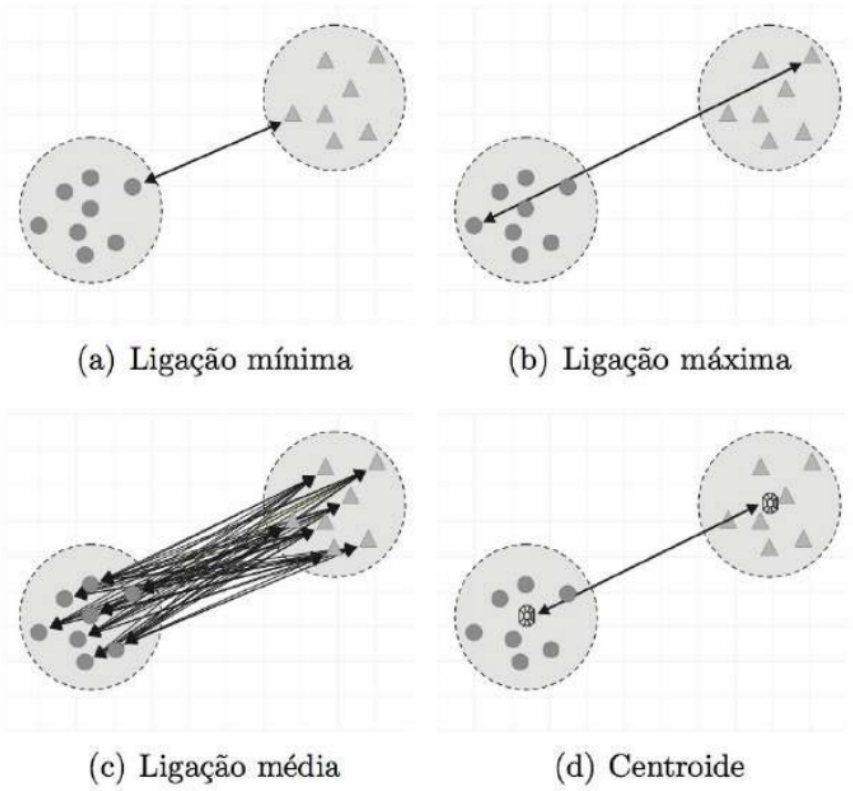
O agrupamento de dados, também conhecido no meio acadêmico como clusterização (*clustering*), é uma das principais técnicas em Aprendizado de Máquina e Mineração de Dados (GERTRUDES, 2019). Essa tarefa consiste em agrupar objetos de um conjunto de dados de modo que entidades semelhantes, de acordo com critérios específicos, sejam alocadas no mesmo grupo (FACELI et al., 2021). Os algoritmos de agrupamento produzem uma ou mais estruturas chamadas partições, que representam os agrupamentos formados.

As partições resultantes podem ser exclusivas, onde cada objeto pertence a apenas um *cluster*, ou não exclusivas, permitindo que um objeto faça parte de múltiplos *clusters* simultaneamente (JAIN; DUBES, 1988). Os algoritmos que geram partições exclusivas são classificados de acordo com a estrutura dos *clusters* produzidos. Existem dois tipos principais de algoritmos de agrupamento exclusivo: particionais e hierárquicos. Os algoritmos particionais baseiam-se na otimização de uma função objetivo que mede a qualidade dos *clusters* formados, resultando em uma partição única e definitiva (JAIN; DUBES, 1988).

Por outro lado, os algoritmos hierárquicos geram uma sequência de partições aninhadas. Esses algoritmos podem ser divididos em dois grupos: divisivos e aglomerativos. Nos métodos aglomerativos, cada objeto é inicialmente tratado como um *cluster* individual. Em seguida, uma série de iterações é realizada para agrupar objetos semelhantes e combinar os *clusters* existentes, resultando em um único *cluster* que contém todos os objetos. Em contraste, os algoritmos divisivos começam com todos os objetos em um único *cluster*, que é progressivamente dividido em *clusters* menores. A cada iteração, o número de *clusters* é reduzido até que cada um contenha apenas um objeto (JAIN; DUBES, 1988).

Os algoritmos aglomerativos utilizam uma matriz de proximidade entre os *clusters*, calculada com base em métricas de ligação específicas. Entre as principais métricas de ligação, destacam-se: a distância mínima (*single-linkage*), que considera a distância entre os objetos mais próximos dos *clusters*; a distância máxima (*complete-linkage*), que utiliza a distância entre os objetos mais distantes; a distância média (*average-linkage*), que é a média das distâncias entre todos os pares de objetos de dois *clusters*; e a métrica de centroide, calculada a partir das distâncias entre dois objetos fictícios que representam a média dos objetos em cada *cluster* (JAIN; DUBES, 1988). A Figura 2.1 ilustra visualmente essas métricas de ligação.

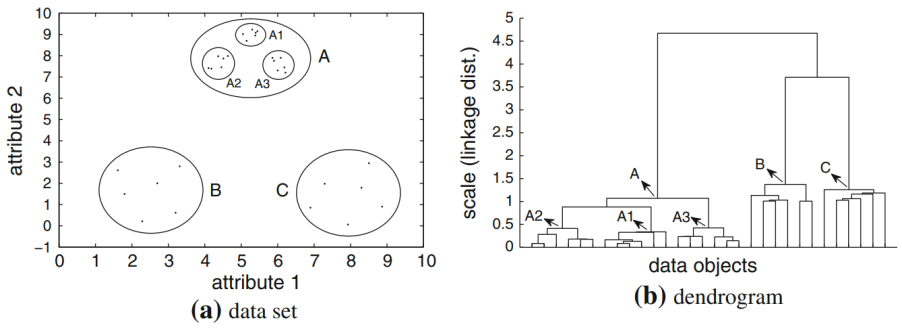
Figura 2.1 – Métricas de ligação.



Fonte: Faceli et al. (2021).

A partir de uma matriz de proximidade, os algoritmos geram uma série de partições aninhadas, representadas por um dendrograma. O resultado final dos algoritmos é obtido realizando um corte horizontal neste dendrograma, para extrair os *clusters* desejados. Contudo, essa abordagem apresenta limitações, pois não contempla todas as possibilidades de agrupamento. Um dos principais problemas é a dificuldade de capturar *clusters* de diferentes densidades em uma única partição, como ilustrado na Figura 2.2, onde um corte horizontal poderia não ser capaz de isolar os *clusters* A1, A2, A3, B e C em um conjunto de dados com variabilidade de densidades.

Figura 2.2 – Exemplo de grupos de diferentes níveis em uma hierarquia de dados.



Fonte: Campello et al. (2013).

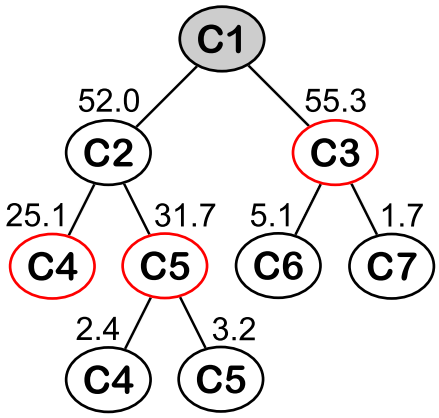
2.2 Framework para Extração Ótima de Clusters (FOSC)

Desenvolvido por Campello et al. (2013), o FOSC (*Framework for Optimal Extraction of Clusters*) é projetado para a extração eficiente de *clusters* de hierarquias em ambientes de aprendizado de máquina não supervisionados e semissupervisionados.

Este *framework* emprega uma estrutura de árvore para modelar as hierarquias de *clusters*, o que facilita tanto a manipulação quanto a conversão para outras representações. Uma forma sugerida por Campello et al. (2013) para obtenção da árvore de *clusters* é por meio da simplificação do dendrograma. Por meio de uma abordagem *top-down*, é possível verificar se as divisões no dendrograma separam os *clusters* em *clusters* significativos ou não. Para isso, utiliza-se de um parâmetro m_{ClSize} que auxilia na simplificação do dendrograma, indicando se a divisão gerou novos *clusters* ou se houve um encolhimento do *cluster* pai. O encolhimento de um *cluster* significa que os objetos pertencentes ao **cluster** pai estão se tornando ruído a partir daquela divisão.

A partir de $\mathbf{P} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$, que representa a hierarquia de *clusters* a ser analisada, onde inicialmente $\mathbf{C}_1 = \mathbf{X}$ (raiz da árvore), e $\mathbf{C}_{i_L}$ e $\mathbf{C}_{i_R}$ são os filhos esquerdo e direito do nó pai, respectivamente[1], a extração de *clusters* é formulada como um problema de otimização. Cada nó da árvore recebe um valor numérico, que é utilizado para maximizar uma função objetivo, como a soma desses valores, considerando a relação hierárquica e a qualidade das partições.

Figura 2.3 – Extração ótima de grupos a partir de uma hierarquia de grupos.



Fonte: Elaborada pelo autor.

A Figura 2.3 ilustra esse processo. Por exemplo, o *cluster* $\mathbf{C}_3$ é selecionado por possuir um valor superior à soma dos valores de suas subárvores $\mathbf{C}_6$ e $\mathbf{C}_7$. De modo semelhante, $\mathbf{C}_4$ e $\mathbf{C}_5$ são escolhidos porque a soma de seus valores (56,8) excede o valor do nó pai $\mathbf{C}_2$ (52,0). Esse é um problema recursivo, para o qual é proposta uma abordagem baseada em programação dinâmica, começando das folhas e progredindo até o nó raiz, a fim de encontrar a solução ótima.

[1] Cada nó $\mathbf{C}_i$ pode se dividir em dois ou mais subclusters, sem afetar o processo de extração. No caso do dendrograma, essa subdivisão ocorre de forma binária devido ao processo aglomerativo.

Para que a extração dos grupos ocorra por meio do processo descrito anteriormente, é necessária a definição de uma medida de qualidade que seja ***aditiva*** (ou seja, o resultado da extração ótima é dado pela soma dos valores dos *clusters* individuais) e ***local*** (cada medida deve ser computada de forma isolada para cada *cluster*).

Existem diversas medidas propostas para a extração ótima de grupos, tanto em contextos semissupervisionados quanto não supervisionados. Para extração semissupervisionada no *framework* FOSC, são implementadas restrições do tipo "é sugerido agrupar" ou "não é sugerido agrupar" (*should link* e *should not link*, respectivamente) (CAMPELLO et al., 2013), que atuam como diretrizes para o cálculo da medida de qualidade. Outra abordagem, proposta por Gertrudes (2019), utiliza informações sobre a distribuição de classes dentro de um grupo para definir a medida de qualidade de um *cluster* com base em rótulos.

Já nos algoritmos não supervisionados, o principal critério avaliado é a estabilidade de cada *cluster*, calculada pela diferença entre o nível em que um objeto entra e o nível em que ele deixa o *cluster*, somando essas diferenças para todos os objetos do grupo. Quanto maior a estabilidade, melhor avaliado é o *cluster*. Além dela, destaca-se a abordagem proposta por Anjos et al. (2018) que realiza o cálculo da medida de qualidade por meio da análise das relações de vizinhança dos objetos dentro de um grupo, utilizando a *Modularidade Q*. Essas medidas complementam as técnicas tradicionais, ampliando as possibilidades de análise e interpretação dos agrupamentos.

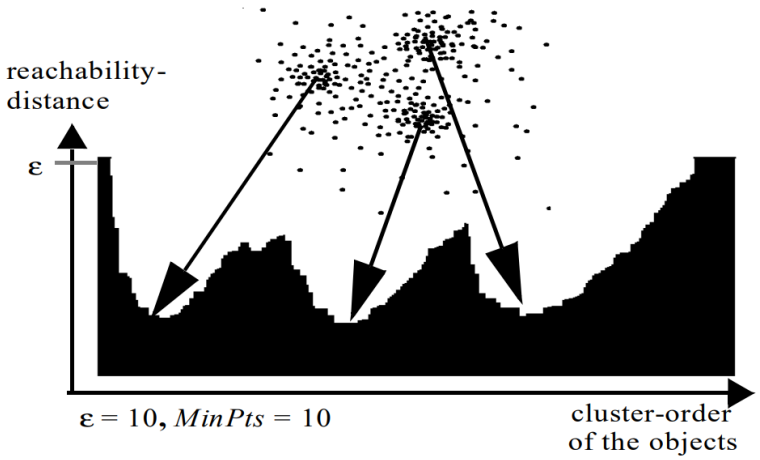
2.3 Visualização de agrupamentos

2.3.1 Gráfico de Alcançabilidade

Ankerst et al. (1999) introduziu uma nova forma de visualização de agrupamentos para conjuntos de dados extensos: o gráfico de alcançabilidade[2]. Essa técnica consiste em um gráfico de barras no qual a altura de cada barra representa a distância de alcançabilidade de um objeto em relação ao seu vizinho mais próximo. Valores mais baixos indicam objetos próximos entre si, enquanto valores mais altos refletem maior distância. Nessa representação, os vales do gráfico correspondem a agrupamentos na hierarquia, pois indicam regiões de alta densidade, enquanto os picos representam separações entre *clusters*, sinalizando dissimilaridade entre grupos. A Figura 2.4 ilustra esse conceito, exibindo três *clusters* principais identificados pela análise dos vales e picos do gráfico.

[2] *Reachability Plot*

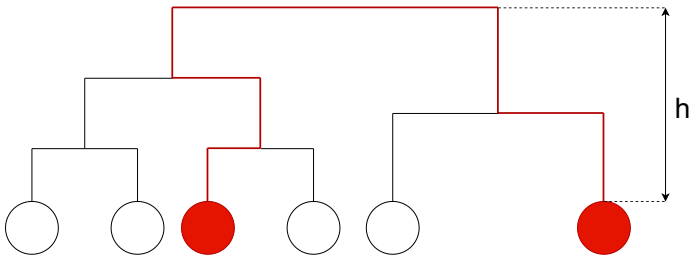
Figura 2.4 – Gráfico de alcançabilidade.



Fonte: Ankerst et al. (1999).

O trabalho de Sander et al. (2003) demonstrou a equivalência entre o gráfico de alcançabilidade (*Reachability Plot*) e o dendrograma, permitindo a conversão entre essas duas representações. A conversão do dendrograma para o gráfico de alcançabilidade requer a compreensão de que, para quaisquer dois nós no dendrograma, a altura do ancestral comum mais próximo entre eles corresponde à medida de alcançabilidade. Essa relação é ilustrada na Figura 2.5, que mostra como a estrutura hierárquica do dendrograma pode ser mapeada diretamente para os padrões de vales e picos do gráfico de alcançabilidade.

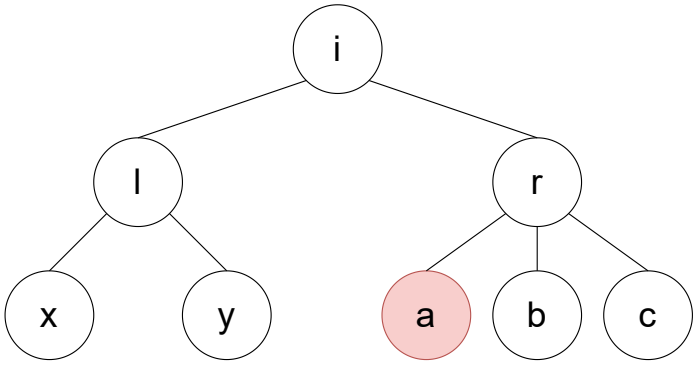
Figura 2.5 – Altura do menor ancestral em comum.



Fonte: elaborada pelo autor.

Além disso, a conversão do dendrograma em um *reachability plot* válido requer a aplicação da seguinte regra. Considere um nó interno i, com filhos esquerdo (l) e direito (r). Ao analisar a subárvore do nó r, o nó mais próximo ao nó l será aquele posicionado mais à esquerda nessa subárvore. Na Figura 2.6, por exemplo, os nós i, l e r são representados esquematicamente, sendo o nó a identificado como o nó da subárvore de r mais próximo a l. Essa condição garante a preservação da relação hierárquica durante a conversão entre as representações.

Figura 2.6 – Condição de alcançabilidade.



Fonte: elaborada pelo autor.

Dessa forma, o *reachability plot* pode ser obtido por meio do seguinte algoritmo: seja $L = \{l_1, l_2, \ldots, l_n\}$ o conjunto de folhas do dendrograma, e $d[l_i]$ o valor de alcançabilidade associado à folha l_i. Inicialmente, o valor de alcançabilidade da primeira folha ($d[l_1]$) é definido como infinito (∞). Em seguida, para cada folha $l_i \in L$ (começando de l_2), calcula-se $d[l_i]$ como a altura do ancestral comum mais próximo entre l_i e a folha anterior l_{i-1}. Esse processo é repetido iterativamente até que todos os valores de alcançabilidade sejam computados.

2.3.2 Gráfico de Silhueta

O gráfico de silhueta, proposto por Rousseeuw (1987), foi desenvolvido para aprimorar a visualização dos resultados de algoritmos de agrupamento. Essa técnica permite avaliar a qualidade dos objetos pertencentes aos seus respectivos *clusters* por meio de um coeficiente denominado $s[i]$, calculado da seguinte forma:

- Para cada objeto pertencente a um *cluster*, calcula-se a distância média ($a(i)$) entre esse objeto e todos os demais elementos do mesmo *cluster*. Em contextos euclidianos, essa medida corresponde à dissimilaridade intra-cluster.
- Em seguida, calcula-se a distância média entre o objeto e os elementos de cada um dos outros *clusters*, selecionando a menor dessas distâncias ($b(i)$), que representa a dissimilaridade inter-cluster mais próxima.

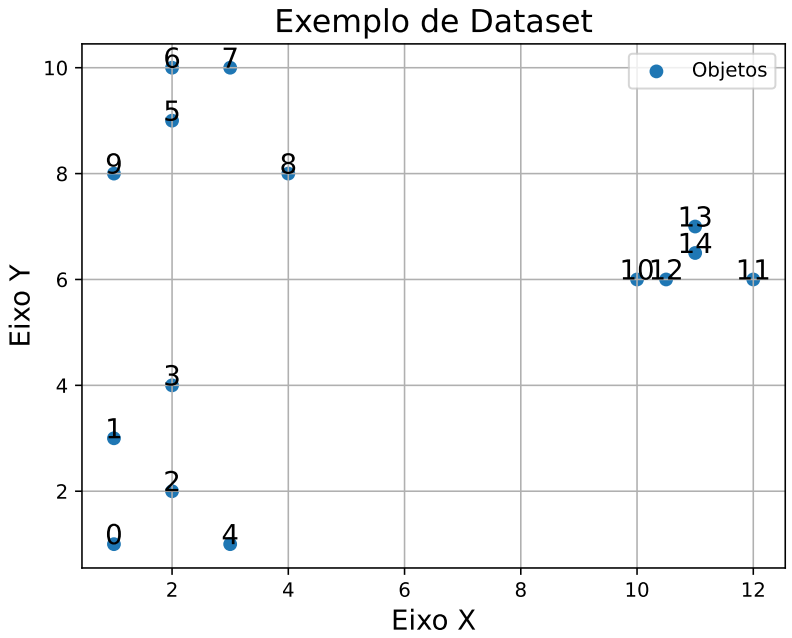
O coeficiente de silhueta para o objeto i é então obtido pela Equação 2.1:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2.1}$$

Este coeficiente varia no intervalo $[-1, 1]$, sendo que valores mais próximos de 1 indicam que o objeto i está bem alocado em seu *cluster*, enquanto valores próximos a -1 sugerem uma

possível má alocação. Para ilustrar a utilidade do coeficiente de silhueta, considere o conjunto de dados exemplar apresentado na Figura 2.7.

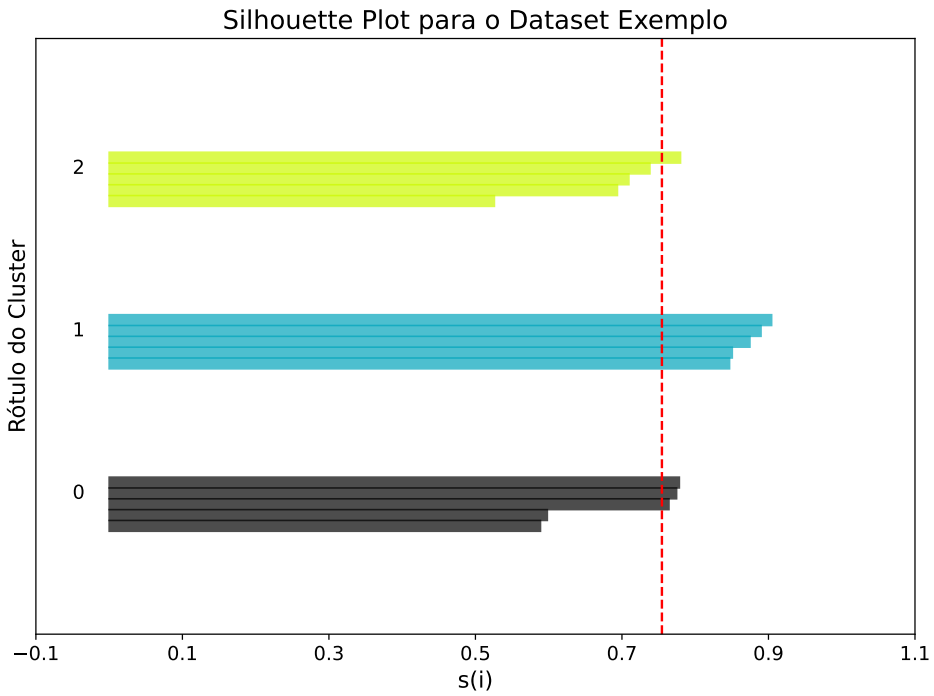
Figura 2.7 – Dataset de exemplo.



Fonte: Elaborada pelo autor.

A partir desse conjunto de dados, pode-se notar três clusters promissores: $\{0, 1, 2, 3, 4\}$, $\{5, 6, 7, 8, 9\}$ e $\{10, 11, 12, 13, 14\}$. Dessa forma, o gráfico de silhueta da Figura 2.8 pode ser obtido:

Figura 2.8 – Gráfico de Silhueta para o Exemplo da Figura 2.7.



Fonte: Elaborada pelo autor.

O gráfico de silhueta, a partir de seus coeficientes, mostram quais objetos são mais ou menos parecidos com seus respectivos clusters. Nesse sentido, as barras de baixo denotam coeficientes menores e, similarmente, objetos mais afastados de seus clusters. Por outro lado, os objetos com maiores coeficientes são aqueles representam melhor seus agrupamentos.

2.3.2.1 Adaptação

O FOSC (CAMPELLO et al., 2013) produz uma hierarquia de dados onde objetos classificados como ruído são atribuídos o valor zero. Essa estrutura hierárquica pode ser representada visualmente através de uma matriz, como demonstrado na Figura 2.9. Essa representação matricial deriva do processo de simplificação do dendrograma original, que incorpora um parâmetro m_{ClSize} (tamanho mínimo do cluster).

O parâmetro m_{ClSize} estabelece o número mínimo de objetos necessários para que um cluster seja considerado válido. Quando um cluster contém menos objetos que m_{ClSize}, isso indica as seguintes situações:

- O cluster pai não sofreu uma subdivisão legítima;
- Ocorreu uma perda de objetos durante sua formação;
- Os objetos restantes podem ser tratados como ruído ou elementos periféricos;

Essa abordagem permite uma distinção clara entre agrupamentos significativos e artefatos do processo de clusterização, facilitando a interpretação da estrutura hierárquica resultante.

Figura 2.9 – Matriz de Hierarquia para o Exemplo da Figura 2.7 com $m_{ClSize} = 2$.

	20.5	11.8	6.4	4.6	4.5	3.9	3.7	3.7	3.5	3.0	2.7	2.1	1.8	1.1	0.0
x1	1	3	4	4	0	0	0	0	0	0	0	0	0	0	0
x2	1	3	4	4	4	4	4	4	4	0	0	0	0	0	0
x3	1	3	4	4	4	4	4	4	4	0	0	0	0	0	0
x4	1	3	4	4	4	0	0	0	0	0	0	0	0	0	0
x5	1	3	4	4	4	4	4	0	0	0	0	0	0	0	0
x6	1	3	5	5	5	5	5	5	5	5	0	0	0	0	0
x7	1	3	5	5	5	5	5	5	5	5	5	0	0	0	0
x8	1	3	5	5	5	5	5	5	5	5	5	0	0	0	0
x9	1	3	5	0	0	0	0	0	0	0	0	0	0	0	0
x10	1	3	5	5	5	5	0	0	0	0	0	0	0	0	0
x11	1	2	2	2	2	2	2	2	2	2	2	2	7	0	0
x12	1	2	2	2	2	2	2	2	0	0	0	0	0	0	0
x13	1	2	2	2	2	2	2	2	2	2	2	2	7	0	0
x14	1	2	2	2	2	2	2	2	2	2	2	2	6	6	0
x15	1	2	2	2	2	2	2	2	2	2	2	2	6	6	0

Fonte: Elaborada pelo autor.

Na Figura 2.9, as linhas representam os objetos, enquanto as colunas correspondem às distâncias nas quais ocorrem mudanças na hierarquia, ou seja, onde novos *clusters* são formados. Assim, os valores das linhas indicam os diferentes agrupamentos aos quais um objeto pertence ao longo da construção da hierarquia.

A partir dessa matriz, é possível gerar um gráfico semelhante ao de silhueta, denominado *silhouette-like plot*, no qual os agrupamentos são representados como "dentes" que perdem elementos gradualmente. Para isso, as colunas da matriz são reordenadas de modo que objetos semelhantes fiquem adjacentes. Essa reordenação é realizada por meio de uma ordenação lexicográfica, proposta por Gupta, Liu e Ghosh (2008), que trata cada coluna como uma *string* lida de baixo para cima. Além disso, cada agrupamento é colorido de forma que *clusters* vizinhos no gráfico recebam cores distintas. O resultado final é o gráfico de silhueta, conforme exemplificado na Figura 2.10:

Figura 2.10 – Matriz apresentada na Figura 2.9 após coloração, reordenação e transposição.

	x12	x14	x15	x11	x13	x1	x4	x5	x2	x3	x9	x10	x6	x7	x8
0.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.1	0	6	6	0	0	0	0	0	0	0	0	0	0	0	0
1.8	0	6	6	7	7	0	0	0	0	0	0	0	0	0	0
2.1	0	2	2	2	2	0	0	0	0	0	0	0	0	0	0
2.7	0	2	2	2	2	0	0	0	0	0	0	0	0	5	5
3.0	0	2	2	2	2	0	0	0	0	0	0	0	5	5	5
3.5	0	2	2	2	2	0	0	0	4	4	0	0	5	5	5
3.7	2	2	2	2	2	0	0	0	4	4	0	0	5	5	5
3.7	2	2	2	2	2	0	0	4	4	4	0	0	5	5	5
3.9	2	2	2	2	2	0	0	4	4	4	0	5	5	5	5
4.5	2	2	2	2	2	0	4	4	4	4	0	5	5	5	5
4.6	2	2	2	2	2	4	4	4	4	4	0	5	5	5	5
6.4	2	2	2	2	2	4	4	4	4	4	5	5	5	5	5
11.8	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3
20.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fonte: Elaborada pelo autor.

3 Revisão bibliográfica

A análise e visualização de grandes conjuntos de dados biológicos têm sido um desafio significativo na área de bioinformática. Para abordar esse problema, Gupta, Liu e Ghosh (2008) propuseram um *framework* robusto para clusterização automática e visualização de dados biológicos em larga escala. O *framework* utiliza algoritmos baseados em densidade para a seleção de *clusters* e ordena os agrupamentos com base em um critério de estabilidade. Para a visualização, foi implementado um software interativo que fornece o gráfico *silhouette-like*, embora não inclua o gráfico de alcançabilidade. A implementação foi realizada em Java, utilizando o *framework* Spring, e a ferramenta foi nomeada Gene-Diver. Entre suas funcionalidades, destacam-se a complexidade de memória $O(n)$, a capacidade de reutilizar resultados de agrupamentos anteriores, a clusterização com base em diferentes distâncias, a filtragem de dados de baixa densidade, o uso de matrizes de distâncias fornecidas pelo usuário, a exploração de genes individuais usando o BioGRID e a capacidade de dar zoom em *clusters* específicos. Dessa forma, o Gene-Diver configura-se como uma solução promissora para clusterização interativa em computadores de recursos modestos. No entanto, atualmente, a ferramenta não está mais disponível para uso. O link do produto disponibilizado no artigo não leva a um website e, utilizando tanto buscadores baseados em indexação, quanto baseados em inteligência artifical, não foram encontradas evidências de implementações na internet.

No contexto de métodos hierárquicos baseados em densidade, Campello et al. (2015) propuseram o HDBSCAN*, um *framework* integrado para análise de *clusters*, detecção de *outliers* e visualização de dados. Essa ferramenta inclui tanto o gráfico *silhouette-like* quanto o gráfico de alcançabilidade. No entanto, detalhes sobre sua implementação e disponibilidade não são fornecidos no artigo. Baseando-se no HDBSCAN*, Neto (2015) desenvolveu um *framework* de visualização para algoritmos semissupervisionados de agrupamento. O software permite gerar gráficos que auxiliam na compreensão de hierarquias de *clusters* obtidas pelo HDBSCAN*, além de oferecer funcionalidades como busca de objetos por ID, obtenção de informações detalhadas (rótulos, descrição, ID), criação de cortes horizontais para extração de grupos e coloração baseada em rótulos. O sistema foi implementado em Java, com uma estratégia eficiente para armazenamento da matriz de hierarquia, e integra três algoritmos: DBSCAN*, SSDBSCAN e Hierarchical Sampling. Infelizmente, assim como o Gene-Diver, essa ferramenta também não está disponível ao público. O texto não cita referências a uma disponibilização da ferramenta e ela não foi encontrada na internet.

Além dessas ferramentas especializadas, é importante destacar implementações disponíveis em bibliotecas amplamente utilizadas para algoritmos e visualização de dados. A biblioteca Scikit-learn, desenvolvida em Python, inclui diversos algoritmos de clusterização, como HDBSCAN, OPTICS (ANKERST et al., 1999) e DBSCAN (ESTER et al., 1996). A implementação

do OPTICS possui um parâmetro `reachability_`, que armazena uma lista de alcançabilidade por objeto, mas não oferece métodos prontos para gerar o gráfico de alcançabilidade. Da mesma forma, as implementações de HDBSCAN e DBSCAN também não incluem essa funcionalidade. Embora a Scikit-learn permita a visualização do gráfico de silhueta original, não é possível gerar o gráfico *silhouette-like* para explorar hierarquias de dados. Além disso, a biblioteca não possui uma implementação do FOSC. Por fim, foram exploradas ferramentas gerais de visualização de dados, como Matplotlib, Seaborn, Plotly, Bokeh e Matlab. Nesse contexto, não foram encontradas soluções que permitam gerar automaticamente os gráficos *silhouette-like* e de alcançabilidade. Ou seja, não há funções prontas que gerem esses gráficos a partir de parâmetros simples.

Tabela 3.1 – Comparação de algoritmos e bibliotecas.

Nome	**AutoHDS**	**HDBSCAN***	**Diferentes bibliotecas**	**Este trabalho**
Tipo de algoritmo	Baseados em Densidade	Baseados em Densidade	Baseados em Densidade	Hierárquicos
Tipos de gráfico	Silhouette-like	Silhouette-like, Alcançabilidade	Silhueta, Alcançabilidade, Dendrograma	Silhouette-like, Alcançabilidade, Dendrograma
Linguagem	Java	Java	Python	Python
Disponibilidade	X	X	V	V

Fonte: Elaborada pelo autor.

Diante deste cenário, é possível extrair algumas conclusões sobre o estado da arte atual. As ferramentas disponíveis foram desenvolvidas com base em algoritmos de clusterização baseados em densidade. Enquanto Campello et al. (2015) e Neto (2015) implementaram tanto os gráficos de alcançabilidade quanto os *silhouette-like*, Gupta, Liu e Ghosh (2008) limitou-se à implementação do gráfico de silhueta. No entanto, Campello et al. (2015) não fornece detalhes sobre a implementação de uma ferramenta de visualização, e as ferramentas desenvolvidas por Neto (2015) e Gupta, Liu e Ghosh (2008), embora implementadas em Java, não estão mais disponíveis para uso.

Essa análise revela uma possível falta de uma ferramenta de visualização que integre os gráficos recomendados para a análise de grandes conjuntos de dados e que esteja disponível para uso público. Além disso, a linguagem Java, utilizada nos trabalhos mencionados, não é a mais adequada para o cenário atual de Inteligência Artificial e Ciência de Dados, áreas nas quais Python se consolidou como a linguagem predominante, devido à sua ampla gama de bibliotecas e recursos disponíveis (HAYES, 2020).

Ao examinar as implementações existentes em bibliotecas Python, como a Scikit-learn, observa-se que nenhuma delas oferece ferramentas prontas para gerar os gráficos propostos neste trabalho, conforme ilustrado na Tabela 3.1. Embora alguns algoritmos, como HDBSCAN, OPTICS e DBSCAN, estejam disponíveis na Scikit-learn, eles não incluem funcionalidades de

visualização adequadas para explorar hierarquias de *clusters*. Adicionalmente, como as ferramentas existentes na literatura focam em algoritmos baseados em densidade, há uma necessidade evidente de uma solução que também suporte algoritmos hierárquicos, permitindo uma análise mais abrangente e aprimorada de *clusters* nesse contexto.

4 Metodologia e Desenvolvimento

4.1 A ferramenta

O presente trabalho propõe a construção de um arcabouço de visualização para a exploração de dados não supervisionados e semissupervisionados. A ferramenta conta com as opções de visualização: dendrograma, silhouette-like plot e gráfico de alcançabilidade. Para alcançar estes feitos, foi feita a implementação do framework FOSC, que auxilia na extração de uma hierarquia de dados que facilita a geração destes tipos de visualizações. Essa hierarquia é armazenada na forma de matriz, e é possível visualizá-la por meio de uma tabela. O arcabouço foi projetado de modo a ser interativo, permitindo o usuário explorar os gráficos e agrupamentos.

4.2 Arquitetura

Para a construção da ferramenta, foi utilizada a linguagem de programação Python. A implementação deu-se início com a tradução do framework FOSC em código. Após isso, foi implementada a parte de visualização. Nesse sentido, utilizando o conceito de Programação Orientada a Objetos, foram criadas classes com seus respectivos métodos para representarem o framework. Cada classe representa um módulo disponibilizado pelo framework. O diagrama de classes pode ser visualizado no UML da Figura 4.1. Para fins de legibilidade, os parâmetros de entrada foram ocultados no diagrama.

Figura 4.1 – Diagrama de Classes do Framework.

Fonte: elaborada pelo autor.

Observando-se a Figura 4.1, é possível ver a relação entre as classes, assim como o detalhamento dessas relações. Entre a classe Plot e Fosc, tem-se uma dependência, já que a classe Plot funciona recebendo uma instância de Fosc. Já para Fosc e DendrogramStructure, existe uma composição pois, dentro de Fosc, existe uma instância de DendrogramStructure. Por fim, Cluster e Fosc possuem uma agregação entre si, o que indica a forte relação entre essas duas classes.

4.2.1 Módulo FOSC

O módulo FOSC é o principal módulo do framework. Esse módulo é responsável pela extração ótima de grupos com base na métrica de estabilidade. A classe recebe como parâmetro de entrada para seu construtor uma matriz de distâncias, característica dos métodos não supervisionados, uma string que representa o método de ligação, e o parâmetro m_{ClSize}. Ao ser instanciada, é possível obter saídas por meio dos seguintes métodos:

- `FOSC()`: Inicializa a classe FOSC, construindo a hierarquia de clusters a partir da matriz de distância e do método de ligação fornecidos. **Parâmetros**:
  - `dist_mat (numpy.ndarray)`: Matriz de distância entre os objetos.
  - `linkage_method (str)`: Método de ligação a ser utilizado no agrupamento hierárquico (e.g., 'single', 'complete', 'average').

  - `mClSize (int, opcional)`: Tamanho mínimo do cluster para que seja considerado válido (padrão: 4).

- `getZ()`: retorna a matriz de ligações;
- `getHierarchy()`: retorna a árvore que representa a hierarquia de clusters;
- `getHierarchyMatrix()`: retorna a hierarquia de clusters em forma de uma matriz.
- `propagateTree()`: executa a extração ótima de clusters a partir da árvore de hierarquia. Retorna `True`, se há algum cluster com estabilidade infinita e `False`, caso contrário.
  Parâmetros:
  - `**kwargs (dict)`: Dicionário contendo argumentos nomeados opcionais. Pode conter:
    * `dict (dict)`: Um dicionário contendo medidas de qualidade para os clusters.
    * `quality_measure (str)`: O nome da medida de qualidade a ser utilizada, presente no dicionário `dict`.
- `findProminentClusters()`: retorna uma tupla contendo a partição (numpy array que representa o resultado da extração ótima de grupos) e um dicionário que contém os objetos extraídos.
  Parâmetros:
  - `rootTree (int)`: O rótulo do cluster raiz da árvore a ser utilizada para encontrar os clusters proeminentes.
  - `infiniteStability (bool)`: Um booleano que indica se há clusters com estabilidade infinita.

4.2.2 Módulo Cluster

Módulo com métodos referentes a criação e propagação de clusters.

- `detachPoints()`: remove o número especificado de pontos do cluster no nível dado, o que atualiza a estabilidade do cluster. Se o cluster for eliminado, o número de restrições satisfeitas pelo cluster filho também é calculado.
  Parâmetros:
  - `numPoints (int)`: O número de pontos a serem removidos do cluster
  - `level (float)`: O nível do dendrograma removido da estrutura do dendrograma
- `propagate()`: propaga o cluster para o cluster pai se o número de restrições satisfeitas é maior que o número de restrições propagadas. Caso contrário, o cluster propaga seus

descendentes propagados. No caso de empate, a estabilidade é examinada. Além disso, o cluster propaga o menor nível de dissolução de qualquer um dos seus descendentes para seu respectivo cluster pai.

- `addPointsToVirtualChildCluster()`: atualiza o cluster após a aparição de um cluster filho virtual.
  Parâmetros:
  - `points (list)`: Lista de pontos a serem adicionados ao cluster filho virtual
- `virtualChildClusterContainsPoint()`: checa se um cluster filho virtual contem um ponto específico.
  Parâmetros:
  - `point`: Ponto a ser verificado no cluster filho virtual
- `addChild()`: atualiza que um nodo satisfez uma resrição.
  Parâmetros:
  - `ch`: Cluster filho a ser adicionado
- `releaseVirtualChildCluster()`: seta o cluster filho virtual para null, salvando memória. Esse método deve ser chamado somente depois de computar o número de restrições satisfeitas pelo cluster filho virtual.
- `releaseCluster()`: desprende o cluster da hierarquia de grupos.

4.2.3 Módulo Dendrogram

Consiste nos métodos relacionados à estrutura do dendrograma, utilizado no framework.

- `DendrogramStructure()`: Inicializa a estrutura do dendrograma a partir da matriz de ligações.
  Parâmetros:
  - `Z (numpy.ndarray)`: A matriz de ligações gerada pelo algoritmo de agrupamento hierárquico.
- `getAffectedNodesAtLevel()`: Retorna os IDs dos nós afetados em um determinado nível.
  Parâmetros:
  - `level (float)`: O nível para o qual os nós afetados devem ser retornados.
- `getFirstObjectAtNode()`: Retorna o primeiro objeto contido em um nó específico.
  Parâmetros:

  - `i (int)`: O ID do nó.

- `getSignificantLevels()`: Retorna uma lista ordenada dos níveis significativos do dendrograma.

- `getNumObjects()`: Retorna o número total de objetos no dendrograma.

- `getNodeSize()`: Retorna o número de objetos contidos em um nó específico.
  Parâmetros:

  - `id (int)`: O ID do nó.

- `getObjectsAtNode()`: Retorna a lista de objetos contidos em um nó específico.
  Parâmetros:

  - `i (int)`: O ID do nó.

- `__str__()`: Retorna uma representação em string da estrutura do dendrograma, incluindo os nós e os clusters afetados por nível.

4.2.4 Módulo Plot

Esse módulo aglomera os métodos relacionados à visualização de dados. Desse modo, obteve-se três métodos. Cada um deles recebe uma string opcional que representa o diretório do arquivo a ser salva a imagem. Se esse parâmetro não é informado, o gráfico é exibido em uma janela.

- `plotDendrogram()`: gera o gráfico de dendrograma.
  Parâmetros:

  - `fosc (FOSC)`: instância da Classe FOSC
  - `title (str)`: título do dendrograma
  - `saveDescription (Optional[str])`: caminho do diretório pra salvar a imagem

- `plotReachability()`: gera o gráfico de alcançabilidade.
  Parâmetros:

  - `fosc (FOSC)`: instância da classe FOSC
  - `partition (Optional[np.ndarray])`: partição de objetos. Se for providenciada, colore o gráfico com base nela.
  - `saveDescription (Optional[str])`: caminho do diretório pra salvar a imagem

- `plotSilhouette()`: gera o gráfico silhouette-like.
  Parâmetros:
  - `fosc (FOSC)`: instância da Classe FOSC
  - `partition (Optional[np.ndarray])`: partição de objetos. Se for providenciada, as silhuetas do gráfico são cortadas, exibindo a extração de grupos.
  - `saveDescription (Optional[str])`: caminho do diretório pra salvar a imagem

4.3 Arcabouço de Visualização

Primeiramente, a biblioteca escolhida para auxiliar na implementação dos gráficos foi Matplotlib. Trata-se de uma biblioteca com ampla customização e documentação, o que facilita o desenvolvimento e possibilita melhorar a experiência de interatividade do usuário.

Dessa forma, o primeiro passo foi transformar a árvore de hierarquia do FOSC para uma matriz de hierarquia, que facilita a criação do gráfico silhouette-like. Em seguida, foi feita a transformação dessa matriz, ilustrada na Figura 2.9, no gráfico em questão. Para a coloração, foi criado um grafo, no qual os vértices representam regiões da matriz, e as arestas representam que duas regiões são adjacentes, o que implicam que não podem receber a mesma cor. Tal fato é ilustrado na Figura 2.10. O grafo obtido é introduzido como entrada para um algoritmo guloso, que recebe também um número de cores n (foi escolhido $n = 4$, no caso) e utiliza a estratégia "*largest first*". Essa estratégia consiste em colorir com o maior número de cores possíveis, que seja menor ou igual a n. O algoritmo então, retorna um dicionário em que as chaves representam as regiões da matriz e os valores correspondem a uma cor de 0 a $n - 1$. Por fim, é feito um mapeamento destes valores para cores reais, neste caso, as cores azul, vermelho, verde e amarelo.

Já para o gráfico de alcançabilidade, foi utilizada a estratégia de Sander et al. (2003), a partir do dendrograma do conjunto de dados. Dessa maneira, o gráfico reachability foi obtido a partir da conversão do dendrograma.

4.3.1 Partições e Extração Ótima de Grupos

Tanto o gráfico silhouette-like, quanto o reachability, foram adaptados para não ilustrar somente a hierarquia total de grupos, mas também mostrar o resultado dos algoritmos de extração de agrupamentos. Nesse sentido, para o gráfico silhouette-like, as silhuetas são cortadas horizontalmente de acordo com os grupos escolhidos, como ilustrado na Figura 2.2.

O gráfico de alcançabilidade é colorido com base na lista de agrupamentos providas pelo algoritmo. As barras dos objetos de um mesmo cluster recebem a mesma cor. Ademais, objetos categorizados como ruído pelo FOSC recebem a cor branca. A estratégia de coloração usada foi a mesma do gráfico silhouette-like, na qual clusters vizinhos no gráfico recebem cores diferentes, e o número de cores é menor ou igual a 4.

4.3.2 Interatividade

Para melhorar a experiência do usuário, os gráficos silhouette-like e reachability contam com uma funcionalidade para a exploração dos agrupamentos. Essa funcionalidade parte do princípio que exibir todos os rótulos de agrupamentos para todos os objetos, como na Figura 2.10, torna o gráfico ilegível para grandes conjuntos de dados. Sendo assim, o procedimento adotado foi usar a técnica de *mouse-hovering*, que consiste em exibir uma informação na tela quando o usuário passa o mouse em cima de um determinado pixel. Nesse contexto, ao posicionar o mouse sob um pixel no gráfico, são mostrados o número do objeto, cluster correspondente e a distância – altura da barra, no caso do reachability, e distância que originou o cluster, no caso do silhouette-like, como mostra a Figura 5.6. Além disso, o usuário ainda tem as opções de interatividade padrão da biblioteca Matplotlib, como salvar uma imagem, dar zoom no gráfico e configurar a escala.

4.4 Experimentos e Datasets

O arcabouço deve ser testado e experimentado para ser validado. Nesse sentido, é necessário testar a ferramenta em grandes conjuntos de dados para avaliar seu desempenho. Para tal, o dataset escolhido foi criado por Handl e Knowles (2007), que consiste em uma base de dados sintéticos de larga escala com diferentes objetos e hierarquias de clusters. O dataset contém metade dos dados de baixa dimensão – 2, 10 – e a outra alta dimensão – 50 e 100. Além disso, os números de clusters variam em: 4, 10, 20 e 40. Os dados foram gerados aleatoriamente, baseando-se em distribuições normais multivariadas e aplicando-se uma técnica de tentativa e erro, para evitar sobreposição de clusters. Além disso, o dataset ainda contém dados elipsoidais, o que colabora com a diversidade do conjunto e permite uma testagem ainda mais ampla. O conjunto de dados pode ser obtido em https://personalpages.manchester.ac.uk/staff/Julia.Handl/generators.html.

4.5 Documentação

Após a criação do framework, foi elaborada uma documentação. Essa visa tanto detalhar os métodos disponíveis na ferramenta, como dar um guia rápido de inicialização ao usuário. A documentação foi feita pela biblioteca *Sphinx*.

4.5.1 Implementação da documentação

Sphinx é uma biblioteca em Python que possibilita a geração automática de páginas web de documentação, a partir de configurações feitas em documentos *RST* (ReStructuredText) ou *Markdown*. Além disso, possibilita a customização de diversos temas, incluindo o tema escolhido *Read the Docs*, que corresponde a uma plataforma de hosteamento de documentações, que é utilizada neste trabalho.

4.6 Publicação

Com tudo elaborado, o último passo é a publicação do framework. *Pip* é o instalador de pacotes padrão para a linguagem Python, sendo assim, o objetivo é que o usuário consiga instalar o arcabouço como se fosse qualquer outro pacote, por meio do comando "*pip install FOSC*". Para atingir esta meta, submete-se o código fonte ao repositório PyPI (The Python Package Index).

5 Resultados

Foi produzido um arcabouço de visualização, com um potencial que será elucidado neste capítulo. Por meio das técnicas de programação e engenharia de software apresentadas, foi obtido um *framework* no qual a extração ótima de grupos proporcionada pelo FOSC pode ser visualizada de duas formas, uma no gráfico de silhueta, e outra no gráfico de alcançabilidade. A Figura 5.1 ilustra o *framework* sendo utilizado para gerar um gráfico de silhueta. O conjunto de dados utilizado nesse exemplo possui 10 clusters e 10 dimensões.

Figura 5.1 – Código para geração do gráfico *Silhouette-like*.

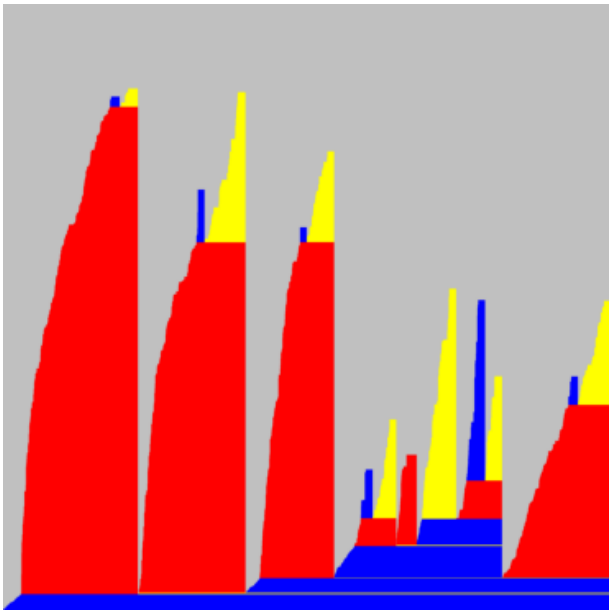
```
1  from FOSC import Plot, FOSC
2  import numpy as np
3  from scipy.spatial import distance_matrix
4
5  file_path = "./datasets/Handl_datasets/10d-10c-no0.csv"
6  mClSize = 30
7  linkage_method = "single"
8
9  mat = np.genfromtxt(file_path, dtype=float, delimiter=',',
10                     usecols=range(0, -1),
11                     missing_values=np.nan)
12 dist_mat = distance_matrix(mat, mat, p=2)
13
14 foscFramework = FOSC(dist_mat, linkage_method,
   ↪  mClSize=mClSize)
15
16 Plot.plotSilhouette(foscFramework)
```

Fonte: Elaborado pelo autor.

O resultado da execução do código da Figura 5.1 pode ser observado na Figura 5.2. Ademais, a codificação apresentada na Figura 5.3 mostra como o framework é utilizado para realizar a extração ótima de grupos do FOSC e visualizá-la.

O resultado da execução do código da Figura 5.3 pode ser observado na Figura 5.4.

Figura 5.2 – Gráfico de Silhueta obtido pelo Arcabouço.



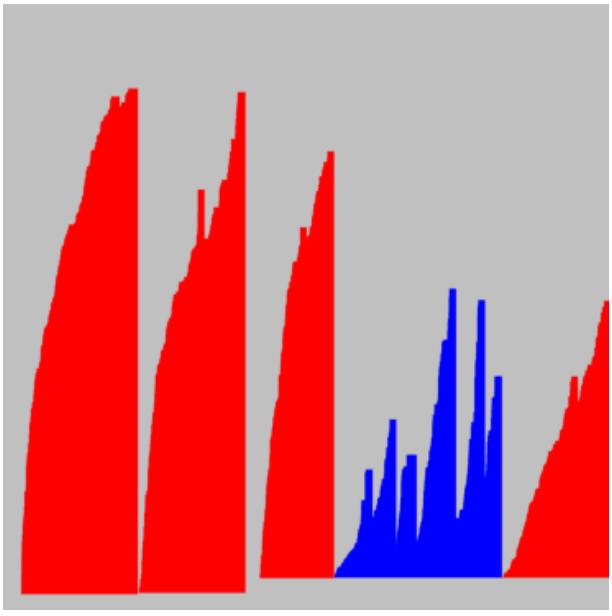
Fonte: Elaborada pelo autor.

Figura 5.3 – Código para geração do gráfico *Silhouette-like*.

```
1 infiniteStability = foscFramework.propagateTree()
2 partition, lastObjects =
  ↪ foscFramework.findProminentClusters(1, infiniteStability)
3 Plot.plotSilhouette(foscFramework, partition)
```

Fonte: Elaborado pelo autor.

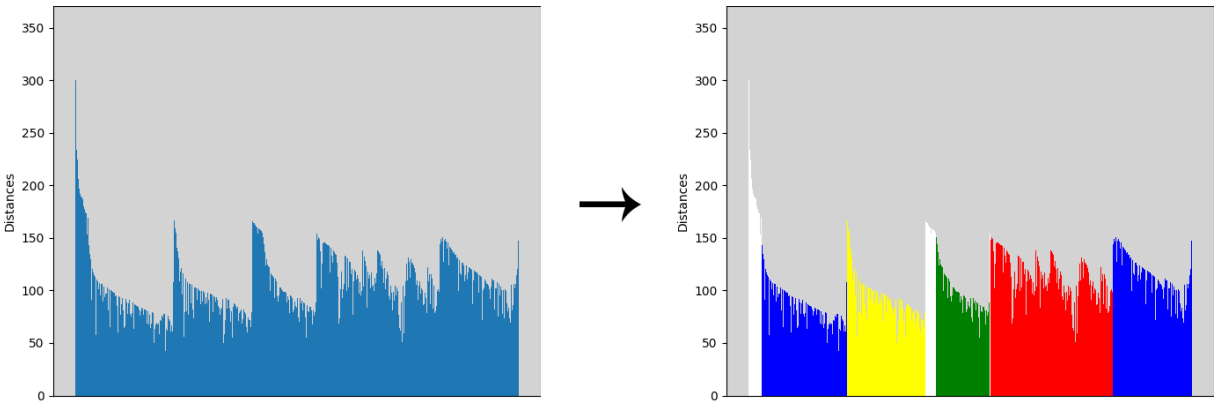
Figura 5.4 – Gráfico de Silhueta com Extração do FOSC.



Fonte: Elaborada pelo autor.

Além disso, o gráfico de alcançabilidade também pode ser obtido, assim como a sua coloração de acordo com a extração de grupos, como é ilustrado na Figura 5.5. Para obter este gráfico, o método `plotReachability()` foi chamado duas vezes, da mesma forma em que o método `plotSilhouette()` foi chamado nos códigos Figura 5.1 e Figura 5.3.

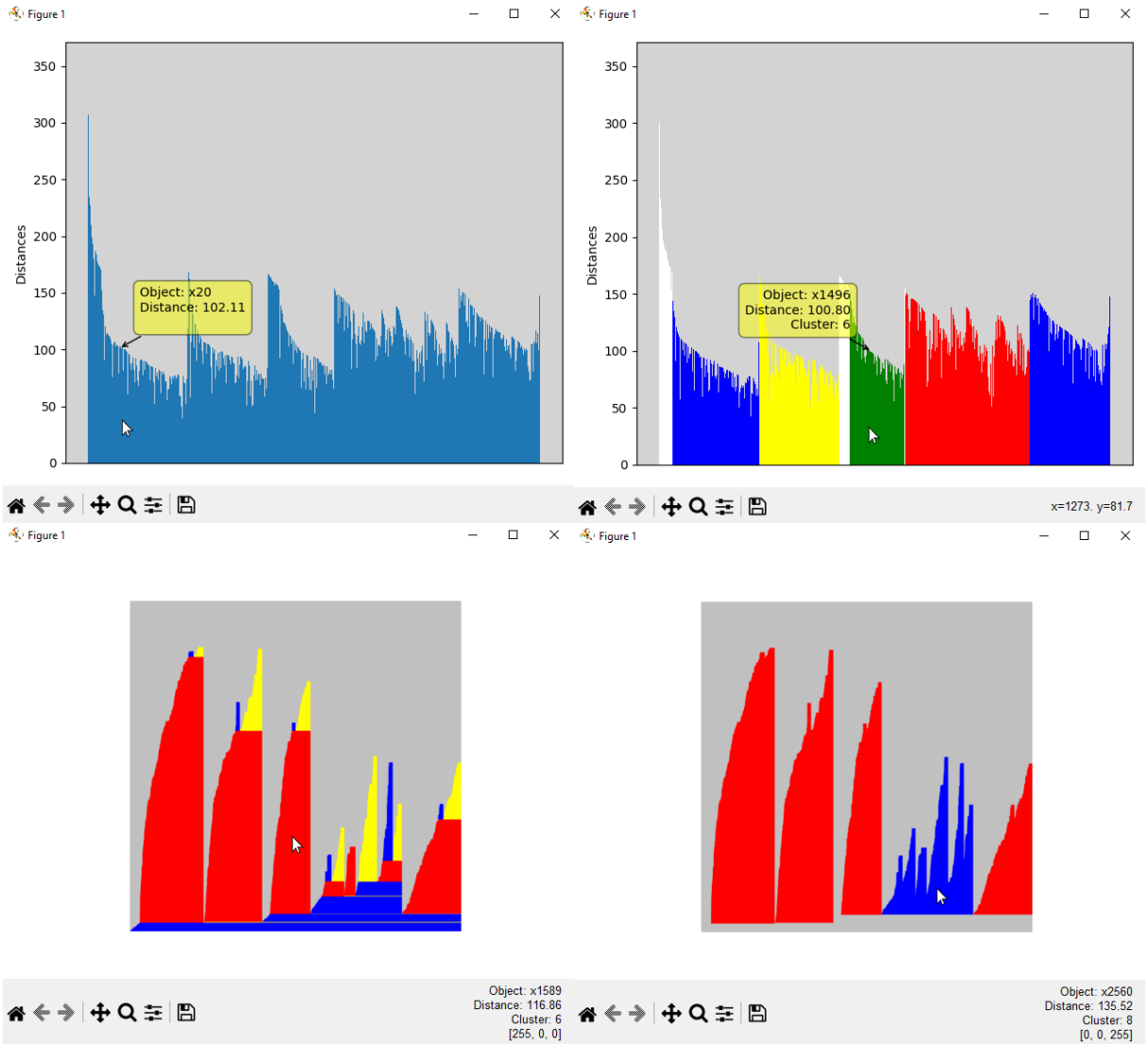
Figura 5.5 – Corte no gráfico de alcançabilidade.



Fonte: Elaborada pelo autor.

Ainda, os gráficos adquiriram a possibilidade de interação com o usuário, de modo a explorar os agrupamentos, como mostra a Figura 5.6. Quando os códigos de Figura 5.1 e Figura 5.3 são executados, uma janela de visualização é aberta, e o usuário, por meio do mouse, pode explorar os objetos, distâncias e *ids* (números) dos *clusters*.

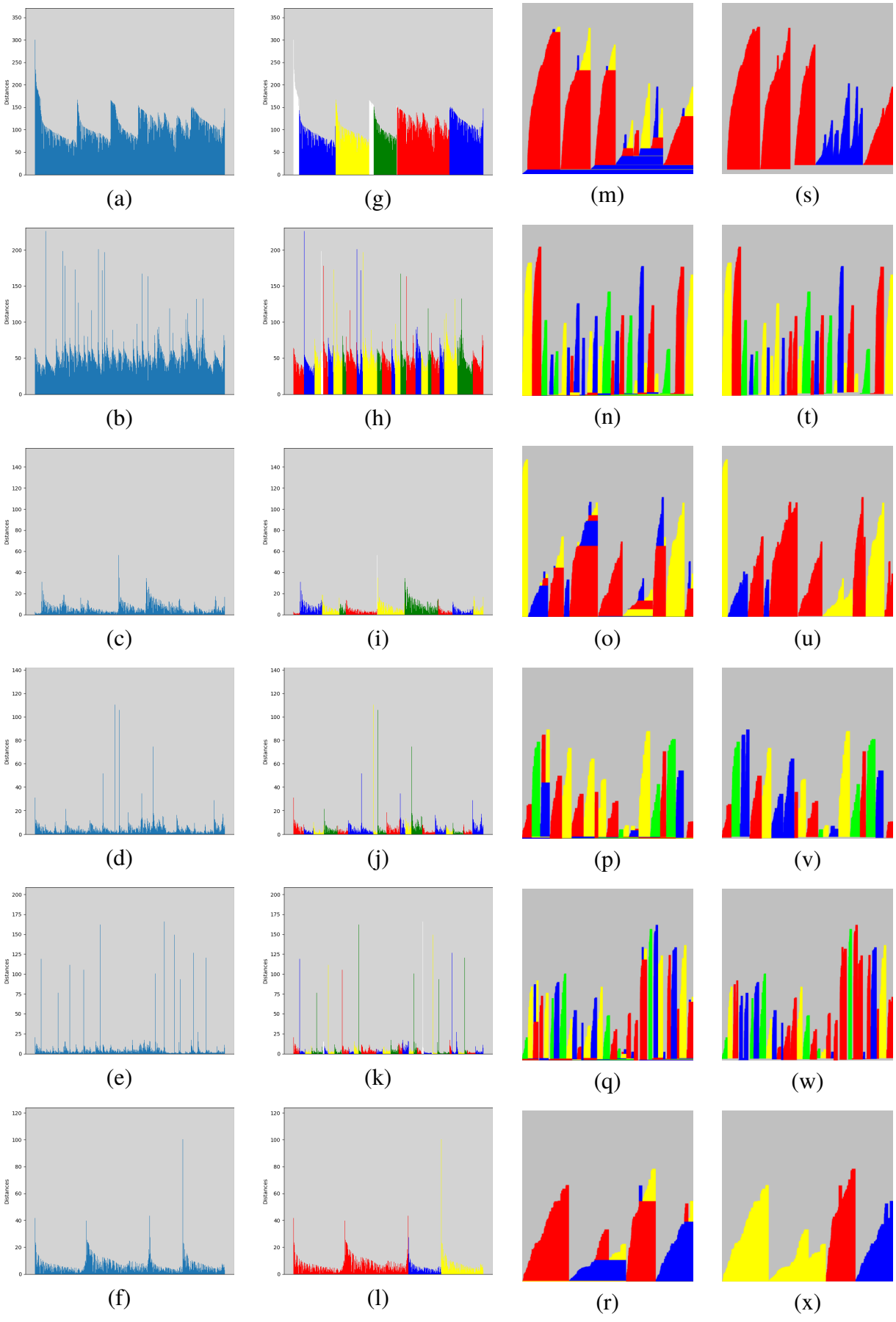
Figura 5.6 – Interatividade com o usuário.



Fonte: Elaborada pelo autor.

Por fim, mais experimentos foram realizados no dataset Handl e Knowles (2007), de modo a obter-se gráficos diferentes que permitem visualizar o potencial da ferramenta. A Figura 5.7 mostra diferentes gráficos obtidos por meio do arcabouço. As figuras de a a f são de gráficos de alcançabilidade, e as imagens de g a l mostram o resultado da extração do FOSC. Ademais, os itens m a r representam gráficos silhouette-like obtidos e o resultado da extração é exibido dos itens s a x. A configuração do FOSC se deu com os parâmetros $MClSize = 30$ e método de ligação *single linkage*. Foram escolhidos conjuntos de dados de diferentes dimensões e número de clusters.

Figura 5.7 – Experimentos com conjuntos de diferentes densidades e número de *clusters*.

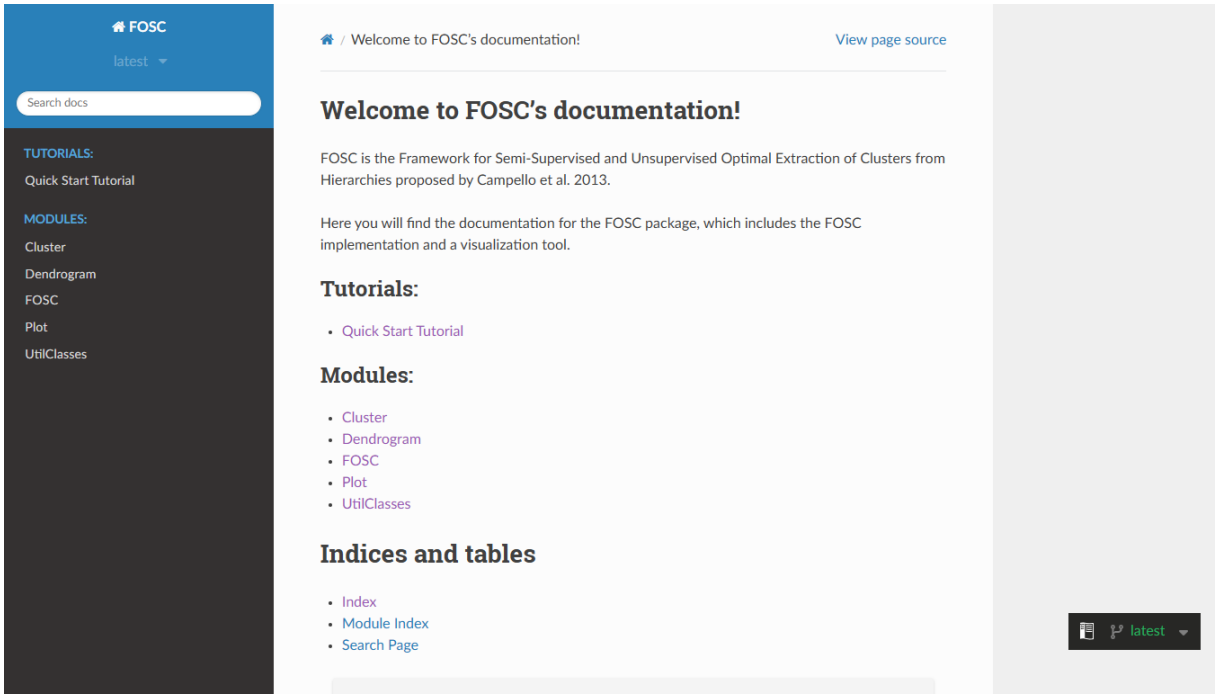


Fonte: Elaborada pelo autor.

5.1 Documentação

A documentação foi implementada e está disponível em https://fosc.readthedocs.io/en/latest/. A página gerada é apresentada na Figura 5.8.

Figura 5.8 – Documentação do arcabouço.



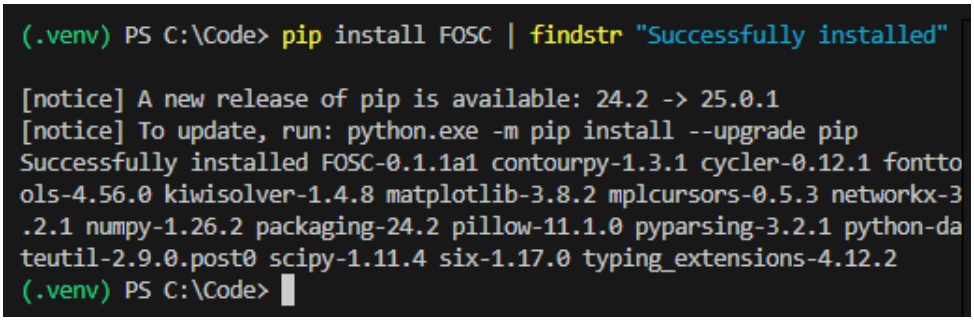
Fonte: Elaborada pelo autor.

A documentação conta com um tutorial rápido de como usar o pacote, além de uma descrição dos módulos disponíveis.

5.2 Disponibilização do pacote

O framework foi submetido para o repositório PyPi e está disponível em: https://pypi.org/project/FOSC/. A Figura 5.9 mostra a instalação do pacote sendo feita com sucesso em um ambiente virtual pelo terminal *PowerShell*. O comando `findstr ‘‘Successfully installed’’` é opcional e é utilizado para filtrar a mensagem final da instalação.

Figura 5.9 – Instalação do pacote.



Fonte: Elaborada pelo autor.

6 Considerações Finais

6.1 Conclusão

Neste trabalho foi feita a criação de um arcabouço de visualização para a exploração de dados não supervisionados e semissupervisionados. Para isso, foi feita uma revisão do framework FOSC (CAMPELLO et al., 2013), suas aplicações e, posteriormente, uma implementação do framework.

Por meio da realização dos testes, conclui-se que a ferramenta mostrou-se bastante eficiente para plotar grandes conjuntos de dados. Dessa forma, o arcabouço configura-se como um protótipo útil para contribuir tanto ao ramo de mineração de dados, quanto inteligência artificial. Nesse sentido, contribuições significativas foram feitas para o estado da arte, sendo elas:

- A criação de uma nova ferramenta implementada em uma nova linguagem e com novos recursos;
- A integração do FOSC a essa nova ferramenta, permitindo explorar, de maneira visual, a extração ótima de grupos do framework;
- A disponibilização da ferramenta, permitindo que a comunidade possa usá-la e testá-la.

Vale ressaltar que parte dos resultados obtidos neste trabalho fazem parte de uma iniciação científica sob o título "Construção de um Arcabouço de Visualização para Exploração de Dados Não Supervisionados e Semissupervisionados".

6.2 Limitações

O presente trabalho possui limitações em relação à parte visual da ferramenta. Nesse sentido, não há uma forma do usuário personalizar as cores que serão utilizadas na geração dos gráficos. Além disso, as cores utilizadas constituem cores básicas e não foram escolhidas pensando em técnicas profundas de experiência do usuário (*UX Design*).

A ferramenta somente estende o FOSC (CAMPELLO et al., 2013) e não agrupa outros algoritmos de aprendizado não supervisionado e semissupervisionado, que poderiam ser incorporados ao pacote, como k-médias, DBSCAN ou técnicas de extração de características, como análise de componentes principais (PCA) e autoencoders.

Ademais, o corte no gráfico de silhueta gerado pela extração ótima de grupos mantém as cores originais do gráfico, o que pode gerar confusões, como grupos diferentes com a mesma cor, como ilustra a Figura 5.4.

6.3 Trabalhos Futuros

Para trabalhos futuros, sugere-se melhorar a aparência da ferramenta de visualização, empregando-se técnicas de UX Design, como cores para daltônicos. Além disso, cabe a implementação de *CI/CD* (*Continous Integration*, *Continous Delivery*), para permitir que a comunidade possa contribuir, de maneira contínua e segura, realizando-se testes automáticos de integração. Ademais, é válido realizar uma nova coloração para o gráfico de silhueta cortado, como mencionado na seção 6.2, para destacar os diferentes grupos extraídos. Por fim, outros algoritmos de aprendizado semissupervisionado e não supervisionado podem ser incorporados à ferramenta, permitindo outros tipos de extração, além da medida de estabilidade do FOSC.

Referências

ABDEL-HAKIM, A. E.; DEABES, W. Handling missing annotations in supervised learning data. *arXiv preprint arXiv:2002.07113*, 2020.

ANJOS, F. de Assis Rodrigues dos; GERTRUDES, J. C.; SANDER, J.; CAMPELLO, R. J. G. B. A modularity-based measure for cluster selection from clustering hierarchies. In: ISLAM, M. R.; KOH, Y. S.; ZHAO, Y.; GRACO, W.; STIRLING, D.; LI, C.; ISLAM, M. Z. (Ed.). *Data Mining - 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, November 28-30, 2018, Revised Selected Papers*. [S.l.]: Springer, 2018. (Communications in Computer and Information Science, v. 996), p. 253–265.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, ACM New York, NY, USA, v. 28, n. 2, p. 49–60, 1999.

CAMPELLO, R. J.; MOULAVI, D.; ZIMEK, A.; SANDER, J. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery*, Springer, v. 27, p. 344–371, 2013.

CAMPELLO, R. J.; MOULAVI, D.; ZIMEK, A.; SANDER, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, ACM New York, NY, USA, v. 10, n. 1, p. 1–51, 2015.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. *Machine learning*, Springer, v. 109, n. 2, p. 373–440, 2020.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proc. KDD*. [S.l.: s.n.], 1996. p. 226–231.

FACELI, K.; LORENA, A. C.; GAMA, J.; ALMEIDA, T. A. d.; CARVALHO, A. C. P. d. L. F. d. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: LTC, 2021.

GERTRUDES, J. C. *Semi-supervised learning approaches with applications in medicinal chemistry*. Tese (Doutorado) — Universidade de São Paulo, 2019.

GUPTA, G.; LIU, A.; GHOSH, J. Automated hierarchical density shaving: A robust automated clustering and visualization framework for large biological data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, IEEE, v. 7, n. 2, p. 223–237, 2008.

HANDL, J.; KNOWLES, J. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 11, n. 1, p. 56–76, 2007. Dataset disponível em: <https://personalpages.manchester.ac.uk/staff/Julia.Handl/generators.html>. Acesso em: 22 mar. 2025.

HAYES, B. *Usage of Programming Languages by Data Scientists: Python Grows while R Weakens*. 2020. Disponível em: <https://businessoverbroadway.com/2020/06/29/usage-of-programming-languages-by-data-scientists-python-grows-while-r-weakens/>. Acesso em: 2024-05-30.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. [S.l.]: Prentice-Hall, Inc., 1988.

JHA, J.; VISHWAKARMA, A. K.; N, C.; NITHIN, A.; SAYAL, A.; GUPTA, A.; KUMAR, R. Artificial intelligence and applications. *2023 1st International Conference on Intelligent Computing and Research Trends (ICRT)*, p. 1–4, 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:259158428>.

NETO, F. S. d. A. *Ferramenta de Visualização para Algoritmos Semi-Supervisionados de Agrupamento e Classificação de Padrões*. São Carlos - SP: Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, Departamento de Sistemas de Computação, 2015. Monografia referente ao projeto de conclusão de curso dentro do escopo da disciplina Projeto de Graduação I do Departamento de Sistemas de Computação do Instituto de Ciências Matemáticas e de Computação - ICMC-USP para obtenção do título de Bacharel em Informática. Área de Concentração: Mineração de Dados.

PARASHAR, A.; GULATI, Y. Survey of different partition clustering algorithms and their comparative studies. *International Journal of Advanced Research in Computer Science*, v. 3, n. 3, 2012.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987.

SANDER, J.; QIN, X.; LU, Z.; NIU, N.; KOVARSKY, A. Automatic extraction of clusters from hierarchical clustering representations. In: SPRINGER. *Advances in Knowledge Discovery and Data Mining: 7th Pacific-Asia Conference, PAKDD 2003, Seoul, Korea, April 30–May 2, 2003 Proceedings 7*. [S.l.], 2003. p. 75–87.

YADAV, P.; YADAV, A.; AGRAWAL, R. Use of artificial intelligence in the real world. *International Journal of Computer Science and Mobile Computing*, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:255361423>.

ZHU, X.; GOLDBERG, A. B. *Introduction to semi-supervised learning*. [S.l.]: Morgan Claypool Publishers, 2009.