UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

THIAGO HENRIQUE ADAMI

DESENVOLVIMENTO DE UM MÉTODO DE *CLUSTERING* BASEADO EM GRAFOS DE SIMILARIDADE

PATO BRANCO

2023

THIAGO HENRIQUE ADAMI

DESENVOLVIMENTO DE UM MÉTODO DE *CLUSTERING* BASEADO EM GRAFOS DE SIMILARIDADE

Development of a clustering method based on similarity graphs

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Ives Renê Venturini Pola

PATO BRANCO

2023

THIAGO HENRIQUE ADAMI

DESENVOLVIMENTO DE UM MÉTODO DE *CLUSTERING* BASEADO EM GRAFOS DE SIMILARIDADE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 20 de junho de 2023

Prof. Dr. Ives Renê Venturini Pola
Pós-Doutorado em Ciências de Matemática e Computação
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dr. Dalcimar Casanova
Doutorado em Física Computacional
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dr. Marco Antonio de Castro Barbosa
Doutorado em Informática
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

PATO BRANCO

2023

AGRADECIMENTOS

Agradeço à minha família, pai, mãe e irmã, e à minha namorada, Juliana, pelo carinho, apoio, paciência e compreensão em todos os momentos e independentemente da distância. Amo vocês.

Agradeço ao meu orientador, Prof. Dr. Ives Renê Venturini Pola pela orientação, pelo incentivo, pelas cobranças e pelas discussões e construção de conhecimento que permitiram a elaboração desse trabalho.

Aos meus amigos, agradeço por todos os momentos de desabafo, de companheirismo e de diversão, que nos fazem escapar da realidade quando necessário.

Por fim, um agradecimento a todos que contribuíram de alguma forma na minha trajetória até o momento da realização desse trabalho.

It's not a question of can or can't. There are some things in life you just do.
(LIGHTNING, Final Fantasy XIII)

RESUMO

ADAMI, Thiago Henrique. Desenvolvimento de um método de *clustering* baseado em grafos de similaridade. 2023. 55 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2023.

O processo de *clustering* consiste em, a partir de um conjunto de dados, obter conjuntos que agrupem dados similares entre si. No entanto, nem todos os algoritmos de *clustering* se preocupam em reduzir ou eliminar a sobreposição entre os agrupamentos identificados. Dessa maneira, o presente trabalho apresenta um método de *clustering* baseado em grafos de similaridade extraídos de espaços métricos contendo vetores de características, com o objetivo de minimizar o fator de sobreposição entre *clusters*. O método de *clustering* desenvolvido é uma combinação das técnicas DBSCAN e consulta por abrangência (*Rq*), e introduz a consulta aos *k*-vizinhos mais próximos (*k*NN*q*) como o limite superior para a vizinhança retornada pela *Rq*. O fator de sobreposição é uma razão que considera o número de elementos que estão na vizinhança de mais de um *cluster*. Nas combinações de parâmetros de entrada utilizados e nos espaços de teste, o método de *clustering* apresentou resultados melhores que ou iguais aos obtidos pelo DBSCAN, tanto nos agrupamentos identificados como nos fatores de sobreposição calculados.

Palavras-chave: *clustering*; grafos de similaridade; dbscan; rangequery; *k*-vizinhos mais próximos.

ABSTRACT

ADAMI, Thiago Henrique. Development of a clustering method based on similarity graphs. 2023. 55 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2023.

The clustering process consists in, starting from a dataset, extracting clusters which contains similar data. On the other hand, not all the clustering methods aim to reduce or remove the overlap between identified clusters. Therefore, the present work proposes the development of a clustering method based on similarity graphs extracted from metric spaces containing feature vectors, which objective is to minimize the overlap between clusters. The proposed clustering method is a combination of DBSCAN and range query (Rq) techniques, and introduces the k-nearest neighbors query (kNNq) as an upper bound of the neighborhood observed by the Rq. The overlap factor is a ratio which considers the number of elements which are in the neighborhood of more than one cluster. Over the input parameter combinations and test spaces, the clustering method presented better or equivalent results to those obtained by DBSCAN, both on the identified clusters and on the evaluation metrics.

Keywords: clustering; similarity graphs; dbscan; rangequery; k-nearest neighbors.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

R_q	Range Query
k-NN	*k*-Nearest Neighbors
k-NNq	*k*-Nearest Neighbors Query
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points To Identify the Clustering Structure
PL/pgSQL	Procedural Language/PostgreSQL
SQL	Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos últimos anos, o volume de informações obtidas cresceu exponencialmente, gerando uma massa de dados que precisa ser armazenada e, dependendo da finalidade, tratada. Esse conjunto massivo de dados é denominado de *Big Data* (ORACLE, 2014). Wang (2017) cita problemas decorrentes do *Big Data* e menciona a análise e processamento dos dados, uma vez que a heterogeneidade é uma das características desse tipo de material. Ainda, existem diferentes tipos de heterogeneidade, o que aumenta a dificuldade ao lidar com tais dados (WANG, 2017). Portanto, é preciso encontrar uma forma de aproximar objetos com características distintas, mas não descartar possíveis características semelhantes que avizinhe cada instância.

Uma das maneiras de realizar esse agrupamento é identificar a estrutura intrínseca de um conjunto de dados para então aproximar cada elemento pertencente à esse conjunto. Esse processo de reconhecimento e concentração de dados recebe o nome de *clustering* (SCHAEFFER, 2007). O *clustering* é uma operação de agrupamento de dados que, segundo Schaeffer (2007, p. 31), dado um determinado *dataset*, visa agrupar os elementos que são afins, gerando núcleos (denominados de *clusters*) espalhados pelo espaço.

A maneira mais simples de se obter a similaridade entre quaisquer objetos é calcular a distância entre eles, de tal modo que distâncias menores representam maior similaridade. O cálculo da distância pode ser feito de maneira linear, utilizando a distância Euclidiana; mas também é possível observar a distância de formas não-lineares, como a distância quadrática, a distância de Mahalanobis e a similaridade de cossenos. Dependendo da finalidade da similaridade dentro da aplicação, deve-se analisar qual é a melhor maneira de realizar esse cálculo.

Por outro lado, é necessário perceber que os dados precisam estar dispostos computacionalmente de alguma maneira que permita realizar qualquer operação sobre eles, como, por exemplo, calcular a similaridade. Uma das formas mais utilizadas para essa representação são grafos. Assim como dados dispersos pelo espaço, grafos também podem ser submetidos ao processo de *clustering*, considerando, além da similaridade, a conectividade entre os vértices. Nesse ponto, o processo de *clustering* se torna próximo de um aprendizado não-supervisionado, embora existam métodos de *clustering* supervisionados e semi-supervisionados (EICK; ZEIDAT; ZHAO, 2004, p. 1).

Tratar a similaridade em grafos nem sempre é um processo simples, pois para entradas com muitos vértices, precisa-se de muito tempo para encontrar a solução ótima (ou próxima da solução ótima) para o problema. Por isso, uma métrica de similaridade compatível com o problema e uma representação adequada dos dados são atributos fundamentais, pois ambas são determinantes para auxiliar o processo da tomada de decisões, possivelmente reduzindo o tempo necessário para obter soluções.

Pola *et al.* (2015) definem os *Similarity Sets* - conjuntos de similaridade -, que são uma maneira de trabalhar com a similaridade em banco de dados. Tais grupos levam em conta a complexidade dos dados, já que aplicações envolvendo dados complexos dificilmente possuem

elementos iguais ou extremamente semelhantes, reafirmando a heterogeneidade em grandes conjuntos de dados. Para obter a semelhança entre cada elemento, Pola *et al.* (2015) utilizam a similaridade de cossenos, conectando vértices similares com distância obtida usando essa métrica. Entretanto, é possível utilizar qualquer métrica de distância que respeite os axiomas do espaço métrico, e que seja possível interpretá-la como a similaridade (ou dissimilaridade) entre os elementos.

Tomando como referência os grafos de similaridade, semelhantes aos abordados por Pola *et al.* (2015), é apresentado neste trabalho o desenvolvimento de um método de *clustering* de maneira que, a partir de um espaço métrico de qualquer dimensão, extraia grafos de similaridade e obtenha conjuntos de *clusters* que representem uma coleção de dados de acordo com a semelhança entre eles. O método de *clustering* proposto irá consultar todo o espaço e os grafos de similaridade obtidos, sendo obrigatório visitar todos os vértices que compõem o grafo para alocá-los nos *clusters* mais adequados.

Nem todos os algoritmos de *clustering* levam em consideração a sobreposição entre os *clusters* identificados, ou a quantidade de elementos que fazem parte de algum *cluster* e estão localizados nas áreas de sobreposição entre os *clusters*. Dessa forma, neste trabalho será introduzido o cálculo do fator de sobreposição, baseado no *fat-factor* das *Slim-trees* (TRAINA *et al.*, 2000). Esse fator relaciona o número de elementos localizados na sobreposição entre dois ou mais *clusters* com o número de elementos dos *clusters* que se sobrepõem.

Partindo da problemática apresentada, o método desenvolvido nesse trabalho combina três técnicas já conhecidas: o Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (ESTER *et al.*, 1996), a R_q e a *k*-Nearest Neighbors Query (*k*-NNq) (ambas abordadas por Pola (2010)), além de aproveitar conceitos e estabelecer paralelos com outros métodos, como a obtenção de focos baseado no método de acesso OMNI (FILHO *et al.*, 2001) e o cálculo de sobreposição das *Slim-Trees* (TRAINA *et al.*, 2000; TRAINA *et al.*, 2002).

A *k*-NNq atua como o limite superior para a vizinhança retornada pela R_q, restringindo a área observada por cada vértice durante as consultas. Originalmente, o DBSCAN possui apenas o limite inferior, que é o número mínimo de pontos na vizinhança, sem especificar um limite máximo para expandir os *clusters* (ESTER *et al.*, 1996). Reduzindo o número máximo de vizinhos, o raio real de abrangência do *cluster* diminui, sendo igual à distância até o k-ésimo vizinho mais distante, tendendo também a diminuir a área de sobreposição.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho foi o projeto, desenvolvimento e implementação de um método de *clustering* baseado em grafos de similaridade, além da obtenção de resultados visuais e medidas de avaliação que possibilitem a comparação entre os resultados finais desse algoritmo e de outros métodos já existentes.

1.2 OBJETIVOS ESPECÍFICOS

- Visualizar os resultados por meio de imagens;
- Obter métricas de desempenho do algoritmo;
- Comparar os resultados obtidos do algoritmo implementado com outros métodos;
- Investigar e sugerir alternativas para possíveis melhorias no método e/ou nos resultados obtidos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este documento está disposto da seguinte maneira:

- No capítulo 2 é apresentada a revisão de literatura que ampara o trabalho desenvolvido. Todos os conceitos abordados por esse trabalho estão apoiados nas referências adotadas.
- No capítulo 3 é apresentado e descrito o método de *clustering*, conectando de forma mais detalhada os conceitos apresentados nos capítulos 2 com o trabalho desenvolvido.
- O capítulo 4 traz a análise e discussão dos resultados obtidos, assim como a avaliação dos resultados baseado em algumas métricas e a comparação dos resultados do método proposto com resultados obtidos utilizando as técnicas tradicionais utilizadas como referência para o trabalho.
- O capítulo 5 apresenta uma conclusão geral sobre o que foi apresentado no trabalho, os resultados obtidos, e também sugere trabalhos futuros utilizando o que foi desenvolvido como ponto de partida.
- Finalmente, são apresentadas as referências bibliográficas utilizadas no trabalho.

2 REVISÃO DE LITERATURA

Neste capítulo, serão abordados os conceitos que serviram de referência para o desenvolvimento do trabalho e o cumprimento dos objetivos apresentados.

2.1 ESPAÇOS MÉTRICOS

De acordo com Lima (1977), um espaço métrico é definido como um par (M, d), onde M representa um conjunto de elementos quaisquer e d é uma métrica que está contida em M . Essa métrica é representada pela relação d: M x $M \rightarrow \mathbb{R}$, relaciona pares ordenados de elementos de M entre si e associa cada par a um número real, que é chamado de distância (LIMA, 1977).

Ainda, também segundo Lima (1977), quando estabelecida a definição de espaço métrico e da relação entre seus elementos, só é possível considerar que o par (M, d) é um espaço métrico se e somente se as seguintes propriedades forem satisfeitas:

1. $d\ (x, x) = 0$;
2. $x \neq y \rightarrow d\ (x, y) > 0$;
3. $d\ (x, y) = d\ (y\ , x)$;
4. $d\ (x, z) \leq d\ (x\ , y) + d\ (y\ , z)$.

2.2 MÉTRICAS DE DISTÂNCIA

Tomando como base as propriedades do espaço métrico e a definição da métrica que relaciona os elementos do espaço, a distância é uma função que recebe dois elementos do conjunto como parâmetros e retorna um número real como resultado (LIMA, 1977).

Conforme Pola, Traina e Traina (2006), existem inúmeros métodos para calcular a distância entre dois elementos quaisquer e que atendem às propriedades dos espaços métricos. Entre os mais conhecidos e mais utilizados, está a distância de Minkowski, que é uma generalização e engloba outras funções de distância, tais como a distância de Manhattan, a distância Euclidiana e a distância de Chebyshev (METCALF; CASEY, 2016).

2.2.1 DISTÂNCIA DE MINKOWSKI

A distância de Minkowski (ou família Minkowski de distâncias) é uma das funções mais comuns para representar a similaridade entre elementos (JAIN; DUBES, 1988). A relação matemática para essa distância está representada na equação a seguir:

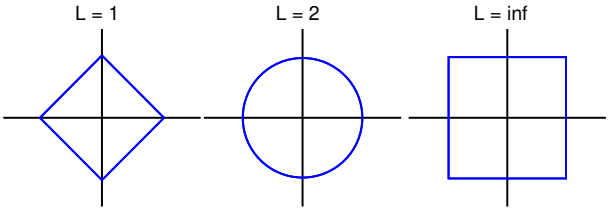
$$d\,(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Devido ao expoente e ao índice da radiciação p, a distância de Minkowski é dita de ordem L^p (METCALF; CASEY, 2016). O valor de p pode variar no intervalo (0, $+\infty$], e para alguns valores específicos, a distância de Minkowski resulta em outras distâncias conhecidas (METCALF; CASEY, 2016; JAIN; DUBES, 1988):

- $p = 1 \longrightarrow$ distância Manhattan;
- $p = 2 \longrightarrow$ distância Euclidiana;
- $p = \infty \longrightarrow$ distância Chebyshev.

Cada expoente afeta o conjunto de pontos equidistantes no mesmo espaço métrico em diferentes maneiras, como é possível enxergar na Figura 1. Substituindo o valor de p na equação da distância de Minkowski pelos valores indicados, a figura formada pelos pontos que estão à mesma distância da origem assume diferentes formatos.

Figura 1 – Casos especiais da distância de Minkowski



Fonte: Autoria própria.

2.3 GRAFO

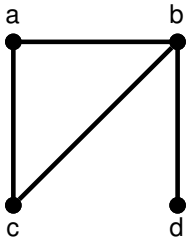
Grafos são estruturas discretas compostas por vértices (pontos ou nós) e arestas (conexões entre os vértices), e podem ser usados para representar e auxiliar na resolução de quase todo e qualquer problema (ROSEN, 2006). Adicionalmente, um grafo é uma abstração matemática da realidade, quando os vértices e arestas recebem um significado intrínseco, relacionado ao nicho do problema (BONDY; MURTY, 1976).

A definição formal de um grafo G, segundo Rosen (2006) e Schaeffer (2007), é um par ordenado $G = (V, E)$, onde V é um conjunto não-vazio de vértices e E é um conjunto de arestas. Bondy e Murty (1976) definem um grafo G como sendo uma tripla ordenada $G = (V, E, \psi_g)$, onde o parâmetro adicional ψ_g é uma função de incidência que associa vértices e arestas; contudo, a

função de incidência nem sempre é abordada ou representada, sendo mais usual a utilização do par ordenado $G = (V, E)$. Um exemplo de grafo é apresentado na Figura 2.

No grafo de exemplo, assumindo a notação $G = (V, E)$, o conjunto de vértices é $V = \{a, b, c, d\}$, e o conjunto de arestas que conecta pares de vértices é $E = \{(a, b), (a, c), (b, c), (b, d)\}$.

Figura 2 – Exemplo de grafo

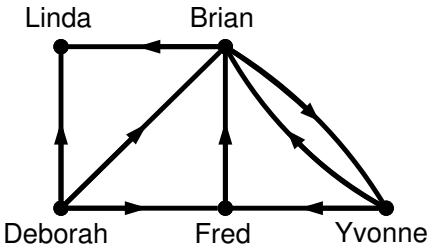


Fonte: Adaptado de (ROSEN, 2006).

Historicamente, o artigo do matemático suíço Leonhard Euler em 1736 sobre as sete pontes de Königsberg é considerada como a primeira publicação relacionada à teoria de grafos, e desde então, principalmente durante o século XX, a teoria de grafos se tornou um importante conjunto de conhecimentos em inúmeras áreas de estudo (BIGGS; LLOYD; WILSON, 1999; NEWMAN, 2003).

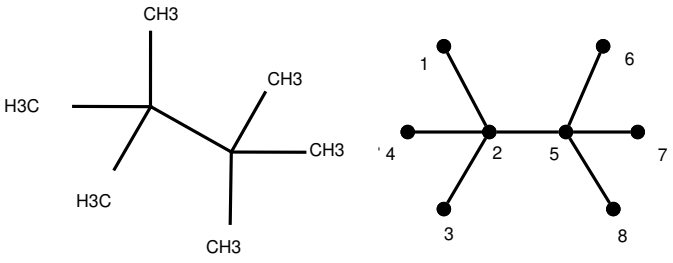
Aplicações dos grafos e da sua teoria incluem áreas como ciências sociais, biológicas e redes de distribuição (NEWMAN, 2003); mapas viários, relacionamentos interpessoais e relações ecológicas (ROSEN, 2006), e diversos usos na química como representações moleculares e enumeração de isômeros (ESTRADA; BONCHEV, 2013; BALABAN, 1985). Apesar da extensa lista de aplicações, a estrutura matemática dos grafos é sempre a mesma, e a teoria de grafos é a formalização de qualquer diferente problema que se deseja visualizar (RUOHONEN, 2013). Exemplos de como os grafos podem ser utilizados nessas aplicações são apresentados nas Figuras 3 e 4, que mostram as relações interpessoais em um grupo de pessoas e a representação molecular.

Figura 3 – Grafo de relações interpessoais



Fonte: Adaptado de (ROSEN, 2006).

Figura 4 – 2,2,3,3-tetrametilbutano e o seu grafo molecular



Fonte: Adaptado de (HOSAMANI, 2016).

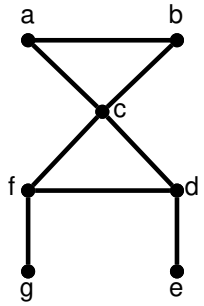
Além da representação, os grafos possuem características que auxiliam na visualização dos problemas. Entre eles, estão a conectividade do grafo e o peso das arestas.

2.3.1 CONECTIVIDADE

A primeira característica relevante dos grafos citada anteriormente é a conectividade. A noção de conectividade em grafos remete à união do grafo e dos vértices entre si, e um grafo conexo é uma única estrutura, não possuindo partes separadas. De forma complementar, a não-conectividade é a divisão do grafo em partes isoladas e não conectadas entre si, chamadas de componentes conexas (BIGGS; LLOYD; WILSON, 1999).

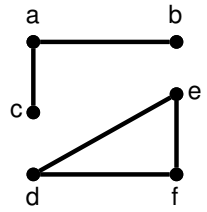
Em resumo, um grafo é dito conexo se, para todo par de vértices distintos $\{(u, v) \mid u \neq v\}$, existe um caminho que permite sair de um vértice e atingir o outro (ROSEN, 2006). Na Figura 5, é possível perceber o grafo é conexo, enquanto o grafo da Figura 6 não atende à condição de conectividade.

Figura 5 – Grafo conexo



Fonte: Adaptado de (ROSEN, 2006).

Figura 6 – Grafo não-conexo

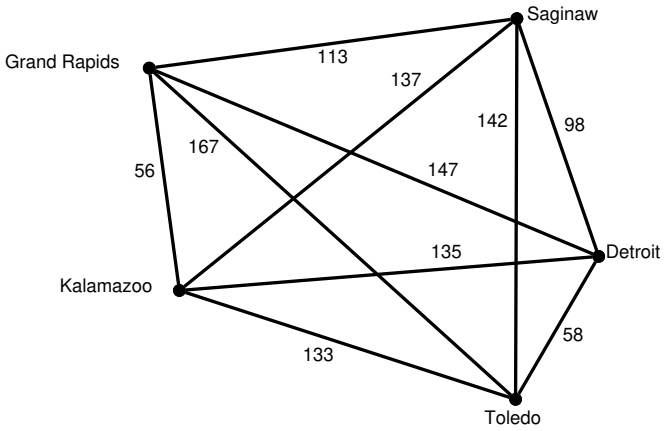


Fonte: Adaptado de (ROSEN, 2006).

2.3.2 PESO

Outra característica relevante dos grafos é o peso. Qualquer grafo pode apresentar nas suas arestas um parâmetro referente ao peso daquela aresta. Exemplos do significado do peso de uma aresta são distâncias, tempos, contagens, intensidade e custos (ROSEN, 2006; BONDY; MURTY, 1976). Diversos problemas representados em grafos utilizam o peso nas arestas como uma métrica de distância física, como o Caixeiro Viajante e o Caminho Mínimo (ROSEN, 2006; BONDY; MURTY, 1976). A Figura 7 traz um exemplo de como o peso das arestas pode ser usado em uma abstração da realidade, como a distância entre algumas cidades.

Figura 7 – Distância entre cidades



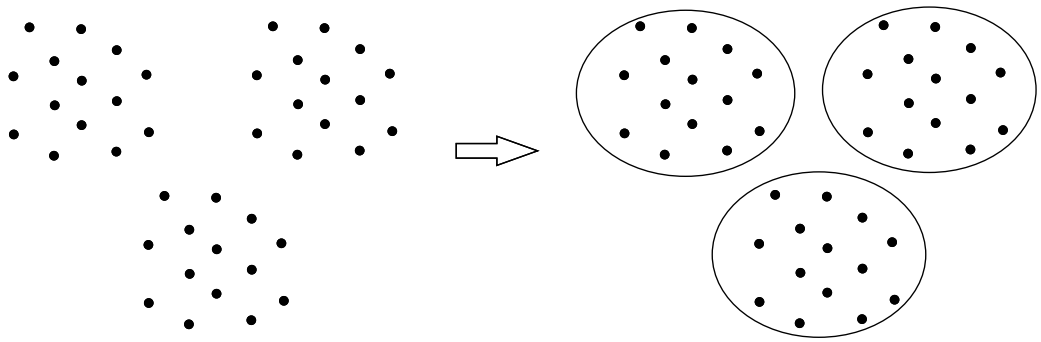
Fonte: Adaptado de (ROSEN, 2006).

2.4 *CLUSTERING*

Clustering é o nome do processo de agrupamento de dados em conjuntos ou grupos (*clusters*), de acordo com algum critério ou objetivo, e é um procedimento fundamental dentro da mineração de dados, mas sua utilização vem crescendo em inúmeras áreas (TAN *et al.*, 2018; OMRAN; ENGELBRECHT; SALMAN, 2007; ROKACH; MAIMON, 2005). Aplicações de métodos de *clustering* incluem aprendizado de máquina, reconhecimento de padrões e inteligência artificial (OMRAN; ENGELBRECHT; SALMAN, 2007); processamento e segmentação de imagens (JAIN; DUBES, 1988), além da utilização em áreas como biologia, psicologia, medicina e climatologia (TAN *et al.*, 2018).

Tradicionalmente, o *clustering* é definido como um método de aprendizado não-supervisionado, que é um método de classificação sem um conjunto de treinamento e sem conhecimento dos objetivos. No entanto, existem abordagens de *clustering* supervisionado, que conhecem alguns parâmetros finais, como o número de *clusters* ou os centroides (FUKUNAGA, 1990; GHAHRAMANI, 2004). A Figura 8 exibe um exemplo de *clustering* que, a partir de dados espalhados no espaço, identifica conjuntos de dados semelhantes e os agrupa.

Figura 8 – Exemplo de *clustering*



Fonte: Autoria própria.

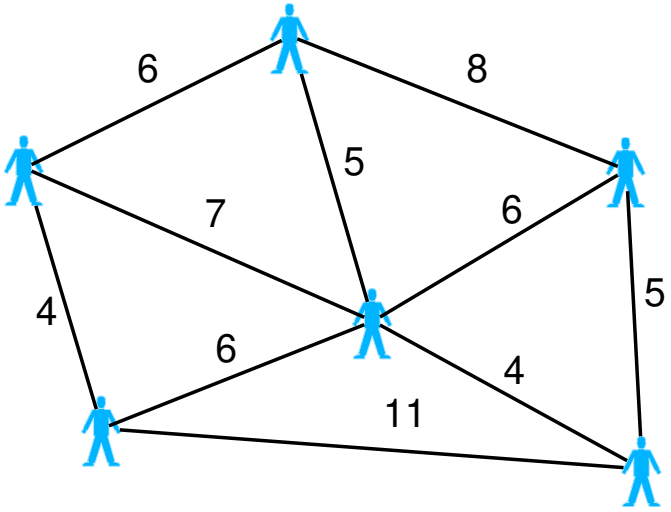
2.4.1 GRAFO DE SIMILARIDADE

Um grafo de similaridade é um grafo cujos vértices representam objetos quaisquer e as arestas representam a presença de similaridade entre os objetos. Além disso, o peso nas arestas é uma forma de quantificar a similaridade (ou dissimilaridade) entre os vértices (POLA *et al.*, 2015; KUANG *et al.*, 2019; SCHAEFFER, 2007; ROKACH; MAIMON, 2005).

A abordagem de problemas utilizando grafos de similaridade pode possibilitar a identificação de estruturas subjacentes e a realização de processos como aprendizado e mineração, e também fornecer estruturas de visualização dos dados (SCHAEFFER, 2007; ROKACH; MAIMON, 2005).

Grafos de similaridade são muito utilizados para resolver problemas em altas dimensionalidades, assim como em problemas com características relacionadas à ciências sociais, como relações interpessoais e redes sociais (BERTOZZI; MERKURJEV, 2019). Na Figura 9, é apresentado um grafo de similaridade que reflete relações em uma rede social, com o peso das arestas representando a similaridade entre diferentes usuários, que pode ser calculada com base no perfil de cada um.

Figura 9 – Exemplo de grafo de similaridade representando uma rede social



Fonte: Autoria própria.

2.5 CONSULTA POR ABRANGÊNCIA (RANGE QUERY)

A R_q, ou consulta por abrangência, é uma consulta por similaridade que é realizada sobre os elementos de um determinado espaço métrico (POLA, 2010). Esse método de consulta recebe dois parâmetros de entrada: um elemento de referência, que pertence ao espaço métrico de interesse e é utilizado como origem para a consulta, e um raio máximo de observação, que é a maior distância possível para que os elementos vizinhos façam parte da vizinhança do elemento

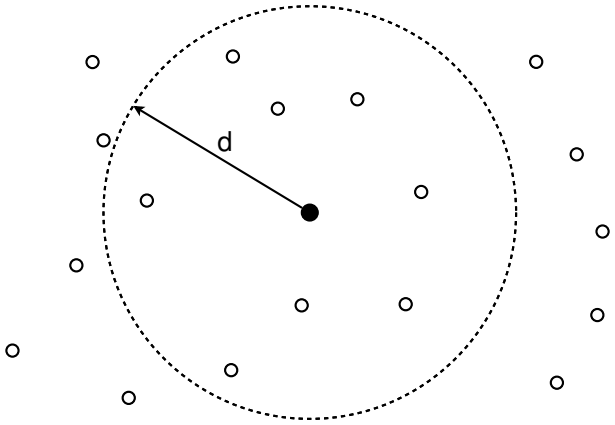
inicial (POLA, 2010). Dessa forma, o resultado de uma consulta por abrangência é uma lista de elementos cuja distância entre eles e o elemento inicial é menor que o raio de consulta.

Utilizando a definição de distância entre os elementos de um espaço métrico, utilizando um elemento de origem x_0 e um raio máximo de observação ε, um elemento qualquer do espaço x_i pertence ao resultado da consulta se ele atende a seguinte relação:

$$d(x_0, x_i) \leq \varepsilon$$

Como representação visual, a Figura 10 demonstra um exemplo de consulta por abrangência com raio d em um espaço bidimensional. Nesse exemplo, a partir de um elemento central (representado por x_0), todos os elementos cuja distância ao ponto de origem seja menor que ou igual ao raio da consulta (ε) fazem parte do conjunto de resposta.

Figura 10 – Exemplo de consulta por abrangência



Fonte: Autoria própria.

2.6 DBSCAN

O DBSCAN (em português, agrupamento espacial baseado em densidade de aplicações com ruído) é um método de análise de agrupamento proposto inicialmente por Ester *et al.* (1996). Partindo de um conjunto de pontos no espaço, esse método tende a agrupar os pontos em regiões de grande densidade, e também consegue identificar ruídos, *outliers* e *clusters* de formatos não-lineares (ESTER *et al.*, 1996).

Esse algoritmo recebe como entrada dois parâmetros: o raio máximo de observação partindo de um determinado ponto do espaço, representado por ε (semelhante ao conceito da consulta por abrangência) e o número mínimo de pontos que devem pertencer à vizinhança definida pelo raio máximo de observação, representado por n (ESTER *et al.*, 1996).

Além de identificar os agrupamentos no espaço, esse método também rotula cada elemento em três categorias: *core* (núcleos), *border* (bordas ou fronteiras) e *noise* (ruídos ou *outliers*). Os pontos *core* são elementos do espaço que possuem (no mínimo) n pontos vizinhos dentro do raio de observação, e representam o núcleo do seu respectivo agrupamento. Os pontos *border* são pontos de fronteira dos *clusters*, e devem estar conectados diretamente em elementos *core*, fazendo parte dos agrupamentos. Os pontos *noise* são considerados ruído no espaço, e não fazem parte de qualquer agrupamento pois não estão localizados no raio ε de um ponto *core*, ou não enxergam ao menos n vizinhos dentro do raio ε para iniciar um novo *cluster* (ESTER *et al.*, 1996).

Como esse método utiliza um conceito semelhante à consulta por abrangência na vizinhança, é possível implementar uma R_q como sub-rotina do DBSCAN.

2.7 CONSULTA AOS *k*-VIZINHOS MAIS PRÓXIMOS

O algoritmo dos *k*-Nearest Neighbors (*k*-NN) (em português, *k*-vizinhos mais próximos) foi proposto inicialmente por Fix e Hodges (1951), e posteriormente estudado por Cover e Hart (1967), e é um método estatístico de classificação ou regressão. O objetivo desse método é categorizar ou estimar o valor de um determinado elemento baseado nos *k* vizinhos mais próximos, onde $k \geq 1$ (FIX; HODGES, 1951; COVER; HART, 1967).

Com um conceito similar, mas um resultado diferente, a consulta aos *k*-vizinhos mais próximos (*k*-NNq) não classifica ou avalia um elemento específico. Essa técnica recebe como parâmetro de entrada um elemento qualquer do espaço, e retorna até os *k* vizinhos mais próximos ($k \geq 1$) (POLA, 2010).

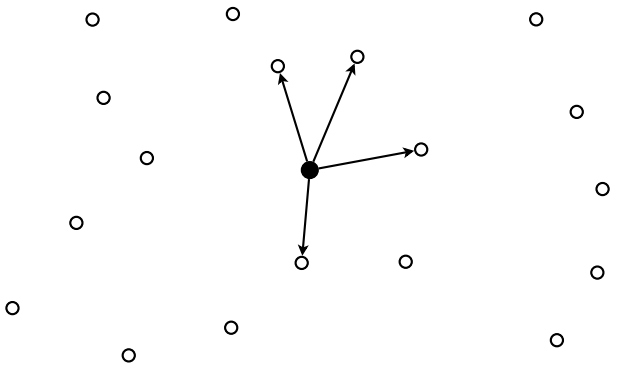
Na Figura 11, é possível observar como a *k*-NNq com $k = 4$ interage com os elementos do espaço, e o retorno ao escolher um elemento qualquer da distribuição. Nesse exemplo bidimensional, os elementos que fazem parte do conjunto de resposta são os k vizinhos mais próximos do elemento central. Ainda vale ressaltar que o elemento central não precisa necessariamente integrar o conjunto de dados, mas deve pertencer ao mesmo domínio (POLA, 2010).

2.8 TÉCNICA OMNI

A técnica OMNI está presente nos trabalhos de Filho *et al.* (2001) e Traina *et al.* (2006), e é um método para otimizar cálculos de distância em uma base de dados.

A ideia central dessa técnica é selecionar certos elementos do conjunto de dados como focos, e utilizar as propriedades dos espaços métricos (principalmente a desigualdade triangular) para reduzir a quantidade de cálculos de distância realizados (FILHO *et al.*, 2001; TRAINA *et al.*, 2006).

Figura 11 – Exemplo de consulta aos k-vizinhos mais próximos



Fonte: Autoria própria.

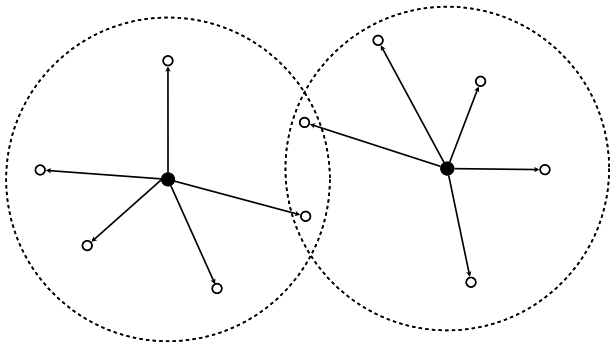
Para selecionar um foco, Filho *et al.* (2001) utiliza o algoritmo *Hull of Foci*, que partindo de um elemento aleatório, escolhe o ponto mais distante desse, e partindo desse novo ponto, escolhe novamente o ponto mais distante.

2.9 FATOR DE SOBREPOSIÇÃO

O fator de sobreposição foi introduzido por Traina *et al.* (2000) como uma forma de mensurar a sobreposição em espaços métricos. Esse fator considera o espaço que é coberto por mais de uma região, ou a área da interseção entre as regiões. Ao invés de calcular áreas no espaço, esse fator considera o número de elementos que estão na região de sobreposição (TRAINA *et al.*, 2000).

Para obter esse valor, a definição do cálculo do fator de sobreposição entre regiões é o número de elementos que está localizado na sobreposição dividido pelo número de elementos nas regiões (TRAINA *et al.*, 2000). Tomando como exemplo a Figura 12, dois *clusters* com 5 elementos cada possuem 2 elementos que estão na área de sobreposição entre eles.

Figura 12 – Sobreposição entre dois *clusters*



Fonte: Autoria própria.

Calculando o fator de sobreposição para os dois vértices na área entre os *clusters*, partindo da definição, o valor obtido é $2/(6+6)$, sendo o numerador correspondente ao número de elementos na sobreposição, e o denominador é a soma do número de elementos em cada *cluster*.

3 O MÉTODO DE *CLUSTERING* PROPOSTO

Partindo dos conceitos introduzidos, o método de *clustering* proposto pretende, a partir de grafos de similaridade extraídos de um espaço métrico, obter *clusters* que agrupam dados similares entre si e dividem os conjuntos de dados que apresentam dissimilaridade.

Além de identificar *clusters* no espaço de entrada, o algoritmo de *clustering* implementado também busca reduzir o fator de sobreposição calculado, limitando o número de elementos que são observados na vizinhança de cada vértice utilizando *k*-NNq. Dessa forma, o raio real de cada *cluster* é reduzido, limitando a área de abrangência de cada agrupamento, e consequentemente, reduzindo a sobreposição entre diferentes *clusters*.

O código completo do algoritmo se encontra em um repositório no GitHub, com os requisitos e um arquivo *readme* com algumas instruções básicas e iniciais para a execução do algoritmo e algumas particularidades. O repositório também conta com um arquivo de pacotes (em versões específicas) que são requisitos para a execução.

As linguagens utilizadas para o desenvolvimento do trabalho foram Python (para a implementação do algoritmo, geração de imagens, cálculo das métricas de avaliação e interação com o banco de dados) e Structured Query Language (SQL) (para consultas, inserções e atualizações no banco de dados).

A linguagem Python foi escolhida pela extensa lista de bibliotecas e a facilidade de acesso ao suporte e documentação, além de ser uma linguagem *open-source* e ter o desenvolvimento realizado pela comunidade (PYTHON SOFTWARE FOUNDATION, 2019). A versão do Python utilizada foi a 3.9.15.

A linguagem SQL é a linguagem padrão para interagir com bancos dados relacionais e manipular registros em seus domínios, sendo tradicional utilizá-la quando se lida com esse tipo de recurso. Nesse trabalho, também foi utilizada uma linguagem particular do PostgreSQL para desenvolvimento procedural, Procedural Language/PostgreSQL (PL/pgSQL). Essa linguagem foi utilizada especificamente para a implementação da função de distância de Minkowski, combinando a sintaxe e facilidade de uso da linguagem SQL e as vantagens computacionais da sua utilização (POSTGRESQL, 2021).

Para utilizar as linguagens SQL e PL/pgSQL, é necessária a conexão com uma instância do PostgreSQL (local ou remota). Para o desenvolvimento do trabalho, foi utilizada uma instância local do PostgreSQL na versão 14. O sistema operacional utilizado para a execução foi o Ubuntu 22.04 LTS. Algumas configurações do banco de dados foram alteradas diretamente na instância para aumentar o desempenho do algoritmo e das conexões, e estão indicadas no repositório como sugestões. As demais interações do código com a instância PostgreSQL (criação de tabelas, inserções, atualizações e buscas) estão centralizadas no algoritmo e não precisam ser feitas externamente. Na seção 3.5 serão apresentadas algumas configurações e ajustes de parâmetros da instância de banco de dados PostgreSQL para melhorar a performance e otimizar algumas operações.

De forma sucinta, a lista a seguir apresenta as etapas de execução do algoritmo, desde a entrada (espaço métrico com vetores de características) até as diversas saídas - a classificação do espaço em *clusters* e a alocação de vértices, as visualizações de cada espaço após a definição dos *clusters* e alocação dos vértices e a avaliação de cada agrupamento distinto.

1. O método proposto recebe como entrada um espaço métrico definido por vetores de características numéricas, normalizados e com o mesmo número de dimensões;

2. A etapa seguinte de execução é a estimativa dos parâmetros, que são utilizados na geração dos conjuntos de similaridade;

3. Após a definição dos parâmetros, os *clusters* são extraídos do espaço inicial de acordo com cada combinação possível dos parâmetros estimados, identificando agrupamentos, alocando os vértices e os classificando;

4. Após a geração de todos os espaços possíveis, caso o espaço seja bidimensional, uma imagem de cada um é gerada para visualização dos resultados;

5. Ao final, todos os conjuntos de similaridade gerados são avaliados, calculando as métricas indicadas para quantificar a qualidade do resultado final de cada agrupamento.

Com as etapas acima brevemente descritas, as seções seguintes irão descrevê-las detalhadamente, explicando os conceitos utilizados e estabelecendo relações com o referencial teórico utilizado como base para o trabalho.

O algoritmo completo do método implementado está apresentado no Apêndice B. O estudo da complexidade do algoritmo está apresentado na Seção 3.2.1, fazendo a análise do pseudocódigo apresentado e estabelecendo uma comparação com algumas decisões específicas da implementação do método proposto.

3.1 ESTIMATIVA DOS PARÂMETROS

A entrada do algoritmo é uma lista com os vetores de características, armazenados em um arquivo (nos testes e para obtenção de resultados, foi utilizado um arquivo no formato *csv*), contendo um identificador para cada vértice e as características numéricas (que serão abstraídas como as coordenadas no espaço métrico). Os vetores devem ter a mesma quantidade de características em todos os elementos, ou seja, estarem normalizados, e podem ter qualquer número de coordenadas. Outro parâmetro de entrada é o índice p da distância de Minkowski, respeitando a regra onde p é um número inteiro e $p \geq 1$.

Seguindo as etapas de execução listadas anteriormente, o próximo passo do método proposto é calcular as distâncias entre todos os pontos (utilizando a distância de Minkowski com expoente p), gerando um grafo completo com todas as distâncias. Esse grafo é armazenado

no banco de dados como uma tabela com três colunas: vértice de origem, vértice de destino e o valor da distância, respeitando aos axiomas do espaço métrico. A escolha do banco de dados para armazenar as distâncias evita que os valores fiquem armazenados em memória (garantindo que o resultado do cálculo não seja perdido) e também elimina a necessidade de calcular novamente uma distância (sendo necessário apenas recuperar do banco de dados).

O código que implementa a função de distância em PL/pgSQL está apresentado integralmente no Apêndice A.

A partir das distâncias calculadas, três listas de parâmetros são determinados com base nas características do próprio espaço. Cada lista possui alguns elementos que são utilizados na etapa de geração dos grafos de similaridade, baseado em combinações de seus valores. Os parâmetros são:

- ε, que é o raio máximo de observação para a vizinhança da consulta por abrangência;
- n, que é o número mínimo de vértices que devem ser encontrados na vizinhança para iniciar ou continuar um *cluster*;
- k, que é o número máximo de vértices mais próximos que serão selecionados da vizinhança utilizando a consulta aos k-vizinhos mais próximos.

Para obter o parâmetro ε, a primeira ação é baseada na técnica OMNI, que utiliza o algoritmo *Hull of Foci* para determinar os focos para minimizar o cálculo de distâncias, utilizando propriedades dos espaços métricos (FILHO *et al.*, 2001).

Seguindo essa técnica, partindo de um ponto aleatório do espaço, a distância para todos os outros pontos é calculada (etapa que já foi realizada anteriormente, e as distâncias estão armazenadas no banco de dados). Dessa lista de distâncias, escolhemos a maior - ou seja, o ponto mais distante do ponto inicial. Então, o processo é repetido a partir do novo ponto, todas as distâncias são calculadas e novamente o ponto mais distante é escolhido.

Na técnica OMNI, o novo ponto selecionado é chamado de foco, e no presente algoritmo ele será a base para extrair os valores de ε. Tomando o novo ponto como referência, os dois primeiros valores selecionados são a média e a mediana da distribuição das distâncias a todos os outros pontos. O restante dos valores é baseado nos percentis da distribuição, considerando os valores de P1 até P5 com incremento de 1%, e os valores de P10 até P45 com incremento de 5%. O valor de P50 é a própria mediana, e por isso não há necessidade de incluí-lo novamente.

A lista de parâmetros de n recebe os valores de $\log_2 t$ e $\ln t$, sendo t o número total de vértices. Como o valor do logaritmo não é inteiro na grande maioria dos casos, o valor é arredondado para cima e para baixo utilizando as funções piso e teto, respectivamente. Caso o valor do logaritmo já seja inteiro, o valor é somado e subtraído de 1 unidade, e os três valores são adicionados à lista.

O parâmetro k considera o valor sugerido pelo próprio método k-NN, que é a $\sqrt{t}$ sendo t o número total de vértices. A fim de obter mais de um valor para a etapa de geração de grafos, o

valor da raiz quadrada é arredondado para os números inteiros mais próximos, também utilizando as funções piso e teto. Caso o resultado da raiz quadrada já seja inteiro, esse valor é adicionado a lista e também são adicionados os resultados da adição e subtração de 1 unidade.

Ao final dessa etapa, três listas com valores variados para cada parâmetro são geradas, e cada grafo de similaridade extraído do espaço utiliza uma combinação dos três parâmetros, entre todas as combinações possíveis. Cada combinação é identificada por um número, que vai de 1 até a multiplicação da quantidade de valores para cada parâmetro.

3.2 EXTRAÇÃO DOS GRAFOS DE SIMILARIDADE

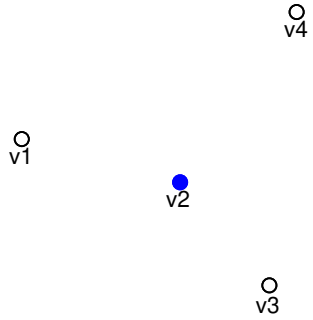
Com as distâncias calculadas e os parâmetros estimados, inicia-se a etapa de extração dos grafos de similaridade, combinando todos os valores dos três parâmetros (ε, n e k) e avaliando o espaço para obtenção dos *clusters* e alocação dos vértices.

O método proposto nesse trabalho utiliza o DBSCAN para identificar os agrupamentos de elementos no espaço métrico, implementando uma consulta por abrangência em cada elemento visitado e uma consulta aos *k*-vizinhos mais próximos como sub-rotinas, com o objetivo de limitar o número de vizinhos e aproximando os elementos mais próximos, eliminando os vizinhos mais distantes da região de abrangência.

A sequência de imagens apresentada nas Figuras 13, 14, 15 e 16 mostra uma iteração do algoritmo, simulando uma execução com os parâmetros de entrada representados por $\varepsilon = d$, $k = 2$ e $n = 2$.

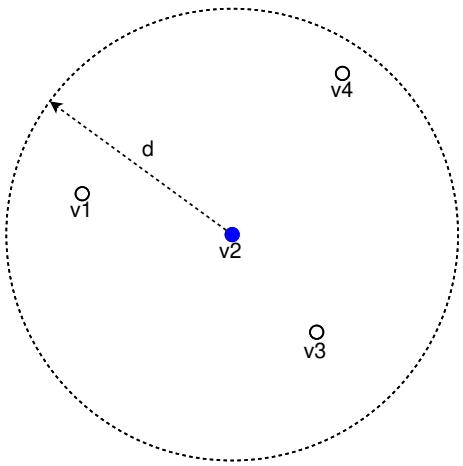
Na Figura 13, o vértice $v2$ é escolhido para iniciar as iterações, sendo destacado na figura em azul. A partir desse vértice, uma consulta por abrangência R_q com raio d é feita, enxergando outros três vértices ($v1$, $v3$ e $v4$), na Figura 14.

Figura 13 – Vértice $v2$ selecionado na iteração



Fonte: Autoria própria.

Figura 14 – Consulta por abrangência com raio d e centro em $v2$

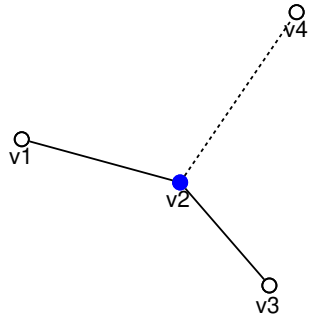


Fonte: Autoria própria.

Com os três vértices selecionados pela R_q, na Figura 15, o vértice $v2$ se torna um núcleo (*core*), pois possui mais vizinhos que o número mínimo de vizinhos definido pelo parâmetro n = 2. No entanto, o *cluster* será iniciado apenas com os dois vizinhos mais próximos, de acordo com o parâmetro k = 2. Ainda na Figura 15, os dois vizinhos mais próximos ($v1$ e $v3$) estão com suas arestas destacadas, enquanto o terceiro vértice ($v4$) pertence à vizinhança mas não será incluído no *cluster*, por isso sua aresta está pontilhada e não será considerada na próxima etapa.

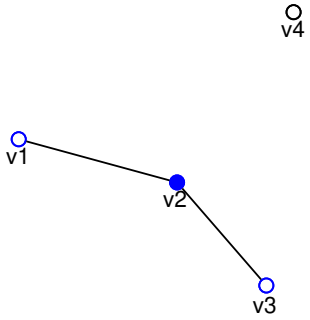
Por fim, na Figura 16, os vértices $v1$ e $v3$ são classificados como bordas (*border*) do *cluster*, e as seguintes iterações irão repetir o mesmo processo para esses vértices, a fim de expandir o *cluster* iniciado em $v2$. Como explicado na Figura 15, o vértice $v4$ não será adicionado nessa iteração pela limitação do parâmetro k.

Figura 15 – Resultado da R_q e a influência de k



Fonte: Autoria própria.

Figura 16 – Resultado final da iteração



Fonte: Autoria própria.

Os grafos de similaridade estão implícitos em cada *cluster*, com arestas que conectam os elementos *core* entre si, ligam vértices *core* e *border* de acordo com os parâmetros estimados, e isola elementos *noise* dos agrupamentos.

Idealmente, cada *cluster* será uma componente conexa do grafo completo baseado no espaço métrico, mas um elemento pode estar em uma região de sobreposição entre dois ou mais *clusters* (posteriormente, o tópico de sobreposição será discutido na seção 3.4.5).

Além disso, considerando que o cálculo de distâncias entre todos os vértices gera um grafo completo, os parâmetros de entrada podem ser interpretados da seguinte maneira:

- ε é o valor máximo da distância entre dois vértices para possibilitar a existência de uma aresta, ou o peso máximo das arestas;
- k é o intervalo de valores que os graus dos vértices podem possuir, variando no intervalo [0, k], sendo 0 o grau de um vértice *noise* e k o maior grau possível dos outros vértices, considerando os vizinhos mais próximos;
- n é o grau mínimo que um vértice deve assumir para ser considerado *core* e iniciar ou continuar um *cluster*.

3.2.1 COMPLEXIDADE DO ALGORITMO

Com base no livro de Toscani e Veloso (2012) e na análise de código, a complexidade estimada de cada algoritmo segue a seguinte ordem, considerando n como o número total de elementos do espaço:

- DBSCAN: todos os vértices devem ser visitados ao menos uma vez (representando uma complexidade inicial $\Omega(n)$), e cada vértice é expandido nas consultas por abrangência internas. Caso as consultas internas visitem todos os vértices, a complexidade observada é $\Theta(n^2)$, sendo esse o pior caso, embora existam variações que permitam atingir uma complexidade de $O(n\log(n))$ (ESTER *et al.*, 1996).

- Consulta por abrangência (R_q): para um determinado elemento de origem, as distâncias para todos os outros elementos devem ser calculadas e comparadas com a distância de corte (raio de observação ε). Por isso, a complexidade desse método é no mínimo $\Omega(n)$, além do cálculo da distância. No método implementado, a distância já foi calculada e armazenada, substituindo todos os cálculos por consultas para recuperar o valor das distâncias diretamente do banco de dados.

- Consulta aos *k*-vizinhos mais próximos: semelhante à consulta por abrangência, para um elemento de origem, também é necessário calcular as distâncias para todos os outros elementos, resultando em uma complexidade $\Theta(n)$, com a mesma observação da distância já calculada e armazenada. No entanto, métodos de ordenação são usualmente utilizados para ordenar a lista de distâncias e então cortá-la no *k*-ésimo elemento. A complexidade média de alguns dos algoritmos de ordenação mais eficientes é $O(n\log(n))$ (BAASE, 1988). O método implementado para cortar o vetor de distâncias no *k*-ésimo maior elemento é um vetor dinâmico de *k* posições (e não *n* posições, reduzindo o número de comparações necessárias para trazer os *k* vizinhos mais próximos), reduzindo a complexidade nos melhores casos para $\Omega(n)$, e nos piores casos, $\Theta(n\log(k))$.

Considerando as complexidades listadas acima, é possível inferir que nos piores casos, o algoritmo implementado possui uma complexidade de $\Theta(n^2)$ do DBSCAN, $\Theta(n)$ da consulta por abrangência e $\Theta(n\log(k))$ da consulta aos *k*-vizinhos mais próximos, resultando em uma complexidade total de $\Theta(n^4\log(k))$, e nos melhores casos possíveis, $\Omega(n^4)$.

Como a maioria das informações que são acessadas repetidamente estão armazenadas no banco de dados (como as distâncias e informações dos elementos), a complexidade de armazenamento em memória e a complexidade de acesso direto não foram consideradas nessa análise.

3.3 VISUALIZAÇÃO DOS GRAFOS DE SIMILARIDADE E *CLUSTERS* IDENTIFICADOS

Para visualizar os resultados obtidos, uma etapa de geração de imagens foi adicionada ao algoritmo. No entanto, essa etapa está condicionada apenas aos espaços com duas dimensões. Também seria possível plotar espaços com três dimensões, mas para facilitar a interpretação visual dos resultados, os espaços tridimensionais não serão explorados de forma gráfica.

Existem algumas maneiras de visualizar mais de 3 dimensões em um gráfico, mas essa ação pode distorcer os resultados e aumentar a complexidade de interpretação, e como a etapa de geração de imagens de espaços tridimensionais foi descartada, não há motivo para gerar a visualização dos resultados para maiores dimensões.

3.4 MÉTRICAS DE AVALIAÇÃO

Após a execução do algoritmo e a etapa de geração de imagens (para espaços bidimensionais), algumas métricas de avaliação foram escolhidas para interpretar de forma numérica os resultados, e determinar de forma objetiva (não apenas visual) os melhores grafos de similaridade gerados.

Apesar dos métodos de cálculo serem os mesmos para todos os espaços, as métricas geradas para grafos diferentes não devem ser comparadas, já que cada espaço possui parâmetros diferentes, distribuições únicas e características distintas. O objetivo do cálculo das métricas é comparar os resultados extraídos de um mesmo espaço, identificando as combinações de parâmetros que resultam nos melhores agrupamentos.

Das métricas utilizadas, foram definidas duas métricas de contagem (de *clusters* e de elementos *noise*), duas somas de distâncias (*intracluster* e *extracluster*) e uma métrica proporcional à sobreposição (fator de sobreposição). As métricas serão abordadas individualmente e explicadas nas subseções seguintes.

3.4.1 NÚMERO DE *CLUSTERS*

A primeira métrica utilizada para avaliar a qualidade do resultado final é o número de *clusters* identificados. A ideia de contar o número de agrupamentos é identificar de maneira simples quais parâmetros resultaram em um número maior de *clusters*. O pior resultado possível para essa métrica é 0, ou seja, quando no espaço não foram identificados *clusters*, então todos os elementos do espaço são *noise*.

Como ela é uma métrica de contagem, os melhores resultados práticos possíveis dependem da distribuição de cada espaço métrico e de cada grafo de similaridade extraído de todas as combinações de parâmetros possíveis, o que dificulta a obtenção de um valor ótimo absoluto ou a quantificação de uma métrica de corte que possa ser traduzida em uma variável qualitativa.

Apesar disso, é possível estabelecer um limite teórico máximo para o valor dessa métrica é o valor inteiro da divisão do número total de elementos (N) do espaço dividido pelo número mínimo de vizinhos (n) para iniciar um *cluster* acrescidos de uma unidade (representando o vértice *core* que irá iniciar o agrupamento). A relação matemática que representa esse valor é $\lfloor N/(n+1) \rfloor$, considerando um espaço ideal onde não existam vértices *noise* e o fator de sobreposição seja 0.

3.4.2 NÚMERO DE ELEMENTOS *NOISE*

Assim como a contagem do número de *clusters*, outra métrica de avaliação de contagem é a quantidade de elementos rotulados como *noise*, que não estão alocados em qualquer agrupamento identificado. Partindo dessa definição, o pior valor possível é o próprio número de elementos do espaço.

O melhor resultado possível é 0, quando nenhum elemento do espaço é ruído, ou todos os elementos estão alocados em seus respectivos *clusters*. No entanto, todos os elementos podem estar alocadas no mesmo *cluster*, e apesar do valor dessa métrica ser o melhor possível, a métrica anterior retorna o segundo pior valor, sendo importante analisar as duas métricas em conjunto.

3.4.3 DISTÂNCIA *INTRACLUSTER*

A distância *intracluster* é a soma absoluta de todas as distâncias entre todos os elementos para todos os elementos do seu respectivo *cluster*. A análise dessa métrica é feita em conjunto com a métrica seguinte, a distância *extracluster*.

O objetivo dessa métrica é atingir o menor valor possível, ou seja, quando os elementos de cada *cluster* estão muito próximos, a distância entre eles será relativamente pequena, diminuindo o valor total da soma.

3.4.4 DISTÂNCIA *EXTRACLUSTER*

A distância *extracluster* é a soma absoluta de todas as distâncias entre todos os elementos para todos os elementos que não pertencem ao seu respectivo *cluster*, incluindo ruídos.

Ao contrário da métrica anterior, o objetivo dessa métrica é atingir o maior valor possível para a soma, representando o afastamento entre os elementos de *clusters* distintos.

As duas métricas de distância são inversamente proporcionais entre si, e podem sofrer variações com diferentes alocações de vértices nos *clusters* com base nos parâmetros de entrada.

3.4.5 FATOR DE SOBREPOSIÇÃO

O fator de sobreposição (que pode ser traduzido como *overlap factor*) é um valor que relativiza o número de elementos do espaço que estão na área de sobreposição entre dois *clusters* (em outras palavras, são enxergados como vizinhos por elementos *core* de mais de um *cluster*).

Esse fator foi derivado do *fat-factor*, proposto por Traina *et al.* (2000). Originalmente, esse cálculo é realizado utilizando parâmetros que quantificam acessos em disco ou são derivados de outras estruturas de dados (como as árvores *slim-trees* utilizadas no trabalho de referência), o que dificulta o seu uso como proposto.

Interpretando o objetivo do cálculo e estabelecendo um paralelo ao algoritmo proposto nesse trabalho, o cálculo do fator de sobreposição foi simplificado. Dessa forma, o cálculo passa a ser realizado para cada elemento (e não para a estrutura completa). Para cada elemento que está na sobreposição de 2 ou mais *clusters*, o fator de sobreposição para esse elemento é o inverso da soma do número de elementos de cada *cluster*.

Por exemplo, se um elemento está na sobreposição entre dois *clusters* com 4 e 5 elementos, o fator de sobreposição desse elemento é $1/(4+5)$, ou seja, $1/9$. O fator final é a soma de todos os fatores de sobreposição individuais.

O melhor resultado possível para o fator é 0, quando nenhum elemento está em áreas de sobreposição. Em teoria, não existe um valor máximo para esse fator, pois ele é uma soma de frações individuais cujo resultado final pode ultrapassar 1. Novamente, esse valor deve ser analisado em conjunto com as outras métricas, pois não existe sobreposição em um espaço com 0 ou 1 *clusters*.

3.5 CONFIGURAÇÕES DA INSTÂNCIA POSTGRESQL

Ao fazer a instalação do PostgreSQL na versão indicada, as configurações da instância estão definidas com os valores padrão. Alguns desses valores influenciam diretamente na performance das operações realizadas durante a execução do algoritmo, e outras trazem melhorias no uso e gerenciamento de recursos. Além das configurações que serão apresentadas a seguir, todas as tabelas utilizadas foram indexadas e as consultas fazem uso dos índices para garantir um ganho de tempo e performance.

As configurações do PostgreSQL estão em um arquivo chamado *postgresql.conf*, e uma maneira de alterar esses parâmetros é editar esse arquivo. Outro modo de alterar os parâmetros é diretamente via SQL, por um usuário que tenha permissão de executar consultas no formato apresentado a seguir:

```
1 ALTER SYSTEM SET parameter_name = parameter_value;
```

Assim, as seguintes alterações foram realizadas na instância utilizando SQL, sendo listadas no formato nome do parâmetro e seu respectivo valor, com um comentário explicando sua função:

```
-- desabilita o scan sequencial, forçando o uso de índices
ALTER SYSTEM SET enable_seqscan = off;

-- aumenta o número máximo de conexões ao banco de dados
ALTER SYSTEM SET max_connections = 100000;

-- utiliza o padrão posix para o uso de memória compartilhada
ALTER SYSTEM SET dynamic_shared_memory_type = posix;

-- configurações de workers para processamento em paralelo
ALTER SYSTEM SET max_worker_processes = 8;
ALTER SYSTEM SET max_parallel_workers_per_gather = 2;
ALTER SYSTEM SET max_parallel_maintenance_workers = 2;
ALTER SYSTEM SET max_parallel_workers = 8;

-- configurações de buffers, cache, logs e memória de trabalho
ALTER SYSTEM SET shared_buffers = 256MB;
ALTER SYSTEM SET work_mem = 256MB;
ALTER SYSTEM SET maintenance_work_mem = 128MB;
ALTER SYSTEM SET wal_buffers = 8MB;
ALTER SYSTEM SET max_wal_size = 1GB;
ALTER SYSTEM SET min_wal_size = 100MB;
ALTER SYSTEM SET effective_cache_size = 768MB;
```

Os resultados apresentados no capítulo 4 foram realizados com as configurações acima, e as que não foram mencionadas continuam com o valor padrão definido na instalação.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Para obtenção dos resultados, foram utilizadas quatro instâncias de teste, sendo três identificadas de acordo com a figura que os elementos formam no espaço, e uma aleatória:

- *Butterfly*, pois a disposição de seus elementos lembra uma borboleta;
- *Donuts*, sendo 2 círculos concêntricos de raios diferentes e um ponto isolado no centro do espaço;
- *Four squares*, com 4 quadrados no espaço (e um elemento extra no centro de cada um), dispostos nos cantos do espaço;
- Aleatória com 64 vértices.

As três instâncias definidas possuem, respectivamente, 40, 49 e 20 elementos no espaço e, para a extração de imagens, todas são bidimensionais. Para obtenção e demonstração dos resultados, alguns exemplos de cada um dos espaços de teste serão utilizados.

Outro parâmetro definido para os testes é a ordem da distância de Minkowski, cujo valor foi estabelecido como 2. Esse valor foi escolhido para facilitar a visualização dos resultados, já que os pontos equidistantes da origem formam um círculo.

4.1 VISUALIZAÇÃO DOS RESULTADOS

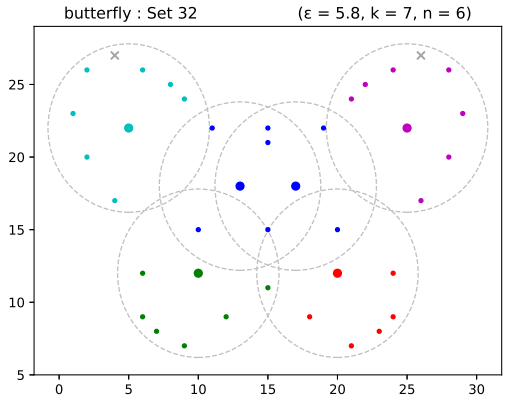
Como já discutido anteriormente, a visualização dos resultados é obtida apenas a partir de espaços bidimensionais, cuja intenção é representar os resultados matemáticos e numéricos de uma maneira humanamente legível e interpretável. Em cada uma das imagens extraídas, estão identificados o espaço de teste, o índice identificador da combinação de parâmetros, e os parâmetros de entrada.

Exemplos de visualização de cada um dos espaços de teste podem ser vistos nas Figuras 17, 18, 19 e 20.

Os três tipos de vértices são representados de maneiras diferentes nas imagens: os vértices *noise* são representados por um x, significando um "descarte", enquanto os vértices *border* e *core* são os círculos preenchidos, sendo os *core* de maior tamanho para destacá-los, já que é a partir deles que os *clusters* são iniciados.

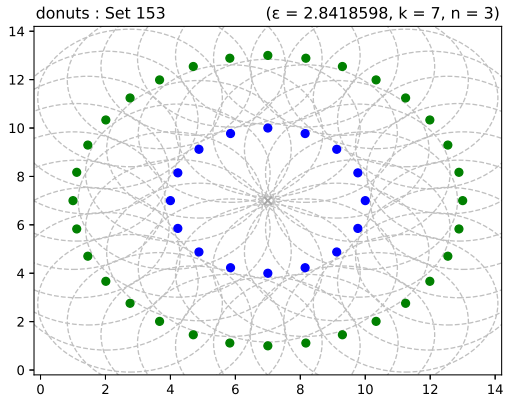
Cada *cluster* é identificado com uma cor diferente, e os raios de observação a partir dos pontos *core* estão delimitados por uma linha pontilhada.

Figura 17 – Instância de teste *Butterfly*



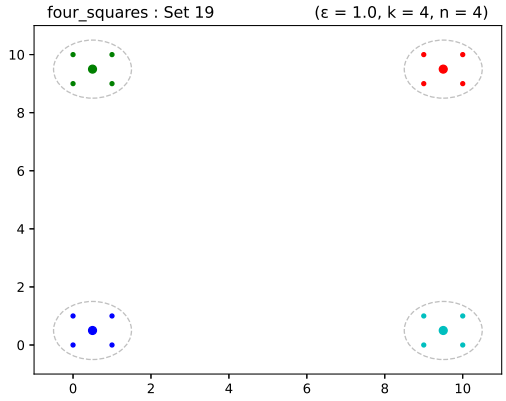
Fonte: Autoria própria.

Figura 18 – Instância de teste *Donuts*



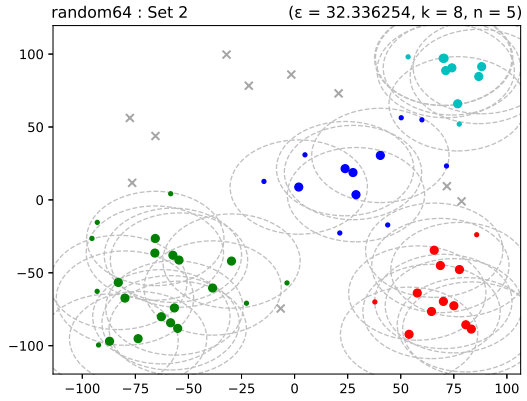
Fonte: Autoria própria.

Figura 19 – Instância de teste *Four squares*



Fonte: Autoria própria.

Figura 20 – Instância aleatória de teste



Fonte: Autoria própria.

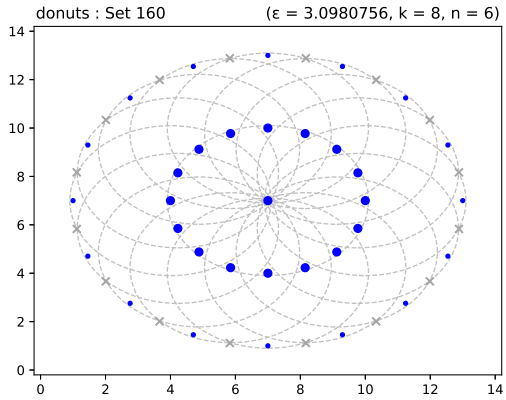
4.2 INFLUÊNCIA DOS PARÂMETROS DE ENTRADA

Como cada combinação possível de parâmetros é testada, cada uma exerce uma influência diferente no algoritmo. Nas seções abaixo, a variação de cada um dos parâmetros será abordada de forma mais detalhada, mantendo os outros parâmetros constantes e avaliando a maneira como cada um interage nas instâncias de teste.

4.2.1 VARIAÇÃO DE ε

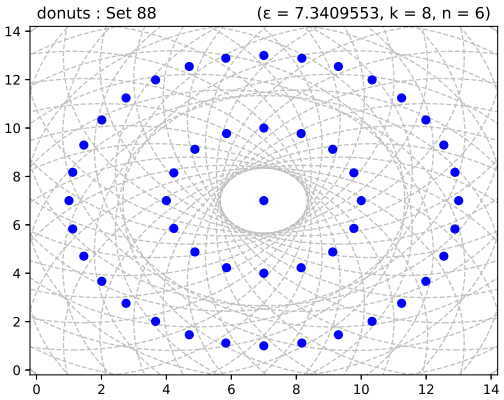
Variar o parâmetro ε aumenta o raio máximo de observação onde cada elemento busca encontrar vizinhos. Mantendo k e n constantes, é possível enxergar casos onde pontos *noise* e *border* passam a ser considerados *core*, como ilustram as Figuras 21 e 22.

Figura 21 – Variação do parâmetro ε



Fonte: Autoria própria.

Figura 22 – Variação do parâmetro ε



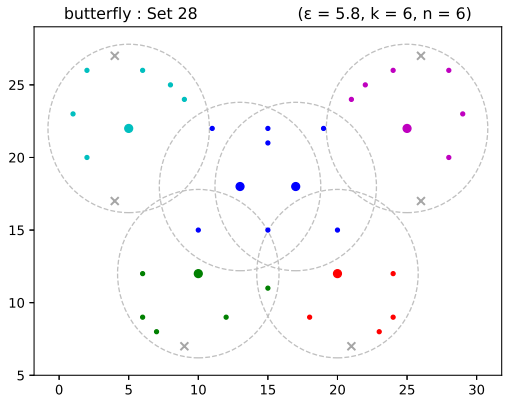
Fonte: Autoria própria.

4.2.2 VARIAÇÃO DE k

A variação de k altera apenas o número máximo de vizinhos mais próximos que cada elemento considera. Esse parâmetro não interfere diretamente impedindo ou favorecendo a formação de *clusters*, mas diferentes valores podem influenciar no resultado da métrica *intracluster*, apresentada na seção 3.4.3.

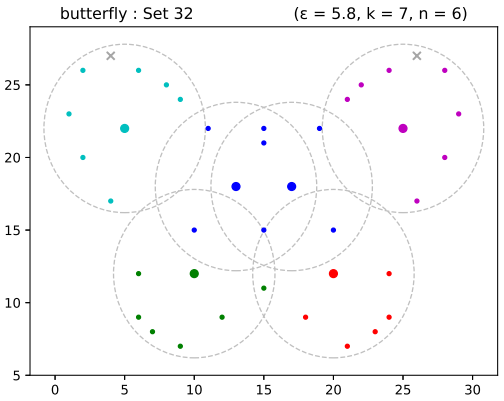
Nas instâncias de teste, um exemplo de alteração nos resultados devido à variação de k está representada nas Figuras 23 e 24, onde alguns pontos são o *k*-ésimo vizinho mais próximo dos elementos *core* e ao diminuir o valor de k, eles são considerados *noise* ao invés de *border*.

Figura 23 – Variação do parâmetro k



Fonte: Autoria própria.

Figura 24 – Variação do parâmetro k

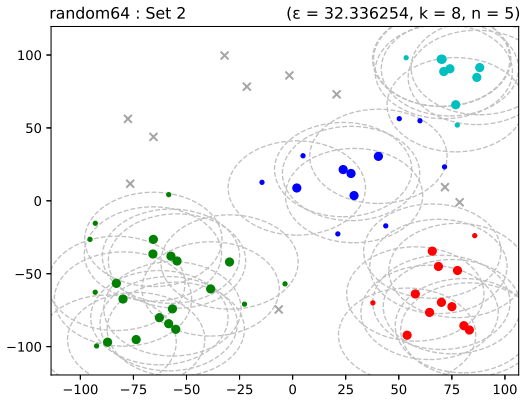


Fonte: Autoria própria.

4.2.3 VARIAÇÃO DE n

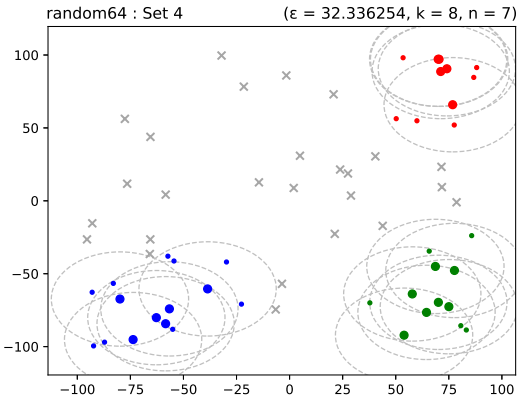
O parâmetro n influencia diretamente na identificação de *clusters* no espaço, pois ele é o número mínimo de vizinhos que devem ser encontrados no raio ε para que um agrupamento seja iniciado. Dependendo do seu valor, impede a formação ou expansão dos *clusters*, e em alguns casos, constrói um único *cluster* contendo todos ou a maioria dos elementos. Nas Figuras 25 e 26, é possível perceber que os parâmetros ε e k favorecem a formação de *clusters* mas a alteração no valor de n reduz o número de elementos do agrupamento no canto esquerdo inferior, e destrói o agrupamento central.

Figura 25 – Visualização com menor valor de n



Fonte: Autoria própria.

Figura 26 – Visualização com maior valor de n



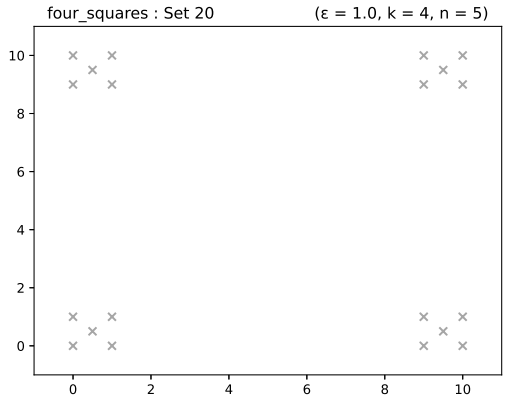
Fonte: Autoria própria.

4.2.4 RELAÇÃO ENTRE k e n

Existe uma relação entre os parâmetros k e n que influenciam diretamente se o algoritmo identificará *clusters* no espaço. Para combinações onde k é menor que n, é impossível que conjuntos sejam formados, já que o número máximo de vizinhos retornado pela k-NNq nunca será maior que ou igual ao valor mínimo de vizinhos, que é obrigatório para iniciar um agrupamento. A relação entre esses parâmetros também pode causar diferenças na categorização dos elementos em *cores* e *borders*.

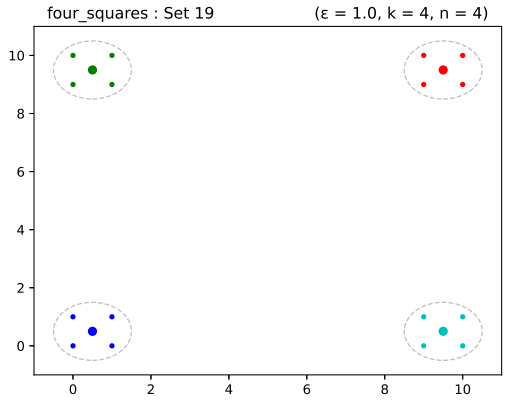
Tais ocorrências podem ser observadas comparando as Figuras 27, 28 e 29, mantendo o parâmetro ε constante e variando a relação entre k e n.

Figura 27 – Visualização para $k < n$



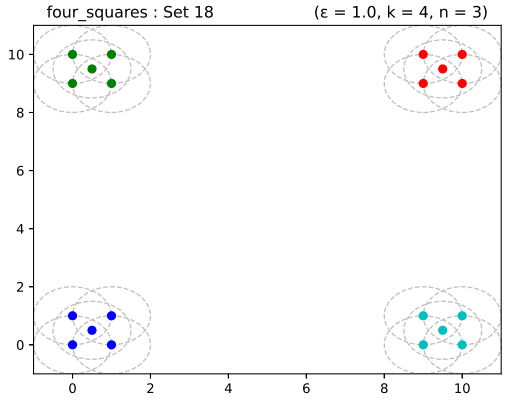
Fonte: Autoria própria.

Figura 28 – Visualização para $k = n$



Fonte: Autoria própria.

Figura 29 – Visualização para $k > n$



Fonte: Autoria própria.

4.3 MÉTRICAS DE AVALIAÇÃO

Para exemplificar como as métricas de avaliação se relacionam com a visualização do resultado, um espaço que possui algum valor representativo em todas as métricas foi escolhido, entre todos os resultados gerados. A Tabela 1 mostra todas as métricas calculadas, e a Figura 30 é a visualização dos *clusters* gerados.

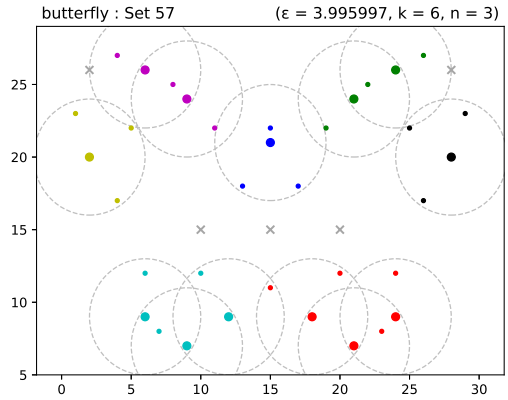
Tabela 1 – Exemplo de métricas calculadas

Instância	*Clusters*	Ruídos	Dist. *Intracluster*	Dist. *Extracluster*	Fator de Sobreposição
57	7	5	657,7898	15.819,722	0,07692308

Fonte: Autoria própria (2023).

Das métricas avaliadas, os 7 clusters e os 5 vértices *noise* são rapidamente identificados. Outra métrica de interpretação rápida é o fator de sobreposição, pois existe apenas um elemento

Figura 30 – Espaço escolhido para visualização das métricas



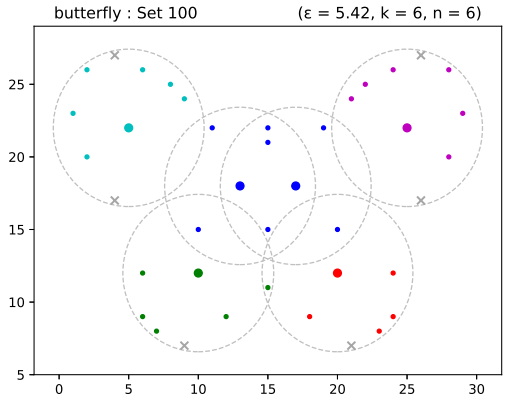
Fonte: Autoria própria.

que está na área de sobreposição entre dois *clusters*, na metade inferior do espaço. O número de elementos dos dois *clusters* é 13 (6 + 7), resultando em um fator de sobreposição de exatamente 1/13, que é o valor calculado pela métrica.

As métricas de distância são as mais complexas de interpretar e comparar o resultado numérico com o visual, mas como os *clusters* são pequenos e possuem poucos elementos, é esperado que a soma das distâncias *intracluster* seja muito menor que a soma das distâncias *extracluster*.

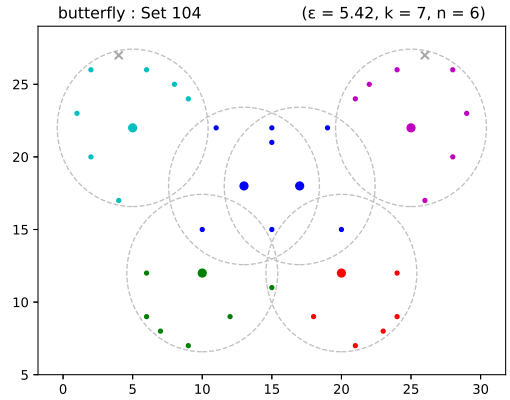
Para efeitos de comparação, outras duas instâncias do espaço da borboleta foram escolhidas para terem suas métricas comparadas. As visualizações estão apresentadas nas Figuras 31 e 32, e as métricas são apresentadas na Tabela 2.

Figura 31 – Instância 100 do espaço *Butterfly*



Fonte: Autoria própria.

Figura 32 – Instância 104 do espaço *Butterfly*



Fonte: Autoria própria.

Embora o resultado visual dos dois conjuntos de *clusters* seja muito próximo, algumas diferenças existem se as métricas calculadas forem observadas. Para a análise das métricas, as instâncias serão identificadas pelos seus respectivos identificadores únicos (100 e 104).

Para as duas métricas de contagem, o número de *clusters* é o mesmo, mas o número de elementos *noise* diminui de forma relevante, de 6 na instância 100 para 2 na instância 104.

Nas métricas de distância, a distância *intracluster* calculada beneficia a instância 100, já que o objetivo é minimizá-la o máximo possível. Por outro lado, a distância *extracluster* favorece a instância 104, pois essa métrica é maior que a calculada para a instância 100. O aumento nas métricas de distância da instância 104 pode ser explicado pela alocação de 4 vértices *noise* em *clusters*, pois a quantidade de distâncias para somar também aumenta.

Por fim, o fator de sobreposição calculado é muito próximo para os dois agrupamentos, mas o fator da instância 104 é ligeiramente menor que o fator da instância 100. Como mais vértices foram alocados, isso significa que os novos *clusters* reduziram o fator, o que melhora a métrica em favor da instância 104.

Com base nessas análises, comparando os dois espaços acima entre si, é possível afirmar que o conjunto 104 é melhor que o conjunto 100, com base nas métricas que foram calculadas.

4.3.1 ANÁLISE DAS MÉTRICAS

Como um dos focos do trabalho é o cálculo das métricas de avaliação, com ênfase na introdução do fator de sobreposição, essa seção será dedicada à análise e discussão das métricas calculadas para algumas instâncias (em conjunto com os parâmetros de entrada e outras métricas de avaliação).

Uma amostra de alguns desses resultados estão apresentadas na Tabela 2, que traz o número identificador da instância, os parâmetros de entrada que geraram esse conjunto de *clusters* (ε, k e n), e algumas métricas de avaliação que serão relevantes para a discussão, que são o número de *clusters*, o número de vertices *noise* (ou ruídos) e o fator de sobreposição calculado. Todos os resultados apresentados na amostra são de espaços com no mínimo 3 *clusters*.

Comparando os fatores de sobreposição apresentados na tabela, o menor valor foi obtido em duas instâncias (92 e 96), cujo fator de sobreposição é igual a zero, que é o melhor valor possível. No entanto, ambas possuem a contagem de ruídos mais alta dessa amostra, e a menor contagem de *clusters* identificados. Isso significa que, apesar dessas instâncias possuírem o menor fator, outras métricas indicam que a qualidade do resultado é inferior em comparação às outras instâncias dessa mesma amostra.

O segundo menor fator de sobreposição é o das instâncias 91 e 95. As duas apresentam a mesma quantidade de *clusters* que as entradas com fator 0, mas a contagem de ruídos é muito menor, sendo a segunda menor contagem de ruídos da amostra. Isso sugere que, apesar de o

Tabela 2 – Fator de comparação para diferentes parâmetros

Instância	ε	k	n	*Clusters*	Ruídos	Fator de Sobreposição
28	5,8	6	6	5	6	0,22900432
32	5,8	7	6	5	2	0,20608975
57	3,995997	6	3	7	5	0,07692308
61	3,995997	7	3	7	5	0,07692308
91	5,04	6	5	3	4	0,06451613
92	5,04	6	6	3	17	0
95	5,04	7	5	3	4	0,06451613
96	5,04	7	6	3	17	0
100	5,42	6	6	5	6	0,22900432
104	5,42	7	6	5	2	0,20608975

Fonte: Autoria própria (2023).

fator apresentado ser ligeiramente maior, a qualidade do resultado do processo de *clustering* tende a ser melhor.

Um comportamento semelhante é observado para o terceiro menor valor de sobreposição da amostra, mas com mais *clusters* e um valor próximo de ruídos. As duas instâncias com esse valor (57 e 61) possuem o raio de observação ε e o número mínimo de vizinhos n menores que o das instâncias anteriores. Apesar da diferença nos parâmetros de entrada, o valor do fator de sobreposição é próximo do segundo menor valor.

Os outros dois valores distintos do fator de sobreposição presentes na tabela são os mais altos, assim como os parâmetros ε do raio de entrada desses espaços. Conforme o raio de observação aumenta, a R_q tende a incluir mais elementos nas vizinhanças de cada vértice da distribuição, o que impacta diretamente o numerador do fator de sobreposição, que é o número de vértices na área de sobreposição entre *clusters*, resultando em um fator de sobreposição maior nesses casos.

Para comparar com as métricas obtidas pelo método implementado, as métricas de avaliação apresentadas no trabalho também foram calculadas para alguns resultados do DBSCAN. A Tabela 3 mostra uma amostra das métricas calculadas para os agrupamentos resultantes do DBSCAN, utilizando os mesmos parâmetros de entrada ε e n:

Tabela 3 – Tabela de métricas do DBSCAN

ε	n	*Clusters*	Ruídos	Fator de Sobreposição
3,995997	3	4	0	0
5,04	5	1	2	-
5,04	6	3	4	0,06451613
5,42	6	1	0	-
5,8	6	1	0	-

Fonte: Autoria própria (2023).

Da tabela, três fatores de sobreposição não podem ser calculados pois o espaço possui apenas 1 *cluster*, e dessa forma, é impossível existir sobreposição. Como as métricas da Tabela

2 indicam que foram identificados mais *clusters* para os mesmos parâmetros de entrada, os resultados obtidos pelo método de *clustering* implementado neste trabalho atingiram resultados com maior qualidade, demonstrando a influência de um limite superior (pela aplicação do parâmetro k e da k-NNq) no método de *clustering*.

Para o resultado cujo fator de sobreposição é zero, na primeira linha da Tabela 3, as duas instâncias apresentadas na Tabela 2 com os mesmos parâmetros de entrada (57 e 61) identificaram mais *clusters* no espaço. Entretanto, alguns vértices não foram alocados, ao contrário do resultado do DBSCAN, em que todos os vértices foram associados aos seus agrupamentos. Os resultados dos dois métodos apresentam vantagens em algumas métricas, mas o fator de sobreposição menor do DBSCAN garante que não existem vértices em áreas de sobreposição entre mais de um *cluster*.

Para o outro resultado da amostra com fator de sobreposição calculado e maior que zero, um resultado com valor igual foi obtido na Tabela 2 para o mesmo ε, mas com n menor. Para os mesmos parâmetros, o fator de sobreposição é 0 mas o espaço possui uma proporção muito alta de ruídos (17 ruídos dos 40 vértices do espaço), o que não é ideal. Apesar das diferentes quantidades de *clusters* identificados e de vértices *noise*, e a variação nos parâmetros de entrada, ambos os métodos atingiram o mesmo fator de sobreposição, o que mostra a equivalência nos resultados.

4.4 TEMPO DE EXECUÇÃO

Um fator importante para analisar na performance de um algoritmo é o tempo de execução. Durante o processo, quatro intervalos de tempo são mensurados utilizando métodos do módulo *time*, que é nativo do Python. Os intervalos são:

- Tempo para estimar os parâmetros de entrada (ε, k e n);
- Tempo para execução do algoritmo de *clustering*;
- Tempo para gerar as visualizações (utilizando espaços bidimensionais);
- Tempo para obtenção das métricas de avaliação.

Cada um dos intervalos de tempo mencionados acima estão apresentados na Tabela 4, cujo valor representa a média de 10 execuções realizadas para aumentar a confiabilidade dos valores obtidos. Além dos valores para as três instâncias de teste (*butterfly*, *donuts* e *four squares*), também foram gerados espaços aleatórios com números específicos de elementos (todos com duas dimensões), para avaliar a performance em cada etapa.

Para as instâncias com mais de 250 elementos, o cálculo do fator de sobreposição foi desconsiderado, baseado no aumento de tempo das instâncias de teste para a instância com 100 elementos.

Tabela 4 – Tempo de execução das etapas do algoritmo

		Tempos (s)				
Instância	Elementos	Estimativa	Execução	Visualização	Avaliação	Avaliação sem F.S.
Four squares	20	0,040	3,014	0,808	5,794	0,890
Butterfly	40	0,040	6,712	0,942	9,968	1,515
Donuts	49	0,041	7,440	0,986	29,738	1,804
Aleatória	100	0,043	11,823	1,708	173,139	4,269
Aleatória	250	0,046	31,150	3,423	-	10,824
Aleatória	500	0,037	40,280	5,201	-	14,025
Aleatória	1000	0,042	90,870	10,315	-	28,225
Aleatória	2500	0,074	334,809	34,306	-	112,116
Aleatória	5000	0,136	503,757	88,116	-	160,066

Fonte: Autoria própria (2023).

Da Tabela 4, percebe-se que a etapa de avaliação apresentou uma piora significativa conforme o número de elementos aumenta. Os tempos de execução do algoritmo e de geração das visualizações também aumentaram, mas em menores proporções. Apenas o tempo de estimativa dos parâmetros se manteve inalterado, o que pode ser decorrente do cálculo prévio das distâncias e armazenamento no banco de dados. Caso o cálculo fosse feito em código, o tempo deveria aumentar conforme o número de elementos aumenta, tal como visto nos outros tempos mensurados.

Ainda, também na Tabela 4, foi adicionada uma nova medida do tempo de avaliação removendo o fator de sobreposição (abreviado como F.S.). Durante os testes, foi identificado que o cálculo dessa métrica é o que mais prejudica o tempo mensurado, e por isso uma nova rodada de avaliação foi feita para cada instância, ignorando o cálculo do fator de sobreposição.

4.5 COMPARAÇÃO DOS RESULTADOS COM TÉCNICAS TRADICIONAIS

As mesmas instâncias utilizadas para obtenção de resultados foram utilizadas como entrada para outros dois métodos de *clustering*: o próprio DBSCAN, e o Ordering Points To Identify the Clustering Structure (OPTICS) (ANKERST *et al.*, 1999). Os dois algoritmos utilizados estão disponíveis na biblioteca de aprendizado de máquina **scikit-learn** em Python (PEDREGOSA *et al.*, 2011).

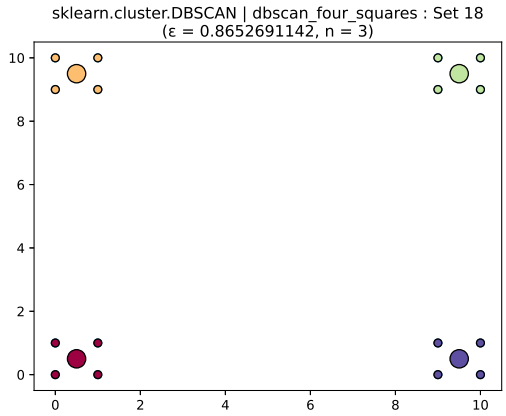
Os resultados que serão comparados nas seções 4.5.1 e 4.5.2 são apenas visualizações, já que tanto a biblioteca utilizada quanto o método desenvolvido nesse trabalho possuem métricas próprias e não são compatíveis de forma nativa.

Assim como o método de *clustering* proposto, a qualidade do resultado obtido com o DBSCAN e o OPTICS também depende muito dos parâmetros de entrada. Para alguns valores de ε e n, o espaço todo foi considerado como *noise* ou como um *cluster* único, tal como percebido no algoritmo desenvolvido. Os resultados apresentados a seguir contemplam apenas os melhores agrupamentos, aproximando as imagens que foram obtidas pelas três técnicas.

4.5.1 COMPARAÇÃO DOS RESULTADOS COM DBSCAN

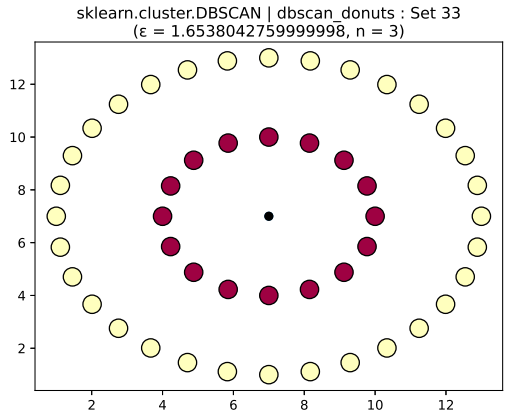
Os resultados obtidos com a utilização do DBSCAN da biblioteca foram muito próximos dos alcançados pelo método de *clustering* desenvolvido. Para as duas instâncias de teste cuja distribuição é mais tendenciosa (nas Figuras 33 e 34), o esperado é que o DBSCAN consiga organizar o espaço de forma perfeita.

Figura 33 – *Four squares* com DBSCAN



Fonte: Autoria própria.

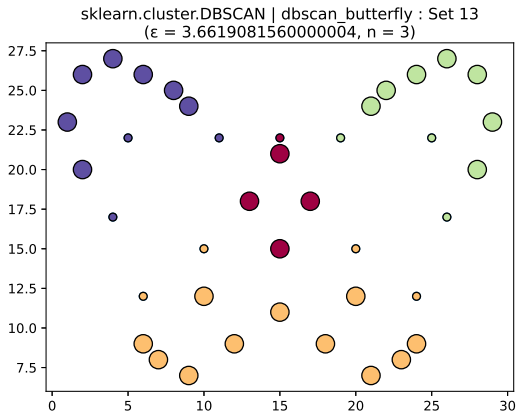
Figura 34 – *Donuts* com DBSCAN



Fonte: Autoria própria.

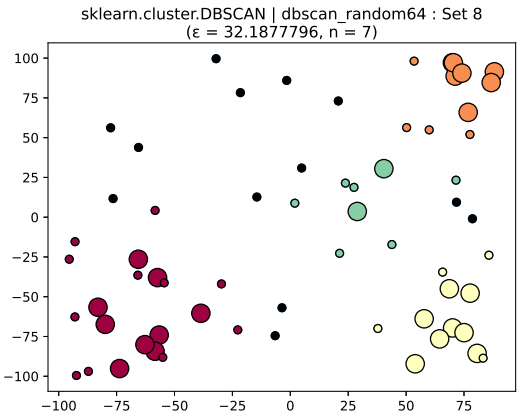
Nas instâncias da borboleta (Figura 35) e aleatória (Figura 36), onde a distribuição dos elementos no espaço não é regular, o DBSCAN gerou visualizações semelhantes às do algoritmo proposto, cujos parâmetros também apresentam valores próximos.

Figura 35 – *Butterfly* com DBSCAN



Fonte: Autoria própria.

Figura 36 – Espaço aleatório com DBSCAN



Fonte: Autoria própria.

Como o método *clustering* desenvolvido é baseado no DBSCAN, é esperado que ambos apresentem resultados parecidos, com exceção em casos específicos gerados pelos valores de

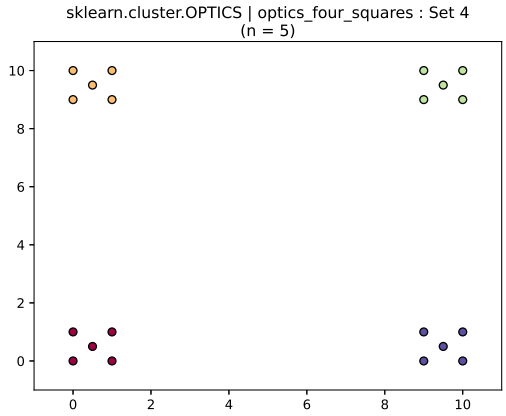
parâmetros de entrada comuns (ε e n), ou com algum corte na lista de proximidade pelo uso do parâmetro k, evitando que um *cluster* continue alocando vértices.

4.5.2 COMPARAÇÃO DOS RESULTADOS COM OPTICS

A abordagem do algoritmo OPTICS é muito parecida com a ideia central do DBSCAN, embora eles apresentem algumas diferenças na execução e no modo como ambos percorrem o espaço (ANKERST *et al.*, 1999).

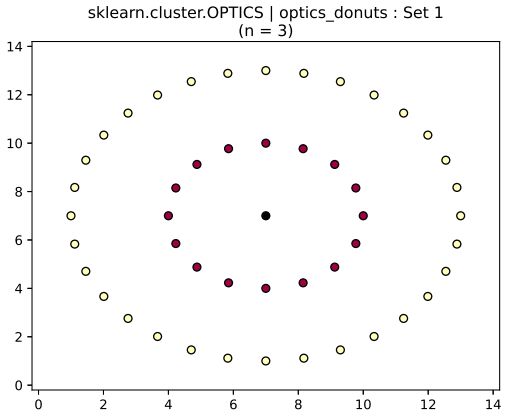
Com essa consideração, para os espaços com mais regularidade (nas Figuras 37 e 38), o OPTICS continua retornando os mesmos *clusters* do DBSCAN e do método proposto, o que é previsível.

Figura 37 – *Four squares* com OPTICS



Fonte: Autoria própria.

Figura 38 – *Donuts* com OPTICS



Fonte: Autoria própria.

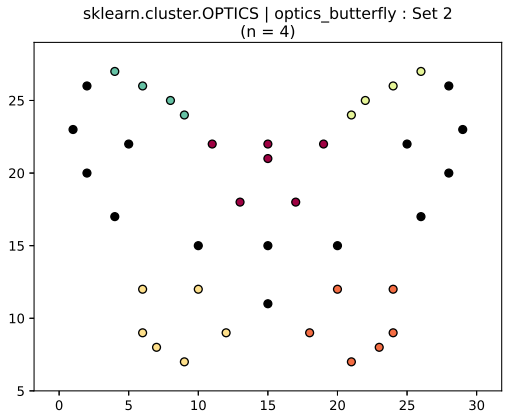
Para os espaços da borboleta e com distribuição randômica, os resultados apresentaram o mesmo comportamento que os do DBSCAN, com leves diferenças entre os agrupamentos gerados com base nos parâmetros de entrada. Os resultados apresentados nas Figuras 39 e 40 mostram algumas das variações na alocação dos vértices nos *clusters*, embora a organização seja majoritariamente a mesma.

4.6 DISCUSSÃO DOS RESULTADOS

A comparação entre as imagens que representam os *clusters* do método desenvolvido e dos métodos DBSCAN e OPTICS demonstra que todos os métodos atingiram resultados muito próximos, com exceção do espaço da borboleta, que apresentou as maiores divergências do OPTICS para o método desenvolvido neste trabalho e o DBSCAN.

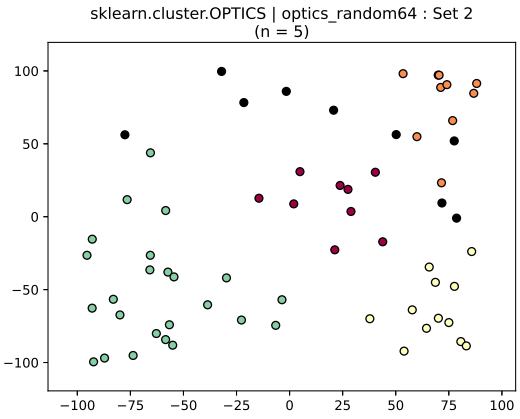
Pelas imagens apresentadas, o DBSCAN identificou um único *cluster* na parte inferior da distribuição quando comparado com os dois agrupamentos percebidos pelo algoritmo desen-

Figura 39 – *Butterfly* com OPTICS



Fonte: Autoria própria.

Figura 40 – Espaço aleatório com OPTICS



Fonte: Autoria própria.

volvido, e alguns vértices foram classificados de forma diferente entre *cores* e *borders* de um método para o outro.

Por sua vez, o OPTICS apresentou os resultados mais divergentes, classificando alguns vértices como ruído que não foram classificados dessa maneira pelo método desenvolvido nesse trabalho e pelo DBSCAN.

Por fim, o espaço com a distribuição aleatória dos elementos apresentou pequenas divergências para os três métodos, principalmente pela organização nos elementos ser mais irregular. Por se tratar de uma distribuição menos tendenciosa, cada método extraiu os resultados de acordo com suas características, a forma como foram implementados e os parâmetros. Todos os resultados são competitivos e extraíram *clusters* que caracterizam a similaridade entre os elementos do espaço.

A comparação das métricas entre o método implementado e o DBSCAN mostrou que, para os mesmos parâmetros de entrada (ε e n), o método de *clustering* atingiu resultados com maior caracterização da distribuição dos elementos, identificando alguns *clusters* em situações onde o DBSCAN identificou apenas 1, o que indica que a presença de um limite superior (representado por k) nas consultas por abrangência impactou positivamente a extração de *clusters*.

Além disso, pela introdução do cálculo do fator de sobreposição, é possível mensurar numericamente a quantidade de vértices que estão em áreas no espaço onde mais de um *cluster* enxerga como vizinhança. Para alguns fatores calculados entre o método deste trabalho e o DBSCAN, em apenas um caso o DBSCAN atingiu um resultado melhor. Em todas as outras combinações dos parâmetros de entrada utilizados, o algoritmo de *clustering* implementado apresentou um fator de sobreposição melhor que ou igual ao fator do DBSCAN.

5 CONCLUSÃO

Este trabalho apresentou uma abordagem de *clustering* não-supervisionado combinando as técnicas DBSCAN, R_q e k-NNq para extrair conjuntos ou grafos de similaridade a partir de espaços métricos de qualquer dimensão. A utilização da k-NNq fornece ao DBSCAN um limite superior, em conjunto com o limite inferior definido pelo parâmetro n (número mínimo de pontos na vizinhança da R_q).

Com base na complexidade estimada do algoritmo e nos tempos de execução mensurados (com exceção da etapa de avaliação do fator de sobreposição), os tempos de execução foram aumentando conforme o número de elementos do espaço também crescia. Novamente, a exceção é o cálculo do fator de sobreposição, que causou uma piora significativa no tempo de execução, e foi desconsiderado para as entradas com maior número de elemento.

Os resultados visuais extraídos foram coerentes, uma vez que ao menos um conjunto de parâmetros entre todos que foram sugeridos resultou em *clusters* adequados para as distribuições de entrada. Possivelmente, um resultado próximo ou idêntico seria atingido se a separação dos espaços mais regulares fosse feita por padrões humanos, sem a aplicação de algoritmos ou métodos matemáticos.

Analisando os resultados numéricos obtidos pelo cálculo das métricas de avaliação, fica claro o impacto da variação dos parâmetros, pois altera a maneira como novos *clusters* são identificados e os vértices são alocados, modificando as características de cada espaço e cada resultado final obtido.

Os resultados visuais obtidos com o método proposto neste trabalho foram similares aos obtidos pelas técnicas tradicionais utilizadas para comparação (DBSCAN e OPTICS), dado que todos utilizam parâmetros em comum e partem do mesmo princípio, que é percorrer o espaço com base na vizinhança de cada elemento.

Considerando as métricas de avaliação, destacando o fator de sobreposição, é possível afirmar que o método proposto atingiu resultados melhores que ou iguais aos resultados do DBSCAN (com exceção de uma combinação de parâmetros), principalmente pela adição da k-NNq, o que diferencia a forma como os algoritmos identificam os agrupamentos.

Com as considerações acima e a comparação com os métodos tradicionais, os resultados obtidos foram condizentes com os objetivos propostos, que foram cumpridos.

Finalmente, o trabalho desenvolvido ainda apresenta alguns pontos de melhoria (como o cálculo do fator de sobreposição, e algumas decisões na criação de *clusters* e alocação de vértices), que serão abordados brevemente a seguir.

5.1 TRABALHOS FUTUROS

Durante o desenvolvimento do algoritmo e pesquisa de trabalhos relacionados, algumas melhorias e pontos de evolução para o tema proposto são:

- Investigar o impacto da utilização da técnica de k-NN reverso para a obtenção de *clusters*;
- Utilizar o algoritmo de *slim-down* para refinar os resultados obtidos, possivelmente diminuindo a sobreposição e alterando a alocação dos vértices nos *clusters*;
- Determinar os valores ou intervalos de valores dos parâmetros para maximizar as métricas de avaliação utilizando redes neurais ou outros métodos computacionais;
- Representar conjuntos de dados complexos em espaços métricos e utilizar como entrada para o método proposto.

REFERÊNCIAS

ANKERST, M. *et al.* OPTICS. **ACM SIGMOD Record**, Association for Computing Machinery (ACM), v. 28, n. 2, p. 49–60, jun. 1999.

BAASE, S. **Computer algorithms**. 2. ed. Upper Saddle River, NJ: Pearson, 1988. (Addison-Wesley series in computer science).

BALABAN, A. T. Applications of graph theory in chemistry. **Journal of Chemical Information and Modeling**, American Chemical Society (ACS), v. 25, n. 3, p. 334–343, ago. 1985.

BERTOZZI, A. L.; MERKURJEV, E. Graph-based optimization approaches for machine learning, uncertainty quantification and networks. *In*: **Handbook of Numerical Analysis**. Elsevier, 2019. p. 503–531. Disponível em: https://doi.org/10.1016/bs.hna.2019.04.001.

BIGGS, N. L.; LLOYD, E. K.; WILSON, R. J. **Graph Theory 1736-1936**. [*S.l.*]: Clarendon Press, 1999. ISBN 0198539169.

BONDY, J. A.; MURTY, U. S. R. **Graph Theory With Applications**. [*S.l.*]: Elsevier Science Ltd/North-Holland, 1976. ISBN 0444194517.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967.

EICK, C.; ZEIDAT, N.; ZHAO, Z. Supervised clustering - algorithms and benefits. *In*: **16th IEEE International Conference on Tools with Artificial Intelligence**. [*S.l.*]: IEEE Comput. Soc, 2004.

ESTER, M. *et al.* A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

ESTRADA, E.; BONCHEV, D. Chemical graph theory. *In*: ____. [*S.l.*: *s.n.*], 2013. p. 1538–1558. ISBN ISBN 9781439880180.

FILHO, R. *et al.* Similarity search without tears: the OMNI-family of all-purpose access methods. *In*: **Proceedings 17th International Conference on Data Engineering**. [*S.l.*]: IEEE Comput. Soc, 2001.

FIX, E.; HODGES, J. Discriminatory analysis, nonparametric discrimination. consistency properties. Technical Report 4, United States Air Force, 1951.

FUKUNAGA, K. **Introduction to Statistical Pattern Recognition (2nd Ed.)**. San Diego, CA, USA: Academic Press Professional, Inc., 1990. ISBN 0-12-269851-7.

GHAHRAMANI, Z. Unsupervised learning. *In*: **Advanced Lectures on Machine Learning**. [*S.l.*]: Springer Berlin Heidelberg, 2004. p. 72–112.

HOSAMANI, S. M. Correlation of domination parameters with physicochemical properties of octane isomers. **Applied Mathematics and Nonlinear Sciences**, Walter de Gruyter GmbH, v. 1, n. 2, p. 345–352, ago. 2016.

JAIN, A. K.; DUBES, R. C. **Algorithms Clustering Data**. Old Tappan, NJ: Prentice Hall, 1988. (Prentice Hall advanced reference series).

KUANG, Z. *et al.* Fashion retrieval via graph reasoning networks on a similarity pyramid. **ArXiv**, abs/1908.11754, 2019.

LIMA, E. L. **Espaços métricos**. [*S.l.*]: Instituto de Matemática Pura e Aplicada, CNPq, 1977. (Projeto Euclides).

METCALF, L.; CASEY, W. **Cybersecurity and Applied Mathematics**. Rockland, MA: Syngress Media, 2016.

NEWMAN, M. E. J. The structure and function of complex networks. **SIAM REVIEW**, v. 45, p. 167–256, 2003.

OMRAN, M. G.; ENGELBRECHT, A. P.; SALMAN, A. An overview of clustering methods. **Intelligent Data Analysis**, IOS Press, v. 11, n. 6, p. 583–605, Nov 2007. ISSN 1571-4128.

ORACLE. **What Is Big Data? | Oracle**. 2014. Disponível em: https://www.oracle.com/big-data/guide/what-is-big-data.html. Acesso em: 30 de agosto de 2019.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

POLA, I. *et al.* Similarity sets: A new concept of sets to seamlessly handle similarity in database management systems. **Information Systems**, Elsevier BV, v. 52, p. 130–148, ago. 2015.

POLA, I.; TRAINA, A.; TRAINA, C. Distance functions association for content-based image retrieval using multiple comparison criteria. *In*: **19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)**. [*S.l.*]: IEEE, 2006.

POLA, I. R. V. **Explorando conceitos da teoria de espaços métricos em consultas por similaridade sobre dados complexos**. 2010. Tese (Doutorado) — Universidade de São Paulo, 2010.

POSTGRESQL. **PostgreSQL: Documentation: 14: 43.1.1. Advantages of Using PL/pgSQL**. 2021. PostgreSQL 14: Documentation. Disponível em: https://www.postgresql.org/docs/14/plpgsql-overview.html#PLPGSQL-ADVANTAGES. Acesso em: 08 jun. 2023.

PYTHON SOFTWARE FOUNDATION. **General Python FAQ — Python 3.8.0 documentation**. 2019. Disponível em: https://docs.python.org/3/faq/general.html. Acesso em: 04 de novembro de 2019.

ROKACH, L.; MAIMON, O. Clustering methods. *In*: ____. **Data Mining and Knowledge Discovery Handbook**. [*S.l.*]: Springer-Verlag, 2005. p. 321–352.

ROSEN, K. **Discrete Mathematics and Its Applications**. [*S.l.*]: McGraw-Hill Education, 2006. ISBN 0073229725.

RUOHONEN, K. **Graph Theory**. [*S.l.*]: Tampere University of Technology, 2013.

SCHAEFFER, S. E. Graph clustering. **Computer Science Review**, Elsevier BV, v. 1, n. 1, p. 27–64, ago. 2007.

TAN, P.-N. *et al.* **Introduction to Data Mining (2nd Edition) (What's New in Computer Science)**. [*S.l.*]: Pearson, 2018. ISBN 0133128903.

TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de Algoritmos: Volume 13**. [*S.l.*]: Bookman, 2012.

TRAINA, C. *et al.* The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. **The VLDB Journal**, Springer Science and Business Media LLC, v. 16, n. 4, p. 483–505, jun. 2006.

TRAINA, C. *et al.* Fast indexing and visualization of metric data sets using slim-trees. **IEEE Transactions on Knowledge and Data Engineering**, Institute of Electrical and Electronics Engineers (IEEE), v. 14, n. 2, p. 244–260, 2002.

TRAINA, C. *et al.* Slim-trees: High performance metric trees minimizing overlap between nodes. *In*: **Advances in Database Technology — EDBT 2000**. [*S.l.*]: Springer Berlin Heidelberg, 2000. p. 51–65.

WANG, L. Heterogeneous data and big data analytics. **Automatic Control and Information Sciences**, Science and Education Publishing Co., Ltd., v. 3, n. 1, p. 8–15, ago. 2017.

APÊNDICE A – Função da distância de Minkowski implementada em PL/pgSQL

Neste apêndice, está o código em PL/pgSQL da função da distância de Minkowski que foi implementada no algoritmo. Durante a execução do método, essa função é chamada no momento de criação das tabelas no banco de dados. O parâmetro p é passado antes da execução, em conjunto com o arquivo de entrada.

```
1  CREATE OR REPLACE FUNCTION distance(from_vertex real[], to_vertex
       real[], exponent real) RETURNS real AS $$
2  DECLARE
3    distance real;
4  BEGIN
5    distance := 0;
6    FOR i IN 1..ARRAY_LENGTH(from_vertex, 1) LOOP
7      distance := distance + POWER(ABS(from_vertex[i] - to_vertex[i]),
       exponent);
8    END LOOP;
9    RETURN POWER(distance, (1.0/exponent));
10 END;
11 $$ LANGUAGE plpgsql;
12 COMMIT;
```

APÊNDICE B – Pseudocódigo do método de *clustering* proposto

Neste apêndice, o método de *clustering* proposto e desenvolvido nesse trabalho está representado em pseudocódigo. Os códigos abaixo contemplam a implementação do DBSCAN utilizando uma R_q como sub-rotina para expandir os vizinhos internos de cada ponto visitado, e uma k-NNq como limitante para a expansão dos *clusters* identificados.

Algoritmo 1: Método de *clustering*

```
Método de clustering(espaço métrico com vetores de características)
base de dados ← vetores de características
lista de ε ← valores estimados de ε
lista de k ← valores estimados de k
lista de n ← valores estimados de n
for cada ε na lista de ε do
    for cada k na lista de k do
        for cada n na lista de n do
            clusters ← DBSCAN(ε, k, n, base de dados)
        end
    end
end
for cada conjunto de clusters gerado do
    if espaço tem 2 dimensões then
        gerar visualização
    end
    calcular métricas de avaliação
end
```

Algoritmo 2: DBSCAN implementado no método de *clustering*

```
DBSCAN(ε, k, n, base de dados)
for ponto na base de dados do
    if ponto não foi visitado then
        vizinhos ← RangeQuery(ponto atual, ε, k, base de dados)
        if número de vizinhos ¡ n then
            ponto ← noise
        end
        else
            ponto ← core
            for vizinho na lista de vizinhos do
                if vizinho não foi visitado then
                    if vizinho é noise then
                        vizinho ← border
                    end
                    else
                        vizinhos internos ← RangeQuery(vizinho, ε, k,
                         base de dados)
                        if número de vizinhos internos ≥ n then
                            vizinho ← core
                            vizinhos ← vizinhos ∪ vizinhos internos
                        end
                        else
                            vizinho ← border
                        end
                    end
                end
            end
        end
    end
end
return vizinhos
```

Algoritmo 3: Consulta por abrangência implementada no método de *clustering*

```
RangeQuery(ponto inicial, ε, k, base de dados)
vizinhos ← lista vazia
for ponto na base de dados do
    if distância entre ponto inicial e o ponto atual ≤ ε then
        vizinhos ← vizinhos + ponto atual
    end
end
return kNNq(vizinhos, k)
```

Algoritmo 4: Consulta aos k-vizinhos mais próximos implementada no método de *clustering*

```
kNNq(vizinhos, k)
vizinhos mais próximos ← vetor com k posições
for vizinho na lista de vizinhos do
    if vizinho está entre os k vizinhos mais próximos then
        remove vizinho mais distante do vetor
        adiciona vizinho no vetor
    end
end
return vizinhos mais próximos
```