Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Natanael Fabrício Dacioli Batista

Eficientes Modelos de Múltiplas Densidades para o Aprendizado em Grandes Conjuntos e Fluxo de Dados

São Carlos
2024

Natanael Fabrício Dacioli Batista

Eficientes Modelos de Múltiplas Densidades para o Aprendizado em Grandes Conjuntos e Fluxo de Dados

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Murilo Coelho Naldi

São Carlos
2024

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Natanael Fabrício Dacioli Batista, realizada em 23/04/2024.

Comissão Julgadora:

Prof. Dr. Murilo Coelho Naldi (UFSCar)

Profa. Dra. Heloisa de Arruda Camargo (UFSCar)

Prof. Dr. Jonathan de Andrade Silva (UFMS)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Este trabalho é dedicado aos desbravadores de aprendizado de máquina e da área de computação em geral.

Agradecimentos

Agradeço ao Programa de Pós Graduação em Ciência da Computação da UFSCar o qual é o apoiador e fomentador de pesquisas e projetos na UFSCar. Agradeço ao programa MAI/DAI do CNPq em parceria com a UFSCar e a FAPESP, Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP - Grant2019/09817-6), os quais apoiaram este projeto com recursos necessários para a pesquisa. Agradecimento especial ao meu Orientador Prof. Dr. Murilo Coelho Naldi que desempenhou papel importantíssimo para esta pesquisa, dando suporte, excelentes orientações e promoveu discussões para meu crescimento na área. Agradeço em especial também ao Bruno Leonel Nunes por todo empenho e ajuda no desenvolvimento desta pesquisa, desde o começo do desenvolvimento dos algoritmos.

Agradeço minha família, minha mãe Lucimar, irmão Maykon e irmã Laryssa, que sempre deram o suporte, apoio e carinho para eu continuar batalhando por meus sonhos. E eles estão sempre lá em minhas conquistas.

Agradeço o Prof. Dr. Diego Saqui por me dar o apoio de entrar no mestrado e todo o incentivo para continuar.

Agradecer também aos que conheci nessa nova etapa da minha vida, pessoas de bem que olharam para mim e já desejaram o melhor.

Agradeço a Deus por toda sabedoria, saúde, alegrias e conquistas que recebi.

"Se, a princípio, a ideia não é absurda, então não há esperança para ela."
(Albert Einstein)

Resumo

O aprendizado de máquina não supervisionado e semi-supervisionado é altamente benéfico em contextos com uso intensivo de dados. O agrupamento hierárquico baseado em densidade oferece uma visão abrangente das estruturas de *clusters* e de valores discrepantes em conjuntos de dados por meio de funções de densidade. Estes algoritmos estabelecem uma hierarquia derivada de uma Árvore Geradora Mínima, onde as arestas representam a densidade máxima necessária para que os dados conectados delineiem *clusters*, dependendo de uma contagem mínima de objetos, denotada como $MinPts$, dentro de uma determinada vizinhança. *CORE-SG*, um gráfico abrangente avançado, gera com eficiência múltiplas soluções hierárquicas com densidades variadas, superando seus antecessores em desempenho computacional. No entanto, algoritmos baseados em densidade necessitam de cálculos de similaridade entre pares, resultando em uma complexidade assintótica de $O(n^2)$ para bases de dados contendo n objetos, tornando-os impraticáveis para cenários que envolvem grandes volumes de dados. Este estudo apresenta modelos hierárquicos de aprendizado de máquina baseados em densidade, visando aliviar custos computacionais com o auxílio de *Data Bubbles*, com foco em agrupamento e detecção de *outliers*. Examina o impacto da sumarização de dados na qualidade de modelos não supervisionados com múltiplas densidades e o ganho em desempenho computacional. A pesquisa garante escalabilidade entre vários métodos de aprendizado de máquina baseados nesses modelos, facilitando o manuseio de grandes volumes de dados sem perda significativa na qualidade resultante. Propomos a aplicação do método de múltiplas hierarquias com *Data Bubbles* em modelos de agrupamento hierárquicos baseados em densidade em fluxos de dados. A pesquisa garante escalabilidade para diversos métodos de aprendizado de máquina baseados nesses modelos onde os dados são infinitos e precisam ser processados em tempo real, além de gerar melhor qualidade do agrupamento.

Palavras-chave: Modelos de Dados Baseados em Densidade; Aprendizagem Não Supervisionada; Agrupamento; Sumarização de Dados; *Big Data*; Fluxo de dados.

Abstract

Unsupervised and semi-supervised machine learning prove highly beneficial in data-intensive contexts. Density-based hierarchical clustering offers a comprehensive insight into cluster and outlier structures within datasets through density functions. These algorithms establish a hierarchy derived from a minimal spanning tree, where the edges represent the maximum density required for connected data to delineate clusters, contingent upon a minimum object count, denoted as $MinPts$, within a given neighborhood. *CORE-SG*, an advanced spanning graph, efficiently generates multiple hierarchical solutions with varying densities, surpassing its predecessors in computational performance. Nonetheless, density-based algorithms necessitate pairwise similarity calculations, resulting in an asymptotic complexity of $O(n^2)$ for datasets containing n objects, rendering them impractical for scenarios involving extensive data volumes. This study introduces hierarchical machine learning models based on density, aiming to alleviate computational costs with the help of Data Bubbles, focusing on clustering and outlier detection. It examines the impact of data summarization on the quality of unsupervised models with multiple densities and the gain in computational performance. The research ensures scalability across various machine learning methods grounded in these models, facilitating the handling of massive data volumes without a significant loss in the resulting quality. We propose the application of the multiple hierarchies method with Data Bubbles in density-based clustering models in data streams. We provide scalability for several machine learning methods based on these models where data is infinite and needs to be processed in real time, in addition to generating better clustering quality.

Keywords: Density-based Data Models; Unsupervised Learning; Clustering; Data Summarization; Big Data; Data Stream.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ARI Adjusted Rand Index

CF Clustering Feature

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DB Data Bubble

FD Fluxo de Dados

HAI Índice de Acordo de Hierarquia

HDBSCAN* Hierarchical DBSCAN*

MST Árvore Geradora Mínima

MC Micro-Cluster

MRD Distância de Alcançabilidade Mútua

o-DB Outlier Data Bubble

o-MC Outlier Micro Cluster

p-DB Potential Data Bubble

p-MC Potential Micro Cluster

Sumário

Capítulo 1

Introdução

Com o rápido aprimoramento das tecnologias, grandes volumes de dados são gerados diariamente e armazenados para identificar padrões, criar modelos preditivos, identificar tendências e tomar decisões autônomas (Zerhari et al., 2015; Blazquez and Domenech, 2018). Em cenários onde os rótulos dos dados são escassos ou caros em situações de *Big Data*, o aprendizado de máquina não supervisionado se mostra vantajoso ao descobrir relacionamentos significativos dentro dos dados sem depender de rótulos (Barlow, 1989). Dentre o conjunto de metodologias não supervisionadas, vale destacar os algoritmos de agrupamento de dados, principalmente aqueles baseados em densidade (Gertrudes et al., 2019). Esses algoritmos possuem a capacidade de identificar *clusters* e *outliers* de forma autônoma, contando apenas com dados internos. Notavelmente, eles exibem flexibilidade, evitando restrições de formato e eliminando a necessidade de uma definição prévia de múltiplos *clusters*, uma característica frequentemente associada a algoritmos de agrupamento paramétricos. As aplicações desses algoritmos são evidentes em periódicos de alto impacto que cobrem diversos domínios de pesquisa, incluindo estudos sobre comportamento humano (Djonlagic et al., 2021), robótica (Savoie et al., 2019), química (Miccio and Schwartz, 2021), genética (Norman et al., 2019), entre outros.

Algoritmos baseados em densidade avaliam a conectividade de objetos da base de dados estabelecendo conexões em cadeia quando um número mínimo de objetos vizinhos, definido como o parâmetro de suavização $MinPts$, estão presentes para definir uma região suficientemente densa para a formação de *cluster*. Os objetos que não atendem a este critério são designados como ruídos. Com um valor $MinPts$ especificado, torna-se viável determinar a densidade e similaridade mínimas necessárias para estabelecer conexões entre qualquer par de objetos na base de dados. Este processo resulta na criação de um

grafo completo G, onde cada nó representa um objeto e cada aresta denota uma medida da densidade de alcance mútuo que os conectam. Por exemplo, o algoritmo HDBSCAN*, reconhecido como o estado da arte em agrupamento baseado em densidade, estabelece uma hierarquia entre objetos e seus *clusters* utilizando a Árvore Geradora Mínima (MST) derivada do grafo completo G. Esta abordagem hierárquica facilita a formação de *clusters* com base na densidade e discrimina *clusters* de ruídos, ao mesmo tempo que é estatisticamente sólida. A sua utilidade estende-se a diversos domínios, incluindo a facilitação da avaliação para a aprovação de projetos de infraestruturas como pontes (Cheema et al., 2022), investigação sobre o câncer (Minussi et al., 2021), sistemas de transporte marítimo (Murray and Perera, 2021), entre outros. Gertrudes et al. (2019) destacaram que vários algoritmos não supervisionados e semi-supervisionados (incluindo HDBSCAN*) podem ser explicados a partir de uma perspectiva baseada em densidade que tem como núcleo comum o processamento de uma MST sobre o grafo G, baseada na densidade mínima $MinPts$. No entanto, determinar o valor ideal de $MinPts$ pode exigir a análise de várias MSTs sobre G, cada um com uma complexidade computacional assintótica de $O(n^2.\, log\, n)$ para um conjunto de dados contendo n objetos.

Em certos cenários, a análise de várias densidades mínimas pode apresentar desafios computacionais devido à necessidade de executar repetidamente o algoritmo de extração da MST no grafo completo G. Recentemente, Neto et al. (2022) propôs uma solução introduzindo um grafo menor chamado *CORE-SG*, que contém todas as MSTs de G para qualquer valor de $MinPts$ menor que um valor máximo especificado $MinPts_{max}$. Extrair a MST de *CORE-SG* tem uma complexidade assintótica de $O(n.log\ n)$, onde $n >> MinPts_{max}$, representando uma redução significativa em comparação com extraí-la de G. A utilização de *CORE-SG* oferece vantagens notáveis, permitindo a análise de resultados para múltiplos valores de $MinPts$ ao custo computacional da obtenção de um único resultado. Isso ocorre porque a construção de *CORE-SG* requer a MST sobre G com densidade estimada como $MinPts_{max}$.

Em aplicações do mundo real, grandes quantidades de dados são geradas, tornando vantajoso computacionalmente o uso do *CORE-SG* para extrair múltiplos resultados com diferentes densidades. No entanto, algoritmos baseados em densidade calculam a estimativa de densidade entre os dados usando cálculos em pares de objetos, que permanecem $O(n^2)$. Devido à complexidade intrínseca destes algoritmos, a construção do *CORE-SG* torna-se proibitiva quando se trata de grandes quantidades de dados, limitando seu uso a conjuntos de dados menores. Existem técnicas aplicadas aos dados que criam estruturas sumárias, reduzindo significativamente a quantidade de informações a serem processadas e mantendo características mais gerais dos dados para criação de modelos. A compactação de dados em estruturas de sumarização é usada para permitir escalabilidade de

algoritmos de *cluster*. Breunig et al. (2001) introduziu *Data Bubble* (DB), uma técnica para sumarizar dados para agrupamento hierárquico por densidade. Este método permite a preservação da qualidade do resultado e, ao mesmo tempo, aumenta significativamente a eficiência computacional. Além disso, os DBs oferecem vantagens sobre outras estruturas de sumarização propostas, pois foram criados especificamente por alguns autores do HDBSCAN* (Campello et al., 2015) para calcular com precisão as distâncias de alcançabilidade com base na densidade dos dados sumarizados.

O avanço, desenvolvimento e disseminação de novas tecnologias tem permitido não só a geração massiva de dados, mas também que estes sejam gerados continuamente através de sensores, monitoramento e entrada de dados do usuário. Esta geração contínua de dados é chamada de Fluxo de Dados (FD) (do inglês, *Data Stream*), onde a entrada dos dados ocorrem de forma sequencial e independente uns dos outros (Gama, 2010). Em um FD o fluxo pode ser ilimitado e a quantidade de informações que serão recebidas não é conhecida a priori (Gama, 2010), portanto armazenar dados de forma persistente é proibitivo, exigindo que os dados sejam processados apenas uma vez. Dessa forma, é criada uma estrutura sumária que sintetiza as informações dos dados de entrada a serem utilizadas na geração de modelos e na extração de conhecimento. Neste cenário, algoritmos baseados em densidade apresentam os mesmos desafios de complexidade para analisar diferentes densidades mínimas, pois também geram MSTs extraídas do grafo completo G sobre a sumarização de dados. Algoritmos hierárquicos baseados em densidade calculam a distância entre pares de objetos para definir a densidade na hierarquia. Os algoritmos do FD calculam a densidade com base nos representantes dos dados sumarizados que, conforme demonstrado em Breunig et al. (2001), podem causar um problema de distorção estrutural, pois as distâncias reais para os pares de objetos não refletem as medidas baseadas em densidade calculadas para os representantes usados na sumarização. Portanto, faltam informações nas estruturas de sumarização para os cálculos de densidades.

1.1 Objetivos e Contribuições

O objetivo desta pesquisa é propor o estabelecimento de *CORE-SG* em conjuntos de dados extensos por meio da sumarização de dados com DBs, facilitando a utilização direta de vários algoritmos não supervisionados e semi-supervisionados para tarefas como agrupamento, detecção de *outliers* e classificação, entre outros (Gertrudes et al., 2019). Algoritmos baseados em densidade são conhecidos por sua precisão de resultados, normalmente limitados a custos computacionais de $O(n^2)$. Este estudo introduz a sumarização de dados em um número significativamente menor de DBs, seguida pela criação de *CORE-SG* sobre os DBs. Chamamos o novo grafo *CORE-SG* sumarizado de *CORE-SSG*. Uma vez formado, *CORE-SSG* permite a extração de resultados hierárquicos utilizando

diferentes valores de parâmetros de densidade $MinPts$ com eficiência computacional de $O(n)$, possibilitando sua aplicação a grandes conjuntos de dados. Tanto quanto os autores sabem, esta é a única literatura onde múltiplos modelos hierárquicos com estimativas de densidade distintas podem ser construídos a partir de um único modelo de dados sumarizados.

Este estudo investiga como a sumarização influencia a qualidade e eficácia de modelos hierárquicos construídos em abordagens baseadas em densidade, com o objetivo de avaliar seu potencial para produzir resultados significativos sem comprometer esses modelos. As descobertas apresentadas não apenas contribuem para aprimorar vários algoritmos documentados na literatura existentes, conforme organizado em Gertrudes et al. (2019), mas também abrem caminho para potencializar novas aplicações de métodos baseados em densidade, particularmente em cenários que envolvem fluxos contínuos de dados.

Propomos neste trabalho o uso do grafo sumarizado *CORE-SSG* em métodos de agrupamento baseados em densidade em FD onde a sumarização com DBs é aplicada aos dados em todo o fluxo para manter a sumarização na fase *Online*. Na fase *Offline*, é construído o grafo *CORE-SSG* permitindo a utilização de diferentes parâmetros de densidade para extrair múltiplas hierarquias com um custo computacional de $O(n)$. Este estudo investiga como a sumarização com DBs influencia a qualidade de algoritmos baseados em densidade em FD e a eficiência do método *CORE-SSG* na geração de múltiplas hierarquias. Experimentos demonstraram que as múltiplas soluções hierárquicas baseadas em densidade sobre a sumarização de dados com DBs obtiveram melhores qualidades que as abordagens de sumarização anteriores. Além do aumento de qualidade do agrupamento, o método *CORE-SSG* em FD permitiu grande redução computacional para a extração de múltiplas hierarquias baseadas em densidade e para diversas chamadas de agrupamento.

Observe que em uma publicação preliminar apresentamos brevemente as definições relevantes para o método *CORE-SSG* e sua capacidade de extrair eficientemente múltiplos resultados de hierarquias baseadas em densidade usando sumarização e sem perda significativa na qualidade do agrupamento. Na presente monografia, apresentamos a pesquisa publicada em Batista et al. (2023) e uma extensão desta publicação, reforçando os conceitos e experimentos abordados, e ao permitir a aplicação de *CORE-SSG* em métodos baseados em densidade no contexto de FD. Introduzimos definições, experimentos e avaliação do novo algoritmo *CoreStream* para gerar múltiplas hierarquias baseadas em densidade de forma eficiente e garantir uma qualidade superior aos algoritmos de FD existentes. Pelo que os autores sabem, este é o único trabalho na literatura que permite a construção de múltiplos modelos hierárquicos com diferentes estimativas de densidade a partir de um único modelo de dados sumarizados em FD.

1.2 Organização do texto

O restante do texto está dividido da seguinte forma: no Capítulo 2 são apresentados os principais trabalhos que norteiam o projeto e outros trabalhos relacionados relevantes da literatura; Capítulo 3 introduz as definições, teoremas e metodologia necessárias para os métodos propostos; no Capítulo 4 são apresentados os experimentos e resultados da pesquisa; no Capítulo 5 apresenta a conclusão baseada nos resultados e experimentos dos métodos propostos.

Capítulo 2

Referencial Teórico

Neste capítulo são abordados os conceitos do algoritmo de agrupamento por densidade hierárquico HDBSCAN*, o método de extração de múltiplas hierarquias baseadas em diferentes níveis de densidade, sumarização de dados e algoritmos de agrupamentos para FD.

2.1 HDBSCAN*

No geral, *clusters* em algoritmos por densidade são regiões densas separados por regiões com baixa densidade entre os dados (Kriegel et al., 2011). O algoritmo HDBSCAN* é uma estrutura para agrupamento por densidade, detecção de *outliers* e oferece visualizações, como o dendrograma da Figura 1 que é uma maneira de visualização da hierarquia de *clusters*, onde os *clusters* são representados pelas linhas pretas na vertical, a divisão do *cluster* (quando um objeto é removido) é representado pelas linhas pretas na horizontal e as linhas vermelhas o momento em que os objetos saem dos *clusters* e se tornam um *cluster* folha, com apenas um objeto (Campello et al., 2015). HDBSCAN* é não-paramétrico e utiliza de estimativas de densidades para formar hierarquias. O algoritmo utiliza do modelo de Hartigan (Hartigan, 1975) de *clusters* de contorno de densidade que melhora os algoritmos de agrupamento baseados em densidade. Como retorno, o HDBSCAN* fornece uma hierarquia de *clusters* completa para todos os possíveis *clusters* baseados em densidade. Da hierarquia são extraídos os *clusters* mais estáveis, usando o conceito de *clusters* rígidos de Hartigan (Campello et al., 2015).

O algoritmo HDBSCAN* utiliza como entrada um único valor de $MinPts$, um parâmetro de suavização de densidade, que está presente em todos os passos da execução. A

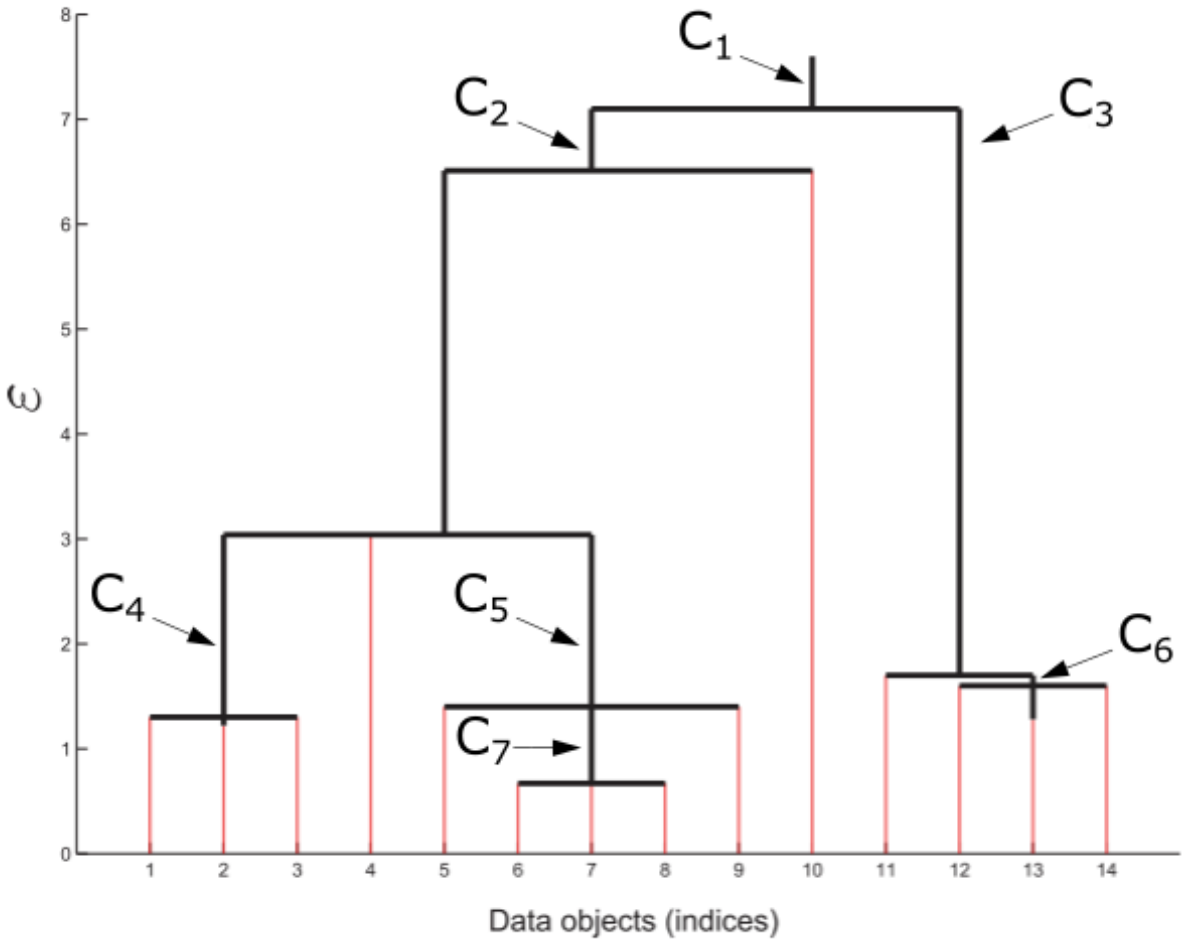


Figura 1 – Dendrograma correspondente a hierarquia HDBSCAN*, onde as linhas pretas simbolizam os *clusters* C_i de 1 à 7 e as linhas vermelhas o momento em que cada objeto foi removido do *cluster* onde era conectado por uma aresta e transformado em um *cluster* folha na hierarquia (Campello et al., 2015).

seguir são apresentados os conceitos principais sobre o HDBSCAN*:

Definição 1: *Distância Core (Core Distance): Dado uma base de dados X a Distância Core ($core_{MinPts}(o)$) de um objeto $o \in X$ é a distância mínima para alcançar os $MinPts$-vizinhos mais próximos. A Distância Core define o raio mínimo ϵ que alcança os MinPts-vizinhos, incluindo o.*

Na Figura 2 os raios apresentados indicam a distância do objeto ao seu $MinPts$-vizinho mais próximo. Note que os raios são de tamanhos diferentes, e que a Distância *Core* de objetos em regiões densas tende a ser menor que os objetos em regiões menos densas. A medida que o parâmetro de suavização $MinPts$ aumenta, as Distâncias *Core* também aumentam, pois será necessário aumentar o raio ao redor do objeto observado para alcançar o $MinPts$-vizinho mais próximo.

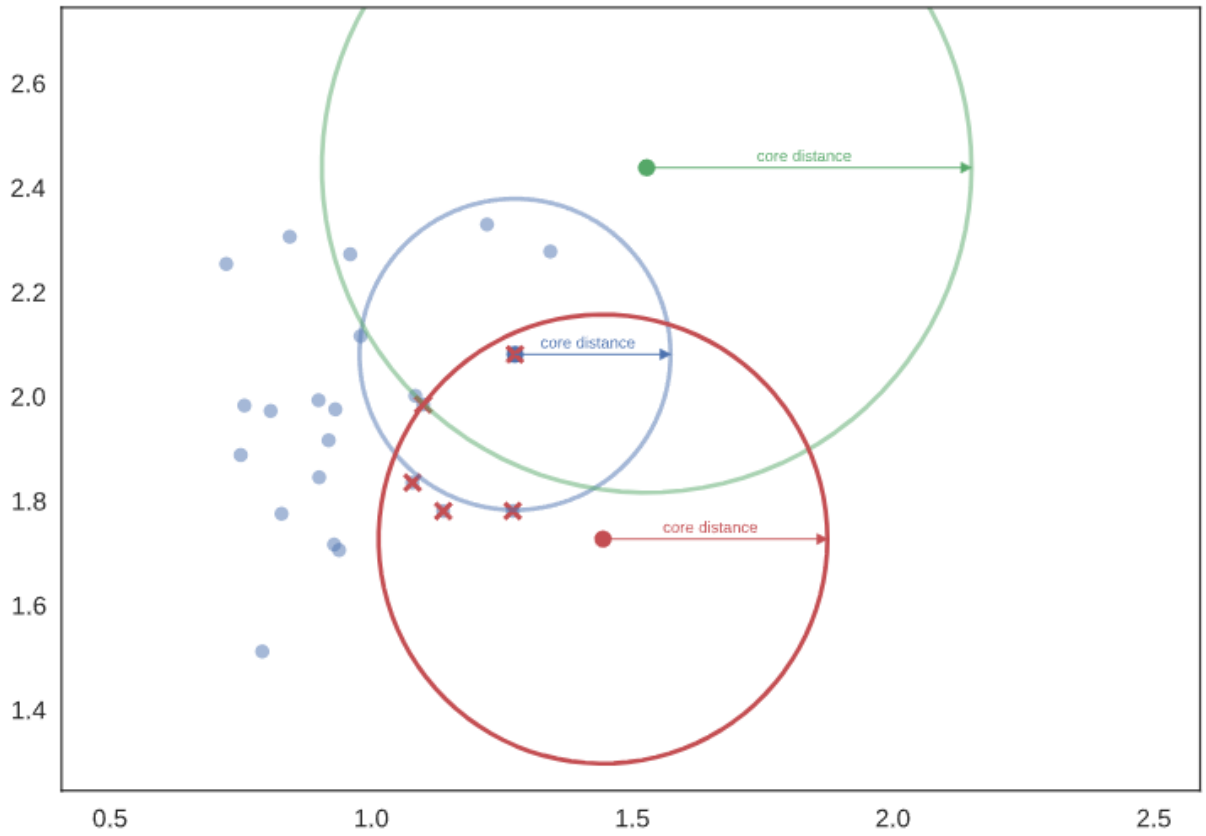


Figura 2 – Exemplo de raios da Distância *Core* para alguns objetos (McInnes et al., 2016).

Definição 2: *Distância de Alcançabilidade Mútua (Mutual Reachability Distance): A Distância de Alcançabilidade Mútua entre dois objetos* o_p *e* o_q *pertencentes ao conjunto de dados* X *é dado pela equação*:

$$mrd_{MinPts}(o_p, o_q) = max\{core_{MinPts}(o_p), core_{MinPts}(o_q), dist(o_p, o_q)\} \quad (1)$$

A $mrd_{MinPts}(o_p, o_q)$ é o raio mínimo ϵ que dado o_p e o_q são ϵ-alcançáveis. Estas definições são de grande importância para o entendimento de hierarquias de *clusters* baseados em densidade. Na Figura 3 demonstramos os três cenários que ocorrem quando se calcula a $mrd_{MinPts}(o_p, o_q)$ entre dois objetos o_p e o_q. No cenário (a) os raios da Distância *Core* de ambos objetos não estão sobrepostos, sendo a distância entre os objetos maior e neste caso a $mrd_{MinPts}(o_p, o_q)$ será a própria distância entre os objetos o_p e o_q. No cenário (b) os raios da Distância *Core* de ambos objetos estão sobrepostos e são menores que a distância entre os objetos o_p e o_q, desta forma a $mrd_{MinPts}(o_p, o_q)$ será a própria distância entre os objetos o_p e o_q. No cenário (c) os raios da Distância *Core* de ambos objetos estão sobrepostos e um ou ambos os raios são maiores que a distância entre os objetos o_p e o_q, desta forma a $mrd_{MinPts}(o_p, o_q)$ será calculada pela máxima entre as Distância Core de o_p e o_q.

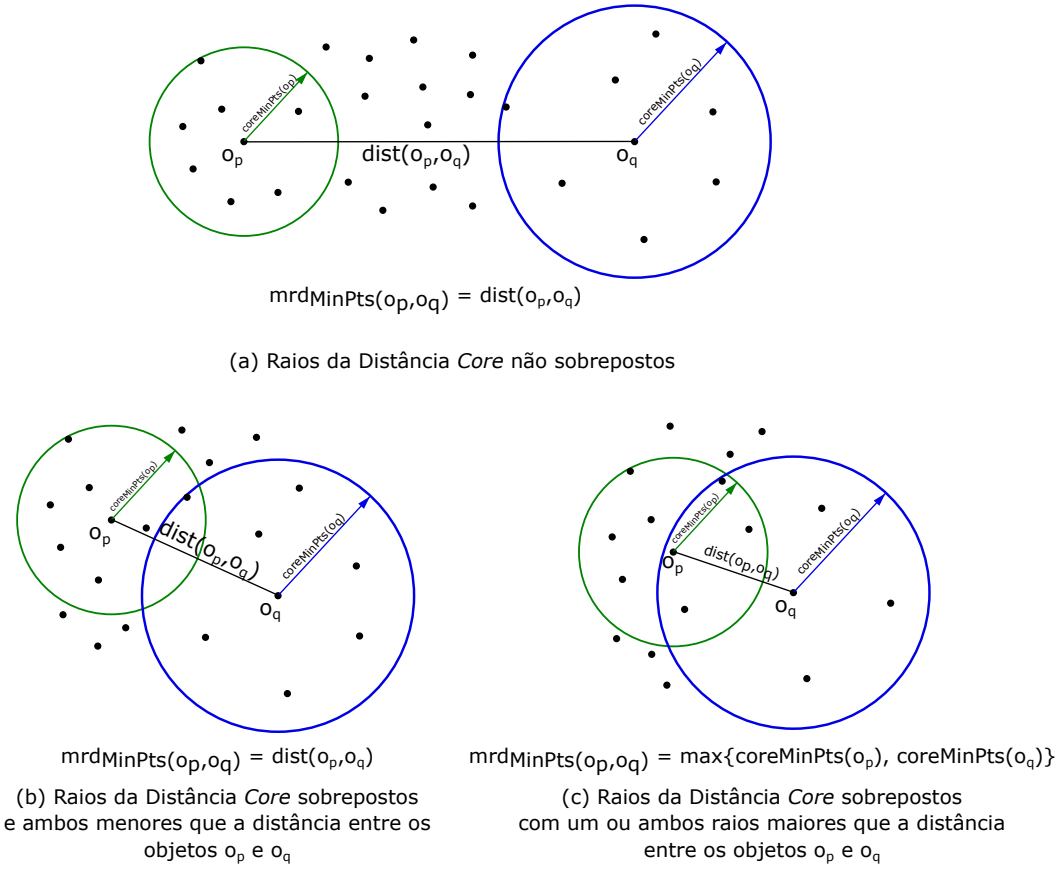


Figura 3 – Cálculo da Distância de Alcançabilidade Mútua em três cenários diferentes.

Definição 3: *Grafo de Alcançabilidade Mútua (Mutual Reachability Graph): Dado um valor de densidade mínima $MinPts$, o Grafo de Alcançabilidade Mútua é um grafo completo $G_{MinPts} = (V, E)$, onde os vértices V são os objetos de X e o conjunto das arestas definida como $E = \{e(o_p, o_q) \,|\, o_p, o_q \in V$ com pesos $w(e) = mrd_{MinPts}(o_p, o_q)\}$.*

Definição 4: *Árvore Geradora: uma árvore T é uma árvore geradora do grafo G se T é um subgrafo de G que contêm todos os vértices de G.*

Definição 5: *Árvore Geradora Mínima (MST, Minimal Spanning Tree): uma MST é um subgrafo conectado de G não direcionado que não contém ciclos e minimiza a soma total dos pesos das arestas.*

HDBSCAN* utiliza do parâmetro $MinPts$ para calcular as Distâncias *Core* de todos os objetos da base de dados. Com estas distâncias o algoritmo calcula as Distâncias de Alcançabilidade Mútua entre todos os pares de dados, para assim calcular o Grafo de Alcançabilidade Mútua G_{MinPts}. O HDBSCAN* extrai a MST de G_{MinPts} e constrói a hierarquia de *clusters* da MST. Para formar a hierarquia, basta remover todas as arestas

da MST em ordem decrescente de pesos, em caso de empate as arestas devem ser removidas simultaneamente. Com isso, ao aumentar a densidade, as arestas de maiores pesos da MST são removidas, e com cada remoção, 2 *clusters* são formados. Cada objeto isolado é considerado ruído e objetos interligados são *clusters*. Sobre a hierarquia usamos o *framework* FOSC (Campello et al., 2013) extraindo a partição final, utilizando o conceito de excesso de massa, que pontua os *clusters* na hierarquia de acordo com suas densidades de contorno.

O HDBSCAN* é o algoritmo estado da arte no quesito agrupamento hierárquico por densidade em dados estáticos, e ganhou o prêmio: *10-year Test of Time Award (PAKDD 2023) PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING*, organizado por NYCU (*National Yang Chiao Tund University*), Academia Sinica, e *Taiwanese Association for Artificial Intelligence*. Os anais da conferência são publicados pela Springer.

2.2 *CORE-SG*

Algoritmos de agrupamento baseados em densidade geram hierarquias baseadas no parâmetro de densidade $MinPts$, que por sua vez atua como um fator de suavização clássico para estimativas de densidade não paramétricas. Escolher o valor de $MinPts$ pode ser um desafio na prática, pois valores muito diferentes de $MinPts$ podem levar a resultados muito diferentes e o usuário normalmente não conhece um valor adequado previamente. Uma abordagem para selecionar o valor de $MinPts$ desejado seria performar uma analise exploratória de múltiplos resultados baseados em densidade, construindo uma MST para cada valor do conjunto $MinPts \in (MinPts_{min}, ..., MinPts_{max})$. Contudo, esta abordagem pode ter um custo computacional proibitivo para grandes bases de dados, porque para isso é necessário a extração de uma MST de um grafo completo G para múltiplos valores de $MinPts$. Para uma base de dados com n objetos, o grafo completo tem $O(n^2)$ número de arestas e a complexidade para extrair cada MST é $O(n^2 \cdot \log n)$.

Para reduzir o custo computacional para geração de múltiplas MSTs baseadas em densidade, os autores Neto et al. (2022) propuseram um grafo muito menor em número de arestas que o grafo completo G, chamado *CORE-SG* (*CORE-Distance Based Spanning Graph*), que garante que todas as arestas das MSTs para valores de $MinPts$ menores que um $MinPts_{max}$ estão presentes. *CORE-SG* é a união de dois gráficos: a $\text{MST}_{MinPts_{max}}$ do $G_{MinPts_{max}}$, onde o peso das arestas são calculadas para o maior valor $MinPts_{max}$; e o *k-Nearest Neighbor Graph* (*k-NNG*), que conecta cada objeto da base com os k vizinhos mais próximos, sendo $k = MinPts_{max}$. A $\text{MST}_{MinPts_{max}}$ e o $MinPts_{max}$-NNG

são extraídas na primeira execução do algoritmo baseado em densidade. Após isso, é construído o $CORE\text{-}SG = MinPts_{max}\text{-}NNG \cup MST_{MinPts_{Max}}$. O número de arestas do grafo *CORE-SG* é $O(n \cdot MinPts_{max})$, assumindo que $MinPts_{max} << n$ (como é esperado na prática), o número de arestas do grafo tem complexidade $O(n)$, muito inferior a quantidade de arestas do grafo completo. Os autores mostram teoricamente e empiricamente que o *CORE-SG* é fácil de ser implementado, é um método eficiente e escalável, e pode ser usado para diversos métodos que extraem MSTs baseadas em densidade. Na Figura 4 temos um exemplo de $k\text{-}NNG$ e $\text{MST}_{MinPts_{max}}$ para $MinPts_{max} = MinPts = k = 4$. O $k\text{-}NNG$ conecta cada objeto com seus k vizinhos mais próximos, garantindo que para $MinPts < MinPts_{max}$ contêm as arestas necessárias para o cálculo da Distância *Core*. O $k\text{-}NNG$ pode ser um grafo não conectado e não contêm uma MST de $G_{MinPts_{max}}$. Na Figura 4, como exemplo, temos o $k\text{-}NNG$ conectado porém não tem todas as arestas da MST_{MinPts}. Ao fazer a união entre o $k\text{-}NNG$ e a $\text{MST}_{MinPts_{max}}$ garantimos que o grafo *CORE-SG* é um grafo conectado e que contêm todas as arestas da $\text{MST}_{MinPts_{max}}$ e das MSTs para $MinPts < MinPts_{max}$.

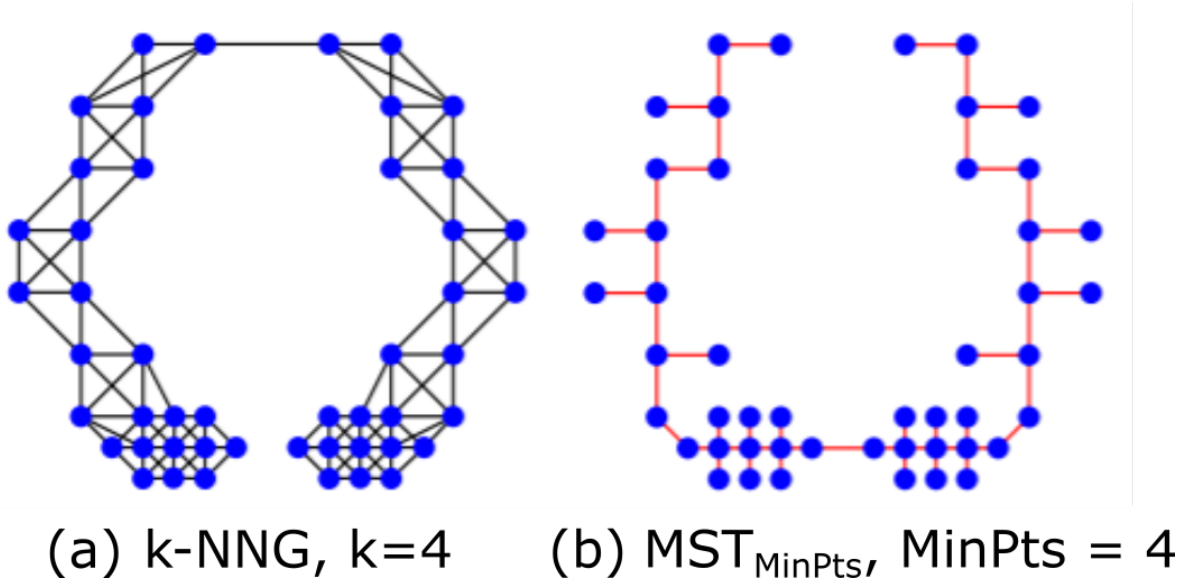


Figura 4 – Exemplo de um k-NNG em (a) e uma MST_{MinPts} em (b) para um determinado G_{MinPts}, onde k = MinPts = 4. (Neto et al., 2022)

2.3 Sumarização de Dados

A sumarização de dados tem sido usada com sucesso para permitir tarefas de aprendizado de máquina em grandes conjuntos de dados ou FD, que são potencialmente infinitos. Para agrupamento hierárquico, um dos algoritmos mais conhecidos é o BIRCH, proposto por (Zhang et al., 1996), que constrói e mantém uma hierarquia de estruturas de compressão de dados chamada *Clustering Features* (CF). Cada CF armazena estatísticas dos

dados sumarizados, formando um *cluster* circular com um raio e um ponto central. As estatísticas são: o número de dados sumarizados, a soma linear e quadrada dos dados. A estrutura do algoritmo BIRCH de CFs forma uma árvore de *subclusters* e é construída de forma incremental, à medida que os objetos são inseridos sequencialmente no CF que melhor os representam. Os parâmetros definem quando novos CFs são adicionados à árvore para aumentar a hierarquia. O uso de estatísticas CFs melhora o desempenho do algoritmo sem impactar significativamente a qualidade do resultado. O uso de CF é muito eficaz para algoritmos de agrupamento do tipo *k-means*. No entanto, para algoritmos hierárquicos baseados em densidade, onde a distância entre os objetos de dados é usada para cálculos de densidade, os CFs têm qualidade limitada. Isto ocorre porque utilizar a distância entre representantes dos CFs não representa bem as distâncias reais entre os objetos.

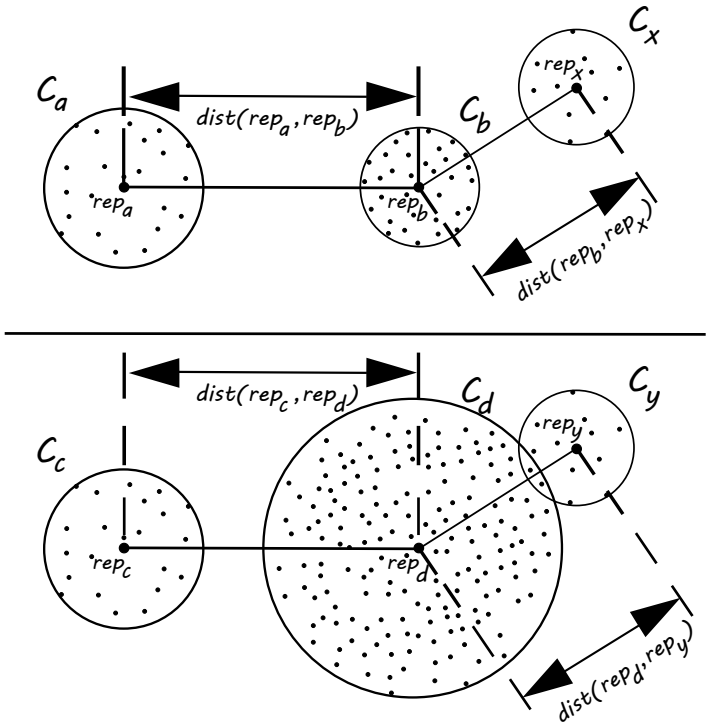


Figura 5 – Ilustração para o problema de distorção no cálculo das distâncias entre CFs. (Breunig et al., 2001)

O CF tradicional sumariza os dados de forma a perder informações sobre seu posicionamento original e seus vizinhos mais próximos, podendo criar distorções quando representado apenas por seus objetos centrais. Essa distorção se torna mais aparente quando uma alta taxa de sumarização é aplicada, ou seja, os CFs possuem uma grande quantidade de dados sumarizados internamente, o que pode levar à sumarização interna de regiões com densidades desiguais. Portanto, a perda de informações impacta a qualidade dos algoritmos baseados em densidade. A Figura 5 ilustra o problema de distorção ao usar distâncias entre representantes dos CFs. A distância $dist(rep_a, rep_b)$ é a mesma que

$dist(rep_c, rep_d)$, e a distância $dist(rep_b, rep_x)$ é a mesma que $dist(rep_d, rep_y)$. A diferença das duas situações é que os CFs C_b e C_d apresentam densidades internas de objetos e raios diferentes, além do que, os CFs C_b e C_y estão sobrepostos. Usando o cálculo de distância entre representantes neste caso, implica que a distância entre os CFs C_a e C_b é a mesma que C_c e C_d. Porém, a distância entre os conjuntos correspondentes de objetos que eles realmente representam é muito diferente, pois os objetos da borda do CF C_d estão mais próximos dos objetos de C_c do que os objetos que estão em C_b em relação aos objetos de C_a. Nos CFs sobrepostos C_d e C_y a proximidade entre os objetos de bordas são mínimas, revelando o quanto os CFs estão densamente conectados. Porém, com a distância entre representantes dos CFs, implica que C_b e C_x possuem a mesma distância que C_d e C_y, mesmo que seus objetos de borda estão mais separados.

Para resolver este problema, foi proposto *Data Bubbles* (DBs) (Breunig et al., 2001) para adicionar informações ao CF necessárias para o cálculo das estimativas de densidade na região coberta pelo DB. Esta informação é um vetor "representativo" para indicar a localização do DB, seu raio e estimativa de distâncias médias entre objetos dentro do DB, essenciais para cálculos de densidade. Os novos cálculos de densidade e estatísticas apresentados no DB permitem uma melhor aproximação das distâncias dos dados originais. DBs têm sido usados com sucesso em uma variedade de trabalhos (dos Santos et al., 2019; Zhou and Sander, 2003; Liu et al., 2006) relacionados a novas abordagens de sumarização e cálculos de densidade. Adicionalmente, DBs foram propostos para aplicação em algoritmos hierárquicos, como foi feito com o OPTICS (Ankerst et al., 1999) e em Breunig et al. (2000) onde o uso de DBs obteve um grande aumento no desempenho computacional, mantendo a qualidade do resultado final. O potencial da aplicação de DBs para análise e visualização de grandes bases de dados também é discutido em Breunig et al. (2000).

Uma característica definidora dos DBs é sua natureza incremental, permitindo atualizações com novos dados. Este aspecto é investigado em um estudo referenciado como Nassar et al. (2004), onde DBs são empregados para permitir incrementalidade dentro de um método denominado *Incremental and Effective Data Summarization* (*IEDS*). Projetado especificamente para gerenciar grandes conjuntos de dados dinâmicos, esse método integra continuamente novos dados em uma árvore hierárquica de *clusters*. A técnica *IEDS* agiliza a construção incremental de dados sumarizados, aproveitando desigualdades triangulares para inserir e excluir DBs. Além disso, o algoritmo avalia os DBs para reter aqueles que contêm dados compactados de alta qualidade.

Durante a inicialização do processo de sumarização, torna-se crucial determinar quais dados devem ser sumarizados dentro de cada DB. Este processo de tomada de decisão tem o potencial de produzir resultados variados e abaixo do ideal, especialmente se os dados de

regiões esparsas forem sumarizados no mesmo DB. Para mitigar o risco do desempenho do algoritmo ser afetado negativamente pela taxa de compressão e pela colocação de objetos sementes que inicializam os DBs, um método para sumarização de dados e inicialização de DB é proposto em Zhang et al. (2016). Este método emprega amostragem baseada em densidade para selecionar um subconjunto de dados para a construção de DBs, evitando assim subconjuntos de dados de regiões de baixa densidade dentro de um único DB.

Em estudos recentes, DBs foram usados como *framework* integral para sumarização de dados, frequentemente em conjunto com o paradigma MapReduce *framework* para melhorar a escalabilidade (dos Santos et al., 2019). Notavelmente, esta estrutura foi aplicada para permitir a implementação escalável de HDBSCAN* (Campello et al., 2015), permitindo o agrupamento hierárquico de dados, incluindo outros DBs, baseados em densidade. Esta metodologia é paralela à abordagem delineada em nossa pesquisa, que também utiliza um algoritmo hierárquico baseado em densidade para análise de dados em larga escala. No entanto, uma distinção fundamental reside no nosso foco em sumarizar estruturas que produzem múltiplas hierarquias caracterizadas por densidades variadas. Este aspecto oferece uma vantagem significativa, particularmente em cenários onde os parâmetros de densidade ideais são desconhecidos a priori, em oposição à hierarquia singular resultante no trabalho anterior Campello et al. (2015).

2.4 Algoritmos de Agrupamento em Fluxo de Dados

Os algoritmos de FD, diferentemente dos algoritmos aplicados sobre toda a base de dados completa, precisam lidar com grandes volumes de dados em evolução, atualizar modelos à medida que novos dados são adicionados e lidar com históricos de fluxo desatualizados. Um FD é uma sequência ordenada e potencialmente ilimitada de instâncias. Um FD S pode ser descrito como $S = \{x_1, x_2, x_3, ..., x_k\}$ onde x_i é a i-ésima instância, há vetores de características de d dimensões e k tende ao infinito. À medida que uma nova instância é adicionada ao fluxo, em um determinado intervalo de tempo t, o algoritmo de agrupamento é aplicado ao FD em evolução. O FD é uma fonte infinita de dados, portanto as instâncias não são armazenadas permanentemente, mas sim sumarizadas em subconjuntos. A velocidade do fluxo nos algoritmos de agrupamentos, representada por v, refere-se ao número de objetos a serem processados dentro do tempo t. Os algoritmos de FD usam v para determinar quão rápido ou lento a sumarização das instâncias será verificada.

Para processar o FD, é mais eficiente processar os dados mais recentes do fluxo do que todo o conjunto de dados de entrada no algoritmo. Existem três tipos de *Window Models* (Modelos de Janela) para auxiliar em processar os dados mais recentes, são eles

(Zubaroğlu and Atalay, 2021): *damped window, landmark window e sliding window*. No modelo *Damped Window*, cada objeto de entrada recebe um peso de acordo com o tempo t de entrada no algoritmo e diminui exponencialmente através da função de decaimento mostrada na Equação 2. A função de decaimento exponencial garante que os objetos mais recentes no fluxo terão pesos mais altos, enquanto os dados mais antigos perdem gradualmente a importância. O valor de λ determina a rapidez com que os objetos perdem peso e se tornam menos relevantes. Em algoritmos de agrupamento de FD que usam sumarização de dados, a função de decaimento exponencial é aplicada às estatísticas que representam o conjunto de objetos. Portanto, os objetos sumarizados que recebem novas entradas, permanecem no fluxo por mais tempo, enquanto os objetos sumarizados com pesos mais baixos desaparecem gradativamente.

$$f(t) = 2^{-\lambda \cdot t}, \text{ onde } 0 < \lambda < 1 \tag{2}$$

No modelo *Landmark Window*, todos os dados entre dois pontos de referência (*landmarks*) possuem o mesmo peso e são incluídas no processo. A quantidade de dados que pertence a uma única janela é chamada de comprimento da janela e geralmente indicada por w. O comprimento da janela pode ser definido como contagem de instâncias ou tempo decorrido. O agrupamento é aplicado a partir de um ponto no tempo inicial até o ponto no tempo atual. O comprimento da janela pode ser definida em termos do número de objetos observados (por exemplo, a cada 1000 objetos) ou em termos de tempo (por exemplo, semanalmente, mensalmente). Quando um novo período de janela começa, todos os objetos mantidos na janela anterior são removidos.

No modelo *Sliding Window*, para cada tempo t contado no fluxo, uma instancia é trocada na janela. A instância mais antiga sai da janela e a instância mais recente entra na janela pelo estilo FIFO (First-In First-Out). Todas as instâncias na janela têm peso igual e as janelas consecutivas se sobrepõem. O comprimento da janela é um parâmetro definido pelo usuário e deve ser decidido de acordo com os dados de entrada. Para o agrupamento todos os dados dentro da janela são considerados.

Algoritmos de agrupamentos de FD foram desenvolvidos posteriormente e introduzem sumarização de dados com Micro-Cluster (MC) que são continuamente mantidos para seguir a distribuição variável dos dados. Periodicamente, MCs são usados para fornecer agrupamentos e ideias de evolução em horizontes de tempo especificados pelo usuário (informações presentes dentro de um intervalo de tempo) (Aggarwal et al., 2003). MCs são uma extensão temporal dos CFs, mantendo estatísticas de dados para definir a localização central e o raio que os delimitam. Além das informações dos CFs, os MCs incluem estatísticas temporais, que são: a soma dos *timestamps* (carimbos de data e hora) e a soma dos quadrados dos *timestamps*, sendo os *timestamps* o tempo de chegada dos dados no fluxo.

Os algoritmos de FD que usam sumarização seguem o modelo de fases *Online-Offline*, também apresentada na Figura 6. Na fase *Online*, a sumarização dos dados em evolução é mantida e continuamente adaptada. Na fase *Offline*, seja por solicitação do usuário ou por uma determinada frequência δt, algoritmos de agrupamentos são aplicados aos dados sumarizados para gerar os *clusters* finais. Como no Clustream (Aggarwal et al., 2003), que na fase *Online* usa MCs para manter estatísticas sobre dados em evolução e na fase *Offline* considera MCs como pseudo-pontos e executa uma variante k-means sobre os MCs. No entanto, assim como as distâncias entre representantes dos CFs não conseguem estimar a densidade entre os objetos sumarizados, os MCs também têm qualidade limitada.

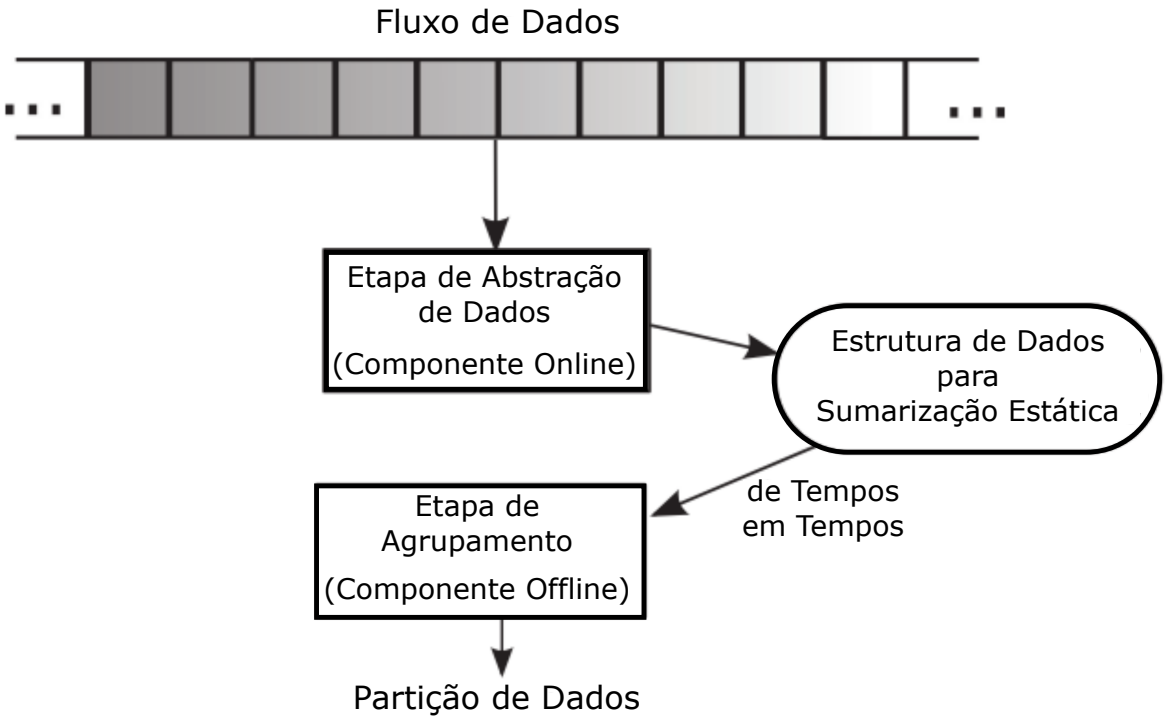


Figura 6 – Estrutura de *cluster* em FD baseada em objetos. (Silva et al., 2013)

DenStream (Cao et al., 2006) é um algoritmo baseado em densidade para FD que usa MCs na fase *Online*. DenStream divide os MCs em potenciais e *outliers*. Ao contrário dos algoritmos baseados em k-means que criam um número limitado de MCs, DenStream define um limite superior nos raios dos MCs, agrupando os dados por proximidade. Antes do DenStream, os algoritmos de agrupamentos em FD resultavam em *clusters* esféricos e não consideravam que os *clusters* poderiam ter formatos arbitrários (Hassani et al., 2014b). DenStream na fase *Offline* usa uma variante do algoritmo DBSCAN (Kriegel et al., 2011) baseado em densidade para agrupar os MCs potenciais de acordo com um certo limite de densidade. Devido ao parâmetro de densidade fixo do DenStream e ser

um algoritmo não hierárquicos baseados em densidade, *clusters* de densidades diferentes não são encontrados e, ao usar MCs, os cálculos de densidade têm qualidade limitada.

O MR-stream proposto em Wan et al. (2009), utiliza de uma estrutura de árvore para representar o FD. O espaço de dados é dividido em células de tamanhos iguais. Estas células são divididas em subcélulas formando uma hierarquia na árvore. A quantidade de subcélulas formadas é limitada pelo parâmetro definido pelo usuário. Na fase *offline*, em alguma altura h da hierarquia, os *clusters* são formados agrupando todas as células densas alcançáveis em um *cluster*. O MR-stream tem problemas de eficiência assim como algoritmos de agrupamento de FD em grade geralmente têm. Outra questão é que o algoritmo introduz um novo parâmetro em que o usuário precisa ter um conhecimento prévio para escolher um valor adequado.

O PreDeConStream (Hassani et al., 2012) tenta encontrar *clusters* em subespaços do FD em evolução. O algoritmo PreDeConStream atua na atualização incremental da saída da fase *Online*, levando em consideração os MCs que vão desaparecendo ou vão surgindo ao longo do tempo. A medida que os MCs mudam, os *clusters* já são atualizados para agilizar na fase *Offline*. A formação dos MCs é baseada na densidade entre os objetos. Mesmo sendo eficiente, o algoritmo precisa de alguns parâmetros de entrada pelo usuário, o que implica diretamente na qualidade do agrupamento, recaindo a responsabilidade no usuário.

SubClusTree é um algoritmo de agrupamento de FD baseado em grade que encontra os *clusters* nos subespaços do FD (Hassani et al., 2014a). Os subespaços que se tornam irrelevantes são eliminados, deixando um número gerenciável de subespaços, cada um representado por uma árvore. Cada árvore está conectada com um vetor de *bits* que representa um subespaço específico. Nos subespaços armazena em níveis diferentes da árvore os MCs de densidades diferentes. O SubClusTree não é hierárquico mas trabalha com árvores em subespaços, o que pode ser observado para entender as estruturas internas de um grafo por meio de árvores.

HDyclee é baseado no DBSCAN e utiliza como parâmetro o limiar de densidade ϵ (Obry et al., 2018). Os *clusters* são visualizados como hipercubos formados pela densidades entre os dados. O HDyclee usa como distância a densidade entre os dados e forma uma árvore binárias com os hipercubos. O algoritmo extrai uma hierarquia de *clusters* que garantem um nível de densidade em cada camada. Para encontrar os *clusters*, HDyclee utiliza o limiar de densidade, onde os *clusters* que estão acima do limiar são considerados para o agrupamento. A ideia de se trabalhar com mais de uma densidade é conseguir identificar os *clusters* que estão em diferentes densidades. No HDyclee além da limitação

do parâmetro ϵ, só são identificados *clusters* que estão acima da densidade ϵ.

A maioria dos algoritmos de fluxo apresentados utilizam sumarização com MCs e uma variante do DBSCAN para formar os agrupamentos, como pode ser observado na Tabela 1. O uso de MC limita a qualidade de algoritmos hierárquicos, pois não apresentam estatísticas suficientes e cálculos adaptados para estimar a densidade entre os dados sumarizados. Algoritmos baseados em DBSCAN possuem um parâmetro fixo de densidade ou acrescentam alguns parâmetros que não são adaptados ao longo do fluxo e não encontram *clusters* de diferentes densidades.

Tabela 1 – Visão geral dos algoritmos de agrupamento de fluxo de dados.

Algoritmo	**Ano**	**Abordagem**	***Window***	**Sumarização**	**Método subjacente**	**Processamento**
DenStream	2006	baseado em densidade	damped	core-micro-cluster	DBSCAN	*Online-Offline*
MR-Stream	2009	baseado em grade	damped	grade/árvore	regiões densas	*Online-Offline*
PreDeConStream	2012	baseado em densidade	damped	core-micro-cluster	DBSCAN	*Online-Offline*
SubClusTree	2014	baseado em grade	damped	core-micro-cluster/árvore	k-means ou DBSCAN	*Online-Offline*
HDyclee	2018	baseado em densidade	damped	micro-cluster	DBSCAN	*Online-Offline*

2.4.1 HASTREAM: Algoritmo hierárquico Baseado em Densidade para Fluxo de Dados

Algoritmos hierárquicos baseados em densidade são conhecidos por sua capacidade de encontrar *clusters* de diversas densidades e formas arbitrárias, assim como o HDBSCAN*. HASTREAM, proposto em Hassani et al. (2014b), é o primeiro algoritmo de FD de agrupamento hierárquico baseado em densidade que aplica os conceitos do HDBSCAN*. O algoritmo HASTREAM mantém a mesma estrutura de MCs que no DenStream na fase *Online*. Na fase *Offline*, uma variante do HDBSCAN* é aplicada aos MCs para gerar o grafo completo, extrair a MST e formar a hierarquia de *clusters*.

Na fase *Online* os MCs são mantidos e atualizados a medida que novos dados chegam no fluxo. Nesta fase é usado o modelo *Damped Window* (Janela Amortecida) com uma função de decaimento exponencial para manter a sumarização atualizada com os novos dados, como especificado na Equação 2 (o parâmetro λ especifica a importância dos dados anteriores). Na fase *Offline* os conceitos do HDBSCAN* são utilizados para extrair os *clusters* sobre os MCs. A seguir são apresentadas algumas definições importantes do algoritmo HASTREAM. Note que algumas delas são adaptações das definições do algoritmo HDBSCAN* (Seção 2.1):

Definição 6: *Micro-Cluster: é uma tupla de informações* $MC(\overline{CF^1}, \overline{CF^2}, w, c, r, t_0)$ *que armazenam estatísticas sumarizadas dos objetos em relação a localização espacial e componentes temporais, onde:* o_j *é o objeto adicionado ao MC no tempo atual* t_j*,* t_u *é o tempo da última atualização do MC,* $\overline{CF^1} = \sum_{j=1}^{n} f(t_j - t_u)o_j$ *é a soma linear ponderada dos objetos,* $\overline{CF^2} = \sum_{j=1}^{n} f(t_j - t_u)o_j^2$ *é a a soma ponderada dos objetos ao quadrado,* $w = \sum_{j=0}^{n} f(t_j - t_u)$ *é o peso do MC referente ao número de objetos sumarizados aplicando a função de decaimento exponencial,* $c = \frac{\overline{CF^1}}{w}$ *é o centro,* $r = \sqrt{\frac{\overline{CF^2}}{w} - (\frac{\overline{CF^1}}{w})^2}$ *é o raio e o* t_0 *é o carimbo de tempo de criação.*

Se um objeto o_p fizer parte de um *Micro-Cluster mc*, as informações do *mc* são atualizadas da seguinte forma: $mc = (\overline{CF^1} + o_p, \overline{CF^2} + o_p^2, w + 1)$. Caso nenhum objeto seja adicionado ao *Micro-Cluster mc* por um certo intervalo de tempo $\delta t = t_c - t_u$, sendo t_c o tempo atual e t_u o tempo da última atualização de *mc*, então ele será atualizado como: $mc = (f(\delta t) \cdot \overline{CF^1}, f(\delta t) \cdot \overline{CF^2}, f(\delta t) \cdot w)$.

Definição 7: *Distância Core (Core Distance): a Distância Core* ($core_{MinPts}(mc_p)$) *de um Micro-Cluster* mc_p *é a menor distância Euclidiana para alcançar os* $MinPts$*-vizinhos mais próximos. A* $core_{MinPts}(mc_p)$ *é o raio determinado pela distância entre o representante de* mc_p *e o* $MinPts$*-vizinho mais próximo.*

Definição 8: *Distância de Alcançabilidade Mútua (Mutual Reachability Distance): a Distância de Alcançabilidade Mútua entre dois Micro-Clusters mc_p e mc_q é definida como o máximo da distância euclidiana entre os representantes dos Micro-Clusters e suas respectivas Distâncias Core, como na Equação 3*:

$$mrd_{MinPts}(mc_p, mc_q) = max\{dist(mc_p, mc_q), core_{MinPts}(mc_p), core_{MinPts}(mc_q)\} \quad (3)$$

Definição 9: *Grafo de Alcançabilidade Mútua (Mutual Reachability Graph): o Grafo de Alcançabilidade Mútua é um grafo completo $G_{MinPts}(V, E)$, onde o conjunto de vértices V são os Micro-Clusters disponíveis no timestamp t. O conjunto de arestas disponíveis no tempo t é definida como $E = \{e(mc_p, mc_q) \mid mc_p, mc_q \in V$ com peso $w(e) = mrd_{MinPts}(mc_p, mc_q)\}$.*

Definição 10: *Estabilidade de Clusters para Micro-Clusters: a estabilidade do cluster (CS) de um cluster C_i composto por Micro-Clusters é calculado de acordo com a Equação 4. $w(mc_j)$ indica o peso do Micro-Cluster mc_j, λ_{max} é o limite de densidade onde mc_j não pertence mais ao cluster C_i e λ_{min} é a densidade mínima em que o cluster C_i foi formado na hierarquia.*

$$CS(C_i) = \sum_{mc_j \in C_i} w(mc_j) \times (\lambda_{max}(mc_j, C_i) - \lambda_{min}(C_i)) \quad (4)$$

Na chamada Offline do HASTREAM, ao ser solicitado o agrupamento hierárquico no tempo t, o algoritmo pega o modelo da fase *Online* no tempo t com as estruturas de MCs para fazer o agrupamento. A seguir é discutido os passos da execução do HASTREAM para a extração dos *clusters* finais (Hassani et al., 2014b):

1. Com o valor de $MinPts$ inserido no início da execução do HASTREAM é calculada a Distância *Core* (Definição 7) de cada MC;
2. São calculadas as Distâncias de Alcançabilidade Mútua entre todos os pares de MC;
3. É gerado o Grafo de Alcançabilidade Mútua $G_{MinPts}(V, E)$ conectando cada MC com os demais, sendo as arestas do grafo G_{MinPts} ponderadas pela mrd_{MinPts};
4. Com o $G_{MinPts}(V, E)$ é extraída a MST_{MinPts};
5. Na MST_{MinPts} as arestas são removidas de ordem decrescente, ou seja, as arestas com maiores pesos são removidas primeiro até que as arestas com menores pesos

sejam removidas. Quando uma aresta é removida dois novos *clusters* se formam, podendo ser um conjunto de MCs ou um MC sozinho, que por sua vez pode ser considerado um *cluster* (caso exceda o limiar de densidade $MinPts$ de objetos sumarizados);

6. Após as remoções é calculada a estabilidade dos *clusters* pela Equação 4;
7. O passo final do HASTREAM é extrair os *clusters* com mais estabilidade na hierarquia, usando uma adaptação do *framework* FOSC.

HASTREAM considera MCs como pseudo-pontos. Os cálculos de densidade utilizam as distâncias entre os representantes dos MCs. Portanto, o HASTREAM também tem a desvantagem de utilizar cálculos de densidade em MCs, onde as distâncias entre os dados sumarizados não representam com qualidade as distâncias reais dos dados. Além disso, a cada requisição de agrupamento na fase *Offline* é gerado um grafo completo G_{MinPts} com complexidade assintótica $O(n^2)$ do número de arestas. A medida que o tempo no FD passa, novos dados chegam, o número de MCs podem aumentar e a frequência das requisições de agrupamento podem aumentar. Desta forma, o custo computacional do HASTREAM será grande, e em alguns casos será proibitivo o cálculo do agrupamento.

Para diminuição do custo computacional de extrair a MST sobre G_{MinPts}, foi proposto o algoritmo I-HASTREAM (Hassani et al., 2016), que atualiza a MST de forma incremental a media que os MCs são criados ou removidos. O I-HASTREAM usa duas listas, armazenando os MCs que foram inseridos e os que foram removidos. Em cada chamada de agrupamento na fase *Offline* estas listas são verificadas e apenas atualizada as arestas e vértices da MST mantida ao longo da fase Online, em vez de calcular G_{MinPts}. No pior caso, onde todos os MCs foram afetados, o I-HASTREAM terá o mesmo custo computacional do HASTREAM, caso contrário, o agrupamento terá um custo menor. O I-HASTREAM também tem o problema de distorção de densidade ao usar os MCs, e caso seja necessário gerar outra hierarquia mudando o parâmetro de densidade $MinPts$, a MST mantida não poderá ser usada, sendo necessário recalcular o G_{MinPts}.

Capítulo 3

Metodologia

Nesta sessão são descritas as definições e teoremas do método proposto *CORE-SSG* para gerar múltiplas hierarquias baseadas em densidade sobre sumarização de dados. São discutidas a adaptação e extensão do método *CORE-SSG* pra aplicação em cenário de FD, sendo desenvolvido o algoritmo CoreStream para geração de múltiplas MSTs baseadas em densidade e em várias requisições de agrupamento. Os índices de avaliação de hierarquias e partições são discutidos.

3.1 *CORE-SG* para Sumarização de Dados

A proposta está dividida em duas etapas: sumarização de dados e obtenção de hierarquias com múltiplas densidades. O DB (Breunig et al., 2001) original usa o parâmetro ϵ, que é incompatível com HDBSCAN* e *CORE-SG*, pois o parâmetro ϵ fixa a densidade ao pesquisar pelos vizinhos mais próximos, que já foi removido desde os métodos por densidades propostos por HDBSCAN*. Portanto, é proposto uma adaptação e novas definições para permitir a aplicação conjunta de DBs e *CORE-SG*. O método usa o fator de suavização clássico nas estimativas de densidade $MinPts$. Os métodos hierárquicos baseados em densidade usam $MinPts$ para estabelecer a densidade mínima para conectar dados. Este parâmetro também é usado nos conceitos de *core* e distâncias de alcançabilidade introduzidos por OPTICS (Ankerst et al., 1999) e adaptados em HDBSCAN* (Campello et al., 2015).

3.1.1 Sumarização Baseada em Densidade

No contexto da análise de um conjunto de dados X compreendendo n objetos, cada um com d dimensões, e dado um inteiro $MinPts$ que especifica o número mínimo de vizinhos necessários para que um objeto seja categorizado como *core*, a abordagem começa empregando um método computacional com uma complexidade de $O(n)$ para particionar X em m subconjuntos, onde $m << n$. O processo de agrupamento em subconjuntos m pode ser executado nesta fase, ou alternativamente, uma técnica de inicialização guiada pode ser empregada (Zhang et al., 2016).

Definição 11 - *Data Bubble (DB): O subconjunto de dados $X_i \subseteq X$ sumarizado em um DB que é definido como uma tupla $B_i = (rep_i, n_i, extent_i, nnDist_i)$, onde rep_i é a posição do representante de X_i, n_i é a cardinalidade $|X_i|$, $extent_i$ é um número natural que define um raio estendido em torno do rep_i que contém os objetos de X_i, e $nnDist_i$ é uma função que denota as distâncias médias estimadas dos k-vizinhos mais próximos dentro do conjunto de objetos X_i para alguns valores k, $k = 1, k = 2, ..., k = MinPts$. Em um espaço euclidiano, as somas estatísticas lineares e quadráticas dos objetos em X_i, LS_i e SS_i, respectivamente, podem ser usadas para calcular as características de B_i. Com essas estatísticas, $rep_i = \frac{LS_i}{n_i}$, $extent_i = \sqrt{\frac{2 \cdot n_i \cdot SS_i - 2 \cdot LS_i^2}{n_i \cdot (n_i - 1)}}$ e a distância esperada do k-vizinho mais próximo é $nnDist_i(k) = (\frac{k}{n_i})^{\frac{1}{d}} \cdot extent_i$, sendo k o k-ésimo objeto sumarizado dentro de B_i, d é o número de dimensões dos objetos em X.*

É importante notar que em um espaço euclidiano, DBs utilizam estatísticas do CF para calcular suas características e determinar seu posicionamento e raio esperado. Além disso, DBs possuem uma nova função que calcula a distância esperada do vizinho mais próximo $nnDist_i$. Esta função calcula uma distância esperada do k-vizinho mais próximo dentro do raio do DB baseando em cálculos estimados sobre uma distribuição uniforme dos objetos dentro de uma esfera, propostos em Breunig et al. (2001). Isto, juntamente com o raio *extent*, pode ser usado para formular uma métrica para medir a distância espacial entre dois DBs. Esse recurso facilita a adequação para agrupamento hierárquico.

Definição 12 - *Distância entre DBs: Se B_i e B_j são dois DBs, a distância direta entre B_i e B_j é definida como*:

$$dist(B_i, B_j) = \begin{cases} 0, \ se \ B_i = B_j \\ dist(rep_i, rep_j) - (extent_i + extent_j) + nnDist_i(1) + nnDist_j(1), \\ \text{se } dist(rep_i, rep_j) - (extent_i + extent_j) \geq 0 \\ max(nnDist_i(1), nnDist_j(1)), \ \text{caso contrário.} \end{cases} \tag{5}$$

Em outras palavras, se $B_i = B_j$, a distância deles será zero. Se eles não se sobrepõem, sua distância é dada pela distância entre seus representantes menos seus raios mais as distâncias esperadas até o vizinho mais próximo. Finalmente, se os DBs se sobrepõem, a distância é o máximo entre suas distâncias esperadas dos vizinhos mais próximos. Os casos descritos estão ilustrados na Figura 7. A definição de distância proposta visa aproximar a distância entre os dois objetos mais próximos dentro dos DBs.

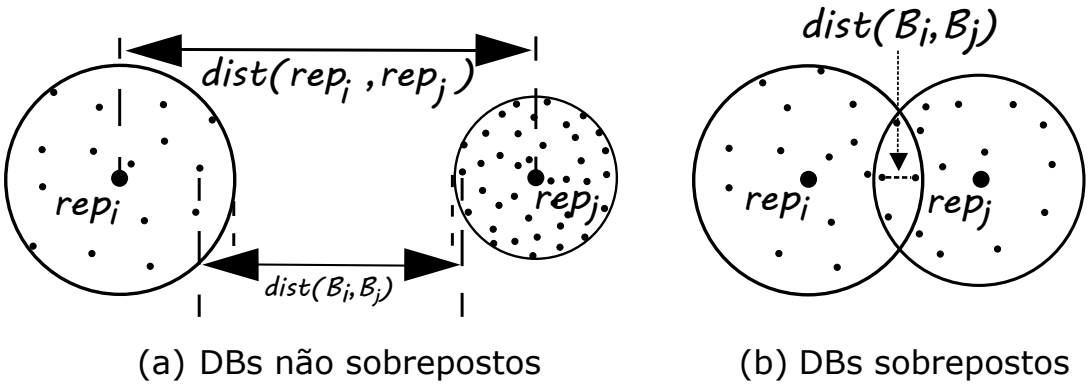


Figura 7 – Distancia entre *Data Bubbles* (Breunig et al., 2001).

Definição 13 - *Distância Core de um DB: Dado um valor de* $MinPts < n$*, a função* $NN(B_i, k)$ *que retorna o* k*-ésimo DB mais próximo para* B_i *ou* B_i *se* $k = 0$*, uma função que retorna o número mínimo de vizinhos de* B_i *necessários para sumarizar* $MinPts$ *objetos, ou seja,* $SNN(B_i, MinPts) = \arg\min_k \sum_{j=0}^{k} n_j \geq MinPts \mid B_j = NN(B_i, j)$*, a distância core de* B_i *é*:

$$core_{MinPts}(B_i) = \begin{cases} nnDist_i(MinPts), \ se \ SNN(B_i, MinPts) = 0 \\ dist(rep_i, rep_k) - extent_k + \\ nnDist_k(MinPts - \sum_{l=0}^{k-1} n_l | B_l = NN(B_i, l)), \text{caso contrário} \end{cases} \tag{6}$$

onde $B_k = NN(B_i, SNN(B_i, MinPts))$ é o k-ésimo vizinho mais próximo de B_i. A distância *core* é a distância mínima de um DB que sumariza pelo menos $MinPts$ objetos.

É importante ressaltar que $core_a(B_i) \geq core_b(B_i)\ \forall\ a \geq b$, ou seja, aumentar o valor de $MinPts$ não pode reduzir a distância *core* de um DB. Podemos calcular as distâncias de alcançabilidade usadas nas estimativas de densidade usando a distância entre dois DBs e suas distâncias *core*.

Em algoritmos hierárquicos, como HDBSCAN*, vemos que a distância *core* representa a distância de um objeto o_p até seu $Minpts$-vizinho mais próximo (incluindo o_p), ou seja, define um raio ϵ no qual garante que pelo menos $MinPts$ objetos estarão presentes. No cálculo da distância *core* proposto em Breunig et al. (2001), quando $SNN(B_i, MinPts) > 0$, é calculado a distância entre B_i e B_k e adicionado uma distância estimada do vizinho mais próximo em B_k. Porém, este cálculo não garante que o raio definido na distância *core* alcance pelo menos $MinPts$ objetos, pois a distância representa a aproximação dos objetos mais próximos entre os DBs e não suas distâncias centrais. Para resolver esta questão e nos aproximarmos da definição da distância *core* dos algoritmos hierárquicos, propomos uma adaptação do cálculo. No caso em que $SNN(B_i, MinPts) > 0$, a distância *core* é a distância entre os representantes de B_i e B_k, menos o raio de B_k, mais uma distância estimada do vizinho mais próximo em B_k. Desta forma, garantimos um raio que vai do rep_i até o objeto $MinPts$ em torno de B_i.

Definição 14 - *Distância de Alcançabilidade Mútua (MRD) entre DBs: Dados os DBs B_i e B_j, o número mínimo de objetos $MinPts$, a distância de alcançabilidade mútua entre B_i e B_j, ou seja, a distância que torna a densidade alcançável um dentro do outro é*:

$$mrd_{MinPts}(B_i, B_j) = max(core_{MinPts}(B_i), core_{MinPts}(B_j), dist(B_i, B_j)) \tag{7}$$

Observe que MRD é o raio mínimo ϵ que garante que B_i e B_j sejam ϵ-alcançáveis. Essas definições são cruciais para a compreensão da hierarquia de *cluster* e extração de múltiplas hierarquias baseadas em densidade.

3.1.2 *CORE-SSG: CORE Summarized Spanning Graph*

A MRD reflete a densidade mínima necessária para conectar dois DBs. Com ela, podemos estimar a distância que conecta todos os pares de DBs no conjunto de dados por densidade, formando um grafo virtual G que representa todas as relações de densidade entre os dados, dado um valor mínimo $MinPts$ de objetos necessários para obter a densidade desejada.

Definição 15 - *Grafo de Alcançabilidade Mútua G: Dado um valor de densidade mínima $MinPts$, o $G_{MinPts} = (V, E)$ é um grafo completo no qual o conjunto de os vértices V representam os DBs e o conjunto de arestas é definido como $E = \{e(B_i, B_j) \mid B_i, B_j \in$*

V *com pesos* $w(e) = mrd_{MinPts}(B_i, B_j)\}$.

A ideia principal por trás dos algoritmos de aprendizado de máquina hierárquico baseados em densidade é extrair um grafo mínimo que conecte todos os objetos no conjunto de dados por densidade (Gertrudes et al., 2019; Campello et al., 2015). Portanto, direta ou indiretamente, estes algoritmos obtêm a MST_{MinPts} do grafo G_{MinPts} e, com ele, constroem uma hierarquia que permite a obtenção de *clusters*, identificação de *outliers* e aplicações. No entanto, diferentes valores de $MinPts$ tendem a derivar resultados diferentes, o que exigiria múltiplas extrações de MSTs a um alto custo computacional. Para evitar isso, dado um limite superior para $MinPts$, definido aqui como $MinPts_{Max}$, podemos substituir o grafo completo G com $O(n^2)$ arestas, pelo grafo *CORE-SG*$_{MinPts_{Max}}$ com $O(n)$ arestas quando aplicado a um conjunto de dados com n objetos, se $MinPts \leq MinPts_{Max} << n$, que é uma suposição natural (Neto et al., 2022).

No entanto, obter o *CORE-SG* de G sobre n objetos usando um algoritmo de extração de MST tradicional é $O(n^2 \cdot log\ n)$, proibitivo para grandes valores de n. No presente trabalho, sumarizamos os dados em DBs e consideramos o grafo completo G sobre m DBs, o que torna o custo para construir o *CORE-SG*$_{MinPts_{Max}}$ $O(m^2 \cdot log\ m)$ e o custo de cada extração de MST_{MinPts} $O(m \cdot log\ m)$. Num cenário com grande volume de dados, é natural considerar que $MinPts_{Max} << m << n$ e, portanto, limitar o custo computacional necessário para nossa proposta a $O(n)$. Em cenários onde esta suposição é inválida, HDBSCAN* pode ser aplicado diretamente sobre os dados.

Como é aplicado sobre DBs, o *CORE-SG* sumarizado será aqui referido como *CORE Summarized Spanning Graph* (*CORE-SSG*) e possui definições e propriedades diferentes de sua versão original (Neto et al., 2022). A seguir apresentaremos seus novos conceitos e propriedades.

Definição 16 - *Grafo do Mínimo Vizinho mais Próximo* $MNNG_{MinPts}$*: Dado um valor de densidade mínima MinPts, o grafo* $MNNG_{MinPts} = (V, E)$ *possui os DBs gerados como um conjunto de vértices* V *sobre os dados e o conjunto de arestas* $E = \{e(B_i, B_j) \mid B_i, B_j \in V \wedge B_j = NN(B_i, k)\ \forall k = [0, SNN(B_i, MinPts)]\}$*. Este grafo conecta os DBs para garantir um número mínimo MinPts de objetos sumarizados conectados. Vemos que* $MNNG_{MinPts}$ *possui duas propriedades importantes, conforme mostrado nos Lemas 1 e 2.*

Lema 1: $\forall\ a \leq b$, $MNNG_a \subseteq MNNG_b$.
Prova: Dado $MNNG_a = (V_a, E_a)$, $MNNG_b = (V_b, E_b)$ e $V_a = V_b = V$, é necessário provar que $E_a \subseteq E_b$ para que o Lema 1 seja verdadeiro. Supondo que $E_a \nsubseteq E_b$, deve haver

uma aresta em E_a que não está em E_b. Por definição, $SNN(B_i, a) \leq SNN(B_i, b)\ \forall\ a \leq b, B_i \in V$. Portanto, o conjunto de valores inteiros do intervalo $[0, SNN(B_i, a)]$ está contido em $[0, SNN(B_i, b)]$, o que torna falsa a afirmação de que existe uma aresta E_a que não está em E_b.

Lema 2: $\forall\ a \leq b : mrd_a(B_i, B_j) = max(core_a(B_i), core_a(B_j)) \Rightarrow e(B_i, B_j) \in MNNG_b$.
Prova: Assumindo que $mrd_a(B_i, B_j) = \max(core_a(B_i), core_a(B_j))$, dada a definição de MRD na Definição 14, segue que $core_a(B_i) \geq dist(B_i, B_j)$ ou $core_a(B_j) \geq dist(B_i, B_j)$, ou ambos. Portanto, pelo menos um desses DBs deve estar na vizinhança do outro, ou seja, $B_j = NN(B_i, k), k = [0, SNN(B_i, a)]$ ou/e $B_i = NN(B_j, k), k = [0, SNN(B_j, a)]$ deve ser válido e implica que $e(B_i, B_j) \in MNNG_a$. Como $a \leq b$, segue pelo Lema 1 que $MNNG_a \subseteq MNNG_b$ e, consequentemente, $e(B_i, B_j) \in MNNG_b$.

O $MNNG_{MinPts}$ é uma adaptação do método k-NNG proposto em Neto et al. (2022). O k-NNG é um grafo que conecta cada objeto de dados com seus k-vizinhos mais próximos, onde k é igual a $MinPts$. Ao sumarizar com DBs, é importante garantir que cada DB esteja conectado aos DBs vizinhos que garantam um mínimo de $MinPts$ objetos sumarizados. Portanto, com $MNNG_{MinPts}$, podemos garantir que os DBs conectados sumarizam pelo menos $MinPts$ objetos em suas vizinhanças, o que é necessário para pesquisas de vizinhos mais próximos para cálculos de densidade.

Definição 17 - *CORE-SG Sumarizado (CORE-SSG): Dado um valor máximo para o parâmetro de densidade mínima $MinPts_{Max}$, o CORE-SSG$_{MinPts_{Max}}$ = $MNNG_{MinPts_{Max}} \cup MST_{MinPts_{Max}}$ é um grafo não direcional que contém as arestas de todas as MST_{MinPts} de G_{MinPts} para cada valor de $MinPts < MinPts_{Max}$. Este recurso permite substituir G (número de arestas $O(n^2)$) por CORE-SSG$_{MinPts_{Max}}$ (número de arestas $O(m \cdot MinPts)$, sendo $m << n$) no processo de extração de um MST_{MinPts}, simplesmente substituindo o peso das arestas $e(B_i, B_j)$ em CORE-SSG$_{MinPts_{Max}}$ por $w(e) = mrd_{MinPts}(B_i, B_j)$ para cada $MinPts < MinPts_{Max}$.*

Teorema 1: *Para dois valores p e q para o parâmetro de densidade $MinPts$, seja M_p o conjunto de todas as MSTs possíveis de G_p, e seja MST_q qualquer MST de G_q, então $\forall\ p < q$: $\exists MST_p \in M_p$: $MST_p \subseteq MST_q \cup MNNG_q$.*
Prova: Vamos considerar uma aresta $e(B_i, B_j)$ em um $MST_p^* \in M_p$ que conecta dois sub-grafos I e J. O peso da aresta $e(B_i, B_j)$ é definido por $mrd_p(B_i, B_j)$, que é o máximo entre $dist(B_i, B_j)$ e suas distâncias *core* $core_p(B_i)$ e $core_p(B_j)$, conforme Definição 14. Devido à propriedade de corte das MSTs, $e(B_i, B_j)$ deve ter o menor peso entre todas as arestas que conectam os conjuntos I e J em G_p. Existem dois cenários:

(1) $e(B_i, B_j) \in MST_q$

(2) $e(B_i, B_j) \notin MST_q$

No cenário (1), como $e(B_i, B_j) \in MST_q$, segue trivialmente que $e(B_i, B_j) \in MST_q \cup MNNG_q$. No cenário (2), MST_q deve ter uma aresta diferente $e(C_i, C_j)$ que conecte os DBs C_i e C_j que estão contidos nos grafos I e J, respectivamente, tendo o menor peso entre as arestas que conectam esses grafos (MSTs são grafos conectados). Assim, as duas afirmações a seguir devem ser verdadeiras (i) $mrd_p(B_i, B_j) \leq mrd_p(C_i, C_j)$ (caso contrário $e(B_i, B_j)$ não poderia ser a aresta com o menor peso conectando I e J em MST_p^*) e (ii) $mrd_q(B_i, B_j) \geq mrd_q(C_i, C_j)$ (caso contrário $e(C_i, C_j)$ não poderia ser a aresta menos ponderada conectando I e J em MST_q). Vimos na Definição 13 que ao aumentar o valor de $MinPts$, as distâncias centrais $core_{MinPts}(\cdot)$ só podem crescer e a distância $dist(\cdot, \cdot)$ entre dois DBs é constante. Portanto, o peso de uma aresta, dado por $mrd_{MinPts}(\cdot, \cdot)$, não pode diminuir quando o valor de $MinPts$ aumenta. Consequentemente, as MRDs (pesos das arestas), definidas na Equação 7, só podem permanecer as mesmas ou aumentar. Portanto, quando $MinPts$ aumenta de p para q ($p < q$ por suposição no teorema), uma das seguintes afirmações deve ser verdadeira:

(a) $mrd_p(B_i, B_j) < mrd_q(B_i, B_j)$

(b) $mrd_p(B_i, B_j) = mrd_q(B_i, B_j)$

Se (a) for verdadeiro, como a distância $dist(\cdot, \cdot)$ entre dois DBs não depende de $MinPts$, a MRD $mrd_q(B_i, B_j)$ deve ser determinada pela distância *core* de B_i ou B_j, ou seja, $mrd_q(B_i, B_j) = max(core_q(B_i), core_q(B_j))$. Do Lema 2 com $MinPts = q$ segue então que $e(B_i, B_j) \in MNNG_q$ e portanto $e(B_i, B_j) \in MST_q \cup MNNG_q$. Se (b) for verdadeiro, então das afirmações (i) e (ii) acima, temos $mrd_q(C_i, C_j) \leq mrd_q(B_i, B_j) = mrd_p(B_i, B_j) \leq mrd_p(C_i, C_j)$. No entanto, como $p < q$ e mrd_{MinPts} não podem diminuir quando $MinPts$ aumenta, segue que $mrd_q(C_i, C_j) = mrd_p(B_i, B_j) = mrd_p(C_i, C_j)$. Neste caso, substituir $e(B_i, B_j)$ em MST_p^* por $e(C_i, C_j)$ resultará em outra $MST_p' \in M_p$, pois o peso total das MST_p^* e MST_p' são iguais. MST_p^* inclui a aresta $e(C_i, C_j)$ e por suposição $e(C_i, C_j) \in MST_q$, e portanto $e(C_i, C_j) \in MST_q \cup MNNG_q$.

Segue que, dado uma MST de G_q, MST_q, e uma MST de G_p, $MST_p^* \in M_p$, com $q > p$, podemos construir um grafo ponderado MST_p' da seguinte forma: (1) inclua todas as arestas e pesos de aresta $e(B_i, B_j) \in MST_p$ que pertencem a $MST_q \cup MNNG_q$ (de acordo com os cenários (1) ou (2)-a); (2) substitua todas as arestas $e(B_i, B_j) \in MST_i$ que não pertencem a $MST_q \cup MNNG_q$ por arestas $e(C_i, C_j)$ que devem existir em $MST_q \cup MNNG_q$(as cenário (2)-b), conectando os mesmos dois subconjuntos de objetos como $e(B_i, B_j)$ e tendo o mesmo peso da aresta como $e(B_i, B_j)$. Este grafo MST_p' é uma MST de G_p, ou seja, $MST_p' \in M_p$ e $MST_p' \subseteq MST_q \cup MNNG_q$. Portanto existe uma $MST_p \in M_p$ tal que

$MST_p \subseteq MST_q \cup MNNG_q$.

O Teorema 1 suporta que o grafo completo $G_{MinPts_{max}}$ pode ser substituído por $MST_{MinPts_{max}} \cup MNNG_{MinPts}$ ao calcular resultados adicionais de MSTs com $MinPts < MinPts_{max}$, com potencial para maior tempo de execução e economia de memória.

Apresentamos no Algoritmo 1, baseado no Teorema 1, um método para calcular eficientemente múltiplas árvores geradoras mínimas baseadas em densidade (MST_{MinPts}) em um intervalo de valores $MinPts$, até um limite máximo $MinPts_{max}$. Na parte (A), a $\text{MST}_{MinPts_{max}}$ é construída usando qualquer algoritmo de MST sobre o grafo "virtual" completo $G_{MinPts_{max}}$. O cálculo de $MNNG_{MinPts_{max}}$ pode ser feito "on-the-fly" durante o cálculo das distâncias *core*, já que a vizinhança necessária para conter os objetos $MinPts$ é calculada para cada DB. À medida que as pesquisas são realizadas, as arestas no grafo $MNNG_{MinPts_{max}}$ são adicionadas. Com a união da $\text{MST}_{MinPts_{max}}$ e $MNNG_{MinPts_{max}}$ podemos construir, sem muito esforço adicional, o $\textit{CORE-SSG}_{MinPts_{max}}$. Este novo grafo é não direcional, onde para um determinado $MinPts$, o peso das arestas é determinado pela MRD entre pares de DBs. Na parte (B) do Algoritmo 1, todos as outras MSTs são extraídas para $MinPts < MinPts_{max}$ de uma versão ponderada de $\textit{CORE-SSG}_{MinPts_{max}}$. Para cada solicitação de $MinPts < MinPts_{max}$, as arestas de $\textit{CORE-SSG}_{MinPts_{max}}$ são reponderadas pela MRD em relação ao $MinPts_i$. Posteriormente, MST_i é extraída do grafo ponderado resultante. Cada extração de MST sobre $MinPts_i$ pode ser feita de forma independente, podendo ser realizadas em qualquer ordem ou até mesmo paralelizadas.

Algoritmo 1 Computação de Múltiplas MSTs baseadas em densidade sobre DBs

Requerido: conjunto de DBs; $MinPts_{max}$

(A) Construir simultaneamente $\text{MST}_{MinPts_{max}}$ e $\textit{CORE-SSG}_{MinPts_{max}}$

1. Encontre e armazene os vizinhos mais próximos para todos os DBs com base em $MinPts_{max}$ e calcule o $MNNG_{MinPts_{max}}$ não direcionado.
2. Compute uma $MST_{MinPts_{max}}$ do grafo de alcançabilidade mutua $G_{MinPts_{max}}$.
3. Construa $\textit{CORE-SSG}_{MinPts_{max}}$ como $MST_{MinPts_{max}} \cup MNNG_{MinPts_{max}}$.

(B) Obtenha as MSTs restantes para $MinPts < MinPts_{max}$

1. For $MinPts$ in {1,...,$MinPts_{max}$ - 1}:
   a) Obtenha $\textit{CORE-SSG}_{MinPts}$ atribuindo distâncias de alcançabilidade mútua em relação ao parâmetro $MinPts$ como pesos das arestas em $\textit{CORE-SSG}_{MinPts_{max}}$.
   b) Encontre a MST_{MinPts} de $\textit{CORE-SSG}_{MinPts_{max}}$.

=0

3.2 *CoreStream*

Propomos uma aplicação de múltiplas hierarquias baseadas em densidade com sumarização de dados em algoritmo hierárquico baseado em densidade em FD, denominada *CORE Summarized Spanning Graph on Data Stream* (CoreStream), que possui definições e propriedades que foram aplicadas ao modelo *Online-Offline*. A proposta é dividida em duas etapas: manutenção de DBs e obtenção de hierarquias com múltiplas densidades ao longo do fluxo.

3.2.1 Fase *Online*: Manutenção dos *Data Bubbles*

Durante a fase *Online* do processamento de dados, os dados de entrada são constantemente processados e sumarizados usando DBs. A função de decaimento exponencial é aplicada às estatísticas do DBs para manter na memória apenas os DBs que possuem pesos altos. Os DBs que possuem pesos menores e não recebem novos objetos são removidos. Em FD as estatísticas de tempo são relevantes para mater um histórico do fluxo atualizado, por isso os DBs precisam ser adaptados com novas estatísticas necessárias para algoritmos de agrupamento em FD.

Definição 18 - *Data Bubbles para Fluxo de Dados: Um DB no tempo t para um grupo de objetos próximos* $X_i = \{o_{j_1}, ..., o_{j_n}\} \subset S$ *com carimbos de data e hora* $T_{j_1}, ..., T_{j_n}$ *é definido como* $B_i = (\overline{CF^1}_i, \overline{CF^2}_i, w_i, rep_i, n_i, extent_i, nnDist_i, t_0)$. o_j *é o objeto adicionado ao* B_i, t_j *é o tempo atual em que o objeto* o_j *entrou no fluxo,* t_u *é o tempo da última atualização do* B_i *antes de* o_j *ser adicionado,* $\overline{CF^1}_i = \sum_{j=1}^{n_i} f(t_j - t_u)o_j$ *é a soma linear ponderada dos objetos,* $\overline{CF^2} = \sum_{j=1}^{n_i} f(t_j - t_u)o_j^2$ *é a soma ponderada dos objetos ao quadrado,* $w = \sum_{j=0}^{n_i} f(t_j - t_u)$ *é o peso,* $rep_i = \frac{\overline{CF^1}_i}{w_i}$ *é a posição do representante de* X_i, $extent_i = \sqrt{\frac{\overline{CF^2}}{w} - (\frac{\overline{CF^1}}{w})^2}$ *é um número natural que define um raio estendido em torno do* rep_i *que contém os objetos de* X_i, *a distância esperada do k-vizinho mais próximo definida como* $nnDist_i(k) = (\frac{k}{w_i})^{\frac{1}{d}} \cdot extent_i$, *sendo d a dimensão dos objetos e k uma constante representando o k-vizinho mais próximo, e* t_0 *é o carimbo de data/hora de criação.*

Observe que $\overline{CF^1}_i$ e $\overline{CF^2}_i$ representam a soma linear e quadrática dos objetos sumarizados (LS e SS, respectivamente) em um espaço euclidiano. Contudo, estes valores agora são ponderados para representar a importância do DB no fluxo. A função de decaimento exponencial é aplicada a estas estatísticas de acordo com a entrada de um novo objeto o ou por um determinado intervalo de tempo δt. Se um novo objeto o_j for inserido, o DB é atualizado da seguinte forma: $B_i = (\overline{CF^1}_i + o_j, \overline{CF^2}_i + o_j^2, w_i + 1)$. Por outro lado, por um certo intervalo de tempo $\delta t = t_c - t_u$, sendo t_c o tempo atual e t_u o tempo da última atualização do DB, o DB é atualizado da seguinte forma: $B_i = (f(\delta t) \cdot \overline{CF^1}_i, f(\delta t) \cdot \overline{CF^2}_i, f(\delta t) \cdot w_i)$. As estatísticas restantes do DB são calculadas usando $\overline{CF^1}_i, \overline{CF^2}_i$ e w_i.

A função de decaimento exponencial reduz gradualmente os valores das estatísticas do DB, já que seu valor está entre $0 \leq f(t) \leq 1$. O modelo *Damped Window* implica que os objetos sumarizados desaparecem gradualmente, o que reduz o raio e a posição de DB no espaço. Na Definição 11, o cálculo de $extent_i$, que foi utilizado no método *CORE-SSG*, exigiu adaptação quando aplicado ao CoreStream, conforme mostrado na Definição 18. O cálculo do raio de DBs na Definição 11 é sensível ao usar a estatística peso w como o número de objetos sumarizados, pois é um número decimal quando aplicado a função de decaimento. A soma ponderada linear e quadrática dos objetos também afetou o cálculo do raio quando reduzido pela função de decaimento, pois passou a apresentar raios imprecisos e inconsistentes com os objetos sumarizados, aumentando o raio mesmo reduzindo o peso do DB. Para resolver essas inconsistências, propomos o uso do cálculo do raio do *Micro-Cluster* (MC) que pode lidar com a redução gradual das estatísticas do DB e calcular raios que sejam mais fiéis aos objetos sumarizados no FD.

À medida que novos objetos são adicionados ao fluxo, eles podem cair em uma região densa e ser adicionados aos DBs existentes ou cair em uma região menos densa sem serem alcançados por qualquer raio dos DBs existentes. Neste caso, um novo DB precisa ser criado e os novos objetos adicionados. A criação e manutenção de DBs devem ser feitas com cuidado para gerar um modelo mais fiel aos dados de entrada. Sem conhecimento prévio dos dados, não sabemos em quais regiões novos objetos poderão cair. Se um DB receber muitos objetos, ele terá muitos dados sumarizados, fazendo com que perca muitas características dos dados e descaracterize os *clusters* finais. Para manter a proporção e distribuição de DBs em regiões de alta e baixa densidade, usamos os conceitos de *potential-Data-Bubbles* e *outliers-Data-Bubbles*, semelhantes a *p-Micro-Clusters* e *o-Micro-Clusters* propostos em Cao et al. (2006).

Definição 19 - *Potential Data Bubble (p-DB): Dado o número mínimo de objetos μ, um DB $B_i = (\overline{CF^1}_i, \overline{CF^2}_i, w_i, rep_i, n_i, extent_i, nnDist_i, t_0)$, o limite do outlier β, $0 < \beta \leq 1$, B_i é um p-DB se $w_i \geq \beta\mu$.*

Definição 20 - *Outlier Data Bubble (o-DB): Dado o número mínimo de objetos μ, um DB $B_i = (\overline{CF^1}_i, \overline{CF^2}_i, w_i, rep_i, n_i, extent_i, nnDist_i, t_0)$, o limite do outlier β, $0 < \beta \leq 1$, B_i é um o-DB se $w_i < \beta\mu$.*

p-DB são os DBs que possuem um mínimo de objetos sumarizados para serem usados para o agrupamento. o-DBs são os DBs que possuem poucos objetos sumarizados e estão presentes em regiões de baixa densidade, sendo considerados *outliers* no agrupamento final. Se nenhum novo objeto for adicionado no p-DB em um determinado tempo, seus

pesos irão decair gradualmente até o ponto em que poderão cair abaixo de $\beta\mu$. Neste caso, significa que p-DB se tornou um o-DB. No caso do o-DB, se novos objetos forem adicionados, seu peso crescerá e poderá ser maior que $\beta\mu$, sendo considerado um p-DB. A mudança de p-DB para o-DB e vice-versa ocorre frequentemente, devido a uma mudança de conceito (*concept drift*), onde regiões densas se movem devido ao fluxo e novos *clusters* aparecem. Portanto, precisamos verificar o peso de cada p-DB e o-DB periodicamente. Para determinar o período que será verificado utilizamos a Equação 8. Esta equação determina o intervalo de tempo mínimo para um p-DB se transformar em um o-DB.

$$T_p = \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu - 1}) \rceil \tag{8}$$

Em cada verificação dos pesos dos DBs, é necessário verificar se os pesos dos o-DBs não caíram abaixo de um peso mínimo calculado na Equação 9. Esta verificação de o-DB permite liberar da memória aqueles que não têm potencial para se tornarem um p-DB. Na equação 9, t_c é o tempo atual e t_o é o tempo em que o-DB foi criado.

$$\xi = \frac{2^{-\lambda(t_c - t_0 + T_p)} - 1}{2^{-\lambda T_p} - 1} \tag{9}$$

A fase Online é descrita no Algoritmo 2, linha 3-9. Objetos de FD são processados assim que entram no algoritmo e são adicionados a um DB existente ou um novo DB é gerado. Os pesos dos DBs potenciais e *outliers* são verificados em um determinado período de tempo T_p, Equação 8. Nesta fase, mantemos os DBs que são relevantes para o fluxo no momento atual t_c e removemos os DBs mais antigo do fluxo, usando o valor calculado da Equação 9.

3.2.2 Fase *Offline*: Múltiplas Hierarquias Baseadas em Densidade

Na fase *Offline*, seja por requisição do usuário ou por um determinado período de tempo, múltiplas MSTs baseadas na densidade são geradas sobre os p-DBs em um tempo corrente t_c. Neste estágio, ele requer apenas o parâmetro de limite de densidade mínimo $MinPts_{max}$. Os passos usados em *CORE-SSG* para gerar múltiplas hierarquias sobre DB são os mesmos usados nesta fase. A fase *Offline* é descrita no Algoritmo 2, linha 10-15.

Primeiro, a distância *core* para cada DB é calculada com base na Definição 13. Em seguida, pesquisas de vizinhos mais próximos são realizadas para cada DB, e construído o grafo $MNNG_{MinPts_{max}}$, as arestas são adicionadas para conectar os DBs aos seus vizinhos mais próximos (Algoritmo 2, linha 11). Então, o grafo de alcançabilidade mútua $G_{MinPts_{max}}$ é construído, conforme Definição 15, com as arestas ponderadas pela MRD (Definição 14). A $MST_{MinPts_{max}}$ é extraída de $G_{MinPts_{max}}$ (Algoritmo 2, linha 12), e o

Algoritmo 2 CoreStream(DataStream *DS*, $MinPts_{max}$, β, μ, λ)

1: calcular a verificação do período de tempo T_p, Equação 8.
2: **repeat**
3: **if** t_c mod $T_p = 0$ **then**
4: verifique os pesos dos *p-DBs* e transforme em *o-DBs* aqueles onde o peso $w < \beta\mu$.
5: verifique os pesos dos *o-DBs* e transforme em *p-DBs* aqueles cujo peso é $w \geq \beta\mu$.
6: verifique os *o-DBs* em que o peso w é menor que o valor da Equação 9.
7: **end if**
8: pegue o próximo objeto $o_j \in DS$ com o *timestamp* atual t_c.
9: insira o_j no *p-DB* ou *o-DB* mais próximo. Caso o_j não estiver próximo a DBs existentes, crie um *o-DB* e insira o_j no novo *outlier*.
10: **if** t_c mod frequenciaDeAgrupamento $= 0$ **then**
11: Encontre e armazene os vizinhos mais próximos para todos *DBs* com base em $MinPts_{max}$ e calcule o $MNNG_{MinPts_{max}}$ não direcionado.
12: Calcule uma $MST_{MinPts_{max}}$ do grafo de alcanlabilidade mútua $G_{MinPts_{max}}$.
13: Construa o *CORE-SSG*$_{MinPts_{max}}$ como $MST_{MinPts_{max}} \cup MNNG_{MinPts_{max}}$.
14: extrairMultiplasHierarquias(*CORE-SSG*$_{MinPts_{max}}$, $MinPts_{max}$)
15: **end if**
16: **until** Fluxo de Dados termina =0

grafo *CORE-SSG* sobre o DBs é então formado pela união da $MST_{MinPts_{max}}$ com o grafo $MNNG_{MinPts_{max}}$ (Definição 17, Algoritmo 2, linha 13). Do grafo *CORE-SSG* extraímos todas $MSTs < MTS_{MinPts_{max}}$ (Algoritmo 2, linha 14).

3.3 Índices de Avaliação de Hierarquias e de Partições

Após ser geradas as múltiplas hierarquias em um intervalo de $MinPts$, primeiro é preciso medir o quão duas hierarquias são similares ou dissimilares. Em Johnson et al. (2013) os autores propõem o Índice de Acordo de Hierarquia (HAI), que identifica o quanto duas hierarquias concordam entre si, analisando a distância entre pares de objetos em ambas as hierarquias. Dado uma base de dados X com m objetos $\{o_1, ..., o_m\}$, a similaridade entre duas hierarquias H_1 e H_2 é definida pela diferença média normalizada das distâncias $d_{H_1}(o_i, o_j)$ e $d_{H_2}(o_i, o_j)$ entre todos os pares de objetos nas hierarquias, de acordo com a Equação 10. n representa o número de *clusters* na hierarquia, e $d_H(a, b)$ representa a distância hierárquica entre a e b. A distância hierárquica $d_H(a, b)$ é calculada sobre o menor *cluster* c na hierarquia que contenha a e b, sendo $d_H(a, b) = |c|/n$. $d_H(a, b)$ será 0 caso c seja um nó raiz da hierarquia.

$$HAI(H_1, H_2) = 1 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}|d_{H_1}(x_i, x_j) - d_{H_2}(x_i, x_j)| \tag{10}$$

Como critério de validação externa para comparar as partições finais dos agrupamentos, foi utilizado o Adjusted Rand Index (ARI) (Hubert and Arabie, 1985). Dado um conjunto de n objetos $S = \{o_1, ..., o_n\}$, e dada duas partições $U = \{u_1, ..., u_n\}$ e $V = \{v_1, ..., v_n\}$, sendo u_i e v_i os grupos finais relacionado a cada objeto o_i de S, supondo que U é nosso critério externo (*ground truth*) e V é o resultado do agrupamento. O ARI assume a distribuição hipergeométrica generalizada como modelo de aleatoriedade, ou seja, as partições U e V são escolhidas aleatoriamente de forma que o número de objetos nas classes e *clusters* seja fixo. Dado n_{ij} o número de objetos que estão em ambas classes u_i e *cluster* v_j. Temos $n_{i.}$ e $n_{.j}$ o número de objetos na classe u_i e *cluster* v_j respectivamente. As anotação são ilustradas na Figura 8 que apresenta a tabela de contingência para comparar duas partições e que suas informações são usadas para o cálculo do ARI mostrada na Equação 11. Os valores de ARI estão entre -1 e 1, onde valores próximos de 1 significam alta similaridade entre as partições e valores próximos de -1 alta aleatoriedade das partições comparadas. $\sum_{i,j}\binom{n_{i,j}}{2}$ define o número de pares de objetos no mesmo *cluster* em U e mesmo *cluster* em V. $\sum_i\binom{n_{i.}}{2} - \sum_{i,j}\binom{n_{i,j}}{2}$ define o número de pares de objetos no mesmo *cluster* em U e em *clusters* diferentes em V. $\sum_j\binom{n_{.j}}{2} - \sum_{i,j}\binom{n_{i,j}}{2}$ define o número de pares de objetos no mesmo *cluster* em V e em *clusters* diferentes em U.

Class \ *Cluster*	v_1	v_2	...	v_C	*Sums*
u_1	n_{11}	n_{12}	...	n_{1C}	$n_{1.}$
u_2	n_{21}	n_{22}	...	n_{2C}	$n_{2.}$
⋮	⋮	⋮		⋮	⋮
u_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R.}$
Sums	$n_{.1}$	$n_{.2}$	...	$n_{.C}$	$n_{..} = n$

Figura 8 – Notação para a tabela de contingência para comparar duas partições

$$ARI = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - \frac{\left[\sum_i\binom{n_{i.}}{2}\sum_j\binom{n_{.j}}{2}\right]}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i\binom{n_{i.}}{2} + \sum_j\binom{n_{.j}}{2}\right] - \frac{\left[\sum_i\binom{n_{i.}}{2}\sum_j\binom{n_{.j}}{2}\right]}{\binom{n}{2}}} \tag{11}$$

Capítulo 4

Experimentos e Resultados

Nesta seção, avaliamos o desempenho e qualidade do uso de *CORE-SSG* para a tarefa de agrupamento, embora possa ser usado em diversas aplicações (Gertrudes et al., 2019), e avaliamos o uso de CoreStream para gerar múltiplas hierarquias baseadas em densidade no FD e também podem ser usado em várias aplicações (Zubaroğlu and Atalay, 2021).

4.1 Avaliação *CORE-SSG*

A implementação do *CORE-SSG*[1] foi em Python/Cython, e os experimentos foram realizados em um computador com 16GB de RAM e 8 núcleos de processamento. As avaliações experimentais são divididas em três partes: avaliação da qualidade das hierarquias e partições extraídas do *CORE-SSG*; as soluções de agrupamento de *CORE-SG* e *CORE-SSG* são comparadas; é feita uma análise do tempo de execução e velocidade dos algoritmos comparados.

4.1.1 Avaliação da Perda de Qualidade Resultante da Sumarização em *CORE-SSG*

Avaliamos as hierarquias geradas do *CORE-SSG* para entender o impacto da sumarização nos *clusters* hierárquicos e a extração de múltiplos resultados baseados em densidade. Avaliamos por meio de validação externa as partições FOSC extraídas do *CORE-SSG* e como o parâmetro de densidade $MinPts$ influencia na seleção dos *clusters* finais. Três bases de dados bidimensionais com tamanhos 38k, 43k e 49k foram sintetizadas para os experimentos de avaliação de qualidade e divididos em 13, 10 e 12 *clusters* com ruído

[1] Implementação em: <https://github.com/natanaelbatista99/CORE-SSG>

(ver Tabela 2). Depois disso, cada base de dados foi dividida em subconjuntos do mesmo volume de modo que o número de subconjuntos totalizasse $n/10$, ou seja, 10% do tamanho das bases de dados. A taxa de sumarização define o número de DBs gerados da base de dados, ou seja, sendo n o número de objetos da base e p a taxa de sumarização, temos que o número de DBs gerados é $(n \cdot p)/100$. Então, os DBs foram criados sobre esses subconjuntos, e um *CORE-SSG* com $MinPts = 200$ foi construído. Do $CORE\text{-}SSG_{200}$, extraímos MSTs baseadas em densidade para $MinPts$ variando de 200 a 2 em etapas decrescentes de 2. Usando as MSTs extraídas, construímos cem soluções de agrupamento HDBSCAN* e calculamos sua similaridade entre pares usando o índice HAI (Johnson et al., 2013). Com base nas suas semelhanças, escolhemos quatro das soluções de agrupamento mais representativas, ou seja, selecionamos quatro hierarquias em que cada hierarquia representa um conjunto de hierarquias $MinPts$ semelhantes entre si, mas ao mesmo tempo apresenta baixa semelhança entre as outras hierarquias representantes, de acordo com o índice HAI. Esses quatro resultados de agrupamento hierárquico são apresentados nas hierarquias de agrupamento da Figura 9, onde os *clusters* mais proeminentes, de acordo com a estrutura FOSC (Campello et al., 2013), estão circulados em vermelho. A estrutura FOSC utiliza o conceito de excesso de massa para extrair a partição mais proeminente de uma solução de *cluster* hierárquica. Conforme feito no artigo HDBSCAN*, assumimos que o valor mínimo do tamanho do *cluster* é igual a $MinPts$, pois não faz sentido um *cluster* ser menor que o número mínimo de objetos necessários para uma região ser definida como *core*.

Tabela 2 – Variáveis para avaliação das partições do *CORE-SG* e *CORE-SSG*.

Variáveis	**Calores**
$MinPts_{max}$	200, step 2
#objetos	38k, 43k and 49k
#dimensão	2
#clusters	13, 10 and 12

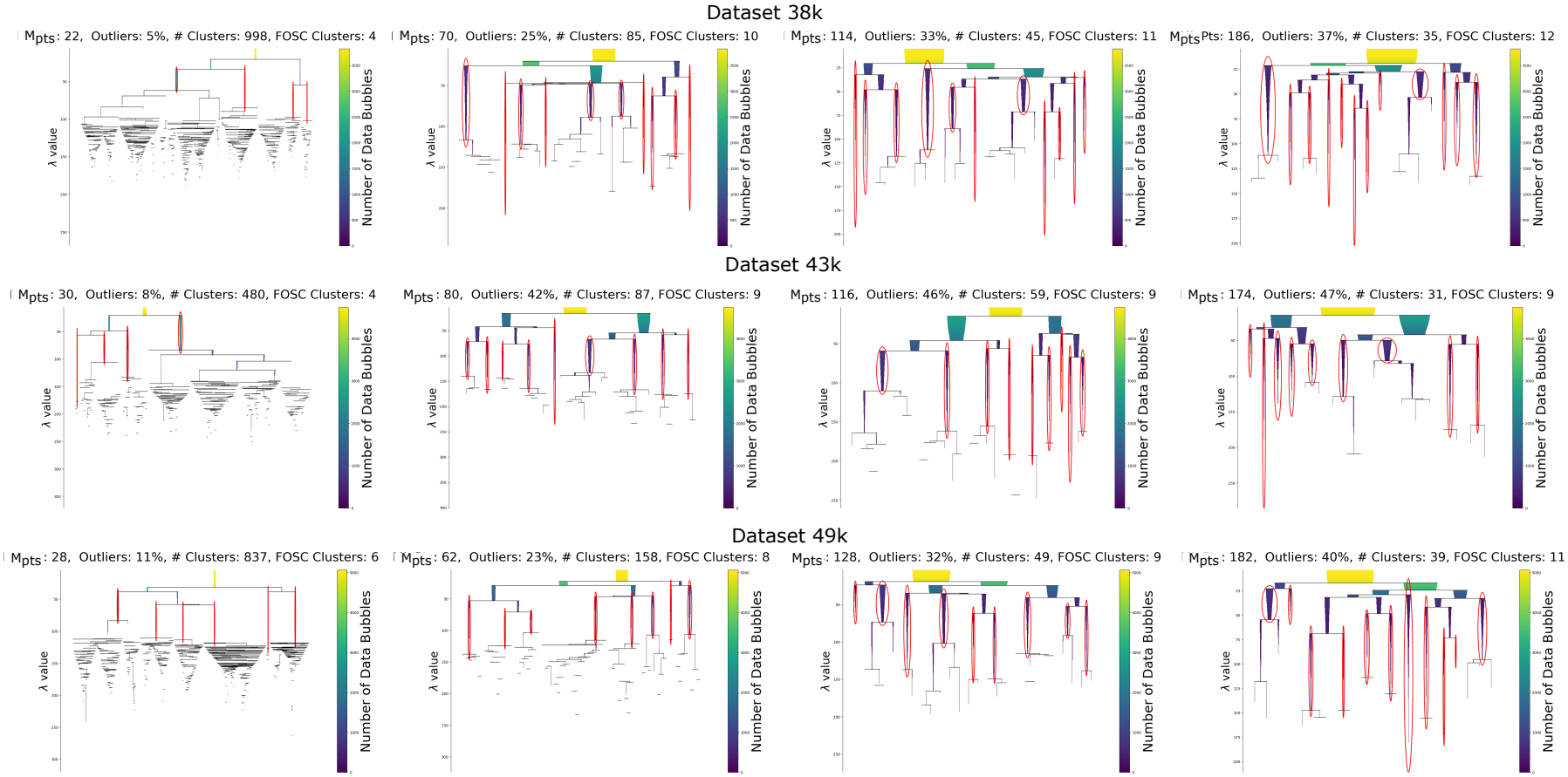


Figura 9 – *Clusters* hierárquicos das soluções mais representativas do *CORE-SSG*.

As hierarquias de *cluster* apresentadas na Figura 9 refletem com precisão o comportamento e variedade do agrupamento HDBSCAN* para um intervalo de valores $MinPts$. Na Figura 9 nas hierarquias $MinPts = 22$ do *Dataset* $38k$, $MinPts = 30$ do *Dataset* $43k$ e $MinPts = 28$ do *Dataset* $49k$ demonstram que para valores pequenos de $MinPts$, a hierarquia está cheia de *clusters* pequenos e altamente detalhados que são difíceis de visualizar, mas podem ser analisados usando índices de validação de *cluster* (Vendramin et al., 2010). Em contraste, à medida que os valores de $MinPts$ aumentam, as hierarquias compreendem um número menor de *clusters* mais robustos que refletem estruturas mais densas, como visto na Figura 9 nas hierarquias $MinPts = 186$ do *Dataset* $38k$, $MinPts = 116$ do *Dataset* $43k$ e $MinPts = 182$ do *Dataset* $49k$. Com $MinPts$ maiores, cada DB precisa de mais conexões para garantir a densidade, mantendo assim os *clusters* mais estáveis na hierarquia. As bases de dados 38k e 49k apresentam *clusters* com maior diversidade de densidades e mostram uma maior variabilidade das hierarquias ao longo dos $MinPts$ e que os *clusters* mais estáveis mudam de acordo com o parâmetro de suavização. Em conjuntos de dados em que os *clusters* estão em densidades semelhantes, como na base de dados 43k, espera-se que as hierarquias sejam muito semelhantes entre si, devido à perda das características mais intrínsecas dos dados na sumarização. No entanto, ao usar cálculos de densidade com DBs, a base de dados de 43k exibiu variações nas hierarquias. Apesar do FOSC selecionar o mesmo número de *clusters*, as hierarquias diferem no tamanho e na altura dos *clusters* e nos *outliers*, apresentados nas hierarquias $MinPts = 30, MinPts = 80, MinPts = 174$ e $MinPts = 116$ do *Dataset* $43k$ da Figura 9. Os resultados imitam perfeitamente o que é esperado de algoritmos de agrupamento baseados em densidade. A diferença reside no fato de que *CORE-SSG* não possui a estrutura hierárquica dos dados sumarizados, ou seja, dentro dos DBs. Na maioria dos casos, os dados sumarizados apresentam um nível de detalhe de menor importância para grandes volumes de dados. Porém, se o usuário necessitar da estrutura dentro dos DBs, o *CORE-SG* pode ser aplicado sobre os dados sumarizados de cada DB de forma independente, e os resultados (que são grafos) combinados. Se os dados sumarizados ainda forem extensos, o *CORE-SSG* pode ser aplicado recursivamente. Para isso, aplicamos a sumarização sobre os DBs existentes, criando macro-DBs (principalmente em regiões com alta densidade e conectividade de sumarização), construímos o grafo *CORE-SSG* sobre os macro-DBs e extraímos as múltiplas soluções hierárquicas baseadas em densidade.

Embora uma hierarquia de *clusters* seja uma forma mais completa e rica de representar a estrutura de dados, ela pode ser muito complexa e extensa para bases de dados muito grandes. Nesses cenários, a extração automática de uma partição de *clusters* pode ser preferida. FOSC é uma excelente escolha para este trabalho (Campello et al., 2013). Na Figura 10, são apresentadas as partições selecionadas (circuladas) pelo FOSC na Figura 9. No gráfico temos a porcentagem de DBs *outliers*, o número de *clusters* selecionados pelo

FOSC e o número de DBs gerados em cada base de dados. Os DBs azuis representam os *outliers* e as demais cores representam os *clusters*. É possível notar que o FOSC extraiu com sucesso *clusters* proeminentes nas bases de dados e separou corretamente o ruído. Além disso, diferentes valores de $MinPts$ resultaram em diferentes seleções de *cluster*, onde valores pequenos favorecem a seleção de *clusters* mais amplos que residem nos níveis mais altos da hierarquia, pois estão mais presentes na hierarquia do que seus vários *clusters* filhos menores aninhados, como pode ser observado na Figura 10 nos gráficos $MinPts = 22$ do *Dataset* $38k$, $MinPts = 30$ do *Dataset* $43k$ e $MinPts = 28$ do *Dataset* $49k$. Em contraste, um valor mais alto de $MinPts$ evita estruturas de *cluster* menores na parte inferior da hierarquia, favorecendo a seleção de *clusters* mais densos aninhados, apresentados na Figura 10 nos gráficos $MinPts = 186$ do *Dataset* $38k$, $MinPts = 174$ do *Dataset* $43k$ e $MinPts = 182$ do *Dataset* $49k$. Os resultados explicitam a importância da análise de múltiplos resultados com diferentes valores de $MinPts$, pois todos eles têm potencial para serem considerados "os melhores" e validam o objetivo principal do presente trabalho. Além disso, sumarizar os dados não teve impacto na escolha dos *clusters* feita pelo FOSC, uma vez que a estrutura detectou os mais proeminentes para diferentes valores de $MinPts$, como esperado.

Para analisar as partições finais das bases de dados, foi utilizado o ARI como critério para validação externa dos *clusters* (Hubert and Arabie, 1985), onde 1 indica que as partições são idênticas e 0 é ajustado para refletir a aleatoriedade. O índice ARI foi utilizado para comparar par-a-par todas as partições do *CORE-SSG*, ou seja, comparando cada partição do *CORE-SSG* com todas as outras partições. Os *heatmaps* (mapas de calor) na Figura 11 mostram a comparação de todas as partições usando o ARI, onde as cores escuras refletem ARIs baixos e as cores claras ARIs altos. Através dos *heatmaps* são observados intervalos de $MinPts$ que possuem similaridade e dissimilaridade, revelando que nas bases de dados criadas há diversidade na densidade dos *clusters*. As áreas escuras no *heatmap* demonstram como os $MinPts$ baixos são diferentes dos $MinPts$ altos. À medida que $MinPts$ aumenta, os *clusters* densos na hierarquia permanecem mais, favorecendo a seleção de *clusters* mais internos pelo FOSC, evidenciado na Figura 9 nas hierarquias $MinPts = 186$ do *Dataset* $38k$, $MinPts = 116$ do *Dataset* $43k$ e $MinPts = 182$ do *Dataset* $49k$. Para $MinPts$ baixo a hierarquia fica cheia de *clusters* pequenos, favorecendo a seleção de *clusters* mais acima na hierarquia, como pode ser observado na Figura 9 nas hierarquias $MinPts = 22$ do *Dataset* $38k$, $MinPts = 30$ do *Dataset* $43k$ e $MinPts = 28$ do *Dataset* $49k$. Esse comportamento é semelhante aos algoritmos de agrupamentos baseados em densidade. Devido à sumarização em que os objetos internos em DBs não são analisados na hierarquia, os *clusters* mais baixos na hierarquia são condensados, reduzindo o número de *clusters* fragmentados em altas densidades.

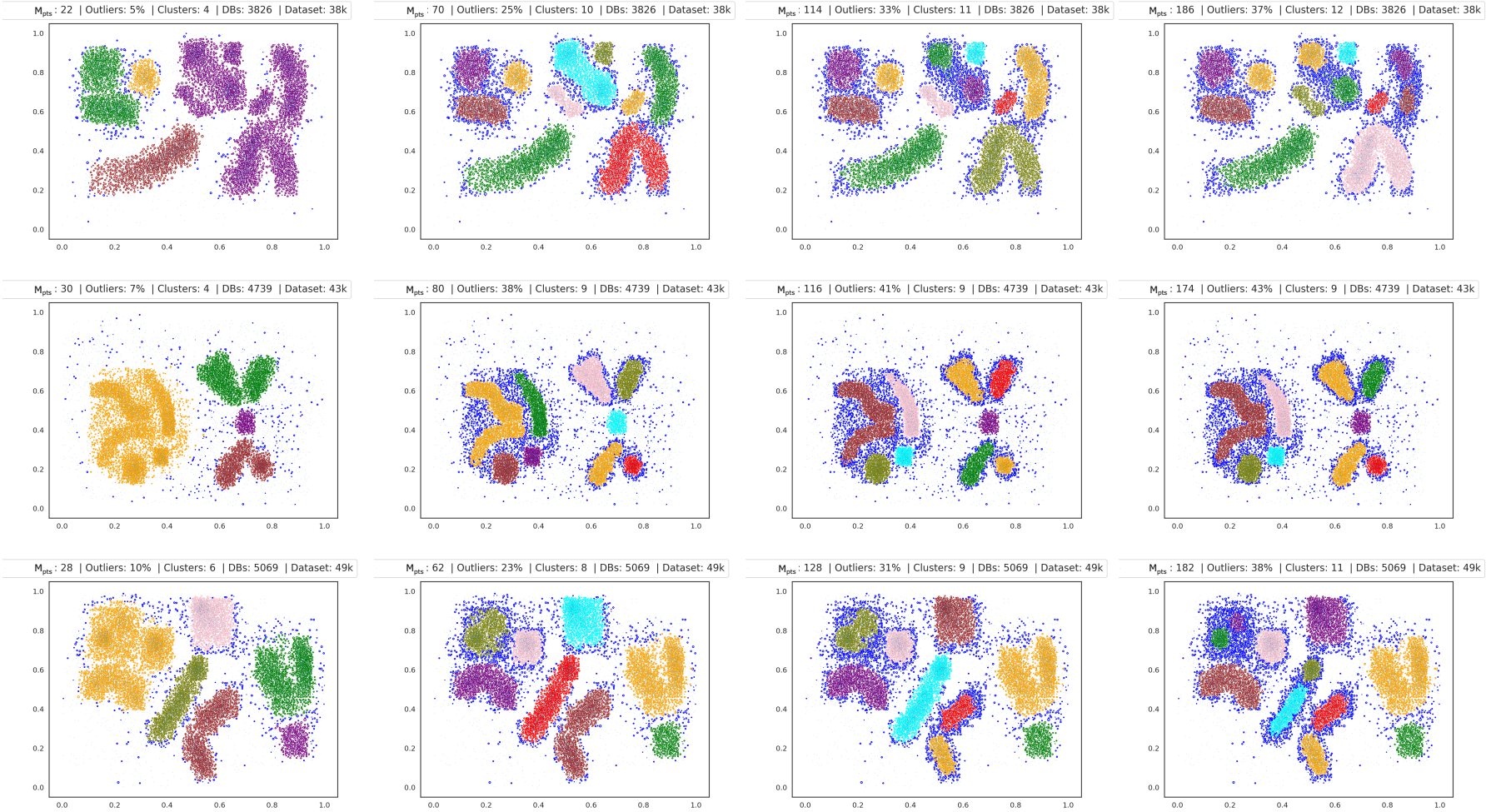


Figura 10 – Gráfico de *cluster* por representante $MinPts$ das bases de dados.

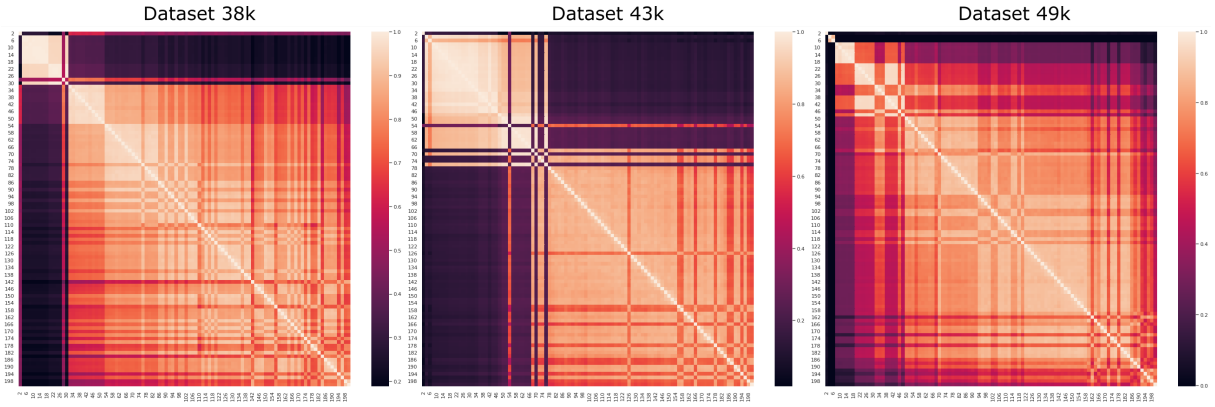
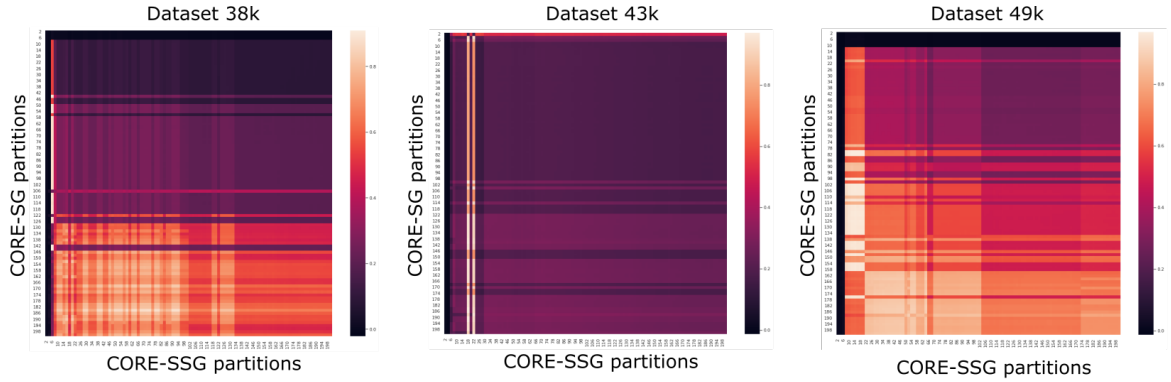


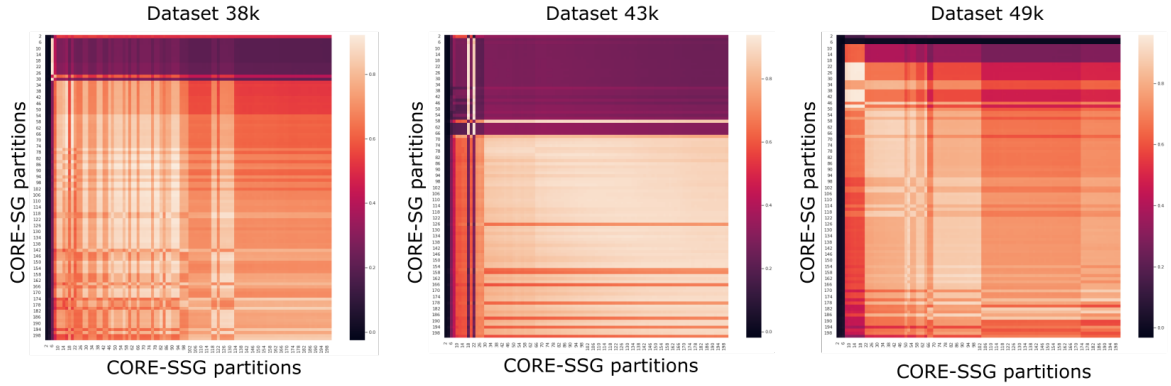
Figura 11 – Heatmap ARI das partições do *CORE-SSG*.

4.1.2 Comparação entre as Partições *CORE-SG* e *CORE-SSG*

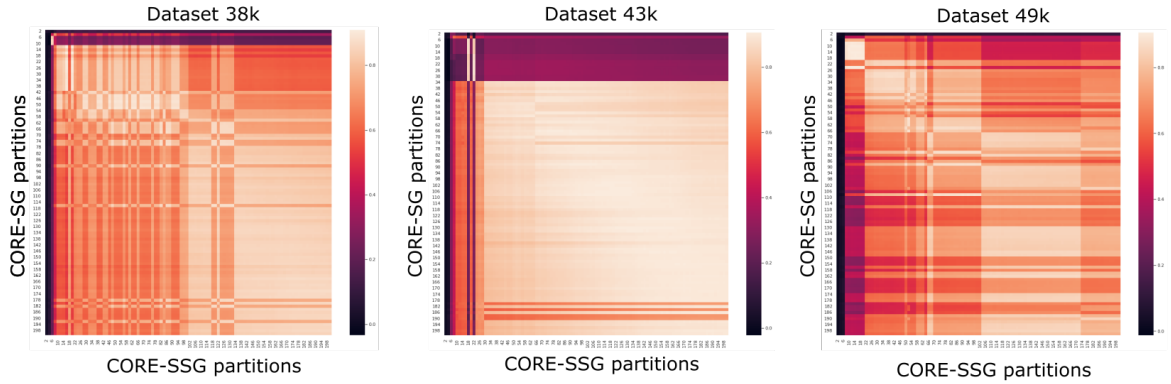
Para avaliar o quão similares ou dissimilares estão as soluções do *CORE-SSG* e *CORE-SG*, avaliamos a similaridade entre os pares de partições de ambos algoritmos usando o índice de validação externa. O índice de comparação entre duas hierarquias faz comparação par-a-par entre os objetos de dados nos *clusters* da hierarquia, buscando o nível hierárquico de maior densidade em que os objetos se encontram. Como as hierarquias de *CORE-SG* e *CORE-SSG* não são comparáveis devido à sumarização, comparamos as partições extraídas do FOSC entre *CORE-SG* e *CORE-SSG* para verificar se os *clusters* proeminentes forem semelhantes. Usamos o ARI para obter as semelhanças entre as partições extraídas. A Figura 12 apresenta os *heatmaps* dos ARIs resultantes da comparação pareada entre as partições de *CORE-SG* e *CORE-SSG* com valores $MinPts$ diferentes. As partições *CORE-SG* são apresentadas no eixo vertical, e as partições *CORE-SSG* estão no eixo horizontal. Os *heatmaps* mostram que, para valores pequenos de $MinPts$, o *CORE-SG* possui uma partição muito mais detalhada, e o *CORE-SSG* tende a unificar pequenos *clusters*, o que é esperado porque as informações dentro dos DBs são sumarizadas e não podem ser divididas. Isso é refletido nas áreas mais escuras em valores pequenos de $MinPts$. Para grandes bases de dados, pequenos valores de $MinPts$ tendem a fragmentar excessivamente a estrutura de *cluster* em uma solução difícil de interpretar. De acordo com o ARI, para valores médios e altos de $MinPts$, a maioria dos *heatmaps* apresentam áreas brilhantes que refletem uma similaridade superior a 0,85. Este resultado mostra que a principal estrutura de *cluster* dos dados está presente nos resultados de ambos os grafos, às vezes para valores distintos de $MinPts$, o que também sustenta a importância de uma análise de múltiplas densidades da base de dados, que representa o objetivo desta dissertação.



(a) 5% do Tamanho Original



(b) 10% do Tamanho Original



(c) 15% do Tamanho Original

Figura 12 – Comparando partições com e sem DBs nas bases de dados com 5, 10 and 15 por cento de sumarização sobre o número de objetos da base.

A Figura 12 mostra as partições *CORE-SG* e *CORE-SSG* geradas usando a taxa de sumarização para 5%, 10% e 15% sobre o tamanho das bases de dados. Os experimentos da Seção 4.1.1 foram realizados com uma taxa de sumarização de dados de 10% (Figura 12(b)), mantendo uma alta similaridade nas partições *CORE-SSG* finais com as partições do *CORE-SG* e menos DBs armazenados na memória. Ao aumentar a taxa de sumarização, o número de DBs aumenta e o número de objetos internos é reduzido. Isso reduz a

perda de informações dos dados originais. Na Figura 12(c), os *heatmaps* representam a taxa de sumarização de 15%. Assim como a taxa de sumarização de 10%, os valores de ARI permaneceram acima de 0.85, e as áreas claras dos *heatmaps* aumentaram, indicando que as partições *CORE-SSG* estavam mais relacionadas com as partições do *CORE-SG*. À medida que a taxa de sumarização aumenta, as áreas claras dos *heatmaps* tendem a aumentar e as áreas escuras tendem a diminuir, conforme mostrado nos *heatmaps* das bases de dados na Figura 12. Com a taxa de sumarização de 5% (Figura 12(a)), menos DBs foram gerados, o que significa que o número de objetos internos sumarizados aumentaram e a qualidade da partição final é impactada. Entretanto, usar DBs reduz esse impacto. Embora existam poucas áreas claras nos mapas de calor, para uma faixa considerável de $MinPts$, *CORE-SSG* encontrou pelo menos uma partição de *CORE-SG* com ARI acima de 0.85 de similaridade, e algumas partições tiveram $ARI > 0.9$.

4.1.3 Análise de Tempo de Execução

Para avaliar o desempenho computacional do *CORE-SSG* em relação ao *CORE-SG*, criamos bases de dados com uma variedade de tamanhos, dimensões e diferentes níveis de sumarização, conforme descrito na Tabela 3. Para avaliar cada característica separadamente, os valores padrão foram fixados e apresentados em negrito na Tabela 3 e também consideramos uma variedade de valores $MinPts_{max}$ usados para construir os grafos. O tempo de execução necessário para a obtenção dos grafos, *CORE-SG* e *CORE-SSG*, somado ao tempo necessário para a extração das múltiplas soluções de densidade hierárquicas, são apresentados na Figura 13.

Tabela 3 – Configurações dos bases de dados para o experimento de tempo de execução do *CORE-SSG* E *CORE-SG*.

Variáveis	**Valores**
$MinPts_{max}$	100, 150, **200**, 250, 300
#objetos	100k, 250k, **500k**, 750k, 1M
#dimensões	2, **4**, 6, 8
#sumarização	5%, **10%**, 15%

A Figura 13(a) mostra que o parâmetro $MinPts$ possui uma correlação linear com o tempo de processamento necessário para rodar o experimento para ambos os grafos, mas *CORE-SSG* apresentou velocidades próximas de 20 vezes melhor que *CORE-SG*. Considerando o tamanho da base de dados n e $m << n$, a Figura 13(b) mostra claramente o comportamento quadrático de *CORE-SG* à medida que o tamanho da base de dados aumenta para um milhão e como o custo computacional é amortizado ao usar DBs, ou seja, para *CORE-SSG*. Os presentes resultados apoiam a principal hipótese de pesquisa deste trabalho: a sumarização de dados permite escalabilidade para soluções de aprendizado de máquina baseadas em densidade, algoritmos de *cluster* em particular. A sumarização

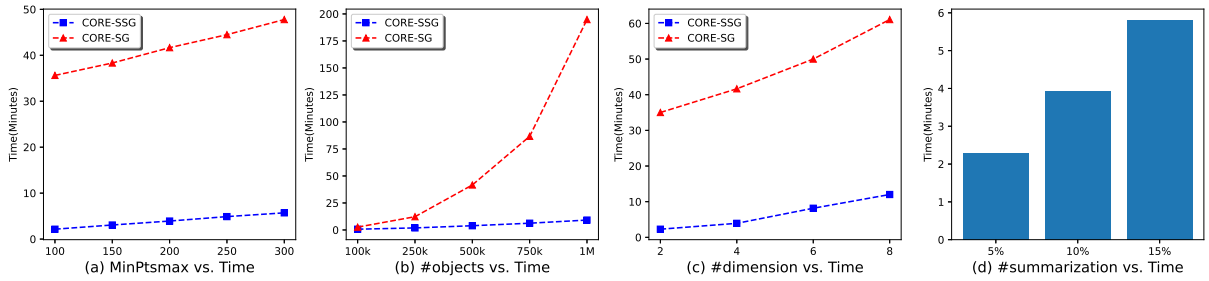


Figura 13 – Tempo de execução para construir os grafos e extrair as soluções hierárquicas baseadas em densidade para *CORE-SG* e *CORE-SSG*.

também teve impacto no tempo de execução total dos algoritmos quando a dimensionalidade dos dados aumenta, conforme mostrado na Figura 13(c), com velocidades de até 15 vezes. Por último, mas não menos importante, o tempo total necessário para a sumarização dos dados, apresentado na Figura 13(d), reflete o aumento linear do tempo computacional relacionado à proporção de DBs utilizados, o que reflete o comportamento esperado para nossa proposta e mostra que a sumarização requer uma pequena fração do tempo total de execução. À medida que a taxa de sumarização aumenta, a qualidade do agrupamento se aproxima dos resultados dos objetos originais. No entanto, o crescimento do tempo é linear para taxas de sumarização, e o algoritmo levará mais tempo para gerar as múltiplas MSTs em taxas de sumarização mais altas. É fundamental destacar que a taxa de sumarização de 10% utilizada nas hierarquias e experimentos de comparação de partições revelou excelente qualidade de agrupamento, e o tempo de execução é mais rápido para geração das múltiplas MSTs.

4.2 Avaliação CoreStream

Nesta seção iremos considerar o cenários de FD e iremos comparar os resultados das múltiplas hierarquias baseadas em densidade do CoreStream com as soluções do algoritmo HASTREAM, avaliando desempenho e qualidade. A implementação do CoreStream[2] e do HASTREAM foram na linguagem Python usando a biblioteca *river* (usada para modelos em FD), e os experimentos foram realizados em um computador com 16GB de RAM e 8 núcleos de processamento. As avaliações experimentais são divididas em duas partes: são comparadas as soluções de agrupamento do CoreStream e HASTREAM; e é feita uma análise do tempo de execução e velocidades dos algoritmos comparados.

[2] Implementação em: <https://github.com/natanaelbatista99/CoreStream>

4.2.1 Comparação entre as Partições CoreStream e HASTREAM

Avaliamos as partições do CoreStream e HASTREAM extraídas pelo FOSC para avaliar a qualidade do agrupamento e o impacto dos modelos de sumarização usados para estimar a densidade.

A Tabela 4 contém as bases de dados usadas para os experimentos de avaliação de qualidade. Três bases de dados sintéticas bidimensionais com 28k, 40k e 100k foram criadas variando o número e a densidade dos *clusters* em cada *timestamp*. Essas bases de dados de controle permitem um melhor monitoramento do comportamento dos DBs em todo o fluxo e a visualização dos *clusters* finais gerados. Bases de dados reais também foram avaliadas, e são: KDD CUP'99[3] *Network Intrusion Detection* de 41 atributos, 34 são contínuos e os atributos nominais transformados em ordinais, e 494021 objetos de dados; Base de dados *Forest Covertype* com 10 atributos contínuos e 581012 objetos de dados; Base de dados *Poker-Hand* (PK) com 10 atributos e 900000 objetos de dados, mas aqui usamos 250000 primeiras instâncias; *Hyper Plane Stream*[4], com 100000 instancias e 10 atributos. O λ presente na Tabela 4 foi escolhido para manter, em cada execução das bases de dados, a quantidade de p-DBs e Potential Micro Cluster (p-MC)s em torno de 10% do valor de entrada em cada *timestamp*, ou seja, sendo n o número de objetos ao entrarem no fluxo antes da chamada *Offline* o número de p-DBs e p-MCs mantidos será em torno de $n/10$ (n é definido para cada base de dados conforme Tabela 4, coluna n° objetos por *timestamp*). A velocidade do fluxo v é definida como 100 objetos por unidade de tempo. O parâmetro β é definido como 0.75, e o número mínimo de objetos armazenados nos DBs $mu = 2$. Na fase Online, CoreStream mantém os p-DBs e o-DBs, e em cada solicitação na fase *Offline*, o grafo *CORE-SSG* com $MinPts_{max} = 200$ é construído sobre os p-DBs. Do *CORE-SSG*$_{200}$, extraímos as MSTs baseadas em densidade para $MinPts$ variando de 200 a 2 em etapas decrescentes de 2. Usando as MSTs extraídas, construímos cem soluções de agrupamento HDBSCAN*. Na fase *Online*, o HASTREAM mantém os p-MC e os Outlier Micro Cluster (o-MC). Em cada solicitação na fase *Offline*, variamos o $MinPts$ de 200 a 2 em passos decrescentes de 2, extraímos a MST$_{MinPts}$ baseada em densidade do grafo completo G_{MinPts} sobre os p-MCs para cada valor de $MinPts$ no intervalo. São geradas cem soluções de agrupamento HDBSCAN*.

Para avaliar os resultados de agrupamento de ambos os algoritmos, usamos o índice ARI para comparar a partição $MinPts_i$ final extraída pelo FOSC da hierarquia do CoreStream, com o $MinPts_i$ *ground truth* (os *clusters* verdadeiros), para um intervalo de $MinPts$. Usamos o índice ARI para comparar a partição $MinPts_i$ final extraída pelo FOSC da hierarquia do HASTREAM, com o $MinPts_i$ *ground truth*, para um intervalo

[3] <https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data/data>
[4] <https://www.cse.fau.edu/~xqzhu/Stream/hyperP.arff>

de $MinPts$. E como *ground truth* usaremos as partições FOSC do HDBSCAN* sobre os objetos sumarizados nos p-DBs e p-MCs de cada algoritmo, ou seja, em cada chamada *Offline* selecionamos os objetos que entraram nos DBs e MCs dentro do *timestamp* para gerar o agrupamento final do HDBSCAN* sobre os objetos. HDBSCAN* é um algoritmo hierárquico baseados em densidade estado da arte, e o objetivo de usar a sumarização com DB é extrair hierarquias que produzam *clusters* semelhantes aos algoritmos de agrupamento hierárquicos baseados em densidade em objetos. Cada algoritmo possui diversas chamadas *Offline*, e em cada chamada, são extraídas cem hierarquias baseadas em densidade. Aplicamos o ARI para cada partição $MinPts$ para todas as cem partições extraídas das hierarquias dentro de cada chamada *Offline*. Selecionamos as partições $MinPts_i$ no tempo corrente t_c extraída do CoreStream, extraímos os objetos sumarizados dentro dos p-DBs que foram usados no agrupamento e aplicamos o algoritmo HDBSCAN* aos objetos em $MinPts_i$, gerando as partições HDBSCAN* $MinPts_i$ por meio do FOSC. O ARI é então aplicado entre cada partição CoreStream $MinPts_i$ e partição HDBSCAN* $MinPts_i$ em t_c. Para cada chamada *Offline* para HASTREAM, aplicamos as mesmas etapas de comparação ARI entre as partições HASTREAM $MinPts_i$ e as partições HDBSCAN* $MinPts_i$, mas os objetos são extraídos dos p-MCs. Os ARIs gerados em cada chamada *Offline* j revelam o quanto as partições CoreStream e HASTREAM em j se assemelham às partições HDBSCAN* em j. Após todas as solicitações *Offline*, aplicamos a média e o desvio padrão de todos os ARIs calculados ao longo do fluxo. A Figura 14 apresenta as médias e desvios padrão dos ARIs para cada partição $MinPts_i$ de cada base de dados.

Tabela 4 – Configurações e características das bases de dados para os experimentos sobre as partições do CoreSream e HASTREAM.

Base de Dados	n° **de objetos por** ***timestamps***[1]	$\lambda_{CoreStream}$	$\lambda_{HASTREAM}$	**dimensões**	**tamanho**	**Tipo**
28k2d	7000	0.035	0.001	2	28000	sintético
40k2d	10000	0.02	0.02	2	40000	sintético
100k15c	10000	0.001	0.001	2	100000	sintético
covertype	20000	0.01	0.003	10	581012	dados reais
kddcup	20000	0.0025	0.0025	41	494021	dados reais
Poker-Hand	5000	0.001	0.012	10	250000	dados reais
Hyper	2000	0.001	0.012	10	100000	dados reais

[1] *timestamp* representa o n° de objetos que entram na fase *Online* até que a fase *Offline* seja chamada.

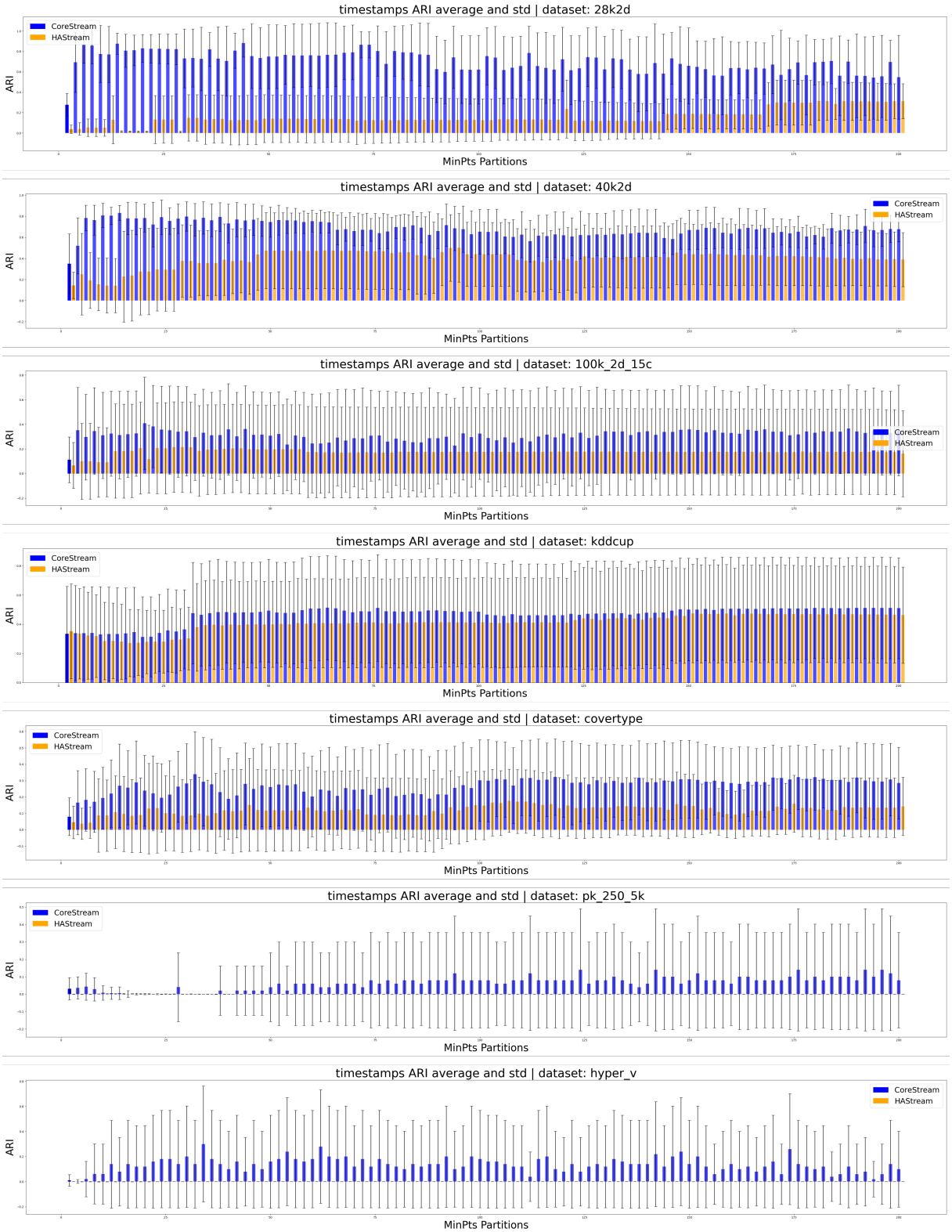


Figura 14 – Média e Desvio Padrão do ARI aplicado aos *timestamps*. Comparando as partições Corestream e HASTREAM.

As barras azuis na Figura 14 representam as médias de ARIs de todas as execuções *Offlines* para cada $MinPts$ da execução do Corestream e as barras laranjas as médias

de ARIs para cada $MinPts$ das execuções do HASTREAM. Como pode ser visto, as médias ARI do Corestream são superiores às do Hastream em todos as bases de dados e os desvios padrão são semelhantes. A qualidade dos *clusters* no CoreStream usando DBs para sumarização de dados foi melhor do que o agrupamento usando MCs. Com os cálculos de densidade do DB, o CoreStream conseguiu encontrar *clusters* mais internos, ou seja, *clusters* mais densos na hierarquia, devido às densidades na hierarquia estarem mais próximas da densidade entre os objetos. Na hierarquia de *clusters* com MCs há uma distorção da densidade entre os objetos sumarizados, pois os MCs são considerados como pseudo-pontos, selecionando na maioria das partições $MinPts$, os *clusters* do topo da hierarquia.

Nas duas bases de dados sintéticos bidimensionais 28k e 40k (Figura 14), a diferença entre os ARIs médios dos algoritmos é muito mais proeminente. Cada chamada para a fase *Offline* nessas bases de dados foram colocadas diferentes densidades de *cluster* e, à medida que novos dados entram no fluxo, os *clusters* se movem no espaço, formando novos *clusters* ou uma união dos existentes. O HASTREAM nestas bases de dados, a partir de um determinado $MinPts$, encontrou apenas os macro *clusters* e não foi capaz de identificar os *clusters* mais baixos na hierarquia, pois a densidade que conectou os MCs para valores altos de $MinPts$ é baixa. CoreStream encontrou melhor os *clusters* internos e mais densos, já que a hierarquia com DBs representa bem as diferentes densidades. Ao contrário dos MCs, que são considerados pseudo-pontos, os DBs utilizam estatísticas internas dos objetos sumarizados que ajudam nas aproximações nos cálculos de densidade. Em bases de dados reais, os ARIs médios de ambos os algoritmos são próximos, pois em algumas chamadas de agrupamento as densidades dos *clusters* não estão bem distribuídas e não apresentam *clusters* bem separados por regiões menos densas. Mesmo assim, o CoreStream revelou uma melhoria na qualidade dos agrupamentos finais, com médias de ARI melhores que o HASTREAM. Nas bases de dados PK e Hyper o HASTREAM obteve médias de ARIs em torno de 0, ocasionando total distorção das densidades dos dados originais ao gerar as hierarquias de MC. O CoreStream conseguiu melhorar os resultados de agrupamento para estas bases, aumentando a relação de densidade entre os dados sumarizados na hierarquia, e em algumas chamadas da fase *Offline* gerou partições com alta similaridade com partições do HDBSCAN* original.

4.2.2 Análise do Tempo de Execução

Para avaliar o desempenho computacional do CoreStream em relação ao HASTREAM, criamos bases de dados com diversos tamanhos e dimensões para avaliar as variáveis descritas na Tabela 5. Para avaliar cada característica separadamente, os valores padrão foram fixados e apresentados em negrito na Tabela 5. Além disso, também consideramos uma variedade de valores $MinPts_{max}$ usados para construir os grafos. Para avaliar o

tempo médio e o desvio padrão de cada chamada *Offline*, e avaliar o tempo total, utilizamos as bases de dados sintéticos usados na Tabela 5 e algumas bases de dados reais com maior número de instâncias e dimensões: KDDCUP'99, Covertype, e dados do *US Census* (1990)[5] com 68 atributos e 2.458.285 instâncias, mas aqui é usado 600.000 primeiras instâncias da base. As características de cada base de dados e os parâmetros utilizados nos algoritmos estão na Tabela 6. A velocidade do fluxo v é definida como 100 objetos por unidade de tempo. O parâmetro β é definido como 0.75, e o número mínimo de objetos sumarizados nos DBs $mu = 2$. O tempo de execução para obter todos os grafos *CORE-SSG* necessários para extrair as múltiplas hierarquias em cada solicitação *Offline* do CoreStream, e o tempo de execução necessário para obter todos os grafos completos e extrair as múltiplas hierarquias em cada chamada *Offline* do HASTREAM, são apresentados na Figura 15.

Tabela 5 – Características e variáveis das bases de dados para avaliar o tempo de execução e velocidades entre CoreStream e HASTREAM.

Variáveis	**Valores**
$MinPts_{max}$	50, **100**, 150, 200
#objetos	40k, 55k, **70k**, 85k, 100k
#dimensões	2, **4**, 6, 8

[5] <https://archive.ics.uci.edu/dataset/116/us+census+data+1990>

Tabela 6 – Configurações das bases de dados para avaliar o tempo de execução do CoreStram e HASTREAM.

Base de dados	n° **objetos por *timestamps***	$\lambda_{CoreStream}$	$\lambda_{HASTREAM}$	**dimensões**	**tamanho**	**tipo**
40k4d	10000	0.015	0.015	4	40000	sintético
55k4d	10000	0.015	0.015	4	55000	sintético
70k2d	10000	0.015	0.015	2	70000	sintético
70k4d	10000	0.016	0.014	4	70000	sintético
70k6d	10000	0.008	0.008	6	70000	sintético
70k8d	10000	0.0025	0.0025	7	70000	sintético
85k4d	10000	0.015	0.020	4	85000	sintético
100k4d	10000	0.015	0.015	4	100000	sintético
covertype	20000	0.01	0.003	10	581012	base real
kddcup	20000	0.0025	0.0025	41	494021	base real
USC1990	20000	0.002	0.004	68	600000	base real

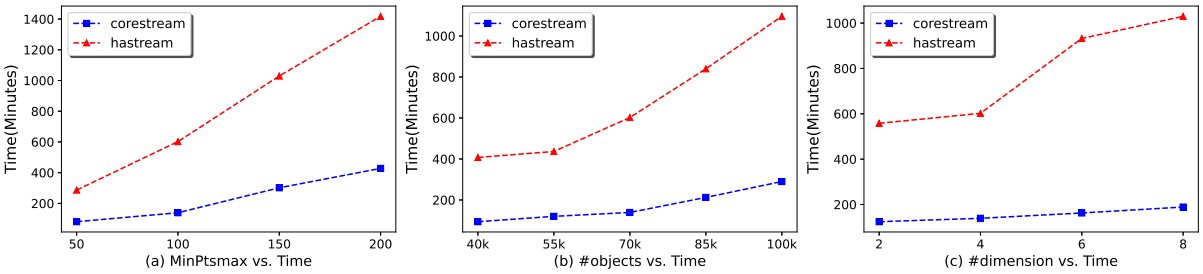


Figura 15 – Tempo de execução para construir todos os grafos e extrair as soluções hierárquicas baseadas em densidade para o CoreStream e HASTREAM para todas as chamadas *Offlines*.

A Figura 15(a) mostra que o parâmetro $MinPts$ possui uma correlação linear com o tempo de processamento necessário para rodar o experimento para ambos os algoritmos, mas o CoreStream apresentou velocidades 3 vezes melhor que o HASTREAM. Observe que a tendência de crescimento do tempo total do HASTREAM é maior que a do CoreStream. A Figura 15(b) mostra que o parâmetro número de objetos tem uma correlação linear com o tempo de processamento necessário para executar o experimento para o CoreStream. HASTREAM mostra uma inicial tendência de ter comportamento quadrático quando o número de instâncias na base de dados aumenta. Em relação ao número de instâncias, a Figura 15(b) mostra velocidades de até 6 vezes. A variação no número de dimensões das bases de dados é mostrada na Figura 15(c). Com velocidades de 6 vezes, o CoreStream tem uma correlação linear com o tempo de processamento necessário para rodar o experimento em relação à dimensão. HASTREAM também apresenta uma correlação linear em relação ao número de dimensões. Assim como no algoritmo *CORE-SSG*, o número de DBs tem uma correlação linear com o tempo para gerar as múltiplas hierarquias. As dimensões afetaram o número de MCs. Em algumas bases de dados, o HASTREAM aumentou consideravelmente o número de MCs ao longo do fluxo, aumentando assim o tempo de execução final. Este comportamento pode ser visto na Figura 15(c) nas bases de dados 70k4d e 70k6d. As duas bases possuem o mesmo número de objetos e em cada *timestamp* o mesmo número de objetos entrando. A diferença nestas bases foi devido ao número de MCs por *timestamp*, onde para a base de 70k4d a média de MCs ficou um pouco abaixo dos 10% de sumarização, e para a base de 70k6d a média de MCs ficou um pouco acima dos 10% de sumarização. Esta diferença de número de MCs impactou diretamente o tempo final de execução do HASTREAM. Os resultados predefinidos apoiam a hipótese deste trabalho: *CORE-SSG* com DBs permite escalabilidade para soluções de aprendizado de máquina baseadas em densidade em cenários de FD.

No HASTREAM, o tempo total para gerar múltiplas hierarquias baseada em densidade em todas as chamadas *Offline* nas bases de dados, apresentou um comportamento

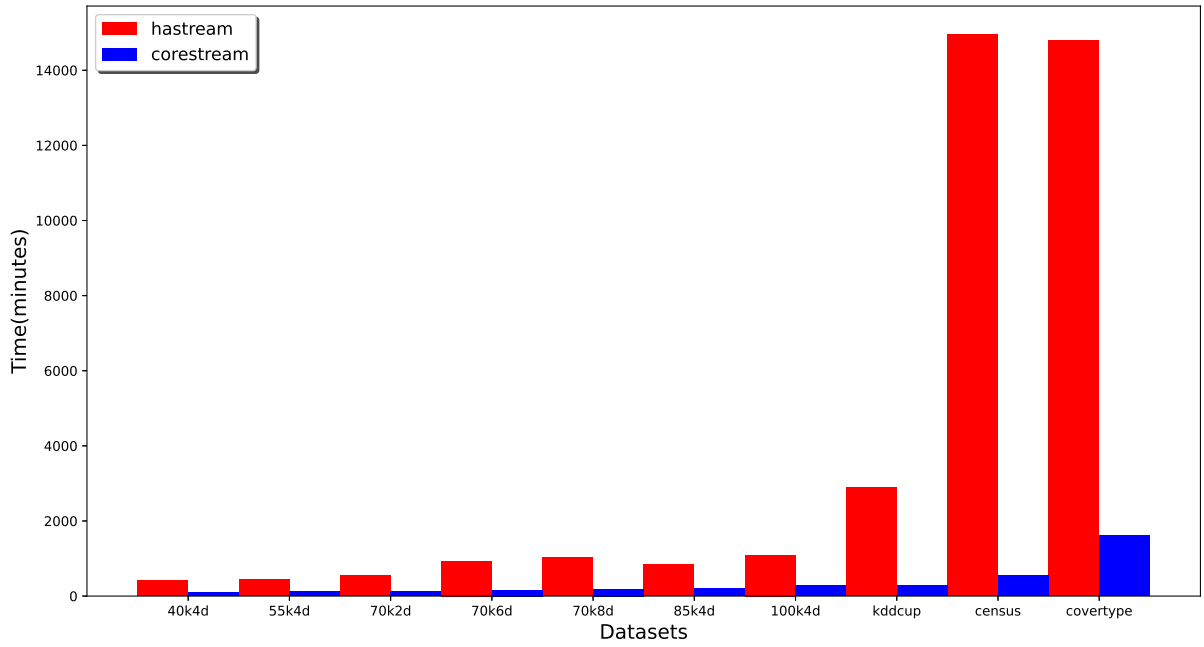


Figura 16 – Total tempo de execução para construir os grafos e extrair as soluções de densidade hierárquica em todas as chamadas *Offline* para Corestream e HASTREAM

quadrático quando é aumentado o número de objetos e o número de dimensões, como pode ser representado na Figura 16. O custo computacional é amortizado ao usar DBs e o grafo *CORE-SSG* para gerar todas as soluções hierárquicas no CoreStream. A Figura 16 mostra velocidades de até 8 vezes. As médias, desvio padrão, *outliers* e distribuições de tempo para executar cada chamada *Offline* do CoreStream e HASTREAM são representados na Figura 17. As amplitudes dos *boxplots* do CoreStream são menores que as do HASTREAM, mostrando que os tempos médios de cada chamada *Offline* no CoreStream são mais estáveis ao longo do stream. O HASTREAM apresenta grandes amplitudes devido à construção dos grafos completos e à variação no número de vértices, pois o número de MCs varia constantemente. Vemos na Figura 17 que a distribuição dos tempos médios do HASTREAM é maior que a do CoreStream, e o HASTREAM apresenta tempos altos que são discrepantes em relação à mediana. Os *outliers* do HASTREAM revelam que em alguns cenários onde o número de objetos está aumentando, é impraticável o cálculo de múltiplas hierarquias em algumas chamadas *Offlines*, pois o tempo de execução será maior do que o processamento de novos dados que entram no fluxo. Ao reduzir o custo computacional para cálculo de múltiplas hierarquias em cada chamada *Offline* pelo CoreStream, nossa proposta obtem resultados eficientes e escaláveis mesmo em bases de dados com grande número de objetos e com alta dimensionalidade.

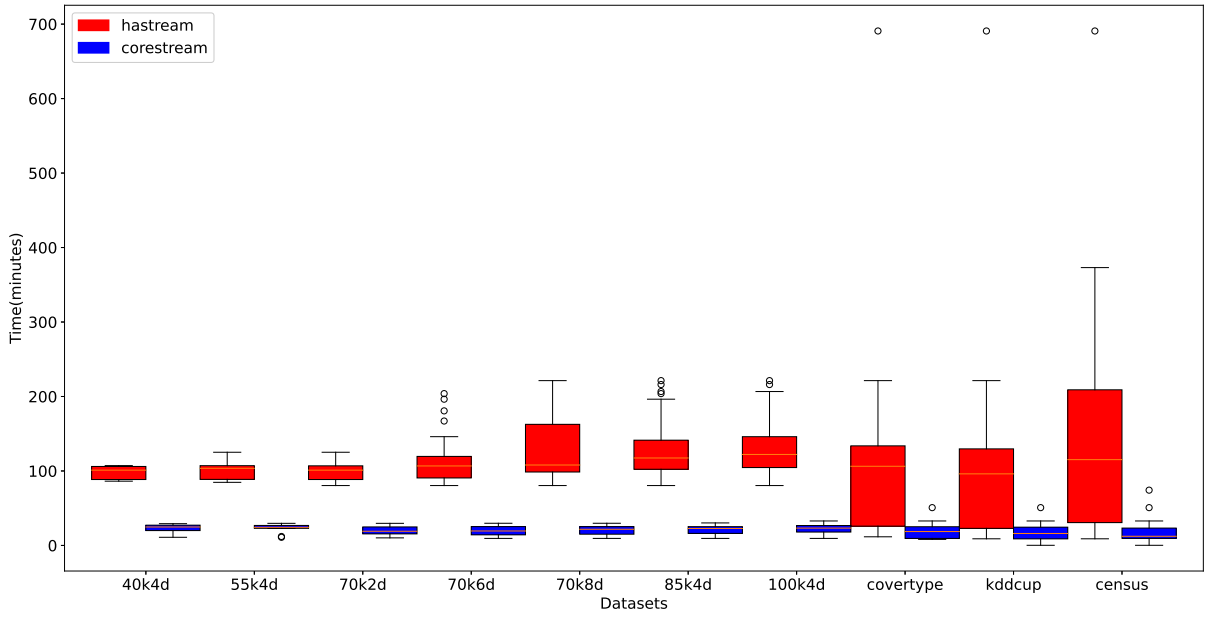


Figura 17 – Média e desvio padrão do tempo para calcular as múltiplas hierarquias baseadas em densidade em cada chamada *Offline* para Corestream e HASTREAM

Capítulo 5

Conclusão

O objetivo principal da presente pesquisa é permitir o uso de *CORE-SG* em grandes conjuntos de dados. Foi alcançado quando mostramos que um único *CORE-SSG* permite a extração precisa de múltiplas soluções hierárquicas baseadas em densidade para parâmetros de densidades diferentes a um custo computacional sub-quadrático. Isso foi demonstrado teoricamente, com a comprovação de que *CORE-SSG* contém as principais características do *CORE-SG*, e empiricamente através de experimentos. Além disso, os experimentos também apoiaram a importância da análise de múltiplos resultados baseados em densidade, pois as bases de dados podem ter estruturas de densidade aninhadas com características diferentes, cada estrutura com sua importância relativa, o que justifica a aplicação de *CORE-SG* em um pleura de tarefas de aprendizado de máquina (Gertrudes et al., 2019), incluindo agrupamento.

Os experimentos mostraram que os *clusters* mais proeminentes (de acordo com o *framework* FOSC) presentes na versão original do *CORE-SG* também estão presentes na nossa solução *CORE-SSG*, obtendo uma redução significativa de custos computacionais. A desvantagem da nossa abordagem é a perda de pequenas estruturas baseadas em densidade sumarizadas dentro dos DBs, que raramente eram consideradas proeminentes pelo *framework* FOSC, pois conectam uma fração muito pequena das bases de dados. Assim, nossa solução é indicada para grandes volumes de dados. Mesmo assim, se estas pequenas estruturas são realmente importantes para a aplicação, recomendamos o uso de *CORE-SG* (ou *CORE-SSG*) sobre os dados sumarizados em cada DB separadamente, seguido da combinação dos grafos resultantes.

Uma nova abordagem chamada CoreStream foi apresentada como uma extensão do

método *CORE-SSG* para agrupamento em FD contínuos. CoreStream permite a extração de múltiplas soluções hierárquicas baseadas em densidade para diferentes valores de $MinPts$ e em diversas requisições da fase *Offline* de agrupamento a um custo computacional sub-quadrático. Os DBs foram adaptados para manter o modelo na fase *Online* considerando o tempo em que os objetos entram no fluxo. Os *clusters* extraídos das múltiplas hierarquias baseadas em densidade do CoreStream tiveram uma similaridade maior com os *clusters* gerados pelo HDBSCAN* em objetos do que as partições geradas do HASTREAM. O CoreStream também forneceu hierarquias mais detalhadas e foi capaz de identificar *clusters* em níveis mais baixos da hierarquia do que o HASTREAM. O HASTREAM, por outro lado, provou ser computacionalmente caro ao extrair múltiplas hierarquias baseadas em densidades variando o parâmetro de densidade $MinPts$, e as principais características dos dados no *cluster* foram perdidas ao sumarizar os dados com MCs.

Ao observar o potencial do método do grafo sumarizado *CORE-SG* tanto para dados estáticos como em FD, como trabalho futuro, uma investigação mais acurada sobre a manutenção dos DBs, adaptação das estatísticas ao entrar novos dados e adaptação dos cálculos de densidade podem ser relevantes para aumentar a qualidade das múltiplas soluções hierárquicas baseadas em densidade.

Divulgação do Trabalho

O artigo Batista et al. (2023) publicado na 12^{th} Brazilian Conference on Intelligent Systems (BRACIS) foi um dos resultados deste trabalho. Concorremos ao melhor artigo da conferência e participamos das apresentações na sessão especial dos melhores artigos.

Referências

aaaa.

Charu C. Aggarwal, S. Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, jun 1999. ISSN 0163-5808.

H. B. Barlow. Unsupervised Learning. *Neural Computation*, 1(3):295–311, 09 1989. ISSN 0899-7667.

Natanael F. D. Batista, Bruno Leonel Nunes, and Murilo Coelho Naldi. Efficient density-based models for multiple machine learning solutions over large datasets. In *Brazilian Conference on Intelligent Systems*, pages 48–62. Springer, 2023.

Desamparados Blazquez and Josep Domenech. Big data sources and methods for social and economic analyses. *Technological Forecasting and Social Change*, 130:99–113, 2018. ISSN 0040-1625.

Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Fast hierarchical clustering based on compressed data and optics. In *Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD 2000 Lyon, France, September 13–16, 2000 Proceedings 4*, pages 232–242. Springer, 2000.

Markus M Breunig, Hans-Peter Kriegel, Peer Kröger, and Jörg Sander. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 79–90, 2001.

Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery*, 27:344–371, 2013.

Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1), jul 2015. ISSN 1556-4681.

Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

P. Cheema, M. Makki Alamdari, K. C. Chang, C. W. Kim, and M. Sugiyama. A drive-by bridge inspection framework using non-parametric clusters over projected data manifolds. *Mechanical Systems and Signal Processing*, 180:109401, 2022. ISSN 0888-3270.

Ina Djonlagic, Sara Mariani, Annette L Fitzpatrick, Veerle MGTH Van Der Klei, Dayna A Johnson, Alexis C Wood, Teresa Seeman, Ha T Nguyen, Michael J Prerau, José A Luchsinger, et al. Macro and micro sleep architecture and cognitive performance in older adults. *Nature human behaviour*, 5(1):123–145, 2021.

Joelson Antônio dos Santos, Talat Iqbal Syed, Murilo C Naldi, Ricardo JGB Campello, and Joerg Sander. Hierarchical density-based clustering using mapreduce. *IEEE Transactions on Big Data*, 7(1):102–114, 2019.

Joao Gama. *Knowledge discovery from data streams*, 2010.

Jadson Castro Gertrudes, Arthur Zimek, Jörg Sander, and Ricardo J. G. B. Campello. A unified view of density-based methods for semi-supervised clustering and classification. *Data Min. Knowl. Discov.*, 33(6):1894–1952, 2019.

John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Marwan Hassani, Pascal Spaus, Mohamed Medhat Gaber, and Thomas Seidl. Density-based projected clustering of data streams. In *International Conference on Scalable Uncertainty Management*, pages 311–324. Springer, 2012.

Marwan Hassani, Philipp Kranen, Rajveer Saini, and Thomas Seidl. Subspace anytime stream clustering. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, pages 1–4, 2014a.

Marwan Hassani, Pascal Spaus, and Thomas Seidl. Adaptive multiple-resolution stream clustering. In *Machine Learning and Data Mining in Pattern Recognition: 10th International Conference, MLDM 2014, St. Petersburg, Russia, July 21-24, 2014. Proceedings 10*, pages 134–148. Springer, 2014b.

Marwan Hassani, Pascal Spaus, Alfredo Cuzzocrea, and Thomas Seidl. I-hastream: density-based hierarchical clustering of big data streams and its application to big graph analytics tools. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 656–665. IEEE, 2016.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2: 193–218, 1985.

David Johnson, Caiming Xiong, Jing Gao, and Jason Corso. Comprehensive cross-hierarchy cluster agreement evaluation. In , 01 2013.

Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3): 231–240, 2011.

Bing Liu, Yuliang Shi, Zhihui Wang, Wei Wang, and Baile Shi. Dynamic incremental data summarization for hierarchical clustering. In *Advances in Web-Age Information Management: 7th International Conference, WAIM 2006, Hong Kong, China, June 17-19, 2006. Proceedings 7*, pages 410–421. Springer, 2006.

Leland McInnes, John Healy, and Steve Astel. How hdbscan works. Disponível em: <https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html>. Acesso em: 06 de maio de 2024, 2016.

Luis A. Miccio and Gustavo A. Schwartz. Mapping chemical structure–glass transition temperature relationship through artificial intelligence. *Macromolecules*, 54(4):1811–1817, 2021.

Darlan C. Minussi, Michael D. Nicholson, Hanghui Ye, Alexander Davis, Kaile Wang, Toby Baker, Maxime Tarabichi, Emi Sei, Haowei Du, Mashiat Rabbani, et al. Breast tumours maintain a reservoir of subclonal diversity during expansion. *Nature*, 592 (7853):302–308, 2021.

Brian Murray and Lokukaluge Prasad Perera. An ais-based deep learning framework for regional ship behavior prediction. *Reliability Engineering & System Safety*, 215:107819, 2021. ISSN 0951-8320.

Samer Nassar, Jörg Sander, and Corrine Cheng. Incremental and effective data summarization for dynamic hierarchical clustering. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 467–478, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138598.

Antonio Cavalcante Araujo Neto, Murilo Coelho Naldi, Ricardo J. G. B. Campello, and Jörg Sander. Core-sg: Efficient computation of multiple msts for density-based methods. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 951–964, 2022.

Thomas M. Norman, Max A. Horlbeck, Joseph M. Replogle, Alex Y. Ge, Albert Xu, Marco Jost, Luke A. Gilbert, and Jonathan S. Weissman. Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793, 2019.

Tom Obry, Louise Travé-Massuyès, and Audine Subias. Computer-aided diagnosis via hierarchical density based clustering. In *29th International Workshop on Principles of Diagnosis (DX 2018)*, page 8p, 2018.

William Savoie, Thomas A. Berrueta, Zachary Jackson, Ana Pervan, Ross Warkentin, Shengkai Li, Todd D. Murphey, Kurt Wiesenfeld, and Daniel I. Goldman. A robot made of robots: Emergent transport and control of a smarticle ensemble. *Science Robotics*, 4(34):eaax4316, 2019.

Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André CPLF de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31, 2013.

Lucas Vendramin, Ricardo JGB Campello, and Eduardo R Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical analysis and data mining: the ASA data science journal*, 3(4):209–235, 2010.

Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. *ACM Transactions on Knowledge discovery from Data (TKDD)*, 3(3):1–28, 2009.

Btissam Zerhari, Ayoub Ait Lahcen, and Salma Mouline. Big data clustering: Algorithms and challenges. In *"International Conference on Big Data, Cloud and Applications"*, 05 2015.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, jun 1996. ISSN 0163-5808.

Yiqun Zhang, Yiu-ming Cheung, and Yang Liu. Quality preserved data summarization for fast hierarchical clustering. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4139–4146, 2016.

Jianjun Zhou and Jörg Sander. Data bubbles for non-vector data: Speeding-up hierarchical clustering in arbitrary metric spaces. In Johann-Christoph Freytag, Peter Lockemann, Serge Abiteboul, Michael Carey, Patricia Selinger, and Andreas Heuer, editors, *Proceedings 2003 VLDB Conference*, pages 452–463, San Francisco, 2003. Morgan Kaufmann. ISBN 978-0-12-722442-8.

Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering: a review. *Artificial Intelligence Review*, 54(2):1201–1236, 2021.