UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Algoritmos rápidos para estimativas de densidade hierárquicas e suas aplicações em mineração de dados

Joelson Antônio dos Santos

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Joelson Antônio dos Santos

Algoritmos rápidos para estimativas de densidade hierárquicas e suas aplicações em mineração de dados

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Rircado José Gabrielli Barreto Campello

USP – São Carlos
Julho de 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

A64a Antônio dos Santos, Joelson
 Algoritmos rápidos para estimativas de densidade
 hierárquicas e suas aplicações em mineração de dados
 / Joelson Antônio dos Santos; orientador Ricardo
 José Gabrielli Barreto Campello. -- São Carlos,
 2018.
 185 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
 em Ciências de Computação e Matemática
 Computacional) -- Instituto de Ciências Matemáticas
 e de Computação, Universidade de São Paulo, 2018.

 1. Agrupamento de dados. 2. MapReduce. 3.
 Sumarização de dados. I. José Gabrielli Barreto
 Campello, Ricardo, orient. II. Título.

Joelson Antônio dos Santos

**Fast algorithms for hierarchical density estimates and its
applications in data mining**

Master dissertation submitted to the Institute of
Mathematics and Computer Sciences – ICMC-USP,
in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Rircado José Gabrielli
Barreto Campello

USP – São Carlos
July 2018

*Este trabalho é dedicado aos meus familiares e amigos.
Em especial, ao meu padrinho Nivaldo (in memoriam), minha tia Maria Helena (in memoriam)
e minha avó Juversina Ana de Jesus (in memoriam).*

AGRADECIMENTOS

Agradeço ao meu orientador, professor Dr. Ricardo, pela oportunidade e presteza para com o desenvolvimento da pesquisa. Também gostaria de agradecer aos colaboradores, professor Dr. Murilo Coelho Naldi e professor Dr. Jörg Sander, pelo apoio constante e paciência durante o desenvolvimento deste trabalho.

Gostaria de agradecer à minha mãe Ivani, meu pai Jovino e minha irmã Luciana por sempre me apoiarem nas minhas decisões e me ajudarem em momentos em que mais precisei, antes e durante o curso de mestrado.

Agradeço também à minha namorada Lohany pelo carinho, paciência e compreensão em momentos importantes de minha jornada acadêmica.

Gostaria de agradecer a todos os amigos e colegas que participaram da minha vida acadêmica durante esses quase três anos de pesquisa. Em especial, gostaria de agradecer aos amigos, Misael, Jonathan, Francisco, Filomen, Evinton, Lucas e Weslei pelos momentos de descontração, trocas de conhecimentos, conversas, risadas e lágrimas às vezes.

Agradeço pelo apoio financeiro da CAPES e pelo suporte tecnológico fornecido pela FAPEMIG durante minha pesquisa de campo na Universidade Federal de Viçosa - campus de Rio Paranaíba - MG.

Agradeço também ao Instituto de Ciências Matemáticas e de Computação (ICMC) pela oportunidade e espaço para que eu pudesse aprender e fazer ciência.

RESUMO

SANTOS, J. A. DOS. **Algoritmos rápidos para estimativas de densidade hierárquicas e suas aplicações em mineração de dados**. 2018. 185 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

O agrupamento de dados (ou do inglês *Clustering*) é uma tarefa não supervisionada capaz de descrever objetos em grupos (ou *clusters*), de maneira que objetos de um mesmo grupo sejam mais semelhantes entre si do que objetos de grupos distintos. As técnicas de agrupamento de dados são divididas em duas principais categorias: particionais e hierárquicas. As técnicas particionais dividem um conjunto de dados em um determinado número de grupos distintos, enquanto as técnicas hierárquicas fornecem uma sequência aninhada de agrupamentos particionais separados por diferentes níveis de granularidade. Adicionalmente, o agrupamento hierárquico de dados baseado em densidade é um paradigma particular de agrupamento que detecta grupos com diferentes concentrações ou densidades de objetos. Uma das técnicas mais populares desse paradigma é conhecida como *HDBSCAN**. Além de prover hierarquias, *HDBSCAN** é um *framework* que fornece detecção de *outliers*, agrupamento semi-supervisionado de dados e visualização dos resultados. No entanto, a maioria das técnicas hierárquicas, incluindo o *HDBSCAN**, possui uma alta complexidade computacional. Fato que as tornam proibitivas para a análise de grandes conjuntos de dados. No presente trabalho de mestrado, foram propostas duas variações aproximadas de *HDBSCAN** computacionalmente mais escaláveis para o agrupamento de grandes quantidades de dados. A primeira variação de *HDBSCAN** segue o conceito de computação paralela e distribuída, conhecido como *MapReduce*. Já a segunda, segue o contexto de computação paralela utilizando memória compartilhada. Ambas as variações são baseadas em um conceito de divisão eficiente de dados, conhecido como *Recursive Sampling*, que permite o processamento paralelo desses dados. De maneira similar ao *HDBSCAN**, as variações propostas também são capazes de fornecer uma completa análise não supervisionada de padrões em dados, incluindo a detecção de *outliers*. Experimentos foram realizados para avaliar a qualidade das variações propostas neste trabalho, especificamente, a variação baseada em *MapReduce* foi comparada com uma versão paralela e exata de *HDBSCAN** conhecida como *Random Blocks*. Já a versão paralela em ambiente de memória compartilhada foi comparada com o estado da arte (*HDBSCAN**). Em termos de qualidade de agrupamento e detecção de *outliers*, tanto a variação baseada em *MapReduce* quanto a baseada em memória compartilhada mostraram resultados próximos à versão paralela exata de *HDBSCAN** e ao estado da arte, respectivamente. Já em termos de tempo computacional, as variações propostas mostraram maior escalabilidade e rapidez para o processamento de grandes quantidades de dados do que as versões comparadas.

Palavras-chave: Agrupamento de dados, Sumarização de dados, *MapReduce*.

ABSTRACT

SANTOS, J. A. DOS. **Fast algorithms for hierarchical density estimates and its applications in data mining**. 2018. 185 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Clustering is an unsupervised learning task able to describe a set of objects in clusters, so that objects of a same cluster are more similar than objects of other clusters. Clustering techniques are divided in two main categories: partitional and hierarchical. The partitional techniques divide a dataset into a number of distinct clusters, while hierarchical techniques provide a nested sequence of partitional clusters separated by different levels of granularity. Furthermore, hierarchical density-based clustering is a particular clustering paradigm that detects clusters with different concentrations or densities of objects. One of the most popular techniques of this paradigm is known as *HDBSCAN**. In addition to providing hierarchies, *HDBSCAN** is a framework that provides outliers detection, semi-supervised clustering and visualization of results. However, most hierarchical techniques, including *HDBSCAN**, have a high complexity computational. This fact makes them prohibitive for the analysis of large datasets. In this work have been proposed two approximate variations of *HDBSCAN** computationally more scalable for clustering large amounts of data. The first variation follows the concept of parallel and distributed computing, known as *MapReduce*. The second one follows the context of parallel computing using shared memory. Both variations are based on a concept of efficient data division, known as *Recursive Sampling*, which allows parallel processing of this data. In a manner similar to *HDBSCAN**, the proposed variations are also capable of providing complete unsupervised patterns analysis in data, including outliers detection. Experiments have been carried out to evaluate the quality of the variations proposed in this work, specifically, the variation based on MapReduce have been compared to a parallel and exact version of *HDBSCAN**, known as Random Blocks. Already the version parallel in shared memory environment have been compared to the state of the art (*HDBSCAN**). In terms of clustering quality and outliers detection, the variation based on *MapReduce* and other based on shared memory showed results close to the exact parallel version of *HDBSCAN** and the state of the art, respectively. In terms of computational time, the proposed variations showed greater scalability and speed for processing large amounts of data than the compared versions.

Keywords: Clustering, Data Summarization, MapReduce.

LISTA DE ILUSTRAÇÕES

Figura 1 – Processos necessários para análise de KDD envolvendo agrupamento de dados (adaptado de (XU; WUNSCH, 2009)).	36
Figura 2 – Tipos de atributos. (Adaptado de (CASTRO; FERRARI, 2016)).	37
Figura 3 – Exemplo de agrupamento hierárquico (dendrograma) de dados baseado nas abordagens: Aglomerativa e Divisiva. (Adaptado de (XU; WUNSCH, 2005)).	43
Figura 4 – Ilustração de grupos “aninhados” baseados em densidade (Adaptado de (ANKERST <i>et al.</i> , 1999a)).	50
Figura 5 – <i>Framework HDBSCAN*</i> (Adaptado de (CAMPELLO <i>et al.</i> , 2015)).	56
Figura 6 – Ilustração de função de densidade, grupos e excesso de massa (Adaptado de (CAMPELLO <i>et al.</i> , 2015)).	66
Figura 7 – Ilustração de solução ótima a partir de uma determinada árvore de grupos (Adaptado de (CAMPELLO <i>et al.</i> , 2015)).	68
Figura 8 – Arquitetura de <i>DiSC</i> em <i>MapReduce</i> (Adaptado de (JIN <i>et al.</i> , 2013a)). . . .	74
Figura 9 – Similaridade entre <i>OPTICS</i> e uma <i>MST</i> construída pelo algoritmo de <i>Prim</i> . (a) O conjunto de dados. (b) A ϵ -vizinhança dos objetos <i>a</i> , <i>b</i> e <i>c</i> sob a perspectiva do algoritmo <i>OPTICS</i> . (c) Vértices adjacentes (vizinhos) dos objetos <i>a</i> , <i>b</i> e <i>c</i> sob a perspectiva do algoritmo de <i>Prim</i> (Adaptado de (PATWARY <i>et al.</i> , 2013)).	75
Figura 10 – Etapas do Algoritmo <i>Random Blocks</i> (Adaptado de (SYED, 2014)).	76
Figura 11 – Exemplo de conjunto de dados composto por grupos com arbitrárias formas e constituído por ruídos.	81
Figura 12 – Exemplo de conjunto de amostras de dados baseando-se no conjunto de dados apresentado na Figura 11.	81
Figura 13 – Exemplo de <i>MST</i> (“self-edges” desconsideradas) criada por <i>HDBSCAN*</i> baseando-se na amostra de dados apresentada na Figura 12.	82
Figura 14 – Grupos mais significativos de amostras de dados baseando-se nas amostras apresentadas na Figura 12.	84
Figura 15 – Inclusão de ruídos aos seus respectivos grupos mais próximos	85
Figura 16 – Grupos mais significativos sobre amostras aleatórias.	87
Figura 17 – Indução de partição sobre o conjunto completo de dados.	87
Figura 18 – (a) Conjunto completo de dados. (b) Boa amostra representativa dos dados. (c) Amostra não representativa dos dados.	88

Figura 19 – (a) Representa dois <i>Data Bubbles</i> B e C que não são sobrepostos (b) Indica dois <i>Data Bubbles</i> B e C que são sobrepostos.	90
Figura 20 – <i>Data Bubble B</i> sumariza pelo menos m_{pts} objetos. Dessa forma, a distância <i>core</i> é representada por $nnDist(m_{pts}, B)$	93
Figura 21 – <i>Data Bubble B</i> não sumariza pelo menos m_{pts} objetos. Neste caso, a área demarcada por P indica que todos os <i>Data Bubbles</i> A , D e E são mais próximos de <i>B</i> do que <i>C</i> , juntos sumarizam pelo menos m_{pts} objetos.	93
Figura 22 – Exemplo de execução de <i>Recursive Sampling</i> em ambiente de computação paralela.	99
Figura 23 – Exemplo de contador de palavras em <i>MapReduce</i>	103
Figura 24 – <i>Framework Apache Hadoop</i>	104
Figura 25 – Estrutura hierárquica de <i>MR-HDBSCAN*</i>	107
Figura 26 – Primeira etapa de <i>MR-HDBSCAN*</i>	109
Figura 27 – Extração de <i>Data Bubbles</i> baseando-se no conceito de <i>MapReduce</i> do algoritmo <i>MR-HDBSCAN*</i>	111
Figura 28 – Exemplo de combinação de <i>MSTs</i> locais e arestas intergrupos de <i>MR-HDBSCAN*</i> para a obtenção da <i>MST</i> global estendida MST_{est} . (a) <i>MSTs</i> locais e arestas intergrupos espalhadas em 4 nós distintos. (b) Combinação em <i>MapReduce</i> de arestas (<i>MSTs</i> locais e arestas intergrupos) em uma única <i>MST</i> global estendida.	115
Figura 29 – Diferenças entre a construção de hierarquias de <i>MR-HDBSCAN*</i> de maneira aglomerativa e divisiva.	117
Figura 30 – Execução de <i>CC-MR</i> : (a) Grafo original; (b) Iteração considerando o vértice 1 como v_{origem} (<i>estado de máximo local</i>); (c) Iteração em que o vértice 2 é v_{origem} (<i>estado de fusão</i>); (d) Iteração considerando o vértice 3 como v_{origem} (<i>estado de fusão otimizado</i>).	121
Figura 31 – Exemplo de execução de <i>CC-MR</i> a partir de uma <i>MST</i> global estendida hipotética.	126
Figura 32 – Exemplo de construção do primeiro nível de uma hierarquia de <i>MR-HDBSCAN*</i>	128
Figura 33 – Exemplo das possíveis situações que podem ocorrer durante a construção de níveis hierárquicos de <i>MR-HDBSCAN*</i>	133
Figura 34 – Esquema de experimentos utilizando memória distribuída e compartilhada.	140
Figura 35 – Esquema de condução de experimento utilizando os algoritmos <i>Random Blocks</i> , $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ sobre um determinado conjunto de dados.	146
Figura 36 – Esquema de condução de experimento utilizando os algoritmos <i>Random Blocks</i> , $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ sobre um determinado conjunto de dados para a avaliação de detecção de <i>outliers</i>	154
Figura 37 – Conjunto de dados - SensorLess	158

Figura 38 – Conjunto de dados - Credit	158
Figura 39 – Conjunto de dados - Magic	158
Figura 40 – Conjunto de dados - Shuttle	158
Figura 41 – Conjunto de dados - Skin	159
Figura 42 – Conjunto de dados - YearPrediction	159
Figura 43 – Esquema de condução de experimento utilizando os algoritmos <i>HDBSCAN*</i> , <i>RS_{bubbles}</i> e <i>RS_{sampling}</i> sobre um determinado conjunto de dados.	163
Figura 44 – Esquema de condução de experimento utilizando os algoritmos <i>HDBSCAN*</i> , <i>RS_{bubbles}</i> e <i>RS_{sampling}</i> sobre um determinado conjunto de dados para a avalia- ção de detecção de <i>outliers</i> pelo método <i>GLOSH</i>	167
Figura 45 – Conjunto de dados - SensorLess	169
Figura 46 – Conjunto de dados - Credit	169
Figura 47 – Conjunto de dados - Magic	170
Figura 48 – Conjunto de dados - Shuttle	170
Figura 49 – Conjunto de dados - Skin	170
Figura 50 – Conjunto de dados - YearPrediction	170

LISTA DE ALGORITMOS

Algoritmo 1 – <i>HDBSCAN*</i> Passo Principal	61
Algoritmo 2 – <i>HDBSCAN*</i> (referente à Linha 8 do Algoritmo 1) utilizando o parâmetro $m_{clSize} \geq 1$	63
Algoritmo 3 – Solução do problema de otimização apresentado pelas Equações 3.5 e 3.6.	69
Algoritmo 4 – Principal passo para calcular os <i>scores</i> de <i>GLOSH</i>	71
Algoritmo 5 – Algoritmo <i>Recursive Sampling</i>	94
Algoritmo 6 – Calcula <i>MST</i> baseada na distância de acessibilidade mútua de <i>HDBSCAN*</i>	95
Algoritmo 7 – Extração de <i>Data Bubbles</i>	96
Algoritmo 8 – <i>HDBSCAN*</i> sobre <i>Data Bubbles</i>	97
Algoritmo 9 – Recupera arestas intergrupos E_{inter}	97
Algoritmo 10 – <i>MapPartitionsToPair</i> : Modelo Local de <i>HDBSCAN*</i>	110
Algoritmo 11 – Combinação local e global	112
Algoritmo 12 – <i>ReduceByKey</i> : Modelo Local de <i>HDBSCAN*</i> a partir de <i>Data Bubbles</i>	113
Algoritmo 13 – <i>Mapper</i> : Indução de Partição	114
Algoritmo 14 – <i>ReduceByKey</i> : Combinação de <i>MSTs</i> locais e arestas intergrupos.	116
Algoritmo 15 – <i>ReduceByKey</i> : Algoritmo CC-MR	122
Algoritmo 16 – Terceira etapa de MR- <i>HDBSCAN*</i> - <i>Mapper</i> : atribuição de objetos à grupos ou determinação de ruído.	124
Algoritmo 17 – <i>ReduceByKey</i> : Construção de Hierarquias de MR- <i>HDBSCAN*</i>	125
Algoritmo 18 – Algoritmo <i>union-find</i> sequencial.	130
Algoritmo 19 – <i>ReduceByKey</i> : Método Aglomerativo de Construção de hierarquia de <i>MR-HDBSCAN*</i>	135
Algoritmo 20 – <i>Reducer</i> : Grupos Significativos de MR- <i>HDBSCAN*</i>	136

LISTA DE TABELAS

Tabela 1	– Características dos conjuntos de dados artificiais.	140
Tabela 2	– Características dos conjuntos de dados reais.	140
Tabela 3	– Conjunto de parâmetros utilizados pelos algoritmos <i>Random Blocks</i> , <i>MR-HDBSCAN*</i> _{bubbles} e <i>MR-HDBSCAN*</i> _{sampling}	143
Tabela 4	– Parâmetro m_{clSize} utilizado pelo método divisivo de construção de hierarquias baseadas em densidade.	145
Tabela 5	– Agrupamentos particionais utilizando <i>ARI</i>	148
Tabela 6	– Testes t pareado em relação aos resultados de agrupamentos particionais feitos pelos algoritmos <i>MR-HDBSCAN*</i> _{bubbles} e <i>MR-HDBSCAN*</i> _{sampling} em ambiente distribuído.	150
Tabela 7	– Desempenho computacional (tempo em minutos de execução) em ambiente paralelo com memória distribuída.	150
Tabela 8	– Teste t pareado em relação aos tempos de execução para construção de <i>MSTs</i> globais estendidas pelos algoritmos <i>MR-HDBSCAN*</i> _{bubbles} e <i>MR-HDBSCAN*</i> _{sampling}	151
Tabela 9	– Avaliação de desempenho computacional de partições obtidas a partir de hierarquias de grupos baseados em densidade construídas por método <i>divisivo</i> e método <i>aglomerativo</i>	153
Tabela 10	– Conjunto de parâmetros utilizados pelos algoritmos <i>MR-HDBSCAN*</i> _{bubbles} e <i>MR-HDBSCAN*</i> _{sampling}	156
Tabela 11	– Conjunto de parâmetros utilizados pelo algoritmo <i>Random Blocks</i>	156
Tabela 12	– Conjunto de parâmetros utilizados pelos algoritmos <i>HDBSCAN*</i> , <i>RS</i> _{bubbles} e <i>RS</i> _{sampling}	162
Tabela 13	– Avaliação de qualidade de agrupamentos particionais utilizando <i>ARI</i>	164
Tabela 14	– Testes t pareado em relação aos resultados de agrupamentos particionais feitos pelos algoritmos <i>RS</i> _{bubbles} e <i>RS</i> _{sampling} em ambiente de memória compartilhada.	165
Tabela 15	– Desempenho computacional em ambiente paralelo com memória compartilhada.	166
Tabela 16	– Teste t pareado em relação aos tempos de execução para construção de hierarquias de grupos baseados em densidade <i>RS</i> _{bubbles} e <i>RS</i> _{sampling}	167
Tabela 17	– Conjunto de parâmetros utilizados pelos algoritmos <i>RS</i> _{bubbles} e <i>RS</i> _{sampling}	168

LISTA DE ABREVIATURAS E SIGLAS

método_{aglomerativo} método aglomerativo

método_{divisivo} método divisivo

CC componentes conectados

CF *Clustering Feature*

MST_{est} *MST* extendida

MST_{local} árvores geradoras mínimas

BIRCH *Balanced Iterative Reducing and Clustering using Hierarchies*

DBSCAN *Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*

DIANA *DIvisive ANAlysis*

GLOSH *Global-Local Outlier Scores from Hierarchies*

HDFS *Hadoop Distributed File System*

KDD *Knowledge Discovery in Databases*

MST *Minimum Spanning Tree*

OPTICS *Ordering Points To Identify the Clustering Structure*

PCA *Principal Components Analysis*

POPTICS Parallel OPTICS

UPGMA *Unweighted Pair Group Method Using Arithmetic Averages*

DiSC *Distributed Single Linkage Hierarchical Clustering*

IC Intervalo de Confiança

MST *Minimum Spanning Trees*

RDDs *Resilient Distributed Datasets*

LISTA DE SÍMBOLOS

- $\mathbf{X}$ — Conjunto de dados multidimensionais.
- C_i — i -ésimo grupo.
- $\mathbf{X}_i$ — i -ésimo subconjunto de dados multidimensionais.
- k — Número de grupos.
- $\mathbf{x}_j$ — j -ésimo objeto multidimensional.
- $d(\mathbf{x}_i, \mathbf{x}_j)$ — Distância entre os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$.
- B — fator de ramificação
- L — fator folha
- T — limiar correspondente ao raio
- H — hierarquia de *HDBSCAN**.
- $\mathbf{S}$ — Conjunto de objetos multidimensionais aleatoriamente amostrados.
- $\mathbf{O}$ — Conjunto de objetos considerados como ruído.
- $P_{\mathbf{S}}$ — Partição extraída por *HDBSCAN** a partir de amostras aleatórias em $\mathbf{S}$.
- $\mathbf{U}$ — Objetos não amostrados.
- LS — Soma linear.
- SS — Soma quadrática.
- n — número de objeto.
- E_{inter} — arestas intergrupos
- $d_{core}(\mathbf{x}_p)$ — distância core do objeto $\mathbf{x}_p$.
- $d_{mreach}(\mathbf{x}_p, \mathbf{x}_q)$ — distância de acessibilidade mútua entre os objetos $\mathbf{x}_p$ e $\mathbf{x}_q$.
- m_{pts} — Número mínimo de objetos em uma determinada vizinhança.
- m_{clSize} — Número mínimo de objetos de qualquer grupo válido.
- p — Número de unidades de processamento independentes.
- m_i — Número de amostras aleatórias do conjunto $\mathbf{S}_i \cup \mathbf{X}_i$.

SUMÁRIO

1	INTRODUÇÃO	29
1.1	Motivação	29
1.2	Metodologia e Objetivos Gerais	30
1.3	Contribuições	32
1.4	Organização do Trabalho	33
2	TÉCNICAS DE AGRUPAMENTO DE DADOS	35
2.1	Pré-processamento de Dados Estruturados	36
2.2	Agrupamento de Dados	38
2.2.1	<i>Medidas de Proximidade</i>	39
2.2.1.1	<i>Medidas de Proximidade - Variáveis Contínuas</i>	40
2.2.1.2	<i>Medidas de Similaridade - Variáveis Binárias</i>	41
2.2.2	<i>Algoritmos Hierárquicos</i>	42
2.2.2.1	<i>Algoritmos Aglomerativos</i>	42
2.2.2.2	<i>Single-linkage</i>	43
2.2.2.3	<i>Complete-linkage</i>	44
2.2.2.4	<i>Group Average-linkage</i>	44
2.2.2.5	<i>Método de Variância Mínima</i>	44
2.2.2.6	<i>BIRCH</i>	45
2.2.3	<i>Algoritmos Divisivos</i>	46
2.2.3.1	<i>DIANA</i>	46
2.2.4	<i>Algoritmos de Agrupamento Baseados em Densidade</i>	47
2.2.4.1	<i>DBSCAN</i>	47
2.2.4.2	<i>OPTICS</i>	49
2.2.5	<i>Validação de Agrupamento de Dados</i>	51
2.2.5.1	<i>Índice Rand</i>	52
2.2.5.2	<i>Índice Jaccard</i>	52
2.2.5.3	<i>Índice Rand Ajustado</i>	53
2.3	Considerações Finais	53
3	FRAMEWORK HDBSCAN*	55
3.1	Definições	56
3.1.1	DBSCAN*	56

3.1.2	<i>Definições de HDBSCAN*</i>	57
3.1.3	<i>Algoritmo HDBSCAN*</i>	59
3.1.4	<i>Simplificação de Hierarquia</i>	60
3.1.5	<i>Complexidade Computacional</i>	63
3.2	<i>Agrupamento não-hierárquico ótimo</i>	64
3.2.1	<i>Estabilidade de Grupo</i>	65
3.2.2	<i>Algoritmo de Otimização de Extração Não-Supervisionada de Grupos</i>	67
3.3	<i>Método GLOSH</i>	69
3.4	<i>Considerações Finais</i>	72
4	TÉCNICAS DE AGRUPAMENTO PARALELO E DISTRIBUÍDO DE DADOS	73
4.1	<i>Algoritmo DiSC</i>	73
4.2	<i>Parallel OPTICS</i>	74
4.3	<i>Random Blocks</i>	75
4.4	<i>Considerações Finais</i>	77
5	ABORDAGEM RECURSIVE SAMPLING	79
5.1	<i>Visão Geral de Recursive Sampling</i>	79
5.2	<i>Extração de Grupos Significativos</i>	82
5.3	<i>Inclusão de Ruído</i>	84
5.4	<i>Arestas Intergrupos</i>	85
5.5	<i>Indução de Partição</i>	86
5.6	<i>Data Bubbles</i>	87
5.6.1	<i>Definições</i>	89
5.7	<i>Implementação de Recursive Sampling utilizando Data Bubbles</i>	93
5.8	<i>Implementação de Recursive Sampling baseada em Memória Compartilhada</i>	98
5.9	<i>Considerações Finais</i>	100
6	IMPLEMENTAÇÃO DE HDBSCAN* USANDO MAPREDUCE	101
6.1	<i>MapReduce</i>	101
6.1.1	<i>Apache Hadoop</i>	104
6.1.2	<i>Apache Spark</i>	105
6.2	<i>Recursive Sampling em MapReduce</i>	105
6.2.1	<i>Primeira Etapa: Modelos Locais de HDBSCAN*</i>	108
6.2.2	<i>Segunda Etapa: Combinação de MSTs Locais e Arestas Intergrupos</i>	114
6.2.3	<i>Terceira Etapa: Construção de Hierarquia de MR-HDBSCAN* sobre o Conjunto de Dados Completo</i>	116
6.2.3.1	<i>Construção Divisiva da Hierarquia</i>	118

6.2.3.1.1	Algoritmo CC-MR	119
6.2.3.1.2	Construção de Hierarquia de MR-HDBSCAN* - Abordagem Divisiva	123
6.2.3.2	Construção Aglomerativa da Hierarquia	129
6.2.3.2.1	Estrutura de Dados - Disjoint Set - Abordagem Paralela	129
6.2.3.2.2	Construção de Hierarquia de MR-HDBSCAN* - Abordagem Aglo- merativa	131
6.2.4	Grupos Significativos e Detecção de Outliers	134
6.2.5	Análise de Complexidade Assintótica	136
6.3	Considerações Finais	138
7	EXPERIMENTOS	139
7.1	Conjunto de Dados	139
7.2	Experimento utilizando Memória Distribuída	142
7.2.1	Algoritmos	142
7.2.2	Configurações de Algoritmos	143
7.2.3	Configurações de Hardware	144
7.2.4	Avaliação de Qualidade de Agrupamento de Dados	145
7.2.4.1	<i>Critério de Validação</i>	145
7.2.4.2	<i>Resultados e Discussão</i>	147
7.2.4.3	<i>Teste de Significância Estatística</i>	149
7.2.5	Avaliação de Desempenho Computacional	149
7.2.5.1	<i>Resultados e Discussão</i>	149
7.2.5.2	<i>Teste de Significância Estatística</i>	151
7.2.6	Avaliação de Métodos de Construção de Hierarquias	152
7.2.7	Avaliação de Detecção de Outliers usando o método GLOSH	153
7.2.7.1	<i>Critério de Validação</i>	153
7.2.7.2	<i>Configurações</i>	155
7.2.7.3	<i>Resultados e Discussão</i>	156
7.2.8	Conclusões	159
7.2.8.1	<i>Limitações</i>	160
7.3	Experimento utilizando Memória Compartilhada	160
7.3.1	Algoritmos	161
7.3.2	Critério de Validação	161
7.3.3	Configurações de Algoritmos	161
7.3.4	Configurações de Hardware	162
7.3.5	Avaliação de Qualidade de Agrupamento de Dados	162
7.3.5.1	<i>Resultados e Discussão</i>	163
7.3.5.2	<i>Teste de Significância Estatística</i>	164
7.3.6	Avaliação de Desempenho Computacional	165
7.3.6.1	<i>Resultados e Discussão</i>	165

7.3.6.2	<i>Teste de Significância Estatística</i>	166
7.3.7	<i>Avaliação de Detecção de Outliers usando o método GLOSH</i>	167
7.3.7.1	<i>Configurações</i>	168
7.3.7.2	<i>Resultados e Discussão</i>	169
7.3.8	<i>Conclusões</i>	171
7.4	<i>Considerações Finais</i>	172
8	CONCLUSÕES E TRABALHOS FUTUROS	173
	REFERÊNCIAS	175

INTRODUÇÃO

1.1 Motivação

O Agrupamento de Dados (ou *Clustering*), que também é conhecido como técnica de aprendizado não supervisionado de dados, é uma técnica capaz de identificar estruturas ou padrões em dados (HAND, 2008). Esses padrões são representados por objetos descritos em grupos, cujo objetos de um mesmo grupo compartilham características mais similares entre si do que objetos de grupos distintos (JAIN; DUBES, 1988).

As técnicas de *Agrupamento de Dados* costumam ser divididas em duas principais categorias: particionais e hierárquicas (JAIN; DUBES, 1988). As técnicas particionais descrevem um conjunto de objetos em um determinado número de grupos (EVERITT *et al.*, 2009; KING, 2014). Um dos algoritmos mais populares dessa categoria é conhecido como k-médias (ou *kmeans*) (MACQUEEN, 1967; JAIN, 2010). Já as técnicas hierárquicas constroem uma sequência aninhada de diferentes agrupamentos particionais. Dessa maneira, as técnicas hierárquicas oferecem resultados mais completos se comparadas aos das técnicas particionais, pois cada agrupamento particional que compõe a hierarquia possui um determinado nível de especificidade e generalidade dos dados (CAMPELLO *et al.*, 2013b).

Além disso, as técnicas hierárquicas podem ser utilizadas para descrever dados com estruturas hierarquicamente naturais (ARNOB, 2016; ZHAO *et al.*, 2005). Por exemplo, na biologia, árvores filogenéticas são representadas pelos relacionamentos entre ancestrais e descendentes em comum de espécies de animais, plantas, bactérias ou outros organismos de maneira hierárquica (ARNOB, 2016). Já em organização de documentos, estruturas hierárquicas são importantes ferramentas para determinar níveis de generalidade de temas e especificidade de seus subtemas relacionados às coletâneas desses documentos (ZHAO *et al.*, 2005). Algumas das técnicas hierárquicas mais populares são: *single-linkage* (FLOREK *et al.*, 1951); *complete-linkage* (JAIN; DUBES, 1988) e método de *Ward* (JR., 1963).

O agrupamento hierárquico de dados baseado em densidade é um tipo especial de agrupamento hierárquico que é capaz de descrever grupos com diferentes concentrações ou densidades de objetos de maneira hierárquica (CAMPELLO *et al.*, 2013a). O conceito de densidade sugere que objetos que formam um determinado grupo são altamente concentrados ou próximos em regiões do espaço de características (atributos) e separados de objetos de grupos distintos por regiões com menor concentração de objetos (ESTER *et al.*, 1996). Objetos que são localizados em regiões de baixa densidade ou concentração de objetos são comumente denotados como ruídos ou *outliers* (KRIEGEL *et al.*, 2011). Geralmente, essas técnicas de agrupamento desconsideram previamente a distribuição dos dados, tornando possível a detecção de grupos com formas arbitrárias (KRIEGEL *et al.*, 2011). Uma das técnicas mais populares deste paradigma é conhecida como *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

O *HDBSCAN** é uma versão hierárquica do clássico algoritmo *DBSCAN*, introduzido em (ESTER *et al.*, 1996) e aprimorado em (CAMPELLO *et al.*, 2013a), que é capaz de produzir infinitos níveis de densidade, de maneira que cada nível hierárquico represente um agrupamento particional baseado em densidade. Em outras palavras, *HDBSCAN** produz todas as soluções possíveis de *DBSCAN* de maneira não parametrizada em forma de sequências aninhadas de agrupamentos particionais (CAMPELLO *et al.*, 2013a). A partir dessas características, o *HDBSCAN** reduz diversas limitações de técnicas clássicas de agrupamento hierárquico de dados, tais como: indução de apenas grupos hiperesféricos (e.g., *average-linkage* (JAIN; DUBES, 1988), *método de Ward* (JR., 1963)), ou a sensibilidade com dados ruidosos (e.g., *single-linkage* (FLOREK *et al.*, 1951; SNEATH, 1957)). Ademais, o método *HDBSCAN** foi estendido para um *framework* de agrupamento semi-supervisionado de dados, além de fornecer um método de detecção de *outliers* (conhecido como *GLOSH*) e visualização de resultados (CAMPELLO *et al.*, 2015).

Embora técnicas de agrupamento hierárquico de dados, incluindo o *HDBSCAN**, forneçam resultados mais completos em relação aos das técnicas particionais, a complexidade computacional assintótica dessas técnicas é ao menos quadrática em relação ao número de objetos de um conjunto de dados (FLOREK *et al.*, 1951; JAIN; DUBES, 1988). Nesse contexto, no presente trabalho de mestrado, foram investigadas novas alternativas para o desenvolvimento de versões exatas e aproximadas de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), que são computacionalmente mais escaláveis para o processamento de grandes quantidades de dados.

1.2 Metodologia e Objetivos Gerais

O escopo deste trabalho de mestrado está inserido no estudo de tecnologias de computação paralela e distribuída aplicadas à tarefa de agrupamento de dados não supervisionado, incluindo a detecção de *outliers*, desempenhada pelo *framework HDBSCAN**. A computação paralela e distribuída pode ser vista a partir de duas principais características: computação para-

lela baseada em memória distribuída e computação paralela baseada em memória compartilhada (LEOPOLD, 2001). O conceito de memória compartilhada está relacionado a arquiteturas compostas por componentes ou computadores individuais munidos de recursos independentes (e.g., memória principal, disco, processadores, etc) (SIMA; KACSUK, 1997). Já a computação paralela baseada em memória compartilhada caracteriza arquiteturas em que múltiplos processadores ou núcleos compartilham dados em uma única memória principal (DAI, 2006).

Um dos principais paradigmas de computação paralela e distribuída é conhecido como *MapReduce* (DEAN; GHEMAWAT, 2008). A ideia desse paradigma é dividir um conjunto de dados em subconjuntos menores de maneira que possam ser processados paralelamente a partir de diferentes unidades de processamento independentes, ou seja, sem comunicação ou compartilhamento de informações entre elas, e, a partir disso, combinar os resultados dos processamentos individuais em uma ou mais soluções gerais (DEAN; GHEMAWAT, 2008).

Entretanto, a falta de comunicação entre diferentes unidades de processamento independentes pode ser considerada um impedimento para o desenvolvimento de técnicas de agrupamento baseadas em *MapReduce*, especialmente técnicas hierárquicas, pois essas são usualmente baseadas em comparações entre pares de objetos do conjunto de dados (JAIN; DUBES, 1988).

Desse modo, poucas técnicas hierárquicas de agrupamento de dados têm sido propostas ou adaptadas na literatura sobre *MapReduce* (HENDRIX *et al.*, 2012; JIN *et al.*, 2013b), especialmente o algoritmo *single-linkage*, que é um caso especial de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). No entanto, essas técnicas possuem limitações críticas: (a) um alto número de subconjuntos de dados criados por alguns desses métodos, a fim de realizar todas as comparações possíveis entre pares de objetos; (b) o processamento duplicado, em que a mesma computação ou comparação entre objetos do conjunto de dados pode ser feita em unidades de processamento distintas (HENDRIX *et al.*, 2012; JIN *et al.*, 2013b).

Neste contexto, uma nova arquitetura de implementação de *HDBSCAN** baseada em *MapReduce* foi proposta no presente trabalho de mestrado. Essa arquitetura consiste em uma versão aproximada de *HDBSCAN** que faz uso de uma eficiente técnica de divisão de dados em diferentes unidades de processamento, conhecida como *Recursive Sampling* (SYED, 2014). Especificamente, *Recursive Sampling* utiliza técnicas de sumarização de dados (e.g., *Data Bubbles* (BREUNIG *et al.*, 2001)) para eficientemente estimar as estruturas dos dados em grupos. Essas técnicas de sumarização são amplamente utilizadas no contexto de agrupamento de dados com o intuito de reduzir a complexidade computacional dos algoritmos desse meio (ZHANG *et al.*, 1996; BREUNIG *et al.*, 2001). Além disso, uma versão aproximada de *HDBSCAN**, também utilizando *Recursive Sampling*, baseada em computação paralela utilizando memória compartilhada de recursos também foi proposta neste trabalho. Para o desenvolvimento de ambas as versões, foram estabelecidos previamente os seguintes objetivos:

- O estudo de eventuais aproximações que possam levar ao desenvolvimento de variantes

com complexidade assintótica subquadrática. Especificamente, a exploração de técnicas sofisticadas de sumarização de dados para auxiliar o agrupamento de dados por *HDBSCAN** de maneira escalável;

- O estudo sobre as possibilidades de paralelização do algoritmo ou partes dele para cálculo exato em cenários envolvendo arquiteturas computacionais de multiprocessamento com memória compartilhada, bem como a avaliação de tais versões paralelas exatas em contraste a possíveis versões aproximadas para cenários de processamento distribuído com *MapReduce*;
- A investigação sobre as possibilidades de computar, mesmo que aproximadamente, a detecção de *outliers* pelo método *GLOSH* contido no *framework HDBSCAN** (CAMPELLO *et al.*, 2015);
- A avaliação e comparação experimental, em conjuntos de dados sintéticos e também em cenários reais de aplicação, dos diferentes algoritmos investigados nos itens anteriores.

1.3 Contribuições

Como citado anteriormente, o escopo deste trabalho de mestrado está relacionado ao estudo de novas versões escaláveis de *HDBSCAN** utilizando tecnologias de computação paralela e distribuída agregadas a um método eficiente de divisão de dados em diferentes unidades de processamento aninhado a técnicas de sumarização de dados. Especificamente, as versões propostas neste trabalho, assim como *HDBSCAN** (CAMPELLO *et al.*, 2015), fornecem um arcabouço de análise não supervisionada de dados composta por hierarquia de grupos, detecção de *outliers* pelo método *GLOSH* e extração de partição composta por grupos significativos dos dados. Desse modo, foram observadas as seguintes contribuições deste trabalho:

- Uma nova arquitetura de implementação de *HDBSCAN** baseada em *MapReduce* utilizando *Recursive Sampling* (SYED, 2014) e técnicas de sumarização de dados (ZHANG *et al.*, 1996; BREUNIG *et al.*, 2001) (Capítulo 6);
- Uma nova implementação paralela de *HDBSCAN** utilizando *Recursive Sampling* e técnicas de sumarização de dados aplicadas ao cenário de computação paralela em ambiente composto por memória compartilhada entre recursos (processadores ou núcleos). Especificamente, essas implementações são apresentadas na Seção 5.8 do Capítulo 5;
- Uma nova maneira de construção de hierarquias baseadas em densidade de *HDBSCAN** para as variantes aproximadas utilizando os conceitos de *MapReduce*. Especificamente, essa contribuição é descrita na Seção 6.2.3 do Capítulo 6;

- A extração de grupos significativos de dados de maneira paralela a partir de hierarquias de *HDBSCAN** construídas pelas variantes desse método utilizando *MapReduce* (Capítulo 6).
- O cálculo aproximado de estimativa (*score*) de *GLOSH* para cada objeto de qualquer conjunto de dados submetido às variantes de *HDBSCAN** baseadas em *MapReduce* (memória distribuída) e variantes baseadas em arquitetura baseada em memória compartilhada propostas neste trabalho (Capítulo 6).
- Avaliações de todas as variantes propostas neste projeto de pesquisa em contraste com versões exatas de *HDBSCAN** propostas na literatura (SYED, 2014) sobre grandes conjuntos de dados artificiais e reais (Capítulo 7).

1.4 Organização do Trabalho

O presente trabalho de mestrado está dividido da seguinte maneira: No Capítulo 2 serão apresentadas técnicas de agrupamento de dados, especificamente, métodos de agrupamento hierárquico de dados e métodos de agrupamento baseados em densidade relevantes para esta pesquisa. O *framework HDBSCAN**, que é o principal objeto de estudo deste trabalho de mestrado, é discutido em detalhes no Capítulo 3. No Capítulo 4, serão apresentados métodos de agrupamento distribuído de dados. No Capítulo 5, são apresentados o método *Recursive Sampling* proposto em (SYED, 2014), os conceitos de sumarização de dados (e.g., *BIRCH* (ZHANG *et al.*, 1996), *Data Bubbles* (BREUNIG *et al.*, 2001)) e implementações de variantes de *HDBSCAN** baseadas em computação paralela em ambientes de memória compartilhada. A nova arquitetura de implementação baseada em *MapReduce* de versões aproximadas de *HDBSCAN** a partir do método *Recursive Sampling* é apresentada em detalhes no Capítulo 6. No Capítulo 7, são apresentados os experimentos realizados sobre diversos conjuntos de dados para avaliar as versões aproximadas de *HDBSCAN** no contexto de memória distribuída e compartilhada. Por fim, no Capítulo 8, são apresentadas as conclusões e trabalhos futuros.

TÉCNICAS DE AGRUPAMENTO DE DADOS

Atualmente, dados são produzidos constantemente por diversos mecanismos ou dispositivos (HAN *et al.*, 2011). Esses dados costumam ser descritos como valores observados ou ocorrência de algum fenômeno específico (TAN *et al.*, 2005). Além de que, conjuntos de dados são separados em três principais categorias: estruturados, não estruturados e semiestruturados (KAUFMAN; ROUSSEEUW, 2009). Os dados estruturados são representados, em sua maioria, por tabelas, planilhas ou bancos de dados relacionais (XU; WUNSCH, 2009). Já os dados não estruturados podem ser representados por textos, *tweets* em redes sociais, imagens entre outros tipos (MOHAMMAD *et al.*, 2013; AGGARWAL; ZHAI, 2012; ANTONIE *et al.*, 2001). Por fim, dados semiestruturados são uma combinação de dados do tipo estruturados e não estruturados (CHAKRABORTY; CHAKI, 2011; SHAW *et al.*, 2016).

Analisar dados é uma tarefa presente no contexto de várias áreas do conhecimento (MELOUN; MILITKY, 2011). Logo, a descoberta de conhecimento em conjuntos de dados (do inglês *Knowledge Discovery in Databases (KDD)*) é uma importante ferramenta para a tomada de decisão baseada em dados, seja ela relacionada ao contexto empresarial, acadêmico ou qualquer outro cenário de aplicação (TAN *et al.*, 2005). Em um contexto geral, *KDD* é comumente representada por uma sequência de processos aplicados aos dados, com o propósito de conseguir informações valiosas (XU; WUNSCH, 2005; XU; WUNSCH, 2009). De maneira simplificada, são quatro passos utilizados sobre o conjunto de dados para a obtenção de conhecimento: o *pré-processamento* dos dados, a(s) técnica(s) de *agrupamento de dados*, a *validação* e a *interpretação dos dados*. Note que, pela Figura 1, esses quatro passos podem formar um ciclo de refinamento dos processos de *KDD* até que exista, de fato, o conhecimento sobre os dados (HAND, 2008).

Figura 1 – Processos necessários para análise de KDD envolvendo agrupamento de dados (adaptado de (XU; WUNSCH, 2009)).

Este capítulo está dividido da seguinte maneira: na Seção 2.1, são apresentados conceitos básicos de pré-processamento de dados; na Seção 2.2, são descritas definições e técnicas de agrupamento de dados; por fim, na Seção 2.3, são apresentadas as considerações finais deste capítulo.

2.1 Pré-processamento de Dados Estruturados

Geralmente, os conjuntos de dados a serem analisados são *brutos* (ou *atômicos*), i.e., dados que ainda não foram devidamente processados para a aplicação de técnicas de agrupamento de dados (KING, 2014). Em um contexto geral, conjuntos de dados brutos podem apresentar dados faltantes ou incompletos adquiridos por algum erro externo provocado por usuários ou máquinas no momento de seu preenchimento. Dentre esses dados faltantes são comuns a falta de algum atributo de um determinado objeto, valor inexistente de algum dos atributos, ou mesmo a falta desse objeto (XU; WUNSCH, 2009). Os dados também podem ser inconsistentes, ou seja, os valores de seus atributos não condizem com a realidade de sua categoria, por exemplo: um médico que tenha apenas três anos de idade. Por fim, podem existir nesses dados a presença de ruídos. Ruídos, neste caso, são dados que fogem da distribuição dos dados considerados “normais” em determinadas situações e muitas das vezes a sua distribuição é desconhecida (AGGARWAL, 2015). Esses tipos de dados citados, sem quaisquer correções de erros, podem influenciar ou atrapalhar o desempenho das técnicas de agrupamento de dados aplicadas sobre eles (HAND, 2008).

Os atributos contidos nos dados são descritos por diferentes tipos de medidas, tais como: numéricos e categóricos (CASTRO; FERRARI, 2016). Os atributos numéricos podem ser discretos (valores inteiros) ou contínuos (valores reais), enquanto que, atributos categóricos são representados por diferentes símbolos (XU; WUNSCH, 2005). Como pode ser visto na Figura 2, tanto os atributos numéricos quanto os atributos categóricos seguem um nível de especificidade. Nos atributos numéricos é comum encontrar, além dos valores discretos e contínuos, os valores indicados como *razão*, valores esses que representam a medida no ponto zero (exemplo: velocidade, peso, altura) (TAN *et al.*, 2005).

No que se refere aos atributos categóricos, os mesmos podem ser representados como binários, nominais e ordinais (TAN *et al.*, 2005). Os atributos binários são representados por

Figura 2 – Tipos de atributos. (Adaptado de (CASTRO; FERRARI, 2016)).

apenas dois valores, {Masculino ou Feminino}, {0 ou 1} e {Verdadeiro ou Falso} são alguns exemplos. Já os atributos nominais são caracterizados por símbolos distintos, por exemplo, dentre os valores que um atributo profissão pode assumir, tem-se: médico(a), administrador(a) entre outros. Por fim, os atributos ordinais assumem valores que expressam uma relação de ordem entre eles, {grande, médio, pequeno}, {graduação, mestrado, doutorado} são alguns tipos de valores assumidos por esses atributos ordinais (ANDERBERG, 1973).

Tendo em vista a variedade de problemas que os dados podem apresentar e os diversos tipos de valores que os atributos podem assumir nos conjuntos de dados, torna-se necessário utilizar técnicas adequadas de tratamento para prepará-los e enviá-los ao processo de agrupamento de dados. Essas técnicas são caracterizadas como *pré-processamento* de dados (AGGARWAL; REDDY, 2013). O *pré-processamento* de dados envolve diversas estratégias para lapidar os dados brutos de forma que os algoritmos de agrupamento de dados possam ter um bom desempenho. Dentre essas estratégias estão:

- **Integração de dados:** é uma tarefa utilizada para unificar dados disponíveis em diferentes repositórios (CASTRO; FERRARI, 2016). Além da unificação de dados em um único local, são incluídas as tarefas de integração, a eliminação de dados redundantes, ou seja, dados iguais que aparecem em diferentes repositórios (TAN *et al.*, 2005).
- **Limpeza de dados:** pode ser utilizada para a remoção ou atribuição de valores à atributos incompletos; remoção ou suavização de dados ruidosos e; correção de valores inconsistentes (WITTEN; FRANK, 2005). Os valores incompletos ou ausentes podem ser preenchidos, em sua maioria, por alguma estatística descritiva a partir dos valores atributo em questão, i.e., pela média, mediana ou moda, a depender do tipo de valor que o atributo assume (CASTRO; FERRARI, 2016). Já os dados ruidosos ou inconsistentes, quando não removidos, podem ser suavizados por técnicas como *binning*, quando identificados a partir dos dados (XU; WUNSCH, 2005; XU; WUNSCH, 2009).
- **Transformação de dados:** é uma tarefa usada para padronizar ou adequar os dados para determinadas técnicas de mineração de dados que são limitadas pelo tipo de dados analisados (HAN, 2005). Além disso, a transformação de dados é comumente aplicada

à dados com diferentes escalas de valores de seus atributos ou objetos (HAND, 2008). A variação da escala de valores de dados pode ser um fator prejudicial para técnicas que são baseadas em medidas de distância (e.g., distância euclidiana) (TAN *et al.*, 2005). Para aliviar este problema, algumas medidas de normalização de dados são comumente utilizadas na literatura (e.g., normalização *max-min* e *score z* (TAN *et al.*, 2005)).

- **Redução de dados:** tem o objetivo de reduzir o número de objetos ou atributos do conjunto de dados, a fim de, melhorar o desempenho computacional de diversas técnicas de *Mineração de Dados* (HAN *et al.*, 2011). Uma das técnicas mais populares de redução do número de atributos é conhecida como *análise de componentes principais* *Principal Components Analysis (PCA)* (JOLLIFFE, 2002). O *PCA* consiste em um processo estatístico de conversão de objetos que possam ter entre si alguma correlação em objetos linearmente não correlatos, chamados de componentes principais (JOLLIFFE, 2002). Já para a redução ou compressão do número de objetos de um conjunto de dados, uma das técnicas mais conhecidas é chamada de *Data Bubbles* (BREUNIG *et al.*, 2001). Os *Data Bubbles* conseguem descrever um conjunto de dados, a partir de uma pequena porção de objetos representativos dos dados, em que, cada objeto representativo é formado por algumas medidas descritivas de seus dados (e.g., média e variância) representados (BREUNIG *et al.*, 2001).

2.2 Agrupamento de Dados

O agrupamento de dados é uma técnica não supervisionada, ou seja, que não necessita de mecanismos ou informações externas aos dados para seus procedimentos, cujo objetivo é descrever os dados em grupos (EVERITT *et al.*, 2009; HAN, 2005). Os objetos contidos em um mesmo grupo compartilham características mais semelhantes entre eles do que objetos pertencentes a outros grupos (JAIN; DUBES, 1988). Essas características são identificadas por medidas de proximidade (Subseção 2.2.1) pela maioria das técnicas. Ainda, as técnicas de agrupamento de dados podem ser divididas em duas principais categorias: particionais e hierárquicas (HARTIGAN, 1975; KING, 2014).

Os algoritmos particionais procuram segmentar um conjunto de dados em um determinado número de grupos. Esses grupos compõem uma partição. Especificamente, existem na literatura vários conceitos de partição (BEZDEK, 1981; XU; WUNSCH, 2005), uma delas, que está presente na maioria das técnicas de agrupamento particional de dados, é a partição rígida (*hard*). Uma partição rígida é uma coleção de k grupos não sobrepostos, ou seja, cada objeto do conjunto de dados pertence a um único grupo da partição. Matematicamente, sendo $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ um conjunto de dados multidimensionais de tamanho n a ser agrupado em k grupos, a partição rígida $\mathbf{P} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$ é denotada pelas restrições $\{\mathbf{C}_1 \cup \mathbf{C}_2 \cup \dots \cup \mathbf{C}_k\} = \mathbf{X}$, $\mathbf{C}_i \neq \emptyset$ e $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ para $i \neq j$ (JAIN; DUBES, 1988; PEDRYCZ, 2005). Existem diversos algoritmos que seguem o

conceito de partição rígida (e.g., k-médias (MACQUEEN, 1967)), k-medóides (KAUFMAN; ROUSSEEUV, 1987), DBSCAN (ESTER *et al.*, 1996).

Já algoritmos hierárquicos constroem sequências aninhadas de agrupamentos particionais em que cada sequência é separada por diferentes níveis de especificidade (LAROSE, 2014; CAMPELLO *et al.*, 2013b). Esse fato reflete em uma maior gama de resultados que podem ser avaliados de diversas formas por especialistas de domínio dos dados agrupados em questão (AGGARWAL; REDDY, 2013).

Adicionalmente, o conceito de grupos é uma questão subjetiva e pode assumir diversas definições devido à natureza dos dados (KRIEGL *et al.*, 2011). Sendo assim, várias técnicas de agrupamento de dados são propostas na literatura com o intuito de identificar estruturas significativas nos dados (XU; WUNSCH, 2009). Algumas técnicas assumem previamente a distribuição dos dados no espaço (e.g., misturas gaussianas (algoritmo *Expectation Maximization*) (DEMPSTER *et al.*, 1977)), há aquelas que desprezam essa distribuição e identificam grupos com formatos arbitrários por meio do conceito de grupos baseados em densidade (e.g., DBSCAN (ESTER *et al.*, 1996)) e outras determinam hierarquias completas sobre os dados (e.g., HDBSCAN* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015)).

Nesse contexto, na Subseção 2.2.2, são descritos alguns dos algoritmos clássicos de agrupamento hierárquico de dados. Por fim, na Subseção 2.2.4, são apresentados alguns algoritmos de agrupamento de dados baseados em densidade.

Diferentes técnicas de agrupamento de dados ou mesmo diferentes configurações de parâmetros de uma única técnica tendem a retornar resultados distintos para um mesmo conjunto de dados (KING, 2014). Além dos diferentes resultados, há a possibilidade de que estes mesmos representem ou não a estrutura dos dados esperada por algum sistema especialista nesses dados. Neste contexto, técnicas de validação de agrupamentos de dados (Subseção 2.2.5) são utilizadas com o intuito de mensurar a qualidade de cada partição ou hierarquia geradas pelas técnicas de agrupamento de dados (EVERITT *et al.*, 2009).

2.2.1 Medidas de Proximidade

Em análise de agrupamento de dados, um dos conceitos básicos de grupo refere-se a objetos similares pertencentes ao mesmo grupo, enquanto os objetos menos similares a estes pertencem a grupos distintos (KAUFMAN; ROUSSEEUV, 2009). À vista disso, medidas de proximidade precisam ser estabelecidas para determinar a semelhança ou diferença entre cada objeto do conjunto de dados (AGGARWAL; REDDY, 2013). Medida de proximidade é um termo generalizado para se referir às medidas de similaridade e dissimilaridade (distância) (JAIN; DUBES, 1988). Dentre as diversas medidas de proximidades encontradas na literatura, algumas dessas medidas lidam com variáveis (atributos) contínuas e variáveis discretas (binárias) (XU; WUNSCH, 2009). Esses dois tipos de medidas são discutidos nas Subseções 2.2.1.1 e 2.2.1.2,

respectivamente.

2.2.1.1 Medidas de Proximidade - Variáveis Contínuas

A medida de dissimilaridade, também denotada como medida de distância é indicada de modo generalizado como um escalar entre dois vetores (objetos) multidimensionais (DUDA *et al.*, 2000). Sendo $d(.,.)$ a distância generalizada entre dois vetores, $\mathbf{a}$, $\mathbf{b}$ e $\mathbf{c}$ vetores multidimensionais, a medida de distância segue quatro propriedades fundamentais:

- **Não-negatividade:** $d(\mathbf{a}, \mathbf{b}) \geq 0$.
- **Reflexividade:** $d(\mathbf{a}, \mathbf{b}) = 0$ se e somente se, $\mathbf{a} = \mathbf{b}$.
- **Simetria:** $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$.
- **Desigualdade triangular:** $d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$.

A *distância Euclidiana* é uma das medidas de proximidade mais conhecidas no contexto de *Mineração de Dados* e que segue todas as propriedades citadas acima (JAIN; DUBES, 1988; XU; WUNSCH, 2009). No contexto de *Agrupamento de Dados*, os grupos encontrados pela *distância Euclidiana* (representada na Equação 2.1) são invariantes à translação e rotação do espaço no plano cartesiano (DUDA *et al.*, 2000).

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^d |x_{il} - x_{jl}|^2 \right)^{\frac{1}{2}} \quad (2.1)$$

A *distância Euclidiana* (também conhecida como norma L_2) faz parte de uma classe de medidas gerais, denominadas medidas de *Minkowski* ou norma L_p (JAIN; DUBES, 1988). Essa medida é representada pela Equação 2.2:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^d |x_{il} - x_{jl}|^p \right)^{\frac{1}{p}} \quad (2.2)$$

Note que, quando $p = 2$, a Equação 2.2 representa a *distância Euclidiana*, para os casos de $p = 1$ e $p = \infty$, essa mesma equação representa as variantes: *distância de Manhattan* (norma L_1) e *distância Suprema* (norma L_∞), respectivamente (AGGARWAL; REDDY, 2013).

Distância de Mahalanobis

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \quad (2.3)$$

Em alguns cenários de aplicação, tais como medicina (HARTIGAN, 1975; MUKAKA, 2012), é interessante analisar a correlação entre as características (ou atributos) de objetos. Para

este propósito, a medida de similaridade conhecida como *Correlação de Pearson* é geralmente utilizada (XU; WUNSCH, 2009). Essa medida busca a interdependência linear entre as características de dois objetos $\mathbf{x}_i$ e $\mathbf{x}_j$. Em termos quantitativos, a *Correlação de Pearson*, apresentada na Equação 2.4, r_{ij} pode assumir valores no intervalo $[-1, 1]$. Quando $r_{ij} = 1$, a correlação linear entre $\mathbf{x}_i$ e $\mathbf{x}_j$ é perfeitamente positiva, ao passo que, quando $r_{ij} = -1$, essa correlação passa a ser perfeitamente negativa. Por fim, se $r_{ij} = 0$, os objetos $\mathbf{x}_i$ e $\mathbf{x}_j$ não são linearmente dependentes, segundo a *Correlação de Pearson*, portanto, não são correlacionados (HAN; KAMBER, 2000; HAN, 2005).

$$r_{ij} = \frac{\sum_{l=1}^d (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)}{\sqrt{\sum_{l=1}^d (x_{il} - \bar{x}_i)^2 \sum_{l=1}^d (x_{jl} - \bar{x}_j)^2}} \quad (2.4)$$

Outra medida para atributos contínuos é a similaridade *cos seno*. Diferente da correlação de *Pearson*, essa medida trata o relacionamento entre dois vetores numéricos de maneira assimétrica e é robusta no tratamento de valores nulos (TAN *et al.*, 2005). Em um contexto de aplicação, a similaridade *cos seno* é bastante utilizada em cenários de mineração de textos (XU; WUNSCH, 2009). Sua fórmula pode ser expressa pela Equação 2.5:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|} \quad (2.5)$$

Em que $\cdot$ representa o produto interno entre os vetores (objetos) $\mathbf{x}_i$ e $\mathbf{x}_j$ e $\|\cdot\|$ representa distância norma L_2 (descrita na Subseção 2.2.1). Uma vez que a medida apresentada na Equação 2.5 possui o intervalo de valores entre $[0, 1]$, essa equação mede o ângulo *cos seno* entre dois vetores e descreve se ambos são maximamente similares, ou seja, são paralelos; se a métrica na Equação 2.5 for igual a 1 ou se um desses vetores é perpendicular ao outro, essa medida descreve que ambos são minimamente similares, portanto, a Equação 2.5 será igual a 0 (XU; WUNSCH, 2009).

2.2.1.2 Medidas de Similaridade - Variáveis Binárias

Há situações em que conjuntos de dados apresentam em seus atributos características binárias, ou seja, assim como descrito na Seção 2.1, os valores de atributos assumidos por um objeto podem ser representados por $\{0, 1\}$, $\{M, F\}$, entre outros. Diante disso, são dispostas na literatura algumas medidas de similaridade capazes de mensurar quantitativamente esses tipos de dados (HAN *et al.*, 2011). Nesta seção, são apresentadas duas medidas de similaridade bastante utilizadas, quais sejam: a similaridade *Casamento Simples* e a similaridade *Jaccard* (XU; WUNSCH, 2009).

Seja $\mathbf{x}_i$ e $\mathbf{x}_j$ dois objetos multidimensionais compostos por valores binários $\{0, 1\}$, por exemplo. O objetivo das medidas de similaridades para variáveis binárias é contar o número de

pares de valores que concordam entre si nos vetores $\mathbf{x}_i$ e $\mathbf{x}_j$, representados por n_{11} e n_{00} , e os pares que não concordam entre si, representados por n_{01} e n_{10} (AGGARWAL; REDDY, 2013).

Neste contexto, a similaridade *Casamento Simples*, apresentada na Equação 2.6, é uma medida simétrica com relação ao grau de importância dado aos casamentos entre pares de valores dos vetores $\mathbf{x}_i$ e $\mathbf{x}_j$, i.e., tanto casamentos 1-1, quanto 0-0 possuem a mesma importância nessa medida (HAN; KAMBER, 2000; HAN, 2005). Além disso, essa medida simétrica é interessante para cenários em que o casamento de pares que concordam entre si não são de interesse comum da aplicação (XU; WUNSCH, 2009).

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}} \quad (2.6)$$

Outra medida bastante comum na literatura é a similaridade *Jaccard* (apresentada na Equação 2.7). Essa medida desconsidera o grau de importância de pares de valores que concordam em relação a um dos valores presentes nos vetores, i.e., a contagem de pares que concordam em relação aos casamentos 0-0 de dois vetores $\mathbf{x}_i$ e $\mathbf{x}_j$. Por essa característica, a similaridade *Jaccard* é denominada como uma medida assimétrica (HAN *et al.*, 2011).

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \quad (2.7)$$

2.2.2 Algoritmos Hierárquicos

Os algoritmos hierárquicos constroem sequências aninhadas de agrupamentos particionais em que cada sequência é separada por diferentes níveis de especificidade (LAROSE, 2014). Tal fato reflete em uma maior gama de resultados que podem ser avaliados de diversas formas por especialistas de domínio dos dados agrupados em questão (AGGARWAL; REDDY, 2013). Em geral, algoritmos hierárquicos são divididos em duas categorias: *aglomerativas* (Subseção 2.2.2.1) e *divisivas* (Subseção 2.2.2.3).

Adicionalmente, os resultados dos algoritmos de agrupamento hierárquico de dados baseados nessas duas categorias podem ser representados graficamente por uma estrutura conhecida como dendrograma, apresentado na Figura 3:

2.2.2.1 Algoritmos Aglomerativos

Algoritmos aglomerativos consideram inicialmente que cada objeto do conjunto de dados é um grupo único (conhecido também como *singleton*) e, iterativamente, une os grupos de acordo com seus inter-relacionamentos¹ (HAND, 2008). A cada união, um novo nível da hierarquia é construído, esse nível é representado pela (dis)similaridade ou outras relações de proximidade

¹ Em técnicas clássicas desse tipo de agrupamento, o inter-relacionamento entre pares de objetos do conjunto de dados é representado por uma matriz de proximidade (similaridade ou distância) (JAIN; DUBES, 1988)

Figura 3 – Exemplo de agrupamento hierárquico (dendrograma) de dados baseado nas abordagens: Aglomerativa e Divisiva. (Adaptado de (XU; WUNSCH, 2005)).

entre os grupos. O processo iterativo de união dos grupos é repetido até que todos esses grupos façam parte de um único grupo (HAN *et al.*, 2011). Alguns algoritmos aglomerativos tradicionais são: *single-linkage* (FLOREK *et al.*, 1951; SNEATH, 1957), *complete-linkage* (JAIN; DUBES, 1988), *Group Average-linkage* (AGGARWAL; REDDY, 2013), *método de variância mínima* (ou método de Ward) (JR., 1963) e *BIRCH* (ZHANG *et al.*, 1996).

2.2.2.2 *Single-linkage*

Single-linkage, que também é conhecido como algoritmo baseado em vizinhos mais próximos (ou *nearest neighbors*), é um dos algoritmos hierárquicos de agrupamento de dados mais simples existentes (FLOREK *et al.*, 1951; SNEATH, 1957). Por meio da analogia de vizinho mais próximo, pode-se medir a distância entre pares de grupos de forma que sejam unidos conforme a distância mínima entre eles. Na Equação 2.8 é expressa a forma de obter a distância mínima entre dois grupos distintos.

$$d(C, C') = \min_{x \in C, y \in C'} d(x, y) \quad (2.8)$$

Em que $d(.,.)$ representa a distância entre grupos, C e C' são dois grupos não vazios e não sobrepostos, e x e y são objetos pertencentes aos grupos C e C' , respectivamente.

Embora seja sensível à presença de ruídos, *single-linkage* é capaz de encontrar grupos com formas arbitrárias e complexas nos dados. Além disso, esse algoritmo é invariante às transformações monótonas (JAIN; DUBES, 1988; JOHNSON, 1967).

Adicionalmente, *single-linkage* possui uma relação com o conceito de árvore geradora mínima (ou *Minimum Spanning Tree - MST*) (JAIN; DUBES, 1988). Desse modo, a hierarquia de *single-linkage* pode ser computada de maneira divisiva por meio da remoção iterativa de arestas de maior peso da *MST*. A cada remoção de arestas, um novo nível hierárquico é construído (CAMPELLO *et al.*, 2013a).

2.2.2.3 Complete-linkage

Diferentemente de *single-linkage*, o algoritmo *complete-linkage* utiliza a maior distância entre dois grupos distintos para atualizar o processo de agrupamento (JAIN; DUBES, 1988). Neste contexto, a Equação 2.9 mostra o princípio básico de atualização entre grupos deste algoritmo.

$$d(\mathbf{C}, \mathbf{C}') = \max_{\mathbf{x} \in \mathbf{C}, \mathbf{y} \in \mathbf{C}'} d(\mathbf{x}, \mathbf{y}) \quad (2.9)$$

Complete-linkage é mais robusto ao tratamento de ruídos e *outliers* e um ótimo candidato para encontrar grupos globulares. Por outro lado, esse algoritmo tende a dividir grupos com maior número de objetos (JAIN; DUBES, 1988; AGGARWAL; REDDY, 2013).

2.2.2.4 Group Average-linkage

Group Average-linkage ou *Unweighted Pair Group Method Using Arithmetic Averages* (UPGMA) (JAIN; DUBES, 1988), atualiza a distância entre grupos pela média de todos os pares de objetos possíveis desses grupos (AGGARWAL; REDDY, 2013). Esse cálculo é expressado pela Equação 2.10.

$$d(\mathbf{C}, \mathbf{C}') = \frac{1}{|\mathbf{C}| |\mathbf{C}'|} \sum_{\mathbf{x} \in \mathbf{C}, \mathbf{y} \in \mathbf{C}'} d(\mathbf{x}, \mathbf{y}) \quad (2.10)$$

Assim como *Complete-linkage*, *Group Average-linkage* é robusto ao tratamento de ruídos e *outliers*. Por outro lado, esse algoritmo tende a encontrar apenas grupos com estruturas globulares (JAIN; DUBES, 1988).

2.2.2.5 Método de Variância Mínima

Diferente das abordagens anteriores, o método de variância mínima (ou método de *Ward*) é um algoritmo que se baseia na minimização da variância intragrupos à medida que eles são unidos, ou seja, a cada partição que surge em determinado nível hierárquico (XU; WUNSCH, 2009). Essa minimização é feita por uma função objetivo baseada no *Critério de Erro Quadrático* (SEQ), representado pela Equação 2.11 (JR., 1963):

$$J = \sum_{i=1}^k \sum_{\mathbf{x}_j \in \mathbf{C}_i} d(\mathbf{x}_j, \bar{\mathbf{x}}_i)^2 \quad (2.11)$$

em que $\bar{\mathbf{x}}_i$ representa o centróide do i -ésimo grupo, k indica o número de grupos a cada nível da hierarquia e $d(\cdot)$ representa distância *Euclidiana*.

Semelhante aos algoritmos *Complete-linkage* e *Group Average-linkage*, o método de *Ward's* é robusto a ruídos e à *outliers*, embora seja limitado geração de grupos globulares (JAIN; DUBES, 1988).

2.2.2.6 BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) é um algoritmo hierárquico aglomerativo proposto para reduzir a complexidade computacional, diminuir os procedimentos de entrada e saída de dados e o controle explícito de memória disponível no momento do agrupamento. Assim, o algoritmo *BIRCH* pode ser capaz de agrupar grandes quantidades de dados de maneira escalável (ZHANG *et al.*, 1996).

Adicionalmente, esse algoritmo foi um dos primeiros nas áreas de banco de dados a estabelecer o tratamento de ruídos nos dados e o primeiro a apresentar de forma sofisticada o conceito de sumarização de dados para aplicação de algoritmos de agrupamento de dados (ZHANG *et al.*, 1996).

BIRCH, em sua essência, utiliza estatísticas suficientes descritas por um vetor conhecido como *Clustering Feature (CF)*, que sumariza informações de subconjuntos de objetos. Dado um subconjunto de objetos multidimensional $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ de tamanho n , um vetor *CF* é descrito por uma tupla $CF = (n, LS, SS)$ de informações sumarizadas sobre esse subconjunto, em que n representa a quantidade de objetos pertencentes ao conjunto $\mathbf{X}_i$, LS corresponde à soma linear dos objetos (descrita pela Equação 2.12) e SS a soma quadrática dos objetos pertencentes a $\mathbf{X}_i$ (Equação 2.13) (ZHANG *et al.*, 1996).

$$LS = \sum_{\mathbf{x}_i \in \mathbf{X}} \mathbf{x}_i \quad (2.12)$$

$$SS = \sum_{\mathbf{x}_i \in \mathbf{X}} \mathbf{x}_i^2 \quad (2.13)$$

Vetores *CF* são inseridos dinamicamente em uma estrutura de dados denominada *CF-tree*. Uma estrutura de dados *CF-tree* contém três parâmetros: B , T e L . Cada nó não folha contém no máximo, B , subnós no formato $[CF_i, child_i]$, cada entrada em um nó folha deve ser menor que o limiar T (ZHANG *et al.*, 1996).

Estatísticas suficientes contidas em vetores *CF* podem ser utilizadas por outros algoritmos de agrupamento de dados. Um exemplo clássico que tem bom desempenho quando submetido a dados sumarizados pelas estatísticas suficientes é o algoritmo *k*-médias (ou *k-means*, do inglês) (MACQUEEN, 1967; GUHA *et al.*, 1998).

2.2.3 Algoritmos Divisivos

Algoritmos divisivos não são encontrados com tanta frequência como ocorre com os algoritmos hierárquicos aglomerativos (JAIN; DUBES, 1988). O motivo disso é que ao considerar um conjunto de dados com n objetos, existem $2^n - 1$ combinações diferentes para dividir grupos. Fato que, aumenta significativamente a complexidade computacional dos algoritmos que seguem essa abordagem (JAIN; DUBES, 1988).

Outro problema comum de alguns algoritmos hierárquicos divisivos é a falta de monotonicidade da hierarquia gerada (XU; WUNSCH, 2005). Nesse caso, dependendo da estratégia utilizada para a divisão de grupos, podem ocorrer variações de tamanhos dos níveis na hierarquia construída (JAIN; DUBES, 1988). Para garantir a monotonicidade nesse algoritmo, é preciso estabelecer ao processo de construção de uma hierarquia que o nível atual nunca será maior do que o nível estabelecido anteriormente (XU; WUNSCH, 2009).

Entretanto, existem na literatura algoritmos hierárquicos divisivos capazes de garantir a monotonicidade na construção de hierarquias de agrupamento e sem a necessidade de empregar todas as combinações possíveis de divisões de grupos. O algoritmo *DIANA* (KAUFMAN; ROUSSEEUW, 2009) é um exemplo que consegue lidar com os problemas citados anteriormente.

2.2.3.1 DIANA

Divisive ANALysis (*DIANA*) é um algoritmo hierárquico divisivo inspirado em uma abordagem clássica proposta em Macnaughton-Smith *et al.* (1964). Inicialmente, esse algoritmo considera todos os objetos de um conjunto de dados como sendo de um único grupo e, a cada passo do algoritmo, um grupo com maior diâmetro é dividido até que restem $n - 1$ grupos, ou seja, até que o algoritmo chegue a uma solução trivial. De maneira complementar, o diâmetro (Equação 2.14) encontrado a cada passo do algoritmo *DIANA* é utilizado como níveis de um dendrograma (XU; WUNSCH, 2005; XU; WUNSCH, 2009).

Considerando que o grupo C_l será dividido em dois novos grupos, C_i e C_j . Nesse contexto, as iterações do algoritmo *DIANA* são apresentadas pela sequência de passo:

1. Inicia o grupo C_i como sendo igual a C_l e C_j como um grupo vazio;
2. Para cada objeto $\mathbf{x}_m \in C_i$:
 - a) Para a primeira iteração, calcula a distância média para todos os outros objetos em C_i pela Equação 2.14:

$$d(\mathbf{x}_m, C_i \setminus \{\mathbf{x}_m\}) = \frac{1}{N_{C_i} - 1} \sum_{\mathbf{x}_p \in C_i, p \neq m} d(\mathbf{x}_m, \mathbf{x}_p) \quad (2.14)$$

em que $C_i \setminus \{\mathbf{x}_m\}$ indica todos os objetos $\mathbf{x}_p \in C_i$, tal que, $m \neq p$ e N_{C_i} o número de objetos em C_i .

- b) Para o restante das iterações, calcula a diferença entre a distância média de C_i e a distância média de C_j pela Equação 2.15:

$$d(\mathbf{x}_m, C_i \setminus \{\mathbf{x}_m\}) - d(\mathbf{x}_m, C_j) = \frac{1}{N_{C_i} - 1} \sum_{x_p \in C_i, p \neq m} d(\mathbf{x}_m, \mathbf{x}_p) - \frac{1}{N_{C_j}} \sum_{x_q \in C_j} d(\mathbf{x}_m, \mathbf{x}_q) \quad (2.15)$$

em que N_{C_i} e N_{C_j} representam o número de objetos nos grupos C_i e C_j , respectivamente.

3. a) Para a primeira iteração, o objeto que contém o maior valor (obtido pela Equação 2.14) para o grupo C_j .
- b) Nas iterações restantes, se o valor máximo obtido pela Equação 2.15 for maior que 0, então, o objeto com maior valor da diferença (obtido pela Equação 2.15) é movido para o grupo C_j e os passos 2 b) e 3 b) são repetidos iterativamente. Caso contrário, a divisão dos grupos é finalizada.

No mais, a abordagem baseada no diâmetro entre grupos faz com que o algoritmo *DIANA* seja sensível a *outliers* (XU; WUNSCH, 2009).

2.2.4 Algoritmos de Agrupamento Baseados em Densidade

Encontrar grupos em dados espaciais é de suma importância para diversas aplicações do mundo real. Entretanto, esses tipos de dados podem ser constituídos por grupos com formas arbitrárias e possuem grandes quantidades de objetos (KRIEGEL *et al.*, 2011). Além dessas características, outro fator relevante para que algumas técnicas de agrupamento não sejam eficientes neste domínio de aplicação é a determinação prévia pelo usuário da quantidade de grupos desejada para ser formada pelos algoritmos. Logo, técnicas como: *DBSCAN* (ESTER *et al.*, 1996), *OPTICS* (ANKERST *et al.*, 1999a), entre outras, foram propostas para minimizar esses problemas. As principais características dos algoritmos *DBSCAN* e *OPTICS* são apresentadas nas Subseções 2.2.4.1 e 2.2.4.2, respectivamente.

2.2.4.1 DBSCAN

Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (*DBSCAN*), proposto por Ester *et al.* (1996), é um algoritmo robusto para encontrar grupos com formas variadas. Basicamente, esse algoritmo consegue distinguir regiões com alta densidade de objetos (denominadas grupos) que são separadas por regiões com baixa densidade de objetos. Adicionalmente, *DBSCAN* também é robusto em relação à detecção de ruídos e *outliers*. Nesta seção, são apresentadas as definições formais de *DBSCAN* tal como descritas em (ESTER *et al.*, 1996).

Neste contexto, seja $\mathbf{X}$ um conjunto de objetos multidimensionais, $\mathbf{x}$ e $\mathbf{y}$ objetos pertencentes a $\mathbf{X}$, têm-se que:

Definição 1. (ε -vizinhança de um objeto): A ε -vizinhança (denotada por $N_\varepsilon(\mathbf{x})$) é definida por $N_\varepsilon(\mathbf{x}) = \{\mathbf{y} \in \mathbf{X} \mid d(\mathbf{x}, \mathbf{y}) \leq \varepsilon\}$.

Em complemento à **Definição 1**, é necessário considerar três tipos de objetos para a formação de grupos: os *core*, os *border* e os *noise* (ou ruído). Objetos *core* precisam ter uma quantidade mínima de objetos (m_{pts}) em suas ε -vizinhanças. Já objetos *border* não contêm pelo menos uma quantidade mínima de objetos em suas ε -vizinhanças mas, por definição, precisam pertencer a ε -vizinhança de pelo menos um objeto *core* (ESTER *et al.*, 1996). Por fim, os objetos *noise* não contêm pelo menos m_{pts} objetos em suas ε -vizinhanças e não são vizinhos de nenhum objeto *core*. A definição a seguir descreve essas características mencionadas:

Definição 2. (diretamente acessível): Um objeto $\mathbf{x}$ é diretamente acessível por um objeto $\mathbf{y}$, com relação a ε e m_{pts} , se:

1. $\mathbf{x} \in N_\varepsilon(\mathbf{y})$
2. $|N_\varepsilon(\mathbf{y})| \geq m_{pts}$ (condição para ser objeto *core*).

Diretamente acessível é simétrica para pares de objetos *core*. Em contrapartida, isto não se aplica se um *objeto core* e um *objeto border* estão envolvidos (ESTER *et al.*, 1996).

Definição 3. (densamente acessível): Um objeto $\mathbf{x}$ é densamente acessível por um objeto $\mathbf{y}$, com relação a ε e m_{pts} , se existe uma sequência de objetos $\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}_1 = \mathbf{y}, \mathbf{x}_n = \mathbf{x}$, em que $\mathbf{x}_{i+1}$ é diretamente acessível por $\mathbf{x}_i$.

Definição 4. (densamente conectado): Um objeto $\mathbf{x}$ é densamente conectado a um objeto $\mathbf{y}$, com relação a ε e m_{pts} , se há um objeto $\mathbf{o}$ tal que ambos, $\mathbf{x}$ e $\mathbf{y}$, são densamente acessíveis por $\mathbf{o}$.

Definição 5. (grupo): Seja $\mathbf{X}$ um conjunto de objetos. Um grupo $\mathbf{C}$, com relação a ε e m_{pts} , é um subconjunto de $\mathbf{X}$ não vazio que satisfaz as seguintes condições:

1. $\forall \mathbf{x}, \mathbf{y} : \text{se } \mathbf{x} \in \mathbf{C} \text{ e } \mathbf{y} \text{ é densamente acessível por } \mathbf{x}, \text{ com relação a } \varepsilon \text{ e } m_{pts}, \text{ então } \mathbf{y} \in \mathbf{C}$. (Maximalidade).
2. $\forall \mathbf{x}, \mathbf{y} \in \mathbf{C} : \mathbf{x} \text{ é densamente conectado a } \mathbf{y}, \text{ com relação a } \varepsilon \text{ e } m_{pts}$. (Conectividade).

Definição 6. (ruído): Seja $\mathbf{C}_1, \dots, \mathbf{C}_k$ grupos de conjunto de dados $\mathbf{X}$, com relação aos parâmetros ε_i e m_{pts_i} , $i = 1, \dots, k$. Assim, define-se ruídos como um subconjunto de objetos em um conjunto de dados $\mathbf{X}$ que não pertencem a nenhum grupo $\mathbf{C}_i$, i.e, ruído = $\{\mathbf{x} \in \mathbf{X} \mid \forall i : \mathbf{x} \notin \mathbf{C}_i\}$.

Basicamente, são necessários dois passos para encontrar grupos pelo algoritmo *DBSCAN*, quais sejam: o primeiro consiste em escolher um objeto aleatoriamente que satisfaça a definição de *objeto core* como semente inicial; o segundo passo consiste em recuperar todos os objetos que são densamente acessíveis por essa semente, obtendo o grupo que a contém. Logo, dois lemas são necessários para validar a corretude do algoritmo *DBSCAN*.

Lema 1: Seja $\mathbf{x}$ um objeto pertencente a $\mathbf{X}$ e $|N_\epsilon(\mathbf{x})| \geq m_{pts}$. Então, o conjunto $\mathbf{O} = \{\mathbf{o} \mid \mathbf{o} \in \mathbf{X} \text{ e } \mathbf{o} \text{ é densamente acessível por } \mathbf{x}, \text{ com relação a } \epsilon \text{ e } m_{pts}\}$, é um grupo.

Lema 2: Seja $\mathbf{C}$ um grupo, com relação a ϵ e m_{pts} , e seja $\mathbf{x}$ algum objeto pertencente a $\mathbf{C}$ com $|N_\epsilon(\mathbf{x})| \geq m_{pts}$. Então, $\mathbf{C}$ é igual ao conjunto $\mathbf{O} = \{\mathbf{o} \mid \mathbf{o} \text{ é densamente acessível por } \mathbf{x}, \text{ com relação a } \epsilon \text{ e } m_{pts}\}$.

2.2.4.2 OPTICS

Conjuntos de dados reais em sua maioria contêm estruturas de grupos com diferentes níveis de densidade em diferentes regiões do espaço (KRIEGEL *et al.*, 2011). Nesse caso, algoritmos de agrupamento baseados em densidade (*e.g.*, *DBSCAN*) que possuem apenas um parâmetro global de densidade (ϵ) não conseguem capturar a real estrutura dos grupos presentes nos dados, pois existe a possibilidade dos grupos conterem diferentes níveis de densidade (ANKERST *et al.*, 1999a).

Existem algumas alternativas para contornar esse problema. A primeira seria utilizar algum algoritmo hierárquico, tal como: *single-linkage*, por exemplo. O problema dessa abordagem é que existem algumas limitações em relação a esse algoritmo. Uma delas é a sensibilidade a ruídos (JAIN; DUBES, 1988). Além disso, hierarquias representadas por dendrograma que contenham centenas ou milhares de objetos podem ser complicadas de interpretá-las (ANKERST *et al.*, 1999b). A segunda alternativa seria utilizar *DBSCAN* com diferentes configurações de parâmetros de níveis de densidade para tentar encontrar a real estrutura dos grupos. Entretanto, essa abordagem se torna custosa em termos computacionais, pois é preciso definir quais configurações testar entre infinitas possibilidades (ANKERST *et al.*, 1999a).

Assim, para lidar com problemas dessa natureza, *Ordering Points To Identify the Clustering Structure (OPTICS)* foi proposto em Ankerst *et al.* (1999a). *OPTICS* é um algoritmo de agrupamento de dados baseado em densidade que pode ser considerado um aprimoramento de *DBSCAN* (ESTER *et al.*, 1996). Nesse algoritmo foi introduzido uma noção de ordenação de grupos baseados em densidade tal que, a partir de um valor constante de m_{pts} , grupos com alta densidade (i.e. valor baixo de ϵ) são contidos (ou “aninhados”) em conjuntos conectados densamente que tenham baixa densidade (i.e. valor alto de ϵ). Na Figura 4 é possível ilustrar essa característica, em que $\mathbf{C}_1$ e $\mathbf{C}_2$ são grupos baseados em densidade com relação a $\epsilon_1 < \epsilon_2$ e $\mathbf{C}$ é um grupo também baseado em densidade, mas com relação a ϵ_1 , que contém completamente os grupos $\mathbf{C}_1$ e $\mathbf{C}_2$.

Figura 4 – Ilustração de grupos “aninhados” baseados em densidade (Adaptado de (ANKERST *et al.*, 1999a)).

OPTICS funciona como uma extensão de *DBSCAN* para infinitos valores de parâmetros de distâncias denotados por ϵ_i , que são menores ou iguais ao limiar (raio) máximo ϵ ($0 \leq \epsilon_i \leq \epsilon$). A diferença é que os membros de grupos não são definidos explicitamente. Ao invés disso, o algoritmo *OPTICS* fornece uma ordenação de estimativas de densidade (distâncias ϵ_i), que facilita a escolha de um parâmetro global de densidade (ϵ), para que os membros dos grupos possam ser recuperados explicitamente pelo algoritmo *DBSCAN* (ANKERST *et al.*, 1999a). Especificamente, as definições de *OPTICS* descritas a seguir, assim com são apresentadas em Ankerst *et al.* (1999a).

Definição 1. (*distância core* de um objeto $\mathbf{x}$): Considere um objeto $\mathbf{x}$ pertencente a um conjunto de dados $\mathbf{X}$, um valor de distância (raio) ϵ , $N_\epsilon(\mathbf{x})$ a ϵ -vizinhança de $\mathbf{x}$, m_{pts} um número natural e $m_{pts_distancia}(\mathbf{x})$ seja a distância a partir de $\mathbf{x}$ ao seu m_{pts} vizinho mais próximo. Então, a *distância core* de $\mathbf{x}$ é definida pela Equação 2.16.

$$distancia_core_{\epsilon, m_{pts}}(\mathbf{x}) = \begin{cases} INDEFINIDO, & \text{se } |N_\epsilon(\mathbf{x})| < m_{pts} \\ m_{pts_distancia}(\mathbf{x}), & \text{caso contrário} \end{cases} \quad (2.16)$$

A *distância core* de um objeto $\mathbf{x}$ diz respeito a menor distância (definida como ϵ') entre $\mathbf{x}$ e algum objeto que pertence a ϵ -vizinhança de $\mathbf{x}$ de maneira que $\mathbf{x}$ pode ser um objeto *core* com respeito a ϵ' .

Definição 2. (*distância de acessibilidade de um objeto $\mathbf{x}$ com respeito a um objeto $\mathbf{o}$*): Considere objetos $\mathbf{x}$ e $\mathbf{o}$ de um conjunto de dados $\mathbf{X}$, seja $|N_\epsilon(\mathbf{o})|$ a ϵ -vizinhança de $\mathbf{o}$, e m_{pts} um número natural. Então a distância de acessibilidade de $\mathbf{x}$ em relação a $\mathbf{o}$ é definida como

$$distancia_acessibilidade_{\epsilon, m_{pts}}(\mathbf{x}, \mathbf{o}) = \begin{cases} INDEFINIDO, & \text{se } |N_\epsilon(\mathbf{o})| < m_{pts} \\ \max(distancia_core(\mathbf{o}), d(\mathbf{o}, \mathbf{x})), & \text{caso contrário} \end{cases} \quad (2.17)$$

A distância de acessibilidade de um objeto $\mathbf{x}$ em relação a um objeto $\mathbf{o}$ é a menor distância, tal que $\mathbf{x}$ seja diretamente acessível por $\mathbf{o}$ se $\mathbf{o}$ é um objeto *core*.

Com base nas definições, *OPTICS* mantém um arquivo de ordenação com as distâncias *core* e de acessibilidade dos objetos de um conjunto de dados. Essas informações correspondem à estrutura implícita dos grupos encontrados.

Além disso, *OPTICS* conta com uma ferramenta de visualização das estruturas de grupos encontradas, conhecida como *reachability plot* (ANKERST *et al.*, 1999a).

2.2.5 Validação de Agrupamento de Dados

Mensurar a qualidade de agrupamentos de dados feitos por diferentes algoritmos ou por um único algoritmo submetido a diferentes parametrizações sobre um mesmo conjunto de dados é uma prática comum na análise de grupos, pois os resultados em sua maioria são distintos devido à intuição que cada algoritmo contém em seus fundamentos (JAIN; DUBES, 1988; XU; WUNSCH, 2009).

Além da intuição de cada algoritmo, as características ou tipos dos dados são fatores cruciais para determinados algoritmos. Por exemplo, o método de *Ward* é um forte candidato para dados que possuem estruturas de grupos globulares ou hiperesféricos. Já para dados que possuem estruturas de grupos com formas arbitrárias, este método pode não ser tão interessante (HAN; KAMBER, 2000; HAN, 2005; HAN *et al.*, 2011).

Isto posto, técnicas de validação de *Agrupamento de Dados* são empregadas para selecionar quais métodos de agrupamento de dados são mais adequados para certos tipos de dados e também demonstrar se existem ou não estruturas de grupos nesses dados (JAIN; DUBES, 1988; AGGARWAL; REDDY, 2013).

Assim como dito anteriormente, é comum encontrar na literatura três tipos de critérios de validação de agrupamento de dados. Entretanto, esta seção descreve apenas alguns dos clássicos critérios externos de validação. Especificamente, os critérios externos avaliam a compatibilidade entre duas partições. Uma dessas partições é denominada como referência, ou seja, uma partição que foi predefinida por um especialista de domínio dos dados. A outra partição corresponde ao resultado do algoritmo de agrupamento de dados (XU; WUNSCH, 2005).

Nesse contexto, seja $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ um conjunto de dados com n objetos e duas partições $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_r\}$ uma partição de referência composta por r grupos e $\mathbf{Q} = \{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_s\}$ o resultado de um agrupamento de dados (partição) obtido por algum algoritmo de agrupamento, composto por s grupos. Assim, algumas grandezas definidas abaixo são utilizadas por muitos critérios de validação:

- a : Número de pares de objetos em $\mathbf{X}$ que pertencem ao mesmo grupo em $\mathbf{P}$ e ao mesmo grupo em $\mathbf{Q}$.

- b : Número de pares de objetos em $\mathbf{X}$ que pertencem ao mesmo grupo em $\mathbf{P}$ e a grupos distintos em $\mathbf{Q}$.
- c : Número de pares de objetos em $\mathbf{X}$ que pertencem a grupos distintos em $\mathbf{P}$ e ao mesmo grupo em $\mathbf{Q}$.
- d : Número de pares de objetos em $\mathbf{X}$ que pertencem a grupos distintos em $\mathbf{P}$ e a grupos distintos em $\mathbf{Q}$.

Pelas definições, os termos a e d são considerados como termos de concordância entre pares, já os termos b e c são pares de discordância entre pares (JAIN; DUBES, 1988).

2.2.5.1 Índice Rand

O Índice *Rand* ou medida *Rand* é uma medida clássica para indicar a semelhança entre duas partições (RAND, 1971). Matematicamente, esse índice, apresentado pela Equação 2.18, tem em seu numerador a soma dos termos concordantes a e b e em seu denominador a soma de todos os termos que corresponde ao número total de pares de objetos em $\mathbf{X}$, i.e., $n(n-1)/2 = a + b + c + d$. Diante disso, o índice *Rand* mede a frequência de ocorrência de pares concordantes sobre o número total de pares (RAND, 1971; XU; WUNSCH, 2009).

$$R = \frac{a + b}{a + b + c + d} \quad (2.18)$$

Adicionalmente, na Equação 2.18, R possui o intervalo de $[0,1]$. Portanto, um *Agrupamento de Dados Q* é considerado completamente inconsistente se $R = 0$, i.e., $a = d = 0$. Ao passo que, se $R = 1$, o *Agrupamento de Dados Q* concorda exatamente com a partição de referência $\mathbf{P}$, ou seja, $c = b = 0$ e $\mathbf{P} \equiv \mathbf{Q}$ (VENDRAMIN *et al.*, 2010).

2.2.5.2 Índice Jaccard

Diferente do índice *Rand*, o índice *Jaccard* é uma medida que desconsidera a importância de um dos termos concordantes, i.e., o termo d (JACCARD, 1901). Essa desconsideração se dá se não fizer sentido a comparação de pares de objetos que são separados tanto na partição de referência, quanto na partição resultante, pois esse índice procura interpretar os grupos de objetos unidos, e a separação entre esses objetos é apenas uma consequência do processo de agrupamento dos dados (DENOEUDE *et al.*, 2005; VENDRAMIN *et al.*, 2010).

$$J = \frac{a}{a + b + c} \quad (2.19)$$

Como pode ser visto, na Equação 2.19, J é referente à contagem de pares concordantes em a dividido pela soma dos termos que têm pelo menos um par de objetos dos dois pares concordantes (DENOEUDE *et al.*, 2005).

2.2.5.3 Índice Rand Ajustado

Os índices *Rand* e *Jaccard* possuem algumas limitações. Uma delas é a não correção por mudança, i.e., dando duas partições completamente aleatórias, o valor do índice de validação esperado não é igual a zero (XU; WUNSCH, 2009). Nesse contexto, o índice *Rand Ajustado* (*ARI* sigla em inglês) resolve essa limitação normalizando o valor esperado para 0 quando as partições são selecionadas aleatoriamente (por mudança) ou para 1 quando existe uma perfeita concordância entre as partições comparadas (HUBERT; ARABIE, 1985; VENDRAMIN *et al.*, 2010). *ARI* é apresentado na Equação 2.20:

$$ARI = \frac{a - \frac{(a+c)(a+b)}{M}}{\frac{(a+c)(a+b)}{M} - \frac{(a+c)(a+b)}{M}} \quad (2.20)$$

em que $M = n(n-1)/2 = a + b + c + d$, i.e., corresponde a todos os pares possíveis de objetos.

2.3 Considerações Finais

Neste capítulo foram abordados conceitos introdutórios de descoberta de conhecimento em bases de dados *KDD* que são de suma importância para este projeto de pesquisa, especificamente, foram introduzidos algumas formas de tratamento e limpeza de dados e os processos de agrupamento de dados. Além disso, foram abordadas algumas medidas de proximidades, em que o uso das mesmas é bastante comum por diversas técnicas de agrupamento de dados, inclusive nas técnicas investigadas e propostas neste trabalho. Foram discutidos alguns algoritmos de agrupamento de dados, especificamente, algoritmos hierárquicos e baseados em densidade. Adicionalmente, cada algoritmo abordado, de alguma forma, possui uma relação com o método investigado neste trabalho de pesquisa, i.e., *HDBSCAN**. Por fim, foram apresentados alguns critérios externos de validação. Alguns desses critérios foram utilizados para a validação de agrupamentos realizados pelas técnicas investigadas e propostas neste projeto de pesquisa.

FRAMEWORK HDBSCAN*

*HDBSCAN** é um *framework* sofisticado de agrupamento não supervisionado e semissupervisionado de dados baseado em densidade, detecção *global* e *local* de *outliers* e visualização dos resultados (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Em suma, *HDBSCAN** tem como inspiração o modelo de contorno de densidade de grupos/árvore (*density-contour clusters* e *density-contour tree*, respectivamente) proposto por *Hartigan* (HARTIGAN, 1975).

Dessa forma, *HDBSCAN**, de maneira praticamente não parametrizada, fornece uma hierarquia de grupos composta por todos os possíveis níveis de densidade nos dados para infinitos limiares de estimativa e também uma árvore de grupos simplificada baseada no conceito de *grupos rígidos* proposto por *Hartigan* (HARTIGAN, 1975).

A partir da hierarquia de *HDBSCAN**, é possível realizar algumas tarefas de pós-processamento de dados (como pode ser visto na Figura 5). Uma delas é a extração de uma partição *flat* composta pelos grupos mais significativos dentro da hierarquia tanto no contexto de agrupamento não supervisionado quanto de agrupamento semissupervisionado de dados.

Diferente de técnicas tradicionais de cortes globais horizontais em níveis hierárquicos para a obtenção de uma partição composta por um determinado número de grupos, *HDBSCAN** utiliza uma estratégia de otimização para determinar pontos de cortes locais nessa hierarquia. Essa estratégia é importante, pois nem sempre um corte global na hierarquia irá representar significativamente a verdadeira estrutura dos dados, uma vez que é difícil determinar esse ponto ótimo de corte na hierarquia de agrupamento.

No contexto de agrupamento não supervisionado de dados, a estratégia de otimização encontra um agrupamento não-hierárquico (particional) ótimo de dados composto pelos grupos significativos a partir hierarquia com base na estabilidade de cada grupo que a compõe. Já no âmbito de agrupamento semissupervisionado de dados, ou seja, quando existem algumas informações (rótulos ou restrições) prévias sobre alguns dados, a estratégia de otimização é acrescida de conceitos de violação e satisfação de restrições (*should-link* e *cannot-link*),

juntamente com a medida de estabilidade.

Como pode ser visto na Figura 5, outra tarefa de pós-processamento que o *framework HDBSCAN** fornece é a de detecção *global* e *local* de *outliers* em uma única definição, conhecido como *Global-Local Outlier Scores from Hierarchies (GLOSH)*. Por fim, *HDBSCAN** provê algumas formas de visualização dos resultados, tais como: dendrogramas (JAIN; DUBES, 1988), *reachability plot* de *OPTICS* (ANKERST *et al.*, 1999a), *silhuete plot* (GUPTA *et al.*, 2010) e árvore simplificada de grupos (CAMPELLO *et al.*, 2015).

Figura 5 – Framework HDBSCAN* (Adaptado de (CAMPELLO *et al.*, 2015)).

Este capítulo tem a seguinte estrutura: na Seção 3.1, assim como apresentadas em Campello *et al.* (2013a), Campello *et al.* (2015), são apresentadas as definições gerais de *HDBSCAN**, incluindo o processo de agrupamento não supervisionado de dados, i.e., construção de hierarquia baseada em densidade; o método de extração de grupos significativos dos dados a partir de hierarquia de *HDBSCAN**, denominado *FOSC*, é apresentado na Seção 3.2; os conceitos do método de detecção de *outliers*, denominado *GLOSH*, são apresentados na Seção 3.3; por fim, na Seção 3.4 são apresentadas as considerações finais deste capítulo.

3.1 Definições

Segundo as definições apresentadas previamente em Campello *et al.* (2013a), *HDBSCAN** pode ser considerado uma melhoria conceitual do algoritmo *OPTICS* (descrito na Subseção 2.2.4.2). Portanto, assim como apresentado em Campello *et al.* (2013a), nesta seção são descritas todas as definições de *HDBSCAN**.

3.1.1 DBSCAN*

Seja $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ um conjunto de objetos multidimensionais composto por n objetos e, apenas conceitualmente, uma $\mathbf{D}$ uma matriz de proximidade $n \times n$ que contém as

distâncias entre pares $d(\mathbf{x}_p, \mathbf{x}_q)$, $\mathbf{x}_p, \mathbf{x}_q \in \mathbf{X}$. Complementarmente, a matriz $\mathbf{D}$ é apenas utilizada quando cabe na memória principal ou é fornecida previamente por algum mecanismo externo ao agrupamento (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Definição 1: (*Objeto Core*): Um objeto $\mathbf{x}_p$ é considerado como objeto *core*, com relação aos parâmetros ε e m_{pts} , se em sua $\varepsilon_{vizinhança}$ contém pelo menos m_{pts} objetos, isto é, se $|N_\varepsilon(\mathbf{x}_p)| \geq m_{pts}$, em que $N_\varepsilon(\mathbf{x}_p) = \{\mathbf{x} \in \mathbf{X} \mid d(\mathbf{x}, \mathbf{x}_p) \leq \varepsilon\}$ e $|\cdot|$ denota a cardinalidade. Um objeto é considerado *ruído* se ele não for um objeto *core*.

Definição 2: ($\varepsilon_{acessível}$): Dois objetos *core* $\mathbf{x}_p$ e $\mathbf{x}_q$ são $\varepsilon_{acessíveis}$, com relação a ε e m_{pts} se $\mathbf{x}_p \in N_\varepsilon(\mathbf{x}_q)$ e $\mathbf{x}_q \in N_\varepsilon(\mathbf{x}_p)$.

Definição 3: (*Densamente Conectado*): Dois objetos *core* $\mathbf{x}_p$ e $\mathbf{x}_q$ são densamente conectados, com relação a ε e m_{pts} se eles são direto ou transitivamente $\varepsilon_{acessível}$.

Definição 4: (*Grupo*): Um grupo $\mathbf{C}$, com relação a ε e m_{pts} é um subconjunto máximo não vazio de $\mathbf{X}$, tal que todo par de objetos em $\mathbf{C}$ é densamente conectado.

Similar às definições de *DBSCAN* introduzidas em Ester *et al.* (1996), *DBSCAN** considera que todos os objetos de um conjunto de dados $\mathbf{X}$ multidimensional sejam identificados como vértices de um grafo, tal que cada par de vértices é adjacente, se e somente se, os objetos envolvidos são $\varepsilon_{acessíveis}$ com relação aos parâmetros ε e m_{pts} , e os objetos que não são denominados *core* são denotados como *ruídos* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Diferentemente de *DBSCAN* (ESTER *et al.*, 1996), que define o conceito de objetos *border*, que, por sua vez, são objetos que pertencem a $\varepsilon_{vizinhança}$ de um ou mais objetos *core*, e consequentemente, pertencem ao mesmo componente conectado (grupo) de um desses objetos *core* correspondentes. Por outro lado, *DBSCAN**, reformulado em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), elimina esse conceito de objeto *border*, porque contém uma visão mais estatística do conceito de grupos formados por componentes conectados em um conjunto de níveis de densidade (HARTIGAN, 1975; CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

3.1.2 Definições de *HDBSCAN**

A versão hierárquica de *DBSCAN** denominada como *HDBSCAN** e, como citado anteriormente, esse algoritmo pode ser considerado uma melhoria do algoritmo *OPTICS* (ANKERST *et al.*, 1999a). Tal motivo está relacionado às definições de distância *core* e distância de acessibilidade definidas por este último (CAMPELLO *et al.*, 2015).

Embora a distância *core* seja definida igualmente por ambos os métodos, ao contrário do que é definido em *OPTICS*, a distância de acessibilidade é reformulada como uma distância simétrica entre os objetos (*distância de acessibilidade mútua*) do conjunto de dados no método

*HDBSCAN**, seguindo a definição apresentada em (LELIS; SANDER, 2009; CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Definição 5: (*Distância Core*): A *distância core* de um objeto $\mathbf{x}_p \in X$ com relação a $m_{pts}, d_{core}(x_p)$ é a distância de $\mathbf{x}_p$ para seu m_{pts} -vizinho mais próximo (incluindo $\mathbf{x}_p$).

Nesse contexto, a distância *core* de um objeto $\mathbf{x}_p$ representa o raio mínimo ε em que sua vizinhança contenha pelo menos um número mínimo de objetos, i.e., $|N_\varepsilon(\mathbf{x}_p)| \geq m_{pts}$ (**Definição 1**).

Definição 6: (*Distância de Acessibilidade Mútua*): A distância de acessibilidade mútua entre dois objetos $\mathbf{x}_p$ e $\mathbf{x}_q$ em $\mathbf{X}$, com relação a m_{pts} é definida como:

$$d_{mreach}(\mathbf{x}_p, \mathbf{x}_q) = \max\{d_{core}(\mathbf{x}_p), d_{core}(\mathbf{x}_q), d(\mathbf{x}_p, \mathbf{x}_q)\}. \quad (3.1)$$

A distância de acessibilidade mútua (d_{mreach}) representa o raio mínimo ε entre os objetos $\mathbf{x}_p$ e $\mathbf{x}_q$, tal que ambos sejam $\varepsilon_{acessível}$. Essa característica garante que seja possível definir apenas conceitualmente um grafo de proximidade no espaço transformado (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Definição 7: (*Grafo de Distância de Acessibilidade Mútua*): É um grafo completo, $G_{m_{pts}}$, em que os objetos de $\mathbf{X}$ são vértices e cada aresta contém um peso que é representado pela distância de acessibilidade mútua entre o respectivo par de objetos.

Seja $G_{m_{pts}, \varepsilon} \subseteq G_{m_{pts}}$ um grafo obtido a partir da remoção de todas as arestas de $G_{m_{pts}}$ que tenham pesos maiores que algum valor de raio ε . Então, a partir das definições de *DBSCAN** baseado nos parâmetros m_{pts} e ε , grupos são definidos como componentes conectados de objetos *core* em $G_{m_{pts}, \varepsilon}$, e os demais objetos são considerados como ruídos (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Diante disso, é possível produzir agrupamentos aninhados de *DBSCAN** para $\varepsilon \in [0, \infty)$ por meio da remoção iterativa de arestas em ordem decrescente de peso a partir de $G_{m_{pts}}$ (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Esse processo iterativo de remoção de arestas de maior peso do que um valor de raio ε decrescente assemelha-se ao que é feito pelo algoritmo *Single-Linkage* em sua formulação baseada em grafos (JAIN; DUBES, 1988). Então, assim como é descrito em Campello *et al.* (2013a) e Campello *et al.* (2015), a *Proposição 8*, juntamente com o *Colorário 9*, descrevem o relacionamento formal entre *DBSCAN** e *Single-Linkage* em um espaço transformado pela distância de acessibilidade mútua.

Proposição 8: Seja $\mathbf{X}$ um conjunto de dados composto por n objetos e descrito em um espaço métrico por pares de distância $n \times n$. O agrupamento desses dados obtido por *DBSCAN** com relação a m_{pts} e algum valor de ε é idêntico a um agrupamento obtido primeiramente pela execução de *Single-Linkage* sobre o espaço transformado pela distância de acessibilidade mútua, posteriormente, realizando um corte horizontal em um nível ε da escala do

dendrograma resultante e considerando todos os *singletons* com $d_{core}(\mathbf{x}_p) > \varepsilon$ como uma classe única representando ruídos.

Colorário 9: Para $m_{pts} = 1$ ou $m_{pts} = 2$, $DBSCAN^*$ com relação a m_{pts} e ε é equivalente a um corte horizontal feito no nível ε do dendrograma de *Single-Linkage* em um espaço original de distância $d(\cdot, \cdot)$, provendo que todos os *singletons* são grupos únicos quando $m_{pts} = 1$ e são considerados como ruído quando $m_{pts} = 2$.

Prova: Pela **Definição 5**, a distância *core* de um objeto $\mathbf{x}_p$ é igual a zero quando $m_{pts} = 1$ e é igual à distância para seu vizinho mais próximo (excluindo o objeto $\mathbf{x}_p$ em questão) quando $m_{pts} = 2$. Com relação a $m_{pts} = 1$, qualquer objeto não conectado em qualquer nível ε é um grupo unitário nesse nível, pois esse objeto é necessariamente um objeto *core*. Para $m_{pts} = 2$, de acordo com as Definições 5 e 6 a $d_{core}(\mathbf{x}_p)$ é igual a $d_{mreach}(\mathbf{x}_p, \mathbf{x}_q)$ mínima entre $\mathbf{x}_p$ e qualquer outro $\mathbf{x}_q$. Em outras palavras, para um dado corte no nível ε , qualquer objeto não conectado será um objeto ruído naquele nível. Adicionalmente, para ambos os valores de $m_{pts} = 1$ e $m_{pts} = 2$, por meio da Definição 6 a $d_{mreach}(\mathbf{x}_p, \mathbf{x}_q) = d(\mathbf{x}_p, \mathbf{x}_q)$, ou seja, o espaço original e transformado dos dados são os mesmos para esses valores de m_{pts} . De acordo com a Proposição 8 descrita anteriormente, a equivalência em $DBSCAN^*$ e *Single-Linkage* é válida mesmo para o espaço original dos dados, i.e., quando $m_{pts} = 1$ ou $m_{pts} = 2$. Por outro lado, o uso de valores relativamente altos de m_{pts} é uma forte estratégia para diminuir a influência do efeito em cadeia que *Single-Linkage* contém pelo alongamento das distâncias entre os objetos que se encontram nas regiões mais dispersas do espaço de dados (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

3.1.3 Algoritmo HDBSCAN*

Como pode ser visto na Proposição 8, a implementação da abordagem hierárquica de $DBSCAN^*$ ($HDBSCAN^*$) pode ser feita pela utilização do algoritmo hierárquico *Single-Linkage* em um espaço transformado (baseando-se na distância de acessibilidade mútua) (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Entretanto, a cada nível hierárquico construído, é necessário atentar-se e definir se os objetos que não são conectados, ou seja, objetos isolados sejam ruídos ou objetos *core* em determinado nível da hierarquia (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Dessa forma, $HDBSCAN^*$ segue um rápido método de construção de hierarquias a partir de um algoritmo divisivo (*top-down*) baseado em *Árvore Geradora Mínima* (ou *Minimum Spanning Tree (MST)*) (JAIN; DUBES, 1988; CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Nesse contexto, uma *MST* corresponde a um subgrafo acíclico composto por um conjunto de vértices conectados por arestas ponderadas por um determinado peso e que a soma total dos pesos dessas arestas seja mínima (GROSS; YELLEN, 2005).

Outrossim, a proposta do algoritmo divisivo baseado em *MST* de *HDBSCAN** é, a partir da remoção iterativa de arestas de maior peso (distância de acessibilidade mútua) em ordem decrescente, construir cada nível da hierarquia que é composta por componentes conectados (grupos) e objetos isolados denominados ruídos (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Ademais, para representar os objetos ruídos em cada nível da hierarquia, adiciona-se “*self-edges*” compostos pela distância *core* de cada vértice (objeto) da *MST*, resultando em uma *MST* estendida (MST_{est}) (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Assim, as arestas que correspondem aos “*self-edges*” também são consideradas no processo de remoção para a construção de cada nível da hierarquia (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

No Algoritmo 1 são apresentados os passos de construção de hierarquia de *HDBSCAN**. Basicamente, esse algoritmo tem como entrada $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, que é um conjunto de dados multidimensional composto por n objetos ou a matriz de proximidades $\mathbf{D}$, que representa a comparação entre os pares de objetos de $\mathbf{X}$, e m_{pts} , que indica o número mínimo de objetos em uma determinada vizinhança. Pelo Algoritmo 1 é possível obter todos os agrupamentos de dados de *DBSCAN** para infinitos valores de ϵ em forma de hierarquia, incluindo os objetos que são considerados como ruídos pelo modelo.

Tecnicamente, na Linha 1, calcula-se a distância *core* de cada objeto em $\mathbf{X}$ (ou a partir de $\mathbf{D}$). Na Linha 2, a *MST* baseada na distância de acessibilidade mútua é obtida a partir de $G_{m_{pts}}$. Na Linha 3, a *MST* para MST_{est} é estendida considerando que cada vértice contenha um “*self-loop*”, ou seja, aresta composta pela sua respectiva distância *core* como peso. Na Linha 4, as arestas de MST_{est} são ordenadas em ordem decrescente de peso. Então, a hierarquia de grupos é construída a partir de MST_{est} pelo laço de repetição, representado na Linha 5. Especificamente, enquanto existem arestas em MST_{est} , na Linha 6, arestas de maior peso são removidas uma a uma (considerando os possíveis empates nos pesos) e, na Linha 7, a escala do dendrograma a ser criado é definida com o peso da aresta corrente na remoção. Na Linha 8, após a remoção de aresta(s) de maior(es) peso(s), rótulos de grupos são determinados aos membros (objetos) de cada componente conectado (denominado como *CC*) que contém os vértices finais das arestas removidas. Neste ponto, se um determinado componente conectado *CC* contém pelo menos uma aresta, então é determinado que esse componente seja rotulado com um rótulo de novo grupo (Linha 10). Caso contrário, esse componente recebe rótulo nulo, pois o mesmo é considerado como ruído (Linha 12).

3.1.4 Simplificação de Hierarquia

Dendrogramas são comumente utilizados no contexto de análise de resultados de agrupamentos hierárquicos (JAIN; DUBES, 1988). Entretanto, dendrogramas tradicionais são considerados ferramentas que dificultam, em determinado contexto, a interpretação de resultados

Algoritmo 1 – HDBSCAN* Passo Principal

Entrada: Seja $\mathbf{X}$ um conjunto de dados multidimensional (ou $\mathbf{D}$ a matriz de proximidade entre os objetos de $\mathbf{X}$) e m_{pts} o número mínimo de objetos em uma determinada vizinhança.

Saída: Hierarquia de HDBSCAN*

- 1: Calcula a distância *core* (utilizando m_{pts}) para todos objetos do conjunto de dados $\mathbf{X}$.
- 2: Calcula a *MST* de um grafo $G_{m_{pts}}$ (baseado na distância de acessibilidade mútua).
- 3: Estende a *MST* para obter MST_{est} , adicionando para cada vértice (objeto) um “*self-loop*” com a distância *core* do objeto correspondente.
- 4: Ordena MST_{est} em ordem decrescente de peso.
- 5: Para a raiz da árvore, atribua a todos os objetos o mesmo rótulo.
- 6: **enquanto** Existem arestas em MST_{est} **faça**
- 7: Remove-se arestas de MST_{est} de maior pesos.
- 8: Antes da remoção, define-se a escala do dendrograma com o valor corrente do nível hierárquico com o peso da aresta a ser removida.
- 9: Após a remoção, atribui-se rótulos para os componentes conectados (*CC*) que contêm os vértices finais da aresta removida para obter o próximo nível hierárquico:
- 10: **se** componente conectado em *CC* contém pelo menos uma aresta **então**
- 11: Atribui-se um rótulo de novo grupo ao componente.
- 12: **senão**
- 13: Atribui-se ao componente um rótulo nulo (*ruído*).
- 14: **fim se**
- 15: **fim enquanto**

(hierarquias de grupos) de conjuntos de dados que contêm dados ruidosos (CAMPELLO *et al.*, 2015).

Ante o exposto, em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015) é apresentada uma importante maneira para simplificação de hierarquia de HDBSCAN* por meio da extração de uma árvore de grupos sumarizada composta pelos grupos “significativos” a partir do dendrograma. Essa simplificação tem como base o conceito de *grupos rígidos* definidos por Hartigan (HARTIGAN, 1975). Especificamente, durante a construção de hierarquias de HDBSCAN*, os autores desse algoritmo observaram que componentes encontrados a cada nível hierárquico podem sofrer apenas três evoluções distintas quando ε é decrementado no contexto de HDBSCAN* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

1. O componente conectado pode encolher, ou seja, o número de objetos que o pertence reduz à medida que os níveis hierárquicos são construídos, mas ainda continua conectado.
2. O componente pode se dividir em subcomponentes menores.
3. O componente pode desaparecer à medida que a hierarquia é construída.

Nesse contexto, a simplificação de hierarquia de HDBSCAN* se dá pela consideração apenas dos níveis hierárquicos em que novos grupos aparecem segundo uma “verdadeira” di-

visão de um grupo já existente na árvore de grupos ou em níveis que grupos desaparecem completamente (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Como é definido por *HDBSCAN**, objetos que são considerados como ruído não constituem grupos válidos na hierarquia de grupos. Esse fato garante que, quando esse tipo de objetos é removido (desconectado) de algum grupo (componente), esse grupo continua com o mesmo rótulo, ou seja, esse grupo apenas encolhe em relação ao número de objetos que o pertence e, portanto, não sofre uma verdadeira divisão.

Em outras palavras, um determinado grupo só será verdadeiramente dividido quando os seus subcomponentes forem grupos válidos sobre alguma perspectiva (e.g., número mínimo de objetos pertencentes a um grupo (m_{clSize})) (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Em algumas aplicações reais, mesmo que os dados sejam desconhecidos, para alguns especialistas há a necessidade de determinar o tamanho mínimo dos grupos que os mesmos gostariam de encontrar nas estruturas dos dados que analisam (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Dessa forma, *HDBSCAN** provê o parâmetro opcional m_{clSize} para determinar essa característica (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Especificamente, a partir do momento que $m_{clSize} \geq 1$, esse fato garante que apenas os componentes que contenham um número de objetos igual ou maior do m_{clSize} sejam reportados e os componentes que não obedecem essa condição sejam denominados como componentes desconectados de grupos, e, portanto, não sejam considerados como uma verdadeira divisão (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

À vista disso, o Algoritmo 2 é um complemento ao Algoritmo 1 (a partir da Linha 8) que refere-se ao uso do parâmetro m_{clSize} . Note que na Linha 1 do Algoritmo 2, cada subcomponente conectado gerado a partir de um grupo existente que possui algum dos vértices finais presentes nas arestas removidas é processado e denominado como grupo “ilegítimo” ou grupo válido à medida que arestas de maior peso são removidas da MST_{est} . Um grupo ilegítimo caracteriza um subcomponente que tem cardinalidade inferior a m_{clSize} , então os objetos desse subcomponente são denominados como ruído (rótulo nulo). Já um grupo válido corresponde a um subcomponente cuja cardinalidade seja superior ou igual a m_{clSize} (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

O parâmetro opcional m_{clSize} pode ser utilizado, em conformidade com m_{pts} , como um fator de suavização da árvore de grupos resultante como uma estimativa de densidade (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Simultaneamente, em Ester *et al.* (1996) o parâmetro m_{pts} controla diretamente o tamanho mínimo dos grupos de maneira que os objetos *borda* são atribuídos ao mesmo grupo de um dos seus objetos *core*, resultando em grupos que tenham mais de m_{pts} objetos (CAMPELLO *et al.*, 2015).

Nessa perspectiva, para tornar *HDBSCAN** mais similar à versão original de *DBSCAN*

Algoritmo 2 – HDBSCAN* (referente à Linha 8 do Algoritmo 1) utilizando o parâmetro $m_{clSize} \geq 1$

Depois de cada remoção (para obter o próximo nível hierárquico), processa-se um grupo de cada vez que contenha apenas a(s) aresta(s) removida(s), através de sua reetiquetagem, resultando em subcomponentes conectados:

- Rotula-se subcomponentes ilegítimos como o ruído, atribuindo-lhes o rótulo **nulo**. Se todos os subcomponentes de um grupo são ilegítimos, em seguida, o grupo desaparece.
 - Caso contrário, se um único subcomponente de um grupo não é ilegítimo, mantenha seu rótulo de grupo original (grupo que apenas encolheu).
 - Caso contrário, se dois ou mais componentes de um grupos não são ilegítimos, atribua novos rótulos de grupos para cada um deles.
-

(ESTER *et al.*, 1996) e simplificar o uso do parâmetro m_{clSize} , utiliza-se $m_{clSize} = m_{pts}$ a fim de tornar m_{pts} um parâmetro único de fator de suavização da estimativa de densidade e, ao mesmo tempo, um parâmetro explícito para controlar o tamanho mínimo dos grupos (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

3.1.5 Complexidade Computacional

A análise de complexidade assintótica de tempo e espaço do *HDBSCAN** descrita pelos Algoritmos 1 e 2 é feita a partir de dois cenários: quando é utilizado como entrada um conjunto de dados $\mathbf{X}$ ou quando é utilizada como entrada a matriz de proximidade $\mathbf{D}$ (contexto de agrupamento de dados relacional (JAIN; DUBES, 1988)) (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Adicionalmente, considera-se que em situações reais de aplicação o parâmetro m_{pts} seja “muito” menor que o número de objetos em $\mathbf{X}$, i.e., $m_{pts} \ll |\mathbf{X}|$.

Primeiro cenário: Quando $\mathbf{X}$ é considerado pelo algoritmo como entrada, a distância $d(.,.)$ entre qualquer par de objeto em $\mathbf{X}$, a complexidade de tempo é dada por $O(a)$, em que a corresponde ao número de atributos dos objetos em $\mathbf{X}$. Nesse sentido, a complexidade de tempo para calcular a distância *core* de todos os objetos do conjunto de dados $\mathbf{X}$ é de $O(an^2)$, uma vez que é necessário realizar consultas de m_{pts} -vizinhos mais próximos para cada objeto. Como sugerido em Campello *et al.* (2013a) e Campello *et al.* (2015), a construção de uma *MST* baseada na distância de acessibilidade mútua pode ser feita por uma implementação do algoritmo de *Prim* (PRIM, 1957). A implementação desse algoritmo pode ser realizada de duas maneiras: considerando o grafo completo composto por $m = n(n-1)/2$ arestas baseadas na distância de acessibilidade mútua ou pelo cálculo de distância de acessibilidade mútua dos objetos em $\mathbf{X}$ por demanda. A partir do grafo completo, a complexidade de tempo para construir a *MST* é de $O(n^2 + m)$. Entretanto, não é necessário calcular o grafo completo explicitamente. Em vista disso, a obtenção por demanda da distância de acessibilidade entre os objetos faz com que a *MST* seja construída em $O(an^2)$ de tempo. Especificamente, a *MST* é composta por $n-1$ arestas, com

adição de n “self-edges”, i.e., $2n - 1$ arestas.

Para o processo de construção de hierarquia de *HDBSCAN** a partir da *MST*, é necessário ordenar suas arestas em ordem decrescente de peso (distância de acessibilidade mútua), procedimento que pode ser feito por $O(n \log n)$ de tempo. Posteriormente, a cada remoção de aresta de maior peso da *MST* para a construção de cada nível hierárquico, como descrito no Algoritmo 1 na Linha 8, é preciso rotular os subcomponentes que serão considerados como grupos. Neste ponto, os subcomponentes só são re-rotulados a cada nível quando os mesmos são parte de uma verdadeira divisão, ou seja, quando se tornam novos grupos. Os subcomponentes que apenas encolhem a cada nível construído nunca são rotulados novamente até que haja uma verdadeira divisão sobre eles. Ainda assim, o pior caso de todo o processo de rotulação dos níveis hierárquicos é de $O(n^2)$. Por conseguinte, a complexidade geral de tempo do algoritmo *HDBSCAN** neste cenário é de $O(an^2)$ (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

No que se refere à complexidade de espaço de *HDBSCAN** neste cenário, para armazenar o conjunto de dados $\mathbf{X}$ é requerido $O(an)$ de espaço. Para armazenar a distância *core* (calculada na Linha 1 do Algoritmo 1) é necessário $O(n)$. A MST_{est} , em um contexto que não é necessário explicitamente calcular o grafo completo de distância de acessibilidade mútua, requer $O(n)$ para armazenamento. Para a construção de cada nível hierárquico deste algoritmo é requerido apenas o armazenamento de informações do nível em questão, o que pode ser feito em $O(n)$ de espaço. Por fim, a complexidade geral de espaço desse algoritmo é de $O(an)$.

Segundo cenário: Quando a matriz de proximidade $\mathbf{D}$ é considerada como entrada, o que muda é a complexidade de acesso de qualquer distância $d(.,.)$, que, por sua vez, pode ser feita em $O(1)$. Diante disso, a dimensão dos dados não interfere na complexidade de tempo nesse cenário, assim, a complexidade geral de tempo desse algoritmo será de $O(n^2)$. Entretanto, a complexidade de espaço será de $O(n^2)$, pois é necessário manter a matriz de proximidade $\mathbf{D}$ em memória de trabalho.

3.2 Agrupamento não-hierárquico ótimo

O agrupamento hierárquico de dados é capaz de produzir uma gama de resultados em diferentes níveis de especificidade (JAIN; DUBES, 1988). Entretanto, a partir da hierarquia de grupos, uma partição composta por um determinado número de grupos mais significativos e não sobrepostos é requerida pelo usuário final em alguns cenários de aplicação (CAMPELLO *et al.*, 2015). O corte horizontal em um determinado nível de um dendrograma é uma estratégia para obtenção da partição. Entretanto, esse tipo de abordagem não expressa resultados satisfatórios no contexto de métodos de agrupamento de dados baseados em densidade, que é o caso de *HDBSCAN**, pois a escolha de um global, ou seja, um limiar global na escala do dendrograma pode causar a perda de grupos que contenham uma grande variação em sua densidade (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Dito isso, em [Campello et al. \(2013a\)](#) e [Campello et al. \(2015\)](#), é proposto um método de extração de grupos significativos a partir de uma hierarquia de *HDBSCAN**. Esse método baseia-se em um problema formal de otimização de maximização da qualidade geral de um conjunto de grupos extraídos por cortes locais na hierarquia de *HDBSCAN**. Para tal extração, [Campello et al. \(2013a\)](#) e [Campello et al. \(2015\)](#) definem o conceito de estabilidade de grupos que expressa a “linha de sobrevivência” de cada grupo na hierarquia de *HDBSCAN**. Esse método é utilizado tanto no contexto de agrupamento de dados não supervisionado quanto semi-supervisionado. Contudo, o foco deste trabalho está no contexto de agrupamento não supervisionado de dados.

3.2.1 Estabilidade de Grupo

Assim como citado anteriormente, *HDBSCAN** tem como inspiração o modelo de *Hartigan* ([HARTIGAN, 1975](#)). Logo, considere que um conjunto de objetos sejam representados por um único atributo expresso por um valor contínuo. Nesse contexto, *Hartigan* define que contornos de densidade de grupos (*density-contour clusters*) de uma dada função de densidade probabilidade $f(x)$ em $\mathfrak{X}$ em um dado nível de densidade λ são subconjuntos máximos conectados do conjunto de nível definido como $\{x \mid f(x) \geq \lambda\}$. No cenário de *DBSCAN**, os contornos de densidade de grupos utilizam o limiar de densidade $\lambda = 1/\varepsilon$ e uma estimativa m_{pts} -NN não normalizada de $f(x)$ representada por $1/d_{core}(x)$ ([CAMPELLO et al., 2013a](#); [CAMPELLO et al., 2015](#)).

Assim como descrito anteriormente, *HDBSCAN** é capaz de produzir todas as possíveis soluções de *DBSCAN** a partir do valor de m_{pts} e todos os limiares $\lambda = 1/\varepsilon$ em $[0, \infty)$ ([CAMPELLO et al., 2013a](#); [CAMPELLO et al., 2015](#)). A partir dessa afirmativa, à medida que o valor de ε decresce, os grupos que compõem a hierarquia de *HDBSCAN** tendem a ser cada vez menores, até que os mesmos desapareçam ou se transformem em novos subgrupos. Essa tendência mostra que os grupos mais significativos conseguem "sobreviver" mais ao longo da hierarquia ([CAMPELLO et al., 2013a](#); [CAMPELLO et al., 2015](#)).

Para mensurar a sobrevivência de grupos ao longo da hierarquia de *HDBSCAN**, [Campello et al. \(2013a\)](#) e [Campello et al. \(2015\)](#) formularam uma medida de estabilidade de grupos baseada na noção de excesso de massa definido em ([MULLER; SAWITZKI, 1991](#)). Nesse cenário, considerando que o nível de densidade λ decresça e que um contorno de densidade de grupo C_i apareça no nível $\lambda_{min}(C_i)$, então é definido que um subconjunto máximo conectado do conjunto de nível é dado por $\{x \mid f(x) \geq \lambda_{min}(C_i)\}$. Assim, o excesso de massa do grupo C_i pode ser definido pela Equação 3.2 (em complemento, ilustrado pela Figura 6).

$$E(C_i) = \int_{x \in C_i} (f(x) - \lambda_{min}(C_i)) dx \quad (3.2)$$

Como pode ser visto na Figura 6, as áreas de três grupos (C_2 , C_4 e C_5) são demarcadas com a cor cinza escuro e representam o excesso de massa desses grupos. O excesso de massa do

3.4:

$$S(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} (\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i)) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\frac{1}{\varepsilon_{\min}(\mathbf{x}_j, \mathbf{C}_i)} - \frac{1}{\varepsilon_{\max}(\mathbf{C}_i)} \right) \quad (3.4)$$

em que $\lambda_{\min}(\mathbf{C}_i)$ representa o nível mínimo de densidade da existência de um grupo $\mathbf{C}_i$, $\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i)$ representa o nível de densidade em que o objeto $\mathbf{x}_j$ não pertence mais ao grupo $\mathbf{C}_i$, $\varepsilon_{\max}(\mathbf{C}_i)$ e $\varepsilon_{\min}(\mathbf{x}_j, \mathbf{C}_i)$ são os valores correspondentes ao limiar (nível de densidade) ε (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

A estabilidade de cada grupo pode ser calculada durante a construção da hierarquia de *HDBSCAN** com pequenos processos adicionais e sem afetar a complexidade assintótica do método (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Basicamente, para cada objeto $\mathbf{x}_j$, as mudanças de seus rótulos são observadas e a expressão $\lambda_{\max}(\mathbf{x}_j, \mathbf{C}_i) - \lambda_{\min}(\mathbf{C}_i)$ é calculada cumulativamente para cada grupo $\mathbf{C}_i$ em relação a cada objeto $\mathbf{x}_j \in \mathbf{C}_i$ (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

3.2.2 Algoritmo de Otimização de Extração Não-Supervisionada de Grupos

O objetivo da extração de grupos é garantir que os grupos mais significativos pertencentes à árvore simplificada de grupos gerada por *HDBSCAN** sejam retornados como um agrupamento particional (solução) composto por grupos não sobrepostos (incluindo objetos considerados como possíveis ruídos) (CAMPELLO *et al.*, 2013a). Logo, seja $\{\mathbf{C}_2, \dots, \mathbf{C}_k\}$ a coleção de todos os grupos pertencentes à árvore de grupos simplificada de *HDBSCAN**, exceto a raiz $\mathbf{C}_1$, e seja $S(\mathbf{C}_i)$ o valor de estabilidade de cada grupo. Então, Campello *et al.* (2013a) formula um problema de otimização (descrito nas Equações 3.5 e 3.6) que tem como objetivo maximizar as estabilidades agregadas gerais dos grupos extraídos.

$$J = \sum_{i=2}^k \delta_i S(\mathbf{C}_i) \quad (3.5)$$

$$\text{sujeito a} \quad \begin{cases} \delta_i \in \{0, 1\}, i = 2, \dots, k \\ \text{Exatamente } \delta_{(\cdot)} = 1 \text{ para cada grupo folha para a raiz.} \end{cases} \quad (3.6)$$

Em que δ_i representa que cada grupo $\mathbf{C}_i$ da árvore é incluído na solução quando $\delta_i = 1$ ou não incluído $\delta_i = 0$. Intuitivamente, as variáveis $\delta_2, \dots, \delta_k$ correspondentes a cada grupo em $\{\mathbf{C}_2, \dots, \mathbf{C}_k\}$ são utilizadas para garantir que grupos no mesmo ramo da árvore não sejam simultaneamente selecionados na mesma solução. Para esse propósito, os grupos devem ser

mutualmente exclusivos e cada objeto pode apenas ter um único rótulo de grupo (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Uma vez que a seleção de grupos em qualquer subárvore representa um subproblema de mesma natureza do problema original (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), esse problema, descrito nas Equações 3.5 e 3.6, pode ser resolvido de maneira global com uma estratégia de programação dinâmica. Essa estratégia objetiva resolver incremental e recursivamente os subproblemas em ordem crescente de tamanho, ou seja, a resolução começa a partir dos nós folha da árvore de grupos e agrega as soluções de nós intermediários até o topo da árvore. A ideia é processar recursivamente todos os nós C_i (com exceção da raiz C_1) iniciando-se pelos nós folha e, em um esquema *bottom-up*, decidir qual ou quais grupos C_i devem ser melhores para a solução. Para que isso seja possível, o valor total de estabilidade $\hat{S}(C_i)$ dos grupos selecionados na subárvore que possui o grupo C_i é propagado e atualizado utilizando a Equação 3.7:

$$\hat{S}(C_i) = \begin{cases} S(C_i) & \text{se } C_i \text{ é um nó folha.} \\ \max\{S(C_i), \hat{S}(C_{il}), \hat{S}(C_{ir})\} & \text{se } C_i \text{ é um nó interno.} \end{cases} \quad (3.7)$$

onde C_{il} e C_{ir} representam os grupos filhos esquerdo e direito, respectivamente, de C_i . Note que, assim como discutido em Campello *et al.* (2013a), Campello *et al.* (2015), a árvore de grupos analisada é binária. Entretanto, essa árvore pode ser facilmente tratada como n -ária. Ante o exposto, na Figura 7 é apresentada uma ilustração de solução ótima obtida pelo problema de otimização descrito anteriormente a partir de uma árvore de grupos:

Figura 7 – Ilustração de solução ótima a partir de uma determinada árvore de grupos (Adaptado de (CAMPELLO *et al.*, 2015)).

Utilizando uma análise *bottom-up* (Algoritmo 3) no exemplo da Figura 7, considera-se inicialmente que todos os grupos $C_2, \dots, C_k$ estejam na solução final. Então, ao avaliar o conjunto de grupos $\{C_7, C_{10}, C_{11}\}$ em que o grupo C_7 é pai dos grupos C_{10} e C_{11} , note que a soma das

estabilidades dos grupos C_{10} e C_{11} é maior que a estabilidade de seu pai C_7 . Portanto, C_7 é removido da solução. Entretanto, ao analisar o conjunto de grupos $\{C_3, C_6, C_{10}, C_{11}\}$ a soma das estabilidades de C_6, C_{10} e C_{11} é menor que a estabilidade do grupo C_3 . Assim, C_6, C_{10} e C_{11} são retirados da solução. Ainda, ao analisar o conjunto de grupos $\{C_5, C_8, C_9\}$, a estabilidade do grupo pai C_5 é maior que a soma das estabilidades de seus filhos, i.e., C_8 e C_9 . Então, C_8 e C_9 são retirados da solução. Ao avaliar o conjunto de grupos $\{C_2, C_4, C_5\}$, a soma das estabilidades dos grupos filhos C_4 e C_5 é maior que a estabilidade do grupo pai C_2 . Então, C_2 é removido da solução final. Por fim, C_3, C_4 e C_5 são os grupos pertencentes à solução ótima (com $J = 22$) desse exemplo.

Algoritmo 3 – Solução do problema de otimização apresentado pelas Equações 3.5 e 3.6.

Entrada: Seja $C_2, \dots, C_k$ a coleção de grupos da árvore simplificada gerada por *HDBSCAN**.

Saída: Partição composta pelos grupos mais significativos.

```

1: Inicializa  $\delta_2, \dots, \delta_k = 1$ 
2: para todo nó folha  $C_h$  faça
3:    $\hat{S}(C_h) \leftarrow S(C_h)$ 
4: fim para
5: Inicializando a partir dos níveis mais profundos da árvore de grupos (com exceção da raiz).
6: se  $S(C_i) < \hat{S}(C_{il}) + \hat{S}(C_{ir})$  então
7:    $S(C_i) \leftarrow \hat{S}(C_{il}) + \hat{S}(C_{ir})$ 
8:    $\delta_i \leftarrow 0$ 
9: senão
10:   $\hat{S}(C_i) \leftarrow S(C_i)$ 
11:  para todo nó da subárvore  $C_i$  faça
12:     $\delta_i \leftarrow 0$ 
13:  fim para
14: fim se

```

Assim como descrito em [Campello et al. \(2015\)](#), os passos das Linhas 11 à 13 do Algoritmo 3 podem ser implementados mais eficientemente em uma abordagem de pós-processamento *top-down*, no qual, para cada ramo da árvore de grupos são avaliados os grupos rasos que não foram retirados da solução, ou seja, com $\delta_{(\cdot)} = 1$. As duas abordagens, i.e., *bottom-up* e *top-down* possuem complexidade assintótica em termos de espaço e tempo igual a $O(k)$, em que k é o número de grupos na árvore ([CAMPELLO et al., 2013a](#); [CAMPELLO et al., 2015](#)).

3.3 Método GLOSH

No caso de detecção não supervisionada de *outliers*, existem dois conceitos que determinam o tipo de *outlier* que um determinado objeto pode ser: local ou global ([AGGARWAL, 2015](#)). A maioria dos métodos de detecção de *outliers* são restringidos a apenas um desses conceitos citados. Por exemplo, o método de detecção local de *outliers*, conhecido como *LOF*, relaciona cada objeto a uma vizinhança local predefinida ([BREUNIG et al., 2000](#)). Por outro lado, alguns métodos são formulados para a detecção global de *outliers*, e.g., o método *KNN-outlier*,

que considera que objetos com valores não usuais em relação a todo conjunto de dados sejam denominados *outliers* (RAMASWAMY *et al.*, 2000). Adicionalmente, um método que pode ser utilizado como detecção global é o *LOCI*. Apesar de ser um método local, o *LOCI* pode ter seu parâmetro largamente estendido, ou seja, estendido até o ponto de englobar o conjunto de dados. Isso o torna um método de detecção global (PAPADIMITRIOU *et al.*, 2003).

Os métodos citados tendem a utilizar um conjunto de referência predefinida para determinar se objetos são ou não *outliers*. Tal característica os limita a um dos conceitos (i.e., local ou global) em algum aspecto. Entretanto, Campello *et al.* (2015) observam que quando se assume um conjunto dinâmico de referência, ou seja, um conjunto de referência em que dados podem ser gerados por um ou mais mecanismos, esses mecanismos são usualmente modelados por grupos. Então, o escopo de conjuntos de referência pode ser ajustado para cada objeto de acordo com o seu grupo mais próximo (em um contexto de grupos baseados em densidade) dentro de uma hierarquia baseada em densidade. Dessa forma, cada objeto pode ser global com relação a alguns objetos ou local para outros. A partir dessa intuição, Campello *et al.* (2015) definem o método *GLOSH* e o integram ao *framework HDBSCAN**.

GLOSH determina a cada objeto do conjunto de dados um grau de anomalia (*degree of outlierness*) que possibilita determinar se um objeto é um *inlier*, *outlier* local ou global em algum aspecto analisado. Nesse contexto, seja um objeto $\mathbf{x}_i$ a ter seu grau de anomalia determinado, o grupo mais próximo de $\mathbf{x}_i$, sob uma perspectiva baseada em densidade, é denotado como $\mathbf{C}_j$, em que $\mathbf{x}_i \in \mathbf{C}_j$ em um nível de densidade correspondente ao de densidade de $\mathbf{x}_i$, i.e., $f(\mathbf{x}_i) = 1/d_{core}(\mathbf{x}_i)$ (ou $\varepsilon = d_{core}(\mathbf{x}_i)$) (CAMPELLO *et al.*, 2015).

A partir dessa informação, ao analisar a hierarquia de *HDBSCAN** em uma visão *bottom-up*, é possível perceber que $\mathbf{C}_j$ é o primeiro grupo em que o objeto $\mathbf{x}_i$ é indexado, ou seja, grupo em que $\mathbf{x}_i$ é modificado de ruído para objeto *core* na hierarquia (CAMPELLO *et al.*, 2015). Logo, a ideia desse método é considerar que o objeto mais “denso” $\mathbf{x}_l$ de um grupo $\mathbf{C}_j$ seja mais *inliers* do que os demais desse grupo, e utilizar $\mathbf{x}_l$ como referência para os outros objetos que também têm $\mathbf{C}_j$ como grupo mais próximo. Em outras palavras, o objeto $\mathbf{x}_l$ é o último “sobrevivente” em relação aos demais objetos do grupo durante o aumento de limiar de densidade ao longo da construção da hierarquia (CAMPELLO *et al.*, 2015).

Seja $f(\mathbf{x}_l) = 1/d_{core}(\mathbf{x}_l)$ o nível de densidade correspondente à densidade do objeto $\mathbf{x}_l$ mais denso pertencente ao grupo $\mathbf{C}_j$, então, define-se $f_{max}(\mathbf{x}_i)$ como um referencial de densidade para todos os objetos $\mathbf{x}_i$ que têm $\mathbf{C}_j$ como grupo baseado em densidade mais próximo. Assim, Campello *et al.* (2015) definem uma medida de *outlierness* (representada na Equação 3.8) para o objeto $\mathbf{x}_i$:

$$\frac{f_{max}(\mathbf{x}_i) - f(\mathbf{x}_i)}{f_{max}(\mathbf{x}_i)} \quad (3.8)$$

em que $f(\mathbf{x}_i) = 1/d_{core}(\mathbf{x}_i)$ e $f_{max}(\mathbf{x}_i) = 1/d_{core}(\mathbf{x}_l)$, sendo $\mathbf{x}_l$ o objeto mais denso do grupo em

que os objetos $\mathbf{x}_i$ são mais próximos na hierarquia.

Intuitivamente, a Equação 3.8 retorna valores entre zero e um. Quando $f_{\max}(\mathbf{x}_i) - f(\mathbf{x}_i)$ é próximo de zero, indica que o objeto $\mathbf{x}_i$ em questão está uma região mais densa de seu grupo mais próximo. Por outro lado, quando $f_{\max}(\mathbf{x}_i) - f(\mathbf{x}_i)$ tende a ser relativamente grande e o resultado da Equação 3.8 é próximo de 1 (grau máximo de *outlierness*), indica que o objeto $\mathbf{x}_i$ está apenas ligeiramente denso conectado ao grupo.

Entretanto, a Equação 3.8 não representa a situação em que é utilizado o parâmetro de tamanho mínimo de grupo $m_{clSize} > 1$. Quando isso acontece, o limite de densidade mais baixo ($\lambda(\mathbf{x}_i)$) em que um objeto é indexado pela primeira vez ao seu grupo baseado em densidade mais próximo e o limite mais alto de densidade $\lambda_{\max}(\mathbf{x}_i)$ que indica que todos os objetos de um grupo tenham sido rotulados como ruído, não são necessariamente os mesmos valores de $f(\mathbf{x}_i)$ e $f_{\max}(\mathbf{x}_i)$, respectivamente (CAMPELLO *et al.*, 2015). Assim, para tornar a medida mais geral para o caso descrito acima, a medida GLOSH é definida pela Equação 3.9:

$$GLOSH(\mathbf{x}_i) = \frac{\lambda_{\max}(\mathbf{x}_i) - \lambda(\mathbf{x}_i)}{\lambda_{\max}(\mathbf{x}_i)} = \frac{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)} - \frac{1}{\varepsilon(\mathbf{x}_i)}}{\frac{1}{\varepsilon_{\max}(\mathbf{x}_i)}} = 1 - \frac{\varepsilon_{\max}(\mathbf{x}_i)}{\varepsilon(\mathbf{x}_i)} \quad (3.9)$$

em que $\varepsilon_{\max}(\mathbf{x}_i)$ e $\varepsilon(\mathbf{x}_i)$ correspondem aos valores de raio ε associados com o limite de densidade $\lambda_{\max}(\mathbf{x}_i)$ e $\lambda(\mathbf{x}_i)$, respectivamente.

Algoritmo 4 – Principal passo para calcular os *scores* de GLOSH.

▷ Durante a construção da hierarquia de HDBSCAN* pelos Algoritmos 1 e 2 (*top-down*).

- 1: **para todo** objeto $\mathbf{x}_i$ **faça**
 - 2: Salva o último grupo $\mathbf{C}_j$ (rótulo) que $\mathbf{x}_i$ pertence, antes de tornar ruído.
 - 3: Salva o raio ε mais baixo (denotado como $\varepsilon(\mathbf{x}_i)$), em que $\mathbf{x}_i$ ainda pertence ao grupo $\mathbf{C}_j$ e abaixo em que $\mathbf{x}_i$ é rotulado como ruído.
 - 4: Salva o raio ε mais baixo (denotado como $\varepsilon_{\max}(\mathbf{x}_i)$), em que o grupo $\mathbf{C}_j$ ou qualquer um dos seus subgrupos ainda existem e abaixo do qual todos os seus objetos são rotulados como ruído.
 - 5: **fim para**
 - 6: **para todo** cada objeto $\mathbf{x}_i$ **faça**
 - 7: Calcula $GLOSH(\mathbf{x}_i)$ utilizando a Equação 3.9 com os valores $\varepsilon_{\max}(\mathbf{x}_i)$ e $\varepsilon(\mathbf{x}_i)$ encontrados nas Linhas 3 e 4.
 - 8: **fim para**
-

A proposta do Algoritmo 4 é obter para cada objeto as informações descritas nas Linhas 2, 3 e 4 enquanto a hierarquia de HDBSCAN* é construída. Assim, ao final da construção da hierarquia, o cálculo dos *scores* pela Equação 3.9 pode ser feito em $O(n)$, em que n representa o número total de objetos do conjunto de dados (CAMPELLO *et al.*, 2015).

3.4 Considerações Finais

Neste capítulo foram apresentados conceitos base do *framework HDBSCAN**, que é o principal método investigado no presente projeto de pesquisa. Em relação a *HDBSCAN**, foram apresentadas suas definições elementares, seguidas da apresentação do método de extração de grupos significativos e conceitos do método de detecção de *outliers*, conhecido como *GLOSH*. Embora *HDBSCAN** abranja conceitos de agrupamento não supervisionado e semi-supervisionado, este projeto limita-se apenas a aspectos de agrupamento não supervisionado de dados por esse *framework*.

TÉCNICAS DE AGRUPAMENTO PARALELO E DISTRIBUÍDO DE DADOS

As técnicas de agrupamento de dados, especialmente os métodos hierárquicos, possuem alta complexidade computacional (JAIN; DUBES, 1988). Por essa razão, a análise de grandes volumes de dados pode ser proibitiva para boa parte desses métodos (TAN *et al.*, 2005). À vista disso, diversas técnicas de agrupamento aninhadas à computação paralela e distribuída foram propostas na literatura para aumentar a escalabilidade de agrupamento de quantidades massivas de dados (XU *et al.*, 1999; OLMAN *et al.*, 2009; ANANDAN, 2015).

Neste capítulo são apresentadas algumas técnicas de agrupamento paralelo de dados que estão relacionadas ao presente trabalho de pesquisa. Especificamente, na Seção 4.1 é apresentado o método *DiSC* (JIN *et al.*, 2013a); na Seção 4.2, é apresentada uma versão paralela de *OPTICS* (PATWARY *et al.*, 2013); por fim, na Seção 4.3, é apresentado o algoritmo *Random Blocks*, que é uma versão paralela de *HDBSCAN** proposta em (SYED, 2014).

4.1 Algoritmo DiSC

Distributed Single Linkage Hierarchical Clustering (DiSC) é uma versão paralela e distribuída do clássico algoritmo *single-linkage* (JIN *et al.*, 2013a). *DiSC* implementa *single-linkage* por meio do conceito de árvores geradoras mínimas *Minimum Spanning Trees* (MST), assim como descrito em (JAIN; DUBES, 1988). A ideia desse algoritmo é dividir um conjunto de dados de entrada em subconjuntos menores distribuídos e sobrepostos em diferentes unidades de processamento. Assim, *MSTs* parciais são obtidas a partir de quaisquer subconjuntos de dados e, posteriormente, combinadas em uma solução geral (JIN *et al.*, 2013a).

O algoritmo *DiSC* é baseado no conceito de programação conhecido como *MapReduce* (explicado em detalhes na Seção 6.1). Com base nesse conceito, *DiSC* é dividido em duas tarefas

compostas por funções de *mappers* e *reducers*. Como pode ser visto na Figura 8, a primeira tarefa de *DiSC* consiste em uma função *mapper Prim* e uma função *reducer Kruskal* (JIN *et al.*, 2013a).

A função *mapper Prim* é responsável pela obtenção de *MSTs* parciais em cada unidade de processamento a partir de cada subconjunto de dados. Já a função *reducer Kruskal* é utilizada localmente para a eliminação de possíveis arestas redundantes obtidas durante a construção parcial das *MSTs* na etapa de *mapper Prim*. Na segunda tarefa, as funções de *mapper Kruskal* e *reducer Kruskal* são responsáveis pelo mapeamento e combinação global de todas as *MSTs* parciais geradas na primeira tarefa de *DiSC*, respectivamente.

Figura 8 – Arquitetura de *DiSC* em *MapReduce* (Adaptado de (JIN *et al.*, 2013a)).

4.2 Parallel OPTICS

Parallel OPTICS (*POPTICS*) é uma versão paralela e distribuída do clássico algoritmo *OPTICS* (ANKERST *et al.*, 1999a) sob uma óptica de conceitos de teoria dos grafos, especificamente, em *Árvore Geradora Mínima (MST)* (PATWARY *et al.*, 2013). Basicamente, para o surgimento desse algoritmo, os autores exploram a similaridade entre *OPTICS* e o conceito de *(MST)* a fim de quebrar a sequencialidade que *OPTICS* possui em sua ordenação de pontos (objetos) segundo suas definições de distância *core* e distância de acessibilidade (ANKERST *et al.*, 1999a). Na Figura 9 é apresentado um exemplo de relacionamento entre o algoritmo *OPTICS* e uma *MST* construída pelo algoritmo de *Prim* (PRIM, 1957).

Assim, para extração de grupos (agrupamento particional via *DBSCAN*) em um conjunto de dados a partir de um determinado valor de ϵ' (definido na Subseção 2.2.4.2), em que $0 \leq \epsilon' \leq \epsilon$, o original algoritmo *OPTICS* produz três informações importantes: a ordenação dos objetos, as distâncias *core* e as distâncias de acessibilidade de cada objeto do conjunto de dados.

A ordenação dos objetos corresponde à ordem em que cada objeto do conjunto de dados foi processado. Já as distâncias *core* são denominadas para cada objeto de duas maneiras, se o objeto é *core*, então a distância *core* deste objeto será a $m_{pts_distância}$ (descrita na Subseção 2.2.4.2), caso contrário, essa distância será indefinida. Por fim, as distâncias de acessibilidade são um conjunto de distâncias mínimas referentes a cada objeto do conjunto de dados (ANKERST

Figura 9 – Similaridade entre *OPTICS* e uma *MST* construída pelo algoritmo de *Prim*. (a) O conjunto de dados. (b) A ϵ -vizinhança dos objetos *a*, *b* e *c* sob a perspectiva do algoritmo *OPTICS*. (c) Vértices adjacentes (vizinhos) dos objetos *a*, *b* e *c* sob a perspectiva do algoritmo de *Prim* (Adaptado de (PATWARY *et al.*, 2013)).

et al., 1999a). Em outras palavras, o conjunto de distâncias mínimas de acessibilidade obtidas por *OPTICS* são muito parecidas com o conceito de *MST* da teoria dos grafos tradicional (ANDERSON, 1991; PATWARY *et al.*, 2013).

Para transformar o clássico algoritmo *OPTICS* em um problema de grafos, na *MST* os objetos serão denotados como vértices e os pesos das arestas serão compostas pelas distâncias de acessibilidade obtidas segundo as definições de *OPTICS* (PATWARY *et al.*, 2013). Então, a utilização de *MST* baseada na distância de acessibilidade de *OPTICS* possibilita a paralelização desse algoritmo, pois existem várias teorias na literatura de como paralelizar algoritmos de construção de *MSTs* (CHONG *et al.*, 2003; NOBARI *et al.*, 2012).

Nesse contexto, a proposta de *POPTICS* é dividir, em diferentes unidades de processamento, um determinado conjunto de dados em subconjuntos de dados menores, obter *MSTs* parciais baseadas na distância de acessibilidade de *OPTICS* e então combinar as *MSTs* parciais resultantes em uma única *MST* global. Por fim, para extrair os grupos sobre a *MST* global (baseada na distância de acessibilidade) a partir de um parâmetro de densidade ϵ' , basta realizar a remoção de arestas que possuírem as distâncias de acessibilidade maiores que ϵ' (PATWARY *et al.*, 2013).

4.3 Random Blocks

Baseado na abordagem de construção paralela e distribuída de *MSTs* no contexto de *Agrupamento de dados* apresentada (OLMAN *et al.*, 2009), *Random Blocks* é um algoritmo paralelo de construção de *MSTs* baseadas na distância de acessibilidade mútua de *HDBSCAN**

(SYED, 2014).

Esse algoritmo procura uma *MST* exata sob a perspectiva de *HDBSCAN** a partir do conjunto completo de dados. Essa procura é feita pela divisão do conjunto de dados em partes menores chamadas de *data blocks*. Cada *data block* é processado independentemente em um ambiente paralelo e distribuído por uma unidade de processamento a fim de obter uma *MST* local (baseada na distância de acessibilidade mútua de *HDBSCAN**) sobre os dados desse *data block*. Posteriormente, as *MSTs* locais geradas para cada unidade de processamento independente são combinadas em uma única *MST* global, *MST* essa que pode ser usada para a construção exata da hierarquia de *HDBSCAN** sobre o conjunto de dados completo (SYED, 2014).

Random Blocks possui quatro etapas para a construção de *MSTs*. Na Figura 10 é apresentado um exemplo simples de todos os processos de *Random Blocks* considerando que o número mínimo de vizinhos de um objeto do conjunto de dados seja igual a dois, i.e., $m_{pts} = 2$:

Figura 10 – Etapas do Algoritmo *Random Blocks* (Adaptado de (SYED, 2014)).

Como pode ser visto na Figura 10, a primeira etapa consiste na divisão do conjunto de dados em s subconjuntos não sobrepostos e de tamanhos aproximadamente iguais entre si. Na segunda etapa, cada *data blocks* é formado por $2 \times m_{pts}$ subconjuntos únicos, ou seja, sem repetição de subconjuntos em um mesmo *data block*. Na terceira etapa, *MSTs* locais são construídas independentemente sobre os dados de cada *data block*. Por fim, na quarta etapa, as *MSTs* locais são combinadas em uma única *MST* global.

Formalmente, no que se refere à divisão dos dados feita na primeira etapa de *Random Blocks* apresentada na Figura 10, sendo $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ um conjunto de dados multidimensional, esse algoritmo parte do princípio de divisão de $\mathbf{X}$ em s subconjuntos não sobrepostos ($\mathbf{S}$),

em que $\mathbf{S} = \mathbf{S}_1 \cup \mathbf{S}_2 \cup \dots \cup \mathbf{S}_s = \mathbf{X}$ e $\mathbf{S}_i \cap \mathbf{S}_j = \emptyset$, para $i \neq j$. O valor s é selecionado seguindo alguns critérios:

1. O valor s selecionado deve ser $s \geq 2 \times m_{pts}$.
2. Número de unidades de processamento disponíveis.

Na segunda etapa do algoritmo *Random Blocks* ilustrada na Figura 10, dado s o número de subconjuntos de dados não sobrepostos e m_{pts} o número mínimo de vizinhos de um determinado objeto, $\binom{s}{2 \times m_{pts}}$ *data blocks* são construídos. Dessa forma, $\binom{s}{2 \times m_{pts}}$ representa o número de combinações possíveis de subconjuntos de dados sem repetição. Assim, cada *data block* é formado por $2 \times m_{pts}$ subconjuntos dos s disponíveis em $\mathbf{S}$. Basicamente, cada *data block* é único em relação à combinação de subconjuntos de dados que o compõe. Os subconjuntos de dados dentro de cada *data block* também são únicos, ou seja, sem sobreposição dentro de um determinado *data block* (SYED, 2014).

De acordo com a prova matemática apresentada pelo trabalho original (SYED, 2014), a proposta de criar *data blocks* utilizando a combinação de subconjuntos de dados sem repetição é de garantir que seja possível calcular as verdadeiras distâncias *core* e distâncias de acessibilidade mútua de cada objeto de um determinado conjunto de dados segundo as perspectivas de *HDBSCAN** em ambientes paralelos e distribuídos (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

A terceira etapa de *Random Blocks*, assim como pode ser visto na Figura 10, é a de construção de *MSTs* locais (baseadas na distância de acessibilidade mútua de *HDBSCAN**) em paralelo por diferentes unidades de processamento. Especificamente, cada unidade de processamento é responsável por um *data block* por vez. Os dados de cada *data block* são transformados em *MSTs* locais calculadas pelo algoritmo de *Prim* (PRIM, 1957).

Por fim, a quarta e última etapa do algoritmo *Random Blocks* (ilustrado na Figura 10) é composta pela combinação das *MSTs* locais criadas na terceira etapa. Especificamente, essa etapa consiste em remover possíveis arestas duplicadas que foram calculadas na etapa anterior e garantir que somente as verdadeiras distâncias *core* e distâncias de acessibilidade mútua de cada objeto do conjunto de dados completos sejam selecionadas para então compor a *MST* global. A *MST* global resultante nessa etapa é ordenada em ordem crescente de peso (distância de acessibilidade mútua). Além disso, a partir da *MST* global, uma hierarquia exata de *HDBSCAN** pode ser encontrada (SYED, 2014).

4.4 Considerações Finais

Neste capítulo foram discutidas algumas técnicas paralelas e distribuídas de agrupamento hierárquico de dados. Os algoritmos apresentados neste capítulo apresentam uma relação com os

métodos propostos neste trabalho de pesquisa (descritos no Capítulo 6), pois os processos de paralelização desses algoritmos também são baseados na divisão de um determinado conjunto de dados em partes menores espalhadas em diferentes unidades de processamento independentes, e de alguma maneira ou intuição, obter a partir das partes menores *MSTs* locais (parciais) dos dados em paralelo, e por fim, computar uma solução global baseando-se nas *MSTs* locais construídas.

ABORDAGEM *RECURSIVE SAMPLING*

Desenvolvido pelos mesmos autores de *Random Blocks* (SYED, 2014) (descrito na Subseção 4.3), *Recursive Sampling* é um algoritmo sofisticado de paralelização de *HDBSCAN**. Esse algoritmo elimina problemas críticos de sobreposições de dados geradas por *Random Blocks*. Tais problemas são resolvidos pelo fato de *Recursive Sampling* utilizar uma abordagem “inteligente” de divisão de dados baseada em uma estrutura presente neles (SYED, 2014).

Este capítulo contém a seguinte divisão: na Seção 5.1 é apresentada a visão geral do algoritmo *Recursive Sampling* utilizando amostras aleatórias em sua composição; na Seção 5.6, são introduzidos conceitos de técnicas de sumarização de dados (e.g., *Data Bubbles*); na Seção 5.7, é apresentada a implementação de *Recursive Sampling* utilizando *Data Bubbles*; uma implementação de *Recursive Sampling* baseada em memória compartilhada é apresentada na Seção 5.8; por fim, na Seção 5.9 são apresentadas as considerações finais deste capítulo.

5.1 Visão Geral de *Recursive Sampling*

Recursive Sampling é um algoritmo dividido em níveis hierárquicos, em que cada um deles corresponde a um agrupamento particional que reflete a estrutura presente nos dados (SYED, 2014). As estruturas desses dados são representadas por grupos mais significativos, segundo as perspectivas de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Diante disso, considerando que um conjunto de dados $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ composto por n objetos multidimensionais no primeiro nível de *Recursive Sampling* não possa ser processado unicamente de uma só vez por uma unidade de processamento, a proposta desse algoritmo é utilizar amostras aleatórias de dados do conjunto $\mathbf{X}$ para estimar, a cada nível hierárquico, a estrutura presente nos dados como um todo (SYED, 2014).

Formalmente, amostras de dados são representadas por $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_m\}$, tal que $1 \leq m \leq n$ e $\mathbf{S} \subset \mathbf{X}$. Os demais objetos de $\mathbf{X}$ que não estão no conjunto de amostras $\mathbf{S}$ são denotados

por $\mathbf{U}$, tal que $\mathbf{X} = \mathbf{U} + \mathbf{S}$. As amostras de dados $\mathbf{S}$ são utilizadas para obtenção de uma partição estimada, denotada por $P_{\mathbf{S}} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k, \mathbf{O}\}$, em relação ao conjunto de dados $\mathbf{X}$, em que $\{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$ representa o conjunto de grupos mais significativos e $\mathbf{O}$ um conjunto de objetos descritos como ruídos pelo *framework HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

A partir dos grupos encontrados nas amostras de dados $\mathbf{S}$, i.e., presentes em $P_{\mathbf{S}}$, os objetos não amostrados em $\mathbf{U}$ são rotulados de acordo com os rótulos de grupos atribuídos às suas amostras aleatórias mais próximas em $\mathbf{S}$ no processo de agrupamento de *HDBSCAN**. Esse procedimento de rotulação tem como resultado uma partição $P_{\mathbf{X}} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k\}$ composta pelos objetos em $\mathbf{U}$ e $\mathbf{S}$ subdivididos em k subconjuntos de dados, i.e., $\mathbf{X}_i$, tal que $1 \leq i \leq k$.

Por fim, cada subconjunto de dados $\mathbf{X}_i$ pode ser processado em paralelo por unidades de processamento distintas, da mesma forma que foi realizada com $\mathbf{X}$ (SYED, 2014).

Assim como definido no Capítulo 4, uma unidade de processamento corresponde a um componente composto por recursos individuais, tais como: memória principal, processadores (ou núcleos), armazenamento em disco, entre outros. De forma complementar, considerando que cada uma dessas unidades possua uma capacidade de processamento τ , que é descrita pela quantidade de objetos que podem ser processados, qualquer subconjunto de dados em $P_{\mathbf{X}} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k\}$ que exceder essa capacidade pode ser recursivamente dividido em subconjuntos menores, utilizando amostras e a aplicação de *HDBSCAN** sobre elas como descrito anteriormente, até que alguma unidade seja capaz de processá-los e, assim, constituir novos níveis da hierarquia de grupos.

Uma vez que os subconjuntos de dados sejam suportados pelas unidades de processamento, são obtidas árvores geradoras mínimas (MST_{local}) baseadas na distância de acessibilidade mútua (descrita na Seção 3.1.2) de *HDBSCAN** sobre os mesmos (SYED, 2014). Cada MST_{local} obtida contém arestas que as conectam entre si durante o processo de *Recursive Sampling*, denominadas *arestas intergrupos* (E_{inter}) (SYED, 2014). Ainda, cada MST_{local} é estendida segundo as definições de *HDBSCAN**, i.e., cada vértice (objeto) de qualquer MST_{local} possui um “self-edge” com sua distância *core* correspondente. Ao final do processo recursivo de *Recursive Sampling*, ou seja, no momento em que não há mais subconjuntos de dados a serem processados, as MST_{local} e as E_{inter} resultantes formarão uma única MST global composta por todos os objetos do conjunto de dados (SYED, 2014).

Com o intuito de ilustrar os processos de *Recursive Sampling* descrito anteriormente, na Figura 11 é apresentado um conjunto de dados bidimensional ruidoso composto por grupos com formas arbitrárias.

Figura 11 – Exemplo de conjunto de dados composto por grupos com arbitrárias formas e constituído por ruídos.

Supondo que o conjunto de dados apresentado na Figura 11 não seja suportado por determinada unidade de processamento, o primeiro passo de *Recursive Sampling* é seleccionar amostras representativas dos dados em questão. Nesse contexto, na Figura 12 é apresentado um exemplo de amostragem obtida a partir do conjunto de dados apresentado na Figura 11:

Figura 12 – Exemplo de conjunto de amostras de dados baseando-se no conjunto de dados apresentado na Figura 11.

Uma vez que as amostras de dados são escolhidas, uma *MST* baseada na distância de acessibilidade mútua de *HDBSCAN** é construída e estendida (denotada como $MST_{est_sampling}$) sobre essas amostras de dados (apresentada na Figura 13). Por simplicidade sem perda de generalidade, o exemplo apresentado na Figura 13 não apresenta as “self-edges” ponderadas pela distância *core* correspondente de cada objeto. Visualmente, na Figura 13, é possível observar que os objetos das amostras de dados (representados pelos pontos pretos) que são conectados por linhas pontilhadas representarão objetos do mesmo grupo (neste contexto, grupos mais significativos).

Ainda nesta figura, os possíveis grupos a serem formados são separados uns dos outros por linhas constantes (que representam as *arestas intergrupos*) que ligam seus respectivos objetos extremos, ou seja, os seus respectivos objetos mais próximos (com relação a distância de acessibilidade mútua) entre os grupos. Vale destacar que os dois objetos das amostras de dados que estão nomeados como *o* são potenciais objetos ruídos deste exemplo.

Figura 13 – Exemplo de *MST* (“self-edges” desconsideradas) criada por *HDBSCAN** baseando-se na amostra de dados apresentada na Figura 12.

5.2 Extração de Grupos Significativos

O processo de extração de grupos significativos do conjunto de amostras de dados é feito pelo algoritmo *HDBSCAN**. Resumidamente, a distância *core* de cada objeto em *S* é calculada. Assim como sugerido em Campello *et al.* (2013a) e Campello *et al.* (2015), as distâncias de acessibilidade mútua de qualquer objeto do conjunto de amostras podem ser calculadas por demanda para a construção da *MST*.

A *MST* pode ser obtida utilizando uma extensão do algoritmo de *Prim* (PRIM, 1957; CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Então, a *MST* é estendida $MST_{est_sampling}$ utilizando as distâncias *core* de cada vértice como “*self-edges*”. Posteriormente, uma hierarquia *H* (árvore de grupos) é construída a partir da remoção iterativa de arestas de maior peso da $MST_{est_sampling}$. A cada remoção da aresta de maior peso, um nível hierárquico é constituído contendo os componentes conectados que denotam os grupos do nível em questão.

Além disso, cada grupo que compõe a árvore de grupos tem sua estabilidade (baseando-se na medida de qualidade de excesso de massa proposto em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015)) calculada no processo iterativo de construção de hierarquia de *HDBSCAN**. Esse processo de construção termina quando não há mais arestas a serem removidas da $MST_{est_sampling}$. Por fim, ao percorrer a árvore de grupos resultante do processo citado, os grupos significativos são determinados segundo características de sua estabilidade.

Como descrito na Subseção 3.2.1, a estabilidade de cada grupo formado pelos objetos de um conjunto completo de dados pode ser obtida e atualizada pela Equação 3.2.1 durante a construção da hierarquia de *HDBSCAN**. Em relação ao uso de amostras de dados, esse procedimento também é realizado. Contudo, para estimar a estabilidade de qualquer grupo de amostras em relação ao conjunto inteiro de objetos, deve-se considerar os objetos a elas atribuídos. Ou seja, é preciso encontrar, para cada objeto do conjunto de dados, o seu objeto mais próximo no conjunto de amostras *S*, contabilizar o número de objetos atribuídos a cada amostra e adaptar a Equação 3.2.1 para que seja considerada a cardinalidade de objetos em cada uma das amostras em *S*. Desse modo, a Equação 5.1 é utilizada para estimar a estabilidade de um grupo de amostras C_i :

$$S(C_i) = \sum_{\mathbf{x}_j \in C_i} n_j (\lambda_{max}(\mathbf{x}_j, C_i) - \lambda_{min}(C_i)) = \sum_{\mathbf{x}_j \in C_i} n_j \left(\frac{1}{\epsilon_{min}(\mathbf{x}_j, C_i)} - \frac{1}{\epsilon_{max}(C_i)} \right) \quad (5.1)$$

em que $\mathbf{x}_j \in C_i$ e n_j representa o número de objetos atribuídos ao objeto $\mathbf{x}_j$.

Então, na Figura 14, uma partição composta pelos grupos mais significativos (usando método de extração de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015)) é obtida a partir da hierarquia *H* construída sobre as amostras de dados. Como dito anteriormente, na Figura 14, três grupos (C_1 , C_2 e C_3) são obtidos e denominados por *HDBSCAN** como sendo os mais significativos do conjunto de amostras. Ainda na Figura 14, dois objetos (nomeados como *o*) são descritos como ruídos no modelo. Portanto, esses objetos não pertencem a nenhum grupo legítimo C_i . Ou seja, esses objetos não pertencem a qualquer grupo C_i , tal que $|C_i| \geq m_{clSize}$, em que $|\cdot|$ denota a cardinalidade de objetos do *i*-ésimo grupo C_i e $1 \leq i \leq k$.

Figura 14 – Grupos mais significativos de amostras de dados baseando-se nas amostras apresentadas na Figura 12.

5.3 Inclusão de Ruído

Assim como definido em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), a partir de um conjunto de amostras de dados $\mathbf{S}$, a partição extraída por *HDBSCAN** é composta por grupos significativos e um conjunto de objetos considerados como ruído. Como citado anteriormente, essa partição é denotada como:

1. $\mathbf{S} = P_{\mathbf{S}} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k, \mathbf{O}\}$ e,
2. $|\mathbf{S}| = (\sum_{i=1}^k |\mathbf{C}_i|) + |\mathbf{O}|$.

em que $\mathbf{C}_i$ representa o i -ésimo grupo legítimo, $\mathbf{O}$ o conjunto de objetos considerados como ruído no modelo e $|\cdot|$ a cardinalidade de $\mathbf{C}_i$ ou de $\mathbf{O}$.

Porém, a abordagem utilizada por *Recursive Sampling* requer que os objetos em $\mathbf{O}$ sejam recolocados no processo hierárquico, ou seja, incluídos aos seus grupos legítimos mais próximos. Pois, a cada nível construído por *Recursive Sampling*, os objetos em $\mathbf{O}$ são considerados como ruído apenas no contexto das amostras aleatórias de dados em $\mathbf{S}$ e não sobre o conjunto completo de dados (SYED, 2014). Nesse contexto, a inclusão desses objetos contém os seguintes passos:

1. Para cada $\mathbf{o} \in \mathbf{O}$ encontra-se o vizinho mais próximo em $P_{\mathbf{S}} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$, em que $NN(\mathbf{o}, \mathbf{S}) = \mathbf{x}_j$ denota o vizinho mais próximo de $\mathbf{o}$ em $\mathbf{S}$, segundo alguma medida de proximidade (e.g., distância euclidiana).

2. Obtenha o grupo C_i de rótulo i que contenha $NN(\mathbf{o}, \mathbf{S}) = \mathbf{x}_j$, ou seja, $NN(\mathbf{o}, \mathbf{S}) = \mathbf{x}_j \in C_i$.
3. Inclui o objeto $\mathbf{o}$ ao grupo C_i , i.e., $C_i = C_i \cup \{\mathbf{o}\}$.

Para elucidar esse processo, a Figura 15 é uma continuação do exemplo apresentado na Figura 14, em que dois objetos determinados como ruídos (indicados pelas setas na Figura 15) pelo modelo são incluídos aos seus respectivos grupos mais próximos.

Figura 15 – Inclusão de ruídos aos seus respectivos grupos mais próximos

5.4 Arestas Intergrupos

À medida que os dados são subdivididos em partes menores para que possam ser processados por diferentes unidades de processamento em paralelo, obter arestas (denominadas como E_{inter}) que conectam diferentes grupos no processo recursivo de *Recursive Sampling* é necessário, pois a criação de uma *MST* global sobre o conjunto completo dos dados depende dessas conexões. Em Syed (2014), são descritos duas maneiras de obter E_{inter} : a primeira corresponde ao uso de todos os objetos do conjunto de dados e a segunda utiliza apenas o conjunto de amostras.

1. **Utilizando todos os objetos:** A cada subdivisão feita por *Recursive Sampling*, novos grupos são formados a partir de amostras aleatórias, i.e., $P_S = \{C_1, C_2, \dots, C_k\}$. Os objetos não amostrados em U são atribuídos aos seus respectivos grupos mais próximos em $P_S = \{C_1, C_2, \dots, C_k\}$. Então, nessa abordagem, para cada par de objetos pertencentes a grupos distintos, é calculada a distância (e.g., usando distância *euclidiana*) entre eles. Por

fim, as menores distâncias que separam pares de grupos distintos em $P_S = \{C_1, C_2, \dots, C_k\}$ formam um grafo completo (denotado por G_{inter}), em que os vértices são os objetos mais próximos entre os grupos distintos e as arestas são as distâncias entre esses objetos (vértices). Logo, uma MST_{inter} é obtida a partir de G_{inter} e, posteriormente, armazenada no conjunto de *arestas intergrupos* (E_{inter}).

2. **Utilizando arestas existentes:** Nessa abordagem, é utilizado apenas o modelo construído a partir das amostras aleatórias de dados para obtenção das arestas, ou seja, a partição com os grupos mais significativos ($P_S = \{C_1, C_2, \dots, C_k\}$) e a MST criada sobre essas amostras. A proposta dessa abordagem é utilizar ambas para diagnosticar as arestas intergrupos E_{inter} a partir da característica de transitividade em grafos (GODSIL; ROYLE, 2013). Essa característica garante que cada vértice da MST seja alcançável a todos outros vértices (SYED, 2014). Dito isso, essa abordagem pode ser dividida nos seguintes passos:

- a) Rotula todos os vértices (objetos) da MST de acordo com os rótulos estabelecidos a cada objeto em $P_S = \{C_1, C_2, \dots, C_k\}$. Dessa forma, obtém-se um multigrafo G_{inter} (grafo que pode conter múltiplas arestas com os mesmos nós finais), em que cada vértice corresponde a um rótulo de grupo (GODSIL; ROYLE, 2013).
- b) Constrói uma MST_{inter} a partir de G_{inter} , eliminando todos pares de vértices em G_{inter} que são do mesmo grupo.
- c) Armazena MST_{inter} em E_{inter} .

5.5 Indução de Partição

A indução de partição é um processo de atribuição de rótulos de grupos determinados a cada amostra de dados aos objetos não amostrados no processo, i.e., aos objetos em U . Em outras palavras, cada objeto pertencente a U recebe o rótulo de grupo atribuído à sua amostra de dados mais próxima. Assim como definido em (SYED, 2014):

1. Para cada $u \in U$ encontra-se o vizinho mais próximo de u em $P_S = \{C_1, C_2, \dots, C_k\}$, em que $NN(u, S)$ denota o vizinho mais próximo de u em S , segundo alguma medida de proximidade (e.g., distância euclidiana).
2. Obtenha o grupo C_i de rótulo i que contenha $NN(u, S)$, ou seja, $NN(u, S) \in C_i$.
3. Atribui o objeto u ao grupo C_i , i.e., $C_i = C_i \cup \{u\}$.
4. Os subconjuntos formados são mutuamente exclusivos, i.e., $C_i \cap C_j = \emptyset, 1 \leq i, j \leq k; i \neq j$.

O resultado da indução neste processo é uma partição $P_X = \{X_1, X_2, \dots, X_k\}$, composta por k subconjuntos de dados. Cada subconjunto de dados X_i contém tanto objetos não amostrados pertencentes a U quanto amostras em S , cujo, possuem inter-relacionamentos entre si.

Adicionalmente, nas Figuras 16 e 17 são apresentados exemplos de indução de partição sobre o conjunto completo de dados (apresentado na Figura 11) utilizando os grupos mais significativos encontrados sobre amostras aleatórias (apresentadas na Figura 12).

Figura 16 – Grupos mais significativos sobre amostras aleatórias.

Figura 17 – Indução de partição sobre o conjunto completo de dados.

A partir desse ponto, cada subconjunto de dados induzido pode ser submetido e subdividido recursivamente por uma unidade de processamento independente até que seja suportado pela mesma e uma MST_{local} obtida sobre o mesmo (SYED, 2014).

5.6 Data Bubbles

Como pode ser visto na Seção 5.1, os procedimentos de *Recursive Sampling* são descritos com a utilização de amostras aleatórias simples dos dados. Entretanto, essa abordagem tem sérias limitações, pois, uma vez que os conjuntos de dados para análise são desconhecidos, existe uma dificuldade de selecionar amostras de dados que representem de maneira satisfatória esse conjunto de dados (TAN *et al.*, 2005). Um exemplo deste problema citado pode ser visto na Figura 18.

Outro problema é que algoritmos de agrupamento hierárquico de dados (e.g., *OPTICS* (ANKERST *et al.*, 1999a)) que utilizam como recurso de sumarização de dados as amostras aleatórias não apresentam bom desempenho em termos de qualidade, pois as medidas de distância (densidade) utilizadas por esses algoritmos não conseguem representar a real distância (densidade) sobre o conjunto de dados completo com o uso apenas de amostras no processo de agrupamento (BREUNIG *et al.*, 2001).

Note que na Figura 18(b) tem-se uma representação aproximada da estrutura real dos dados apresentados na Figura 18(a). Por outro lado, na Figura 18(c) os dados do grupo dois (C2)

Figura 18 – (a) Conjunto completo de dados. (b) Boa amostra representativa dos dados. (c) Amostra não representativa dos dados.

são praticamente eliminados da amostra, causando, assim, uma perda significativa da estrutura dos dados originais (BREUNIG *et al.*, 2001; TAN *et al.*, 2005).

- As medidas de distância entre amostras aleatórias, que são objetos representativos dos objetos não amostrados, não expressam bem a distância entre os objetos que essas amostras representam.
- As medidas de estimativas densidade aplicadas apenas em amostras aleatórias não apresentam bom desempenho em termos de qualidade quando comparadas ao conjunto completo dos dados (ANKERST *et al.*, 1999a).

Perante isso, técnicas sofisticadas de sumarização de dados foram propostas para reduzir os problemas citados anteriormente (ZHANG *et al.*, 1996; BREUNIG *et al.*, 2001). Uma das primeiras utilizadas para esse propósito é a sumarização de dados por estatísticas suficientes $CF = (n, LS, SS)$ introduzidas no algoritmo *BIRCH* (ZHANG *et al.*, 1996). Como descrito na Subseção 2.2.2.6, cada vetor CF é capaz de armazenar características de subconjuntos de dados, tais como: número de objetos, soma linear (LS) e soma quadrática (SS) de cada objeto, de maneira que a representação do conjunto de dados como um todo seja melhor em relação ao uso de simples amostras aleatórias dos dados (ZHANG *et al.*, 1996).

Pelo uso de estatísticas suficientes CF como alternativa de sumarização de dados, alguns algoritmos particionais possuem bom desempenho qualitativo quando submetidos a essa técnica (e.g. *kmeans* (MACQUEEN, 1967)). Entretanto, o mesmo não ocorre para algoritmos hierárquicos (e.g. *OPTICS* (ANKERST *et al.*, 1999a)), pois esses são baseados na proximidade entre cada par de objetos do conjunto de dados, e essa medida pode não ser bem representada apenas pela distância entre os objetos representados por CF s, provocando também distorções estruturais de conjuntos de dados. Logo, o método *Data Bubbles* foi proposto para minimizar esses problemas (BREUNIG *et al.*, 2001).

5.6.1 Definições

A principal proposta do método *Data Bubbles* é combinar a utilização de estatísticas suficientes (CF), introduzidas em [Zhang et al. \(1996\)](#), com medidas de dissimilaridades eficientes para resolver problemas de distorções estruturais presentes nos métodos descritos anteriormente ([BREUNIG et al., 2001](#)). À vista disso, serão apresentados detalhes sobre este método de sumarização de dados.

Seja $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ um conjunto de dados composto por n objetos multidimensionais, a extração de *Data Bubbles* a partir de $\mathbf{X}$ é feita pela escolha de $\mathbf{S}$ amostras aleatórias de tamanho m . Então, a amostra $\mathbf{s}_i \in \mathbf{S}$ é o objeto representativo de um objeto não amostrado $\mathbf{u} \in \mathbf{U}$ quando $\arg\min_{\mathbf{s}_i \in \mathbf{S}} d(\mathbf{s}_i, \mathbf{u})$ ([BREUNIG et al., 2001](#)).

Definição 1. (*Data Bubbles*): Considere $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ um conjunto de dados composto por n objetos. Um *Data Bubble* a partir de $\mathbf{Y}$ é definido por uma tupla $B_{\mathbf{Y}} = (rep, n, extent, nnDist)$, em que:

- rep : caracteriza objeto representativo de $\mathbf{Y}$.
- n : indica o número de objetos contidos em $\mathbf{Y}$.
- $extent$: representa um número real onde "a maioria" dos objetos de $\mathbf{Y}$ estão localizados dentro de um raio em torno de rep .
- $nnDist$: é uma função que denota a distância média estimada de k -vizinhos mais próximos dentro de um subconjunto de objetos em $\mathbf{Y}$ para um valor de $k = 1, \dots, k = m_{pts}$. Uma particular ($knn - distancia$) esperada em $B_{\mathbf{Y}}$ é denotada por $nnDist(k, B_{\mathbf{Y}})$.

Definição 2. (Distância entre dois *Data Bubbles*): Seja $B = (rep_B, n_B, e_B, nnDist_B)$ e $C = (rep_C, n_C, e_C, nnDist_C)$ dois *Data Bubbles*. Então, a distância entre B e C é definida como:

$$d(B, C) = \begin{cases} 0 & \text{se } B = C \\ d(rep_B, rep_C) - (e_B + e_C) + nnDist(1, B) + nnDist(1, C) & \text{se } d(rep_B, rep_C) - (e_1 + e_2) \geq 0 \\ \max(nnDist(1, B), nnDist(1, C)) & \text{caso contrário} \end{cases} \quad (5.2)$$

Basicamente, quando um *Data Bubble* B é igual a um outro *Data Bubble* C , a distância entre eles é igual a 0. De maneira complementar, se a distância entre dois objetos representativos rep_B e rep_C de dois *Data Bubbles* B e C menos a soma de seus respectivos raios (i. e., e_1 e e_2) for maior ou igual a 0, então esses *Data Bubbles* não apresentam sobreposição entre eles. Logo, a distância entre o *Data Bubble* B e o *Data Bubble* C é dada pela subtração entre a distância

dos objetos representativos rep_B e rep_C e seus respectivos e_B e e_C somados a suas respectivas distâncias estimadas dos vizinhos mais próximos. Por outro lado, se a distância entre dois objetos representativos rep_B e rep_C de dois *Data Bubbles* B e C menos seus respectivos raios e_1 e e_2 for menor do que 0, a distância entre esses *Data Bubbles* é definida pelo máximo entre a distância média estimada de B e C . Nesse caso, os *Data Bubbles* B e C são sobrepostos. A Figura 19 é apresentada como forma de ilustração das duas situações citadas anteriormente:

Figura 19 – (a) Representa dois *Data Bubbles* B e C que não são sobrepostos (b) Indica dois *Data Bubbles* B e C que são sobrepostos.

Quando *Data Bubbles* são agrupados por algoritmos hierárquicos, tal como: *Single-linkage* (HAND, 2008); as definições apresentadas até o presente momento são suficientes para o agrupamento de *Data Bubbles*. Todavia, para a utilização de *Data Bubbles* por algoritmos de agrupamento de dados mais sofisticados, neste caso o *HDBSCAN**, será necessário definir outras noções apropriadas de *distância core* e *distância de acessibilidade mútua*. Basicamente, estas definições a seguir são modificações das definições apresentadas em Ankerst et al. (1999a).

O método *Data Bubbles* é capaz de fornecer uma consistente ferramenta para a aplicação de diversos métodos hierárquicos, especificamente, os métodos *OPTICS* (ANKERST et al., 1999a) e *HDBSCAN** (CAMPELLO et al., 2015). Nesse contexto, a utilidade desse método estende-se para aplicações sobre vetores no espaço Euclidiano utilizando estatísticas suficientes (n, LS, SS) (BREUNIG et al., 2001).

Dado um *Data Bubble* $B_Y = (rep, n, extent, nnDist)$, é necessário descrever seus componentes. Diante disso, em um objeto representativo (*Data Bubble*) B_Y , rep indica a média (centro) dos vetores de Y , n indica a cardinalidade de um conjunto Y e $extent$ representa a dispersão dos objetos em relação à média do conjunto Y . Sob essa mesma perspectiva, a distância de k -vizinhos mais próximos esperados pode ser calculada segundo o Lema 1 (BREUNIG et al., 2001).

Lema 1. (distância de k -nn esperados de um vetor no espaço Euclidiano) Seja Y um conjunto com n objetos d -dimensional. Se os n objetos estão uniformemente distribuídos em torno de

uma esfera com centro c e raio r . Então a distância esperada de k -vizinhos mais próximos dentro de $\mathbf{Y}$ é igual a:

$$\left(\frac{k}{n}\right)^{1/d} \cdot r \quad (5.3)$$

Prova. O volume de uma esfera d -dimensional de raio r é igual a:

$$V_S(r) = \frac{\sqrt{\pi^d}}{\Gamma(\frac{d}{2} + 1)} \cdot r^d \quad (5.4)$$

em que, Γ é a função *Gamma*. Se os n objetos estiverem uniformemente distribuídos em torno dessa esfera, espera-se um objeto no volume $V_S(r)/n$ e k objetos no volume $kV_S(r)/n$. Assim, a distância esperada de k -vizinhos mais próximos é igual a um raio r' de uma esfera com este volume $kV_S(r)/n$ (BREUNIG *et al.*, 2001). Por uma simples transformação algébrica, tem-se que:

$$r' = \left(\frac{k}{n}\right)^{1/d} \cdot r \quad (5.5)$$

através dessas noções descritas anteriormente, *Data Bubbles* sob vetores no espaço Euclidiano pode ser definido como:

Definição 3. (*Data Bubbles no espaço Euclidiano*): Considere $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ sendo um conjunto de dados multidimensional. Logo, um *Data Bubble* $B_{\mathbf{Y}}$ de $\mathbf{Y}$ é dado como uma tupla $B_{\mathbf{Y}} = (rep, n, extent, nnDist)$, no qual

$$rep = \frac{\sum_{i=1}^n \mathbf{y}_i}{n} \quad (5.6)$$

rep representa o centro do conjunto $\mathbf{Y}$.

$$extent = \sqrt{\frac{\sum_{i=1}^n \sum_{j=1}^n (\mathbf{y}_i - \mathbf{y}_j)^2}{n(n-1)}} \quad (5.7)$$

$extent$ representa o raio de $\mathbf{Y}$ e $nnDist$ é denotada por,

$$nnDist(k, B) = \left(\frac{k}{n}\right)^{1/d} \cdot extent \quad (5.8)$$

Corolário. Seja um *Data Bubble* $B_Y = (rep, n, extent, nnDist)$ para um conjunto de dados $Y = y_1, y_2, \dots, y_n$ composto por n objetos, considere LS como a soma linear e SS a soma quadrática dos objetos de Y . Logo, tem-se:

$$rep = \frac{LS}{n} \quad (5.9)$$

$$extent = \sqrt{\frac{2 \cdot n \cdot SS - 2 \cdot LS^2}{n \cdot (n - 1)}} \quad (5.10)$$

Definição 4. (*Distância Core de um Data Bubble B*): Considere $B = (rep_B, n_B, extent_B, nnDist_B)$ como um *Data Bubble*, m_{pts} um número natural e N o conjunto de todos os *Data Bubbles*. Então, a *distância core* é definida como:

$$d_{core}(B) = d(B, C) + nnDist(k, C) \quad (5.11)$$

em que C e k são dados como: $C \in N$ contém distância máxima $d(B, C)$ tal que $\sum_{Y \in N} n < m_{pts}$, tal que $d(B, Y) < d(B, C)$ e $k = m_{pts} - \sum_{Y \in N} n$, tal que $d(B, Y) < d(B, C)$.

Basicamente, se B sumariza pelo menos m_{pts} objetos, então $B = C$ e $d(B, C) = 0$. Logo, a distância *core* de B é representada por $nnDist(m_{pts}, B)$. Por outro lado, se B não sumariza pelo menos m_{pts} objetos, estima-se m_{pts} -distance em relação a rep_B considerando as distâncias entre todos os *Data Bubbles* $Y \in N$ e B , em que cada Y esteja mais próximo de B do que de C . Nestes casos, todos os *Data Bubbles* mais próximos de B , com exceção de C que contém distância máxima $d(B, C)$ definida na Equação 5.11, precisam ter juntos pelo menos m_{pts} objetos. Então, adiciona-se em C uma distância de k -vizinhos mais próximos estimada, em que k é obtido pela subtração de m_{pts} , o número de objetos de todos os *Data Bubbles* mais próximos de B do que de C . As Figuras 20 e 21 representam as situações descritas anteriormente.

Diferentemente da definição de distância *core* de um *Data Bubble* B proposta para o método *OPTICS* em Breunig et al. (2001), na definição de distância *core* do método *HDBSCAN** não é necessário considerar o parâmetro ε como limiar de densidade, uma vez que esse parâmetro não existe nesse algoritmo.

Definição 5. (*Distância de Acessibilidade Mútua entre Data Bubble B e Data Bubble C*): Dado $B = (rep_B, n_B, extent_B, nnDist_B)$ e $C = (rep_C, n_C, extent_C, nnDist_C)$ com relação a m_{pts} , pode-se definir a *distância de acessibilidade mútua* $d_{mreach}(B, C)$ entre dois *Data Bubbles* B e C , tal como:

$$d_{mreach}(B, C) = \max(d_{core}(B), d_{core}(C), d(B, C)) \quad (5.12)$$

Figura 20 – *Data Bubble B* sumariza pelo menos m_{pts} objetos. Dessa forma, a distância *core* é representada por $nnDist(m_{pts}, B)$.

Figura 21 – *Data Bubble B* não sumariza pelo menos m_{pts} objetos. Neste caso, a área demarcada por **P** indica que todos os *Data Bubbles A, D e E* são mais próximos de *B* do que *C*, juntos sumarizam pelo menos m_{pts} objetos.

Ao contrário da definição de distância de acessibilidade introduzida em [Ankerst et al. \(1999a\)](#), a distância de acessibilidade mútua, com relação a d_{mreach} proposta em [Campello et al. \(2013a\)](#) considera a simetria entre objetos. Tal característica possibilita a criação de um grafo de proximidade em um espaço transformado e.r.c d_{mreach} . Com isso, assumindo $\mathbf{y}_p \in \mathbf{Y}$ e $\mathbf{y}_q \in \mathbf{Y}$, a $d_{mreach}(\mathbf{y}_p, \mathbf{y}_q)$ é obtida pelo máximo entre as distâncias de $\mathbf{y}_p$ em relação a $\mathbf{y}_q$, a $d_{core}(\mathbf{y}_p)$ e a $d_{core}(\mathbf{y}_q)$. Analogamente, a distância de acessibilidade mútua entre *Data Bubbles* segue esse mesmo raciocínio.

5.7 Implementação de Recursive Sampling utilizando Data Bubbles

A técnica de *Data Bubbles* pode ser facilmente adaptada ao contexto do algoritmo *Recursive Sampling* ([SYED, 2014](#)). Assim, os passos técnicos desse algoritmo utilizando essa abordagem de sumarização de dados podem ser descritos pelo Algoritmo 5.

Como pode ser visto, no Algoritmo 5 é descrito um procedimento chamado *RECURSIVESAMPLING*, que tem como entrada um conjunto de dados $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ composto por n objetos multidimensionais, o parâmetro m_{pts} que indica o número mínimo de objetos em uma determinada vizinhança, m_{clSize} o número mínimo de objetos pertencentes a um grupo legítimo (válido), m o número de *Data Bubbles* a serem extraídos a partir de $\mathbf{X}$ e τ a capacidade de processamento de cada unidade de processamento.

A partir dessas entradas, é verificado dentro desse procedimento se $\mathbf{X}$ é suportado por alguma unidade de processamento (Linha 2). Assim, se essa condição é satisfeita, é construída

Algoritmo 5 – Algoritmo *Recursive Sampling*

Entrada: Seja $\mathbf{X}$ um conjunto de dados multidimensional, m_{pts} o número mínimo de objetos em uma determinada vizinhança, m_{clSize} o número mínimo de objetos pertencentes a um grupo legítimo (Opcional), m o número de *Data Bubbles* a serem extraídos a partir de $\mathbf{X}$ e τ a capacidade de processamento de uma unidade de processamento.

Saída: Árvores Geradoras Mínimas (MST_{local}) e Arestas Intergrupos E_{inter} .

```

1: procedimento RECURSIVESAMPLING( $\mathbf{X}$ ,  $m_{pts}$ ,  $m_{clSize}$ ,  $m$ ,  $\tau$ )
2:   se  $|\mathbf{X}| < \tau$  então
3:      $MST_{local} \leftarrow \text{CALCULA\_MST}(\mathbf{X}, m_{pts})$ 
4:     Salva  $MST_{local}$  em Saída_Arestas.
5:   senão
6:      $S_{\mathbf{X}} \leftarrow \text{RETIRA\_AMOSTRAS}(\mathbf{X}, m)$ 
7:      $B_{\mathbf{X}} \leftarrow \text{EXTRAI\_BUBBLES}(\mathbf{X}, m)$ 
8:      $P_B = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k, O\} \leftarrow \text{HDBSCAN}^*\_BUBBLES(B_{\mathbf{X}}, m_{pts}, m_{clSize})$ 
9:     Para todo  $\mathbf{o} \in O \mid B_{\mathbf{X}}(\mathbf{o}) = \mathbf{x}_i$  e  $\mathbf{x}_i \in \mathbf{C}_j$ , tal que  $1 \leq j \leq k$ .
10:     $E_{inter} \leftarrow \text{ARESTAS\_INTERGRUPOS}(MST, P_B = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\})$ 
11:    Salva  $E_{inter}$  em Saída_Arestas.
12:    Induz uma partição  $P_{\mathbf{X}} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k\}$ , tal que,  $\mathbf{X}_i = \{\mathbf{x} \in \mathbf{X} \mid B_{\mathbf{X}}(\mathbf{x}) \in \mathbf{C}_i\}$ , em
    que  $1 \leq i \leq k$ 
13:   fim se
14:   para todo  $\mathbf{X}_i$  faça
15:     RECURSIVESAMPLING( $\mathbf{X}_i$ ,  $m_{pts}$ ,  $m_{clSize}$ ,  $m$ ,  $\tau$ )
16:   fim para
17: fim procedimento

```

uma MST_{local} a partir de $\mathbf{X}$ utilizando o Algoritmo 6 (Linha 3). Na Linha 4, a MST_{local} resultante é salva em *Saída_Arestas*, que representa um abstração de um sistema de arquivo ou banco de dados.

Ainda no Algoritmo 5, se a condição apresentada na Linha 2 não for satisfeita, então, na Linha 7, um conjunto de *Data Bubbles*, denotado como $B_{\mathbf{X}}$, é extraído a partir de $\mathbf{X}$ utilizando o Algoritmo 7. Os *Data Bubbles* em $B_{\mathbf{X}}$, dessa forma, são utilizados para construir uma hierarquia de *HDBSCAN** com o Algoritmo 8 (Linha 8).

Na Linha 9 acontece o processo de inclusão de ruídos (*Data Bubbles*) aos seus grupos mais próximos. Esse procedimento é semelhante ao que é feito para amostras aleatórias (descrito em detalhes na Subseção 5.3).

Na Linha 10 as *arestas intergrupos* E_{inter} são recuperadas utilizando a MST e $P_B = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$ construídos sobre *Data Bubbles*. Os passos de recuperação de E_{inter} é descrito pelo Algoritmo 9. As arestas em E_{inter} são salvas em *Saída_Arestas* (Linha 11).

Na linha 12, uma partição é induzida sobre o conjunto de dados $\mathbf{X}$ (denotada por $P_{\mathbf{X}} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k\}$). O processo de indução de partição de *Recursive Sampling* utilizando *Data Bubbles* é o mesmo quando se é utilizado apenas amostras aleatórias de dados (detalhes na Subseção 5.5), i.e., para cada objeto conjunto de dados $\mathbf{X}$, é atribuído o rótulo de grupo definido

ao seu *Data Bubbles* em B_X .

Por fim, na Linha 15, para cada subconjunto de dados $X_i \in X$, o processo recursivo é feito pela invocação do procedimento *RECURSIVESAMPLING*. Intuitivamente, o processo do Algoritmo 5 termina quando não há mais subconjuntos de dados a serem divididos recursivamente. Ou seja, quando for possível obter *MSTs* locais e E_{inter} a partir desses subconjuntos. Assim que as *MSTs* locais são obtidas, as mesmas podem ser utilizadas para compor uma *MST* global e, consequentemente, uma hierarquia de *HDBSCAN** a partir do conjunto completo de dados.

No Algoritmo 6 são apresentados os detalhes de implementação da construção de *MST* baseada na distância de acessibilidade mútua de *HDBSCAN**:

Algoritmo 6 – Calcula *MST* baseada na distância de acessibilidade mútua de *HDBSCAN**

Entrada: Seja X um conjunto de dados multidimensional e m_{pts} o número mínimo de objetos de uma determinada vizinhança.

Saída: MST_{local}

- 1: **função** CALCULA_MST(X, m_{pts})
 - 2: Calcula as distâncias *core* para cada objeto em X .
 - 3: Calcula as distâncias de acessibilidade mútua por demanda e constrói-se a MST_{local} .
 - 4: Obtém MST_{est} a partir da inserção de “self-edges” ponderadas pelas distâncias *core* correspondentes aos objetos na MST_{local} .
 - 5: **fim função**
-

Nota-se que o Algoritmo 6 tem como entrada o conjunto de dados X e o número mínimo de objetos em uma determinada vizinhança, i.e., m_{pts} . A saída desse algoritmo corresponde a uma MST_{local} estendida baseada na distância de acessibilidade mútua de *HDBSCAN** construída a partir de X . Basicamente, na Linha 2, são calculadas as distâncias *core* de cada objeto contido em X . Na Linha 3, as distâncias de acessibilidade mútuas entre todos os pares de objetos em X são calculadas por demanda e a MST_{local} é construída. Por fim, a partir de MST_{local} , na Linha 4 uma MST_{est} localmente é obtida.

Já no Algoritmo 7 são apresentados os passos para a extração de *Data Bubbles* a partir de um conjunto de dados X . Esse algoritmo tem como entrada o conjunto de dados X e o parâmetro m que indica a quantidade de amostras de dados que devem ser retiradas de X . A saída desse algoritmo corresponde a um conjunto de *Data Bubbles* denotados como $B_X = (rep, n, extent, nnDist)$, assim como descritos na Seção 5.6.

Nesse contexto, na Linha 2 do Algoritmo 7, são retiradas m amostras aleatórias S de X . Em sequência, na Linha 3, um conjunto composto por m estatísticas suficientes CF são geradas. Para a construção de cada CF , cada objeto x pertencente ao conjunto de dados X deve ser atribuído a sua amostra mais próxima em S , dessa forma, a soma linear LS , a soma quadrática SS e o número de objetos atribuídos a cada amostra mais próxima (n) são calculados a cada atribuição. O conjunto de CFs resultantes são utilizadas para a extração de *Data Bubbles* em B_X na Linha 4. Intuitivamente, cada *Data Bubble* é extraído segundo as definições apresentadas na

Algoritmo 7 – Extração de *Data Bubbles*

Entrada: Seja $\mathbf{X}$ um conjunto de dados multidimensional e m o número de *Data Bubbles* a serem extraídos de $\mathbf{X}$.

Saída: $B_{\mathbf{X}} = (rep, n, extent, nnDist)$.

- 1: **função** EXTRAI_BUBBLES($\mathbf{X}, m_{pts}$)
- 2: Retira m amostras aleatórias $\mathbf{S}$ de $\mathbf{X}$.
- 3: Gera estatísticas suficientes $CF = \{cf_1, cf_2, \dots, cf_m\}$, em que: $cf_i = (n, LS, SS)$ para cada subconjunto de dados formado a partir de $\mathbf{S}$ tem-se $\{\mathbf{x} \in \mathbf{X} \mid NN(\mathbf{x}, \mathbf{S}) = \mathbf{x}_i\}$
- 4: $B_{\mathbf{X}} = (rep, n, extent, nnDist)$ a partir de $CF = \{cf_1, cf_2, \dots, cf_m\}$.
- 5: **fim função**

Seção 5.6.

No Algoritmo 8 são apresentados os passos para a obtenção de hierarquia de *HDBSCAN** a partir de *Data Bubbles*. Esse algoritmo tem como entrada $B_{\mathbf{X}}$ um conjunto de *Data Bubbles* extraídos a partir de um conjunto de dados $\mathbf{X}$, m_{pts} o número mínimo de objetos (nos *Data Bubbles*) em uma determinada vizinhança de *Data Bubbles* e m_{clSize} o número mínimo de objetos pertencentes a um grupo legítimo, este último, assim como no algoritmo *HDBSCAN** original, aplicado a *Data Bubbles* também é um parâmetro opcional. A saída desse algoritmo é uma hierarquia de *HDBSCAN** sobre *Data Bubbles*, i.e., árvore de grupos e grupos significativos de *Data Bubbles*.

Em resumo, o Algoritmo 8 retorna uma hierarquia de grupos representada pela árvore de grupos (como descrito no Capítulo 3) e uma partição composta pelos grupos significativos e pelo conjunto de *Data Bubbles* considerados como ruídos no modelo (denotados por $P_B = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k, O\}$). Assim como é feita a construção de hierarquia (árvore) e cálculo de estabilidade de qualquer grupo sobre amostras aleatórias (como descrito na Subseção 5.2), para *Data Bubbles* o mesmo é feito para a construção de hierarquia e para o cálculo de estabilidade, considera-se a cardinalidade de cada *Data Bubble* pertencente a um determinado grupo, portanto, a Equação 5.1 também é utilizada nesse contexto.

Como descrito na Seção 5.6, algumas adaptações para a aplicação desse algoritmo sobre *Data Bubbles* são necessárias. Então, na Linha 2, para cada *Data Bubble* em $B_{\mathbf{X}}$, é calculada a distância *core*, segundo a Equação 5.11 descrita em detalhes na Seção 5.6. Em seguida, na Linha 3, uma *MST* baseada na distância de acessibilidade de *HDBSCAN** (segundo a Equação 5.12) é obtida a partir dos *Data Bubbles* $B_{\mathbf{X}}$. Na Linha 4, uma $MST_{est_bubbles}$ sobre *Data Bubbles* é obtida. Assim, a hierarquia de *HDBSCAN** (H) utilizando a $MST_{est_bubbles}$ resultante (Linha 6) é construída. Por fim, na Linha 6, os grupos significativos de *Data Bubbles* são extraídos da hierarquia H .

No Algoritmo 9 tem-se os procedimentos realizados para recuperar as arestas intergrupos E_{inter} a partir de uma partição $P_B = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$ em conjunto com uma $MST_{est_bubbles}$ ¹,

¹ Os “self-edges” de todos os vértices podem ser desconsiderados nessa abordagem de recuperação de

Algoritmo 8 – *HDBSCAN** sobre *Data Bubbles*

Entrada: Seja B_X um conjunto de *Data Bubbles* extraídos a partir de um conjunto de dados X , m_{pts} o número mínimo de objetos (*Data Bubbles*) em uma determinada vizinhança (de *Data Bubbles*) e m_{clSize} o número mínimo de objetos pertencentes a um grupo legítimo (parâmetro opcional).

Saída: hierarquia de *HDBSCAN** sobre *Data Bubbles*.

- 1: **função** *HDBSCAN*_BUBBLES*(B_X, m_{pts}, m_{clSize})
- 2: Calcula a distância *core* de cada *Data Bubble* B_X utilizando a Equação 5.11.
- 3: Calcula as distâncias de acessibilidade mútua (Equação 5.12) para qualquer B_X e constrói a *MST*.
- 4: Obtém a *MST* estendida $MST_{est_bubbles}$ a partir de *MST* utilizando as distâncias *core* de cada B_X .
- 5: Constrói hierarquia de *HDBSCAN** (H) a partir de $MST_{est_bubbles}$.
- 6: Extrai grupos mais significativos $P_B = \{C_1, C_2, \dots, C_k, O\}$ a partir de H .
- 7: **fim função**

ambos obtidos sobre *Data Bubbles*. Então, esse algoritmo tem como entrada uma $MST_{est_bubbles}$, construída a partir de *Data Bubbles*, e a partição B_X e uma partição $P_B = \{C_1, C_2, \dots, C_k\}$ composta pelos grupos significativos do conjunto de *Data Bubbles*.

Algoritmo 9 – Recupera arestas intergrupos E_{inter}

Entrada: Seja $MST_{est_bubbles}$ construída a partir de *Data Bubbles* (B_X) e $P_B = \{C_1, C_2, \dots, C_k\}$ os grupos significativos extraídos sobre B_X .

Saída: E_{inter}

- 1: **função** *ARESTAS_INTERGRUPOS*($MST_{est_bubbles}, B_X$ e $P_B = \{C_1, C_2, \dots, C_k\}$)
- 2: Rotula todos os vértices de $MST_{est_bubbles}$ correspondentes aos objetos em $P_B = \{C_1, C_2, \dots, C_k\}$
- 3: Cria um multigrafo G_{inter} a partir da $MST_{est_bubbles}$ rotulada.
- 4: Obtém E_{inter} a partir G_{inter} eliminando pares de vértices que contêm o mesmo rótulo de grupo.
- 5: **fim função**

Nota-se que pelo Algoritmo 9, na Linha 2, um processo de rotulação dos vértices da $MST_{est_bubbles}$ é feito com base na partição $P_B = \{C_1, C_2, \dots, C_k\}$. Nesse caso, cada vértice (objeto) da $MST_{est_bubbles}$ correspondente na partição $P_B = \{C_1, C_2, \dots, C_k\}$ recebe o rótulo de grupo estabelecido na partição (assim como descrito na Seção 5.4). Na Linha 3, um multigrafo G_{inter} é obtido a partir da $MST_{est_bubbles}$ rotulada. Como dito anteriormente, G_{inter} é composto por vértices (objetos) representados pelos seus rótulos de grupos. Então, na Linha 4, uma E_{inter} é obtida a partir de G_{inter} por meio da eliminação de pares de vértices que contêm o mesmo rótulo de grupo. Por fim, o Algoritmo 9 tem como saída as arestas intergrupos representadas por E_{inter} .

5.8 Implementação de *Recursive Sampling* baseada em Memória Compartilhada

Assim como descrito na Seção 5.1, o algoritmo *Recursive Sampling* é um algoritmo sofisticado de paralelização de *HDBSCAN** (SYED, 2014). Esse algoritmo serve como um *framework* que pode ser adaptado para diferentes arquiteturas de computação paralela, tais como: a arquitetura de memória compartilhada (VALDURIEZ, 2009).

Nessa arquitetura, recursos de memória primária e memória secundária são compartilhados com várias unidades de processamento independentes (EL-REWINI; ABD-EL-BARR, 2005). Dessa forma, os núcleos podem trabalhar independentemente em paralelo a partir dos mesmos dados armazenados em memória primária ou secundária, aproveitando ao máximo esses recursos.

O uso de computação paralela permite a escalabilidade de diversos processos de agrupamento de quantidades massivas de dados (KIM, 2009). Isso não é diferente para o algoritmo *Recursive Sampling*, pois, a paralelização permite que *Recursive Sampling* processe vários subconjuntos de dados a partir de sua intuição apresentada na Seção 5.1. Diante disso, é proposta, neste trabalho de mestrado, uma implementação de *Recursive Sampling* aplicada ao cenário de computação paralela baseada em memória compartilhada. Adicionalmente, essa implementação segue as duas abordagens de sumarização de dados apresentadas na 5.1, i.e., amostras aleatórias e *Data Bubbles*.

A ideia da implementação de *Recursive Sampling* proposta neste trabalho consiste em considerar que cada subconjunto de dados possa ser exclusivamente processado por uma unidade de processamento. Logo, dado um subconjunto de dados $\mathbf{X}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ composto por n objetos multidimensionais, $\mathbf{X}_i$ é atribuído a uma unidade de processamento P_i e, assim, se P_i for capaz de processar localmente todos os objetos de $\mathbf{X}_i$, com relação à sua capacidade de processamento τ , então uma *MST* local baseada em densidade é obtida a partir $\mathbf{X}_i$, como descrito na Seção 5.1. Se o número de objetos pertencentes a $\mathbf{X}_i$ exceder a capacidade de processamento τ de P_i , então um modelo local de *HDBSCAN** é construído sobre amostras aleatórias ou *Data Bubbles* extraídos a partir de $\mathbf{X}_i$ e novos subconjuntos de dados são induzidos a partir de $\mathbf{X}_i$ e submetidos ao mesmo processo realizado em $\mathbf{X}_i$, recursivamente.

Na Figura 22, há um exemplo de execução da implementação de *Recursive Sampling* em ambiente paralelo, proposta neste trabalho de pesquisa, a partir de um conjunto hipotético de dados $\mathbf{X}_1$. Basicamente, o exemplo da Figura 22 consiste em seis unidades de processamento, denotados por P_i . Nesse exemplo é desconsiderada a maneira com que essas unidades de processamento compartilham recursos de memória, justamente para simplificar o processo de execução de *Recursive Sampling*. Nesse contexto, como pode ser visto na Figura 22, *Recursive Sampling* é iniciado a partir de $\mathbf{X}_1$ sendo processado pela unidade de processamento P_1 . Note que nesse exemplo, $\mathbf{X}_1$ não pode ser processado totalmente em uma iteração de *Recursive Sampling*. Então,

Figura 22 – Exemplo de execução de *Recursive Sampling* em ambiente de computação paralela.

um modelo local de *HDBSCAN** é construído a partir de amostras aleatórias (ou *Data Bubbles*) extraídas de X_1 . Esse modelo é composto pelos grupos C_2 , C_3 e C_4 . Adicionalmente, a cada modelo local construído, as arestas intergrupos também são calculadas, assim como descrito na Seção 5.4.

Ainda na Figura 22, os subconjuntos de dados X_2 , X_3 e X_4 são induzidos pela unidade de processamento P_1 a partir dos C_2 , C_3 e C_4 , respectivamente, assim como descrito na Seção 5.5. Então, esses novos subconjuntos de dados são atribuídos a novas unidades de processamento disponíveis para o processamento paralelo dos mesmos. Note que, a partir dos subconjuntos X_3

e $\mathbf{X}_4$, são obtidas *MSTs* locais, pois a capacidade de processamento (τ) das unidades P_3 e P_4 não é superada pelos subconjuntos de dados. Já o subconjunto de dados $\mathbf{X}_2$ é subdividido em dois novos subconjuntos de dados, ($\mathbf{X}_5$ e $\mathbf{X}_6$), e, a partir dos mesmos, são obtidas outras *MSTs* locais. Por fim, todas as *MST locais* obtidas são armazenadas no sistema de arquivo².

Para a construção de hierarquia de *HDBSCAN** a partir do conjunto completo de dados, existe a necessidade de combinar as arestas das *MSTs* locais e arestas intergrupos computadas a cada iteração de *Recursive Sampling*. A combinação consiste em ordenar as arestas em ordem decrescente de peso (distância de acessibilidade mútua). Na implementação apresentada nesta seção, o processo de combinação de arestas é feito como descrito em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Após a combinação das arestas, a hierarquia de *HDBSCAN** é obtida sequencialmente pela remoção iterativa de arestas de maior peso, assim como descrito no Capítulo 3.

Adicionalmente, uma implementação paralela de *Recursive Sampling*, assim como descrito nesta seção, é disponibilizada na linguagem de programação R.

5.9 Considerações Finais

Neste capítulo foram apresentados conceitos fundamentais do algoritmo *Recursive Sampling*, o qual faz parte do núcleo principal deste projeto de mestrado. Especificamente, este capítulo apresenta uma relação direta com o Capítulo 3, pois, *Recursive Sampling* é uma maneira eficiente de paralelização de *HDBSCAN**. Diante disso, esse algoritmo proporciona a este projeto de mestrado uma investigação de alternativas sofisticadas de paralelização de *HDBSCAN**.

² São armazenadas também as arestas intergrupos obtidas a cada modelo local construído sobre amostras ou *Data Bubbles*

IMPLEMENTAÇÃO DE HDBSCAN* USANDO MAPREDUCE

A paralelização do método *HDBSCAN** é uma importante estratégia para aumentar a escalabilidade do processo de agrupamento de dados. Entretanto, esse procedimento é um tanto desafiador, pois é requer-se a comparação par a par de objetos do conjunto de dados, com o objetivo de construir uma *MST* baseada em densidade (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), como descrito no Capítulo 3.

Diante disso, é necessário utilizar maneiras sofisticadas de divisão dos dados em diferentes unidades de processamento independentes (máquinas ou nós) para que possam ser processados separadamente em paralelo. Uma dessas maneiras é baseada no algoritmo *Recursive Sampling*, introduzido primeiramente em (SYED, 2014) e discutido no Capítulo 5 deste trabalho. Com base nisso, o foco deste capítulo é fornecer uma das contribuições deste trabalho, que é uma nova arquitetura de implementação de *Recursive Sampling* baseada no conceito de *MapReduce* (DEAN; GHEMAWAT, 2008).

Assim, para contextualizar a nova arquitetura de implementação de *Recursive Sampling* proposta neste projeto de pesquisa, este capítulo é dividido da seguinte maneira: na Seção 6.1, são apresentados breves conceitos e aplicações de *MapReduce*; a implementação de *Recursive Sampling* utilizando o conceito de *MapReduce* é apresentada na Seção 6.2; por fim, na Seção 6.3, são apresentadas as considerações finais deste capítulo.

6.1 MapReduce

MapReduce é um modelo de programação desenvolvido para computação distribuída e paralela de quantidades massivas de dados (DEAN; GHEMAWAT, 2008). Desenvolvido pelo Google e adotado atualmente pelo projeto *Apache*, *MapReduce* utiliza em seu núcleo conceitos de processamento paralelo e distribuído de dados em múltiplas máquinas interconectadas (cluster

de computadores). Adicionalmente, esse modelo é núcleo de plataformas consolidadas de análise de grandes quantidades de dados, tais como: *Apache Hadoop* e *Apache Spark*; que, por sua vez, são ferramentas bastante utilizadas pela academia e indústria (LAM, 2010; KARAU *et al.*, 2015).

O modelo *MapReduce*, em sua essência, tem o papel de dividir um dado problema em problemas menores que possam ser resolvidos por nós paralelos e independentes e, então, combinar as partes resultantes em uma ou mais soluções. Assim sendo, os nós (também conhecidos como *workers*) representam unidades de processamento interconectadas por uma topologia de redes de computadores e cada uma delas possui memória principal, processadores (ou núcleos) e outros recursos de maneira independentes das outras unidades. O conjunto de unidades de processamento são denominadas na literatura por *cluster* de computadores. Em outras palavras, o modelo de *MapReduce* assemelha-se à estratégia de divisão e conquista, a qual é um fundamento elementar em ciência da computação (CORMEN *et al.*, 2001; MINER; SHOOK, 2012).

Uma característica importante e atrativa de *MapReduce* é que - diferente de tecnologias, tais como *OpenMP* (CHANDRA *et al.*, 2001) e *MPI* (PACHECO, 1996) - os desenvolvedores não necessitam saber explicitamente como é feita a distribuição ou mesmo a comunicação entre as tarefas de processamento de dados (WHITE, 2009). Dessa forma, o desenvolvedor precisa se preocupar apenas com aspectos teóricos de suas aplicações e com o tratamento dos dados que serão processados por esse modelo (WHITE, 2012).

MapReduce possui aspectos de programação funcional, tais como: as linguagens de programação *Lisp* (GRAHAM, 1993) e *Scala* (ODERSKY *et al.*, 2011). Dito isso, a programação funcional é um paradigma em que funções podem aceitar como argumentos outras funções. Nesse contexto, dado um elemento de entrada em uma determinada função, a saída resultante para esse elemento será única (MICHAELSON, 1989).

Em outras palavras, esse paradigma de programação está relacionado aos conceitos de funções matemáticas, por exemplo, dados dois conjuntos distintos de elementos, A e B , respectivamente, em que os elementos de A correspondem aos elementos de entrada (domínio) da função e B (contradomínio) corresponde à sua saída. Cada elemento em A possuirá uma única imagem no seu contradomínio em B (LANE, 1986).

Dessa forma, várias funções são comuns nesse tipo de paradigma de programação, dois exemplos desse tipo de função são: *mapper* e *fold*, que são funções de transformação e agregação, respectivamente (DEAN; GHEMAWAT, 2008). Dada uma lista de elementos como argumento, a função de *mapper* tem o objetivo de transformar ou representar de outra forma (por uma função) os elementos dessa lista. Já a função *fold* tem a responsabilidade de agregar os elementos da lista segundo algum critério de junção, por exemplo, agregação por meio da soma de todos os elementos da referida lista (GRAHAM, 1993).

De maneira similar às linguagens de programação funcional, *MapReduce* possui fases de transformação (*mapper*) e agregação (*reducer*) de dados (LIN; DYER, 2010). Além dessas duas

fases citadas anteriormente, esse modelo também possui fases de entrada, divisão, mistura (ou *Shuffle*) e saída dos dados (descritos em detalhes nos itens abaixo).

Logo, para o funcionamento dessas fases, o modelo *MapReduce* utiliza como recurso a estrutura de dados baseada em pares do tipo $\langle \text{Chave}, \text{Valor} \rangle$. Nesse contexto, a *Chave* corresponde à identificação de cada dado a ser processado e o *Valor* corresponde à informação que esse dado agrega (LI *et al.*, 2016). Um exemplo de dados modelados por essa estrutura seriam por exemplo grafos, em que os vértices são denominados como *Chave* e suas arestas que ligam a seus vértices adjacentes são denotados como *Valor* (VENNER, 2009). Além disso, o uso desse tipo de estrutura de dados possibilita que dados sejam processados distribuídos e em paralelo por unidades de processamento distintas (HOLMES, 2012).

- **Entrada:** Corresponde ao processo de leitura e escrita dos dados em algum sistema de arquivos distribuídos (do inglês, Distributed File System (*DFS*)).
- **Divisão:** Consiste em dividir os dados dispostos no repositório em partes menores para o envio de cada uma dessas partes para uma função *mapper* específica.
- **Mapper:** Transforma as partes de dados designadas em pares do tipo $\langle \text{Chave}, \text{Valor} \rangle$.
- **Mistura:** Organiza ou ordena os pares resultantes da função *mapper* e envia-os à função *reducer*.
- **Reducer:** Agrega os dados organizados pela função de embaralhamento e retorna novos pares resultantes do tipo $\langle \text{Chave}, \text{Valor} \rangle$.
- **Saída:** Salva a saída da função *reducer* no *DFS*.

Figura 23 – Exemplo de contador de palavras em *MapReduce*.

Outro exemplo clássico de aplicação do modelo *MapReduce* é o contador de palavras (WHITE, 2012). Nesse exemplo, apresentado na Figura 23, o modelo parte de início pela *entrada*

de dados. Essa entrada é composta por algumas palavras dispostas em um ou mais arquivos ou fluxo de dados. A partir desse arquivo, o modelo *divide* os dados em porções menores para que sejam analisados separadamente.

Note que cada porção de dados é submetida à um *mapper* para as devidas transformações nesses dados. Cada *mapper* acionado recupera palavra por palavra e a saída correspondente a cada palavra é um par do tipo $\langle \text{Chave}=\text{palavra}, \text{Valor}=1 \rangle$, em que 1 representa a frequência de determinada palavra transpassada a um *mapper* qualquer.

O processo de *mistura* organiza as palavras iguais em um mesmo fluxo de dados. As saídas do processo de *mistura* ativam *reducers* de acordo com o número de chaves distintas nos dados. É nesse processo que são somadas as ocorrências de cada palavra contida nos dados iniciais de entrada. Por fim, os dados são agregados em uma saída resultante que pode ser armazenada no *DFS*.

6.1.1 Apache Hadoop

Apache Hadoop é uma plataforma (ou *framework*) de código aberto utilizada para criar e executar aplicações paralelas e distribuídas desenvolvidas a partir do conceito de *MapReduce* (VENNER, 2009). Especificamente, esse *framework* analisa em paralelo grandes quantidades de dados em *clusters* de computadores interconectados por rede ou por *computação em nuvem*. O conceito de computação em nuvem é referente a computadores ou servidores interconectados pela internet para designarem diversas tarefas computacionais (ERL *et al.*, 2013). Na Figura 24 é apresentada uma abstração do *framework Apache Hadoop*:

Figura 24 – Framework Apache Hadoop.

Como pode ser visto na Figura 24, o *Apache Hadoop* funciona por meio de requisição

e execução de múltiplas tarefas em paralelo. A requisição parte dos computadores clientes que enviam tarefas independentes ao *cluster* de computadores, os quais recebem essas tarefas, executam determinadas ações em paralelo e retornam as respostas aos clientes (VENNER, 2009).

Uma característica preponderante do *framework Apache Hadoop* é o seu sistema de arquivos distribuídos, conhecido como *Hadoop Distributed File System (HDFS)* (HOLMES, 2012). Os dados que são armazenados no *HDFS* são divididos em blocos de dados menores e espalhados em sobre o *cluster* de computadores. No mais, os dados dos blocos de dados são replicados em diversos computadores do *cluster* a fim de compor um mecanismo consistente de recuperação por falhas (WHITE, 2012).

6.1.2 Apache Spark

Apache Spark é um *framework* de análise massiva de dados distribuídos, também baseado nos conceitos reformulados de *MapReduce* (ZAHARIA *et al.*, 2012). Adicionalmente, esse *framework* implementa um conceito de dados distribuídos resilientes (ou *Resilient Distributed Datasets (RDDs)*). *RDDs*, em sua essência, é considerado como uma abstração de memória primária distribuída que possibilita a implementação de aplicações para gerenciar em memória principal grandes quantidades de dados espalhados no *cluster* de computadores (KARAU *et al.*, 2015).

Diferentemente do modelo de *MapReduce* tradicional implementado no *framework Apache Hadoop*, o *RDD* do *Apache Spark* funciona como uma memória principal distribuída (ZAHARIA *et al.*, 2012). Essa característica permite que os dados sejam recuperados com baixa latência e processados por fases de *mappers*, *reducers* (ZAHARIA *et al.*, 2012).

6.2 Recursive Sampling em MapReduce

Recursive Sampling pode ser facilmente adaptado para conceitos de computação paralela, especificamente o *MapReduce* (descrito na Seção 6.1) (DEAN; GHEMAWAT, 2008). Nessa perspectiva, em Syed (2014) foi proposta uma implementação de *Recursive Sampling* aplicada ao conceito de *MapReduce*, conhecida como *PHDBSCAN*. Basicamente, *PHDBSCAN* transforma o problema de *Recursive Sampling* em uma abordagem iterativa de divisão dos dados para obtenção de um conjunto de *MSTs locais* baseadas em densidade (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Entretanto, *PHDBSCAN* é restrito ao processamento sequencial de cada subconjunto de dados gerado a cada iteração desse algoritmo. Tal fato gera um gargalo em relação ao refinamento de subconjuntos de *Recursive Sampling*. Ainda, esse algoritmo consegue construir apenas uma *MST global* dos dados, deixando de fornecer uma hierarquia de grupos baseados em estimativas de densidade em forma de dendrograma e árvore de grupos simplificada (assim como descrito no Capítulo 3) utilizando o conjunto completo de dados. O ponto crítico de se construir apenas a

MST global é que a partir dela, sem uma hierarquia de grupos, fica impossível realizar etapas de pós-processamento, i.e., extrair grupos significativos e estimar a detecção de *outliers* com base no conjunto completo de dados.

Para eliminar esses problemas, é proposto neste trabalho uma implementação escalável e iterativa de *Recursive Sampling*, intitulada *MapReduce HDBSCAN** (*MR-HDBSCAN**), utilizando o conceito de *MapReduce*. *MR-HDBSCAN** segue o mesmo conceito de hierarquia de níveis de *Recursive Sampling* e também possui uma arquitetura totalmente reformulada em relação ao *PHDBSCAN* para o processamento paralelo de (sub)conjuntos de dados que vão surgindo a cada iteração desse método.

Para que o processamento em paralelo seja possível, *MR-HDBSCAN** utiliza uma sofisticada estratégia de mapeamento e processamento de cada subconjunto de dados induzido a partir de modelos locais de *HDBSCAN** aplicados sobre seus respectivos objetos representativos (*Data Bubbles* ou amostras aleatórias) (descrito no Capítulo 5).

Além disso, *MR-HDBSCAN** fornece não somente uma *MST global* dados, mas, a partir desta, esse algoritmo é capaz de produzir de maneira paralela uma hierarquia de grupos. Ainda, a partir da hierarquia, *MR-HDBSCAN** realiza em paralelo a extração de grupos significativos e detecção *outliers* pelo método *GLOSH* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

O algoritmo *MR-HDBSCAN** é aplicável sobre as duas abordagens de sumarização de dados utilizadas por *Recursive Sampling*, i.e., abordagem baseada em amostras aleatórias ou *Data Bubbles* (ambas descritas no Capítulo 5). Adicionalmente, a implementação de *MR-HDBSCAN** pode ser dividida em quatro principais etapas (apresentadas em alto nível na Figura 25): (1) a de construção de modelos locais de *HDBSCAN** ou *MSTs* locais baseadas na distância de acessibilidade mútua de *HDBSCAN** (Subseção 6.2.1) e a de indução de partição sobre um determinado (sub)conjunto de dados; (2) a de combinação de *MST locais* e *arestas intergrupos* em uma única *MST global* (Subseção 6.2.2) e; (3) a de construção de hierarquia de *MR-HDBSCAN** a partir da *MST global* (descrita em detalhes na Subseção 6.2.3).

Figura 25 – Estrutura hierárquica de MR-HDBSCAN*.

6.2.1 Primeira Etapa: Modelos Locais de HDBSCAN*

Como descrito no Capítulo 5, a proposta de *Recursive Sampling* é subdividir um subconjunto de dados $\mathbf{X}$ em uma partição $\mathbf{P}_{\mathbf{X}}$, que representa grupos (subconjuntos) não são sobrepostos a partir de $\mathbf{X}$. E então, processar, independentemente em paralelo, cada subconjunto em $\mathbf{P}_{\mathbf{X}}$, até que não haja mais subconjuntos a serem processados. Neste contexto, a primeira etapa de *MR-HDBSCAN** consiste em realizar, iterativamente, esse processo paralelo de subdivisão dos dados.

Como pode ser visto na Figura 26, a primeira etapa de *MR-HDBSCAN** pode ser dividida em quatro fases. Note que na ilustração apresentada na Figura 26, para fins de simplicidade, é apresentado o procedimento feito apenas sobre um subconjunto de dados. Entretanto, como pode ser visto na Figura 25, *MR-HDBSCAN** é capaz de processar múltiplos subconjuntos de dados em paralelo à medida que os níveis hierárquicos são construídos.

A primeira fase da primeira etapa de *MR-HDBSCAN** consiste no uso de uma função, conhecida como *MapPartitionsToPair*, apresentada no Algoritmo 10. Diferente do conceito tradicional de *MapReduce*, em que invocações de funções *mapper* são restritamente capazes de processar apenas um registro (objeto) por vez. A função *MapPartitionsToPair* permite processar em paralelo um conjunto de objetos de uma só vez a partir de uma única invocação (DEAN; GHEMAWAT, 2008). Desse modo, a função *MapPartitionsToPair* (Algoritmo 10) processa em paralelo cada subconjunto de dados $\mathbf{X}_{id}$, em que cada objeto $\mathbf{x}_j \in \mathbf{X}_{id}$ é representado por um par do tipo $\langle id, \mathbf{x}_j \rangle$.

Especificamente no Algoritmo 10, se um subconjunto de dados $\mathbf{X}_{id}$ contiver um número de objetos que seja suportado localmente em memória em qualquer unidade de processamento, i.e., $|\mathbf{X}_i| < \tau$, em que $|\mathbf{X}_i| = n$. Então, as distâncias *core* e de acessibilidade mútua (d_{mreach}) são calculadas para cada objeto de $\mathbf{X}_{id}$. Uma *MST* baseada em densidade (denotada como MST_{mreach}) de *HDBSCAN** é obtida. Assim, a MST_{mreach} é estendida de forma que cada vértice (objeto) tenha uma “self-edge” composta pela sua respectiva distância *core*, obtendo então, uma *MST* local estendida (MST_{local}). Cada aresta da MST_{local} é extraída como um vetor $[u, v, d_{mreach}(u, v)]$, em que, u e v são vértices conectados por uma aresta ponderada pelo peso $d_{mreach}(u, v)$, que representa a distância de acessibilidade mútua entre u e v . Por fim, cada vetor (aresta) da MST_{local} é emitida como um par do tipo $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$, em que a chave 1 é utilizada para combinar em uma única solução todas as arestas geradas localmente.

Por outro lado, se o subconjunto de dados $\mathbf{X}_{id}$ não for suportado localmente, então, é retirado um subconjunto de amostras aleatórias $\mathbf{S}_{id} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_m\}$ desse subconjunto de dados¹, e então, cada objeto de $\mathbf{X}_{id}$ é atribuído a sua amostra aleatória mais próxima em $\mathbf{S}_{id}$. Como pode ser visto no Algoritmo 10, cada objeto é mapeado com relação a sua amostra mais

¹ Se vários subconjuntos de dados (não suportados em memória) são processados em paralelo ao mesmo tempo, os subconjuntos de amostras retiradas de cada um deles devem ser compartilhadas para qualquer instância de *MapPartitionsToPair*

próxima pelos pares $\langle \text{AmostraId}, \{\mathbf{x}_j, id\} \rangle$, em que AmostraId representa a identificação da amostra mais próxima de $\mathbf{x}_j$.

Figura 26 – Primeira etapa de $MR-HDBSCAN^*$.

Além da entrada dos pares do tipo $\langle id, \mathbf{x}_j \rangle$ correspondentes a um determinado subconjunto de dados $\mathbf{X}_{id} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, o Algoritmo 10 contém como entrada o número mínimo de objetos de uma determinada vizinhança (m_{pts}) e a capacidade de processamento (τ) de qualquer unidade de processamento. Adicionalmente, é dada como entrada ao Algoritmo 10 um subconjunto de amostras aleatórias $\mathbf{S}_{id}$ ². Cada objeto $s_l \in \mathbf{S}_{id}$, é representado por um par $\langle id, s_l \rangle$, em que o id de qualquer amostra representa o mesmo valor de id do subconjunto de dados em que foi obtida.

Após a criação de pares do tipo $\langle \text{AmostraId}, \{\mathbf{x}_j, id\} \rangle$ sobre os objetos de um subconjunto de dados $\mathbf{X}_{id}$ não suportado em memória, um processo de combinação desses pares é necessário para calcular informações referentes à técnica de sumarização de dados, previamente escolhida³, i.e., amostras aleatórias ou *Data Bubbles*. O termo local corresponde a atualização de informações em cada unidade de processamento independente sem a comunicação entre elas. Já a combinação global é utilizada para combinar, em uma solução geral, as partes computadas localmente por todas as unidades de processamento. Desse modo, no contexto de amostras alea-

² Qualquer subconjunto de dados $\mathbf{X}_{id}$ só terá um subconjunto de amostras aleatórias $\mathbf{S}_{id}$ extraídas se o número de objetos pertencentes a $\mathbf{X}_{id}$ for maior que a capacidade de processamento τ .

³ A escolha da técnica de sumarização de dados é feita apenas uma vez no início do processo e não pode ser modificada durante a execução de $MR-HDBSCAN^*$.

Algoritmo 10 – *MapPartitionsToPair*: Modelo Local de HDBSCAN*.

Entrada: Seja $\mathbf{X}_{id}$ um (sub)conjunto de dados em que cada objeto $\mathbf{x}_j$ em $\mathbf{X}_{id}$ é representado por um par $\langle id, \mathbf{x}_j \rangle$; $\mathbf{S}_{id}$ subconjunto de amostras aleatórias, em que cada objeto $\mathbf{s}_l \in \mathbf{S}_{id}$ é representado por um par $\langle id, \mathbf{s}_l \rangle$; m_{pts} o número mínimo de objetos em uma determinada vizinhança; e τ a capacidade de processamento de qualquer unidade de processamento.

```

1: procedimento MapPartitionsToPair( $\mathbf{X}_{id}, \mathbf{S}_{id}, m_{pts}, \tau$ )
2:   se  $|\mathbf{X}_{id}| < \tau$  então
3:      $d_{core} \leftarrow \text{DistanciaCore}(\mathbf{X}_{id}, m_{pts})$ .
4:      $d_{mreach} \leftarrow \text{DistanciaAcessibilidadeMutua}(d_{core})$ .
5:      $MST_{mreach} \leftarrow \text{ConstroiMST}(d_{mreach})$ 
6:      $MST_{local} \leftarrow \text{EstendeArestas}(MST_{mreach}, d_{core})$ 
7:      $Arestas \leftarrow \text{ExtraiArestas}(MST_{local})$ 
8:     retorna todos os pares  $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$  em Arestas
9:   senão
10:    para todo  $\langle id, \mathbf{x}_j \rangle \in \mathbf{X}_{id}$  faça
11:       $AmostraId \leftarrow \text{AmostraMaisProximaId}(\mathbf{x}_j, \mathbf{S}_{id})$ 
12:      Cria par  $\langle AmostraId, \{\mathbf{x}_j, id\} \rangle$ 
13:    fim para
14:    retorna todos os  $\langle AmostraId, \{\mathbf{x}_j, id\} \rangle$ 
15:  fim se
16: fim procedimento

```

tórias, as combinações locais e globais são utilizadas para contabilizar o número de objetos de um determinado subconjunto de dados $\mathbf{X}_{id}$ atribuídos, em diferentes unidades de processamento, às suas respectivas amostras mais próximas em $\mathbf{S}_{id}$. Já no contexto de *Data Bubbles*, essas combinações são utilizadas para computar as estatísticas suficientes $CF = (n, LS, SS)$ (descritas no Capítulo 5) para a construção dos mesmos.

Considerando que a técnica de sumarização de dados escolhida inicialmente seja a de *Data Bubbles*, na Figura 27 é apresentado um exemplo de combinação local e global de estatísticas suficientes $CF = (n, LS, SS)$ e, conseqüentemente, a construção de *Data Bubbles*. Especificamente, o fluxo de execução da Figura 27 é iniciado a partir de quatro funções *mappers* (uma em cada unidade de processamento hipotética), em que todas têm como entradas objetos $\mathbf{x}_j \in \mathbf{X}_{id}$ mapeados com seus respectivos *ids* (representados pelos pares $\langle id, \mathbf{x}_j \rangle$). Note que existem dois valores de *ids* (1 e 2). Portanto, são dois subconjuntos de dados (e.g., $\mathbf{X}_1 = \{\mathbf{x}_1, \dots, \mathbf{x}_5\}$ e $\mathbf{X}_2 = \{\mathbf{x}_6, \dots, \mathbf{x}_9\}$) a serem processados em paralelo. Observe também a existência de dois subconjuntos de amostras oriundas de $\mathbf{X}_1$ e $\mathbf{X}_2$, representados por $\mathbf{S}_1 = \{\mathbf{s}_1, \mathbf{s}_2\}$ e $\mathbf{S}_2 = \{\mathbf{s}_3, \mathbf{s}_4\}$, respectivamente.

Observe na Figura 27, que as amostras dos subconjuntos $\mathbf{S}_1$ e $\mathbf{S}_2$ são acessíveis por todas as funções *mappers* do exemplo, i.e., por todos os objetos $\{\mathbf{x}_1, \dots, \mathbf{x}_9\}$ desse exemplo. Dessa maneira, os objetos amostrados em $\mathbf{S}_1$ e $\mathbf{S}_2$ podem ser comparados com todos os objetos representados pelos pares $\langle id, \mathbf{x}_j \rangle$. A comparação é feita com base no valor de *id* entre os objetos amostrados ($\mathbf{S}_1$ e $\mathbf{S}_2$) e os não amostrados ($\mathbf{X}_1$ e $\mathbf{X}_2$), ou seja, objetos amostrados de $\mathbf{S}_1$

são comparados apenas com objetos de X_1 e os objetos amostrados de S_2 com os objetos de X_2 . Então, no exemplo da Figura 27, o objeto x_1 é atribuído à sua amostra mais próxima s_2 de identificação igual a 2 e uma tupla é gerada para esse objeto, que é representada por $\langle 2, \{x_1, 1\} \rangle$. O mesmo procedimento é realizado para os demais objetos analisados e suas respectivas amostras mais próximas.

Figura 27 – Extração de *Data Bubbles* baseando-se no conceito de *MapReduce* do algoritmo *MR-HDBSCAN**.

Ainda, como pode ser visto na Figura 27, o processo de combinação é iniciado a partir das entradas de tuplas $\langle Amostrald, \{x_j, id\} \rangle$ (e.g., $\langle 2, \{x_1, 1\} \rangle$). Então, com a combinação local de cada tupla $\langle Amostrald, \{x_j, id\} \rangle$ do exemplo da Figura 27, são retornadas tuplas do tipo $\langle Amostrald, \{CF_{amostrald}, id\} \rangle$, em que $CF_{amostrald}$ representa a estatística suficiente referente à amostra identificada como $Amostrald$. Um exemplo dessa atualização local da Figura 27, que ocorre na Unidade 1 seria a tupla $\langle 2, \{CF_{21,2}, 1\} \rangle$, que combina os atributos dos objetos x_1 e x_2 para a computação das estatísticas suficientes, i.e., soma linear LS , soma quadrática SS e o número de objetos n , referentes à amostra de dados s_2 de identificação igual a 2.

A combinação global de cada estatística suficiente $CF_{amostraId}$ da Figura 27 é realizada pela atualização dessas estatísticas de acordo com as chaves das tuplas $\langle AmostraId, \{CF_{amostraId}, id\} \rangle$. Por exemplo, a tupla $\langle 2, \{CF_{2,1,2}, 1\} \rangle$ (Unidade 1) é combinada globalmente com a tupla $\langle 2, \{CF_{2,5}, 1\} \rangle$ (Unidade 2), dando origem a um *Data Bubble* $b_{2,1,2,5}$ representado por um par $\langle 1, b_{2,1,2,5} \rangle$, em que 1 corresponde ao *id* referente à amostra (2) que originou o *Data Bubble* $b_{2,1,2,5}$. O mesmo procedimento acontece com cada tupla que contém a chave 1, 3 ou 4 e, consequentemente, originando três novos *Data Bubbles* representados por $b_{1,3,4}$, $b_{3,6,8}$ e $b_{4,9}$, respectivamente.

Em termos técnicos, no caso de escolha de *Data Bubbles* como abordagem de sumarização de dados, as combinações locais e globais de estatísticas suficientes são feitas pela Equação 6.1. O objetivo da Equação 6.1 é somar os campos LS , SS e n das CF s parciais (locais) que contêm a mesma identificação (*AmostraId*) a partir do teorema de adicionalidade (ZHANG *et al.*, 1996). A partir das estatísticas atualizadas local e globalmente, *Data Bubbles* são extraídos por meio delas, segundo as definições apresentadas na Seção 5.6.

$$CF_{1,2} = CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2) \quad (6.1)$$

No Algoritmo 11 são apresentados detalhes de como a combinação local e global é realizada para a criação de *Data Bubbles*. Esse algoritmo representa uma abstração da implementação das fases de combinação descritas anteriormente⁴. Especificamente, o Algoritmo 11 tem como entrada tuplas $\langle AmostraId, \{x_j, id\} \rangle$ referente a cada objeto x_j e à identificação de sua amostra aleatória mais próxima (*AmostraId*) como chave. A partir da combinação de tuplas de mesma identificação, estatísticas suficientes (CF) são calculadas local e globalmente. Depois que as estatísticas são computadas, os *Data Bubbles* são extraídos das mesmas. O resultado desse processo de combinação apresentado no Algoritmo 11 é um *Data Bubble* representado como $\langle id, b_{AmostraId} \rangle$.

Algoritmo 11 – Combinação local e global

Entrada: Sejam as tuplas $\langle AmostraId, \{x_j, id\} \rangle$ referentes a cada objeto x_j e sua respectiva amostra mais próxima.

- 1: **procedimento** *Combinacao*($\langle AmostraId, \{x_j, id\} \rangle$)
 - 2: **para todo** $\langle AmostraId, \{x_j, id\} \rangle$ **faça**
 - 3: $CF_{AmostraId} \leftarrow \text{calculaLocalGlobalCF}(AmostraId, x_j)$ ▷ Equação 6.1.
 - 4: $B_{AmostraId} \leftarrow \text{extraiDataBubble}(CF_{AmostraId})$.
 - 5: Emite par $\langle id, b_{AmostraId} \rangle$.
 - 6: **fim para**
 - 7: **fim procedimento**
-

Em sequência ao exemplo de fluxo de execução apresentado pela Figura 26, uma vez que a abordagem de sumarização de dados é determinada inicialmente, i.e., *Data Bubbles* ou

⁴ O conceito de combinação local e global é implicitamente implementado com funções *ReduceByKey* do reformulado *MapReduce* (KARAU *et al.*, 2015).

apenas amostras aleatórias, o próximo passo de $MR-HDBSCAN^*$ é construir um modelo local de $HDBSCAN^*$ sobre a abordagem escolhida.

Caso *Data Bubbles* sejam utilizados nessa fase do fluxo, um modelo local de $HDBSCAN^*$ é construído por meio de invocações de funções *ReduceByKey*, como nos passos descritos no Algoritmo 12. Esse algoritmo tem como entrada os pares de mesma chave $\langle id, b_{AmostraId} \rangle$ correspondente a cada *Data Bubble* extraído na fase de combinação descrita anteriormente, o parâmetro m_{pts} que indica o número mínimo de objetos de uma determinada vizinhança e m_{clSize} que representa o número mínimo de objetos de um grupo válido.

Consequentemente, o retorno do Algoritmo 12 para cada modelo local de $HDBSCAN^*$ a partir de um subconjunto de dados $\mathbf{X}_{id}$ é uma tupla $\langle id, \mathbf{P}_B \rangle$, em que $\mathbf{P}_B$ representa a “melhor” partição obtida a partir de *Data Bubbles* e um conjunto de arestas intergrupos (E_{inter}) recuperadas em cada modelo, em que cada aresta é representada por um par $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$.

Além disso, como $MR-HDBSCAN^*$ processa vários subconjuntos de dados e, em consequência, vários modelos locais de $HDBSCAN^*$ podem ser criados ao mesmo tempo em paralelo, os grupos das partições $\mathbf{P}_B$ geradas em paralelo devem ter rótulos únicos, ou seja, *ids* exclusivos para cada grupo criado independente da partição que for gerada.

Ao considerar amostras aleatórias, o procedimento de construção de um modelo de $HDBSCAN^*$ é praticamente o mesmo do que para *Data Bubbles*, exceto pela construção da *MST*. Pois, diferente do que acontece com *Data Bubbles*, para a abordagem de amostras, são utilizadas as definições originais de distância *core* e distância de acessibilidade mútua para a construção da *MST* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Portanto, para simplificar o Algoritmo 12, são apresentados apenas os passos para a construção de modelos locais sobre *Data Bubbles*. Porém, esse processo é de fácil adaptação para o uso de amostras aleatórias.

Algoritmo 12 – *ReduceByKey*: Modelo Local de $HDBSCAN^*$ a partir de *Data Bubbles*.

Entrada: Sejam os pares $\langle id, b_{AmostraId} \rangle$ que representam *Data Bubbles* e seus respectivos *ids*; m_{pts} o número mínimo de objetos em uma vizinhança; e m_{clSize} o tamanho mínimo de qualquer grupo (opcional).

```

1: procedimento ReduceByKey( $\langle id, b_{AmostraId} \rangle$ ,  $m_{pts}$ ,  $m_{clSize}$ )
2:   Agrega todos  $\langle id, b_{SampleId} \rangle$  para construir  $\mathbf{B}_{id} = \{b_1, \dots, b_m\}$ 
3:    $(MST_{id}, H_{id}) \leftarrow HDBSCAN^*(\mathbf{B}_{id}, m_{pts})$ 
4:    $\{\mathbf{C}_1, \dots, \mathbf{C}_k, \mathbf{O}\} \leftarrow FOSC(H_{id}, m_{pts}, m_{clSize})$ 
5:   Induz partição  $\mathbf{P}_B = \{\mathbf{C}'_1, \dots, \mathbf{C}'_k\}$ :
6:      $\mathbf{C}'_i = \{b \in \mathbf{C}_i \cup \mathbf{O} \mid GrupoMaisProximo(b, \mathbf{C}_i)\}$ 
7:    $Arestas' \leftarrow arestasIntergrupos(MST_{id}, \mathbf{P}_B)$ 
8:   para todo  $[u, v, d_{mreach}(u, v)] \in Arestas'$  faça
9:     emite  $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$ 
10:  fim para
11:  retorna  $\langle id, \{\mathbf{P}_B\} \rangle$ 
12: fim procedimento
```

Por fim, a quarta fase da primeira etapa de $MR-HDBSCAN^*$ consiste em realizar uma

indução de partição que corresponde à atribuição de cada objeto não amostrado $\mathbf{x}_j \in \mathbf{X}_{id}$ ao mesmo grupo atribuído ao seu objeto representativo mais próximo (*Data Bubbles* ou amostras aleatórias) no modelo local de HDBSCAN*. Nesse contexto, para a indução de partição via *MapReduce*, é necessária apenas uma função *mapper* descrita pelo Algoritmo 13.

Na indução de partição, apresentada no Algoritmo 13, são utilizadas as tuplas $\langle \text{AmostraId}, \mathbf{x}_j, id \rangle$ computadas pelo Algoritmo 10, e os modelos locais de HDBSCAN* (sobre *Data Bubbles*/amostras), construídos a partir de cada subconjunto de dados $\mathbf{X}_{id}$ do nível hierárquico em questão (Algoritmo 12), em que cada modelo é representado por um par do tipo $\langle id, \mathbf{P}_B \rangle$. A ideia dessa função é fazer com que, para cada objeto $\mathbf{x}_j$, o valor de id é utilizado para identificar a partição $\mathbf{P}_B$ e o respectivo grupo que contém o *Data Bubble* ou amostra que representa esse objeto e determinar o mesmo rótulo de grupo a ele. O retorno do Algoritmo 13 é um par do tipo $\langle \text{GrupoId}, \mathbf{x}_j \rangle$ para cada objeto $\mathbf{x}_j$. Em que *GrupoId* é o rótulo de grupo atribuído ao mesmo⁵.

Algoritmo 13 – *Mapper*: Indução de Partição

Entrada: Seja $\langle \text{AmostraId}, \{\mathbf{x}_j, id\} \rangle$ os pares referentes a cada objeto $\mathbf{x}_j$ e $\langle id, \mathbf{P}_B \rangle$ pares correspondentes a diferentes modelos de HDBSCAN* construídos sobre *Data Bubbles*.

- 1: **procedimento** *Mapper*($\langle \text{AmostraId}, \{\mathbf{x}_j, id\} \rangle$, $\langle id, \mathbf{P}_B \rangle$)
 - 2: **para todo** $\langle \text{AmostraId}, \{\mathbf{x}_j, id\} \rangle$ **faça**
 - 3: $\text{GrupoId} \leftarrow \text{grupoIdBubbleMaisProximo}(\mathbf{P}_B, \mathbf{x}_j, id, \text{AmostraId})$
 - 4: Emite par $\langle \text{GrupoId}, \mathbf{x}_j \rangle$
 - 5: **fim para**
 - 6: **fim procedimento**
-

6.2.2 Segunda Etapa: Combinação de MSTs Locais e Arestas Inter-grupos

A segunda etapa consiste em combinar *MSTs* locais e arestas intergrupos E_{inter} obtidas na primeira etapa de *MR-HDBSCAN**, descrita na Subseção 6.2.1, para obter uma única *MST* global estendida (denotada como MST_{est}) composta por todos os objetos do conjunto de dados. Essa etapa é iniciada pelo carregamento no *DFS* das arestas computadas (representadas pelas tuplas $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$) pelos Algoritmos 10 e 12. Os valores u e v representam objetos (vértices) de *MSTs* locais e de arestas intergrupos E_{inter} . Já o valor d_{mreach} representa o peso (distância de acessibilidade mútua) da aresta entre os vértices u e v .

Em essência, as arestas de *MSTs* locais e arestas intergrupos E_{inter} estão espalhadas em diversas unidades de processamento, porém, fazem parte implicitamente de uma única *MST* global estendida (MST_{est}). Na Figura 28(a) é apresentada uma abstração de várias arestas (*MSTs* locais e arestas intergrupos) que compõem a MST_{est} espalhada em quatro unidades de processamento distintas.

⁵ Cada subconjunto de dados gerado e identificado com *GrupoId*, podem ser subdivididos e processados em paralelo por diferentes instâncias de *MapPartitionsToPair* em iterações futuras de *MR – HDBSCAN**

Figura 28 – Exemplo de combinação de $MSTs$ locais e arestas intergrupos de $MR-HDBSCAN^*$ para a obtenção da MST global estendida MST_{est} . (a) $MSTs$ locais e arestas intergrupos espalhadas em 4 nós distintos. (b) Combinação em `MapReduce` de arestas ($MSTs$ locais e arestas intergrupos) em uma única MST global estendida.

Então, para a combinação da MST_{est} única, é designada uma função `ReduceByKey` representada pelo Algoritmo 14. Essa função tem como entrada as tuplas $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$. Assim como citado anteriormente, a *Chave*=1 indica que o processo de combinação das $MSTs$ locais e arestas intergrupos será direcionado para um único *reducer*.

Na Figura 28(b) é representado um exemplo de execução do Algoritmo 14 sobre as arestas apresentadas na Figura 28. Especificamente nessa figura, são mostrados vértices (objetos) enumerados de 1 a 13 e arestas ponderadas que os conectam. A representação de cada aresta do exemplo da Figura 28 é considerada no formato de tuplas do tipo $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$. Dessa forma, essas são combinadas pelas suas chaves e ordenadas em ordem decrescente de peso (d_{mreach} distância de acessibilidade mútua) parcialmente em paralelo passo a passo até que todas elas estejam combinadas em uma única solução, i.e., na MST_{est} .

Tecnicamente, na Linha 2 do Algoritmo 14, é criada uma lista de arestas vazia denominada MST_{est} . Na Linha 3, enquanto houver arestas $([u, v, d_{mreach}(u, v)])$ a serem recuperadas, insere $\{v, u, d_{mreach}\}$ na lista *Arestas*. Após esse processo, as arestas contidas em MST_{est} são ordenadas em ordem decrescente de peso (distância de acessibilidade mútua). Nesse contexto, a saída desse algoritmo é a *MST Global Estendida* (representada por MST_{est}) ordenada e representada por tuplas do tipo $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$.

Algoritmo 14 – *ReduceByKey*: Combinação de *MSTs* locais e arestas intergrupos.

Entrada: Seja $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$ a tupla correspondente cada aresta das *MSTs* locais ou das *arestas intergrupos*.

```

1: procedimento ReduceByKey( $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$ )
2:   Arestas =  $\emptyset$ 
3:   para todo  $[u, v, d_{mreach}(u, v)]$  faça
4:     Arestas  $\leftarrow$  insereAresta( $[u, v, d_{mreach}(u, v)]$ ).
5:   fim para
6:   MSTest  $\leftarrow$  ordenaEmOrdemDecrescente(Arestas)
7:   Retorna MSTest (pares  $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$ ).
8: fim procedimento

```

6.2.3 Terceira Etapa: Construção de Hierarquia de MR-HDBSCAN* sobre o Conjunto de Dados Completo

Assim como descrito nas seções anteriores, *MR-HDBSCAN** retorna uma *MST* global estendida baseada na distância de acessibilidade mútua a partir de um determinado conjunto de dados. Contudo, no contexto de agrupamento hierárquico de dados, existe a necessidade de representar os dados agrupados de maneira mais intuitiva para posteriores análises. É interessante mostrar as estruturas dos dados em forma de hierarquia (dendrograma) ou árvore de grupos simplificada, assim como é definido para o original *HDBSCAN** em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Em suma, *HDBSCAN** trabalha de maneira sequencial e centralizada sobre cada aresta de uma determinada *MST* baseada em densidade. A cada remoção de arestas de maior peso da *MST* é determinado um novo nível hierárquico rotulando os subcomponentes conectados que contêm os vértices finais das arestas afetadas pela remoção.

No contexto original de *HDBSCAN**, a determinação dos subcomponentes conectados a cada nível pode ser feita de maneira simples, uma vez que os vértices da *MST* fazem parte do mesmo repositório centralizado de dados. Todavia, no caso de aplicações distribuídas de dados, que é o caso de *MR-HDBSCAN**, a *MST* gerada por esse algoritmo é inerentemente distribuída em unidades de processamento distintas (como pode ser visto no exemplo da Figura 28), fato que torna a construção de hierarquia de *MR-HDBSCAN** complexa.

Neste trabalho de mestrado, são apresentados dois métodos de construção de hierarquia de *MR-HDBSCAN** a partir de *MST* global estendida distribuída em diferentes unidades de processamento independentes utilizando o conceito de *MapReduce*.

O primeiro método consiste na construção de hierarquia utilizando a abordagem divisiva (*top-down*) de maneira similar a *HDBSCAN** Campello *et al.* (2013a), Campello *et al.* (2015). Já o segundo método, utiliza uma abordagem aglomerativa (*bottom-up*) para a construção da hierarquia. Ambos os métodos retornam uma hierarquia em forma de dendrograma e árvore de grupos simplificada, como descrito em Campello *et al.* (2013a), Campello *et al.* (2015).

Ainda, os métodos propostos são capazes de retornar, em uma atividade de pós-processamento, uma partição composta pelos grupos mais significativos e os graus de anomalia de cada objeto do conjunto de dados pelo método *GLOSH* (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Assim como *HDBSCAN** determina grupos válidos e ruídos ao longo da construção de hierarquias baseadas em densidade, os métodos propostos neste trabalho de pesquisa também são capazes de realizar essa tarefa. Entretanto, existem diferenças substanciais entre esses métodos para identificação de grupos e ruídos durante a construção de hierarquias.

No método divisivo de construção, novos grupos são formados assim que arestas de maior peso são removidas da *MST* baseada em densidade, sejam eles grupos legítimos (válidos) ou grupos compostos por objetos ruidosos, com relação ao parâmetro m_{clSize} . Quando um grupo C_i legítimo “aparece”⁶, as primeiras informações sobre esse grupo são coletadas (e.g., nível de aparecimento representando o valor corrente da escala do dendrograma, além do número de objetos em C_i).

Em contrapartida, no método aglomerativo, assim que um grupo C_i legítimo aparece, a primeira informação que é obtida desse grupo é o nível hierárquico em que o mesmo esteve “válido” pela última vez. Para esse grupo, o nível de seu aparecimento acontecerá quando o mesmo se unir a outro grupo legítimo na hierarquia. Na Figura 29 é apresentado um exemplo com o intuito de explicitar as diferenças entre construção divisiva e aglomerativa de hierarquias de *MR-HDBSCAN**.

Figura 29 – Diferenças entre a construção de hierarquias de *MR-HDBSCAN** de maneira aglomerativa e divisiva.

Como pode ser visto na Figura 29, é feita uma construção de hierarquia hipotética de *MR-HDBSCAN** a partir de oito objetos e considerando $m_{pts} = m_{clSize} = 3$. Inicialmente, de

⁶ Assim como descrito no Capítulo 3, um grupo deve possuir ao menos m_{clSize} para ser considerado um grupo legítimo (válido).

maneira divisiva, todos os objetos são parte de um único grupo (C_1) no nível w_6 . É nesse nível que o grupo C_1 é dividido em dois grupos legítimos (C_2 e C_3), com relação a m_{clSize} , no nível w_6 . Em seguida, no nível w_5 , o grupo C_3 é dividido em dois novos grupos ruidosos. É nesse ponto que o grupo C_3 desaparece na hierarquia. O grupo C_2 encolhe no nível w_4 , pois o objeto 1 se desliga desse grupo e se torna um ruído. Na sequência, o grupo C_2 se divide em dois grupos de ruídos e, conseqüentemente, desaparece. Por fim, todos os objetos são considerados como ruídos a partir do nível w_3 na construção de hierarquia de maneira divisiva.

Em oposição à abordagem divisiva, na construção de hierarquia feita de forma aglomerativa na Figura 29, todos os objetos são considerados como ruído no primeiro nível w_1 . Então, dois objetos são unidos (3 e 4) no nível w_2 , porém, esses objetos serão considerados ruídos porque o número mínimo de objetos para legitimar um grupo é $m_{clSize} = 3$. Novos grupos de ruídos ($\{5,6\}$ e $\{7,8\}$) são formados no nível w_4 . No nível w_2 os objetos (2, $\{3,4\}$) são unidos e dão origem ao primeiro grupo legítimo da hierarquia (C_2). O grupo C_2 aumenta de tamanho no nível w_5 com a inclusão do objeto 1 a ele. Nesse mesmo nível, o grupo legítimo C_3 é formado pela união dos grupos de ruídos ($\{5,6\}$ e $\{7,8\}$). Por fim, no nível w_6 da hierarquia, todos os objetos são unidos em um só grupo C_1 . Logo, o nível de “aparecimento” dos grupos legítimos na hierarquia construída de maneira aglomerativa será no nível w_6 .

Nesse contexto, os detalhes de implementação dos métodos de construção de hierarquia de *MR-HDBSCAN** propostos neste trabalho de mestrado são apresentados nas seções seguintes. Especificamente, o método divisivo de construção é apresentado na Subseção 6.2.3.1 e o método aglomerativo de construção é apresentado na Subseção 6.2.3.2.

6.2.3.1 Construção Divisiva da Hierarquia

A ideia por trás do método divisivo de construção de hierarquia de *MR-HDBSCAN** é considerar inicialmente que todos os objetos (vértices da MST_{est}) que constituem o conjunto de dados estejam em um grupo único espalhado em diferentes unidades de processamento. Então, de maneira divisiva e iterativa, novos grupos vão surgindo à medida que as arestas de maior peso da MST_{est} são removidas sequencialmente. Em cada remoção também podem ser consideradas arestas de maior peso empatadas.

Ante o exposto, a cada remoção, subcomponentes conectados são encontrados e denominados como novos grupos no nível em questão da hierarquia. Os grupos encontrados podem ser denominados como: grupos legítimos (ou válidos), grupos de ruídos ou grupos que apenas se fragmentaram e continuam se propagando conforme os níveis vão surgindo, como descrito no Capítulo 3. Adicionalmente, esse processo é repetido até que não haja mais arestas na MST_{est} a serem removidas. O resultado desse processo iterativo é a hierarquia completa de *MR-HDBSCAN** representada por um dendrograma e árvore de grupos simplificada (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Considerando que as arestas da MST_{est} estão distribuídas em diferentes unidades de

processamento, para definir os níveis da hierarquia é preciso saber quais são e onde estão localizados todos os objetos (vértices) que compõem os grupos à medida que esses níveis são construídos. Logo, para formar esses grupos nível a nível, é necessário utilizar técnicas para detectar componentes conectados em grafos (MST_{est}) de maneira paralela, independentemente da localização de seus vértices. Para tal propósito, o algoritmo *CC-MR* (SEIDL *et al.*, 2012) foi utilizado como base no método proposto neste projeto de mestrado. Adicionalmente, o algoritmo *CC-MR* é uma abordagem rápida de detecção de componentes conectados em grafos baseada em *MapReduce*.

6.2.3.1.1 Algoritmo *CC-MR*

O algoritmo *CC-MR* (do inglês, *Connected Components - MapReduce*) foi proposto baseado em *MapReduce* para detectar componentes conectados em grafos de larga escala (SEIDL *et al.*, 2012). Para tal, esse algoritmo relaciona componentes conectados a grafos estrela (GODSIL; ROYLE, 2013). Grafos estrela são árvores compostas por n vértices, em que um desses vértices é denominado um nó interno da árvore e os demais vértices ($n - 1$) são seus vértices (nós) folhas (GODSIL; ROYLE, 2013).

Desse modo, a ideia inicial de *CC-MR* é, a partir de um grafo qualquer, determinar índices (*ID*) individuais para cada um de seus vértices (índices ordinais) e alterar a vizinhança de cada um desses vértices de maneira que vértices com maiores *IDs* sejam apontados pelo seu vértice diretamente alcançável (vizinho) de menor *ID* (SEIDL *et al.*, 2012).

Especificamente, a alteração da vizinhança de cada vértice corresponde a adição e a remoção iterativamente de arestas de um determinado grafo (SEIDL *et al.*, 2012). Essa estratégia de alteração de grafos faz com que *CC-MR* faça um número menor de iterações para encontrar todos os componentes conectados de um grafo em relação aos métodos (COHEN, 2009; KANG *et al.*, 2009).

Para identificar um (sub)componente, é preciso lidar com duas situações: se o subcomponente já é máximo localmente ou se existem subcomponentes no grafo que possam futuramente ser fundidos a ele (SEIDL *et al.*, 2012). Um (sub)componente máximo localmente corresponde a um grafo cujos vértices de maiores *IDs* já são apontados por algum vértice diretamente alcançável de menor *ID*. Nesse caso, não há o que alterar nesse (sub)componente em uma determinada iteração de *CC-MR*. Já um (sub)componente não máximo é aquele em que existem vértices de maiores *IDs* que ainda não são apontados diretamente pelo vértice de menor *ID*. Diante disso, adições e remoções de arestas são necessárias nesse (sub)componente de maneira que o mesmo se torne, em iterações futuras de *CC-MR*, um (sub)componente máximo (SEIDL *et al.*, 2012).

Formalmente descrito em Seidl *et al.* (2012), considere um grafo $G = (V, E)$, em que V é um conjunto de vértices rotulados com *IDs* pertencentes ao conjunto dos números inteiros $\mathbb{Z}$ e $E = \{(v_{origem}, v_{destino})\} \in V^2$ é um conjunto de arestas não ponderadas. Então, a proposta

de *CC-MR* para identificar componentes é verificar para cada vértice v se o mesmo contém ou não o menor *ID* em relação aos seus vértices adjacentes ($adj(v) = \{u_1, u_2, \dots, u_m\}$), em que m corresponde ao número de vértices adjacentes de v .

Assim, se o *ID* de v seja o menor em relação aos *IDs* de seus vértices adjacentes, então, todos os adjacentes ($u \in adj(v)$) são apontados por v . Neste caso, o componente em que v pertence é máximo localmente (denotado por *estado de máximo local*). Caso contrário, se existe um vértice $u \in adj(v)$ com o *ID* menor do que o próprio v , existe uma situação de *estado de fusão*. Portanto todos os adjacentes de v , inclusive v são apontados por u (SEIDL *et al.*, 2012).

Intuitivamente, o procedimento de identificar esses dois estados faz com que o algoritmo *CC-MR* transforme um vértice com menor *ID* em um vértice central de um determinado componente de um grafo G . O ponto de parada deste algoritmo acontece quando não existe nenhuma situação de *estado de fusão* para ser realizada (SEIDL *et al.*, 2012).

O algoritmo *CC-MR* utiliza em sua implementação, conceitos simples de arestas para frente (ou do inglês *forward edges*) e arestas para trás (*backward*) (SEIDL *et al.*, 2012). Em comparação com os *IDs* de cada vértice de um determinado grafo, uma aresta para frente ($v \rightarrow u$) é identificada quando $v < u$. Conceitualmente, uma aresta para frente $v \rightarrow u$ indicará que u é parte do componente em que v pertence. Já quando uma aresta para trás é identificada, i.e., $u > v$, isso indica que existe um componente de um vértice w que é conectado a um componente do vértice u através de v . Em outras palavras, a conexão de w e u será determinada como uma “ponte” entre os componentes que envolvem esses vértices (SEIDL *et al.*, 2012).

Ainda, no conceito de *MapReduce*, esses dois tipos de conceitos são representados por pares (v, u) , i.e., <Chave, Valor>, na implementação de *CC-MR* (SEIDL *et al.*, 2012). Então, o algoritmo *CC-MR* é composto por apenas uma função de *ReduceByKey*, descrita pelo Algoritmo 15. À vista disso, para ilustrar os procedimentos feitos por *CC-MR* apresentados no Algoritmo 15, na Figura 30 é exposto um exemplo de execução desse algoritmo sobre um grafo acíclico composto por cinco vértices e quatro arestas.

Basicamente, o Algoritmo 15 tem como entrada pares do tipo $\langle v_{origem}, adj(v_{origem}) \rangle$, em que $adj(.)$ representa os vértices adjacentes de v_{origem} . Os vértices adjacentes de v_{origem} ($adj(v_{origem})$) são previamente ordenados em ordem ascendente de índices. Isto posto, o Algoritmo 15 tem como objetivo identificar três situações distintas:

1. *Estado de máximo local* (Linhas 5 à 6 e 14 à 15);
2. *Estado de fusão*: (Linhas 16 à 18 e 23 à 25);
3. *Estado de fusão otimizada* (Linhas 16 à 18).

Logo, na Linha 1 do Algoritmo 15 existe uma variável global (*noveIteracao*) que indica se para um determinado vértice necessitará ou não de uma nova iteração para encontrar seu

Figura 30 – Execução de CC-MR: (a) Grafo original; (b) Iteração considerando o vértice 1 como v_{origem} (estado de máximo local); (c) Iteração em que o vértice 2 é v_{origem} (estado de fusão); (d) Iteração considerando o vértice 3 como v_{origem} (estado de fusão otimizada).

componente máximo local (*estado de máximo local*). Na Linha 2, a variável *estadoMaximoLocal* indica que determinados vértices encaixam na situação de *estado de máximo local* de um (sub)componente. Na Linha 3, a variável $v_{primeiro}$ obtém o primeiro vértice adjacente de v_{origem} , i.e., u_1 . Se $v_{origem} < u_1$, então, é atribuído um valor verdadeiro à variável *estadoMaximoLocal* (Linha 5) e emitida uma aresta para frente ($v_{origem} \rightarrow u_1$) (Linha 6) indicando que u_1 é parte do componente em que v_{origem} pertence. Na Figura 30 (b) é apresentada uma situação de *estado de máximo local*, em que o vértice de ID 1 é considerado como vértice origem (v_{origem}) e, em uma única iteração, é criada uma aresta entre seu único vértice vizinho de índice 2.

Na Linha 8 do Algoritmo 15, a variável $v_{destinoVelho}$ é responsável por eliminar arestas duplicadas durante a execução da função *ReduceByKey*. Das Linhas 9 à 22, são realizados três procedimentos: no primeiro, na Linha 11, é verificado se não existem arestas duplicadas para serem emitidas; na Linha 12, continua o processo se isso for verdadeiro; o segundo, na Linha 14, se o valor de *estadoMaximoLocal* for verdadeiro, existirá outro *estado máximo local*. Isso quer dizer que o vértice origem (v_{origem}) é menor do que todos os seus adjacentes. Então, uma aresta “para frente” é emitida (Linha 15); no terceiro, nas Linhas 17 e 18, duas arestas são emitidas com o objetivo de alterar o grafo por meio de adição dessas arestas. A primeira emitida corresponde a uma aresta para frente, em que $v_{primeiro} < v_{destino}$. Já a segunda emitida corresponde a uma

Algoritmo 15 – *ReduceByKey*: Algoritmo CC-MR**Entrada:** Sejam os pares $\langle v_{origem}, adj(v_{origem}) \rangle$ arestas da MST_{est}

```

1:  $novaIteracao \leftarrow \text{falso}$  ▷ Variável global
2:  $estadoMaximoLocal \leftarrow \text{falso}$ 
3:  $v_{primeiro} \leftarrow u_1$ 
4: se  $v_{origem} < v_{primeiro}$  então
5:    $estadoMaximoLocal \leftarrow \text{verdadeiro}$ 
6:    $emite(v_{origem}, v_{primeiro})$ 
7: fim se
8:  $v_{destinoVelho} \leftarrow v_{primeiro}$ 
9: para todo os vértices adjacentes  $u_i$  de  $v_{origem}$  faça
10:    $v_{destino} \leftarrow u_i$ 
11:   se  $v_{destino} == v_{destinoVelho}$  então
12:     continue
13:   fim se
14:   se  $estadoMaximoLocal$  então ▷ Estado máximo local
15:      $emite(v_{origem}, v_{destino})$  ▷ aresta para frente
16:   senão ▷ Estado de fusão, estado de fusão otimizada
17:      $emite(v_{primeiro}, v_{destino})$  ▷ aresta para frente
18:      $emite(v_{destino}, v_{primeiro})$  ▷ aresta para trás
19:      $novaIteracao \leftarrow \text{verdadeiro}$ 
20:   fim se
21:    $v_{destinoVelho} \leftarrow v_{destino}$ 
22: fim para
23: se  $v_{origem} < v_{destino} \ \&\& \ !estadoMaximoLocal$  então
24:    $emite(v_{origem}, v_{primeiro})$  ▷ aresta para trás
25: fim se
26: Retorna Componentes Conectados (pares  $\langle v_{origem}, v_{conectado} \rangle$ )

```

aresta para trás. Assim, as operações realizadas nas Linhas 17 e 18 indicam uma situação de *estado de fusão*.

Para representar um estado de fusão descrito nas Linhas 17 e 18 do Algoritmo 15, na Figura 30 (c) é apresentada uma situação em que o vértice de índice 2 é considerado como vértice de origem (v_{origem}). Note que o vértice 2 contém como vizinhos os vértices 1, 3 e 4 (Figura 30 (a)), inicialmente. Então, na Figura 30 (c) o vértice 1 ($v_{primeiro}$) é considerado como o menor índice e, consequentemente, são adicionadas arestas “para frente” $v_{primeiro} \rightarrow v_{destino}$ entre o vértice 1 e o vértice 3 e entre o vértice 1 e o vértice 4, na Linha 17 do Algoritmo 15.

Na Linha 18, é emitida uma aresta para trás no contexto de estado de fusão, i.e., $v_{origem} > v_{primeiro}$. Pelo exemplo apresentado na Figura 30 (c), arestas do vértice 4 para 1, e do vértice 3 para o vértice 1 são emitidas. Nesse caso, essas arestas emitidas (para trás) servem como pontes, pois, em iterações futuras, os vértices 4 e 3 podem ser conectados a outro vértice de menor *ID* do que o vértice 1 (e.g., vértice 0) no componente.

Considerando a pré-ordenação em ordem ascendente de índices dos vértices adjacentes

$adj(v_{origem})$ do vértice v_{origem} , note que a Linha 24 do Algoritmo 15 só será executada quando $v_{origem} < v_{destino}$ e não exista um estado de máximo local relacionado a v_{origem} . O $v_{destino}$, neste caso, é o último vértice adjacente de v_{origem} a ser processado nesse ponto desse algoritmo. Então, com a intuição de criação de “pontes” por meio da emissão de arestas para trás descrita anteriormente, na Figura 30 (c), uma aresta para trás do vértice 2 (v_{origem}) é emitida para o seu primeiro vértice vizinho, i.e., o vértice 1 ($v_{primeiro}$).

Como pode ser visto no Algoritmo 15, as instruções apresentadas das Linhas 16 à 18 também são descritas como um *estado de fusão otimizada*. Nesse estado, o número de arestas duplicadas é reduzido a cada iteração do Algoritmo 15. Assim, na Figura 30 (d) é apresentada uma situação de estado de fusão otimizada que é iniciada pelo vértice de índice 3 como vértice de origem (v_{origem}). Os vértices 0 e 2 são vizinhos do vértice 3 na Figura 30 (a), inicialmente. Portanto, duas arestas (para frente e para trás) são criadas entre o vértice 0 e o vértice 2 (Figura 30 (d)). A redução do número de arestas no exemplo da Figura 30 (d) ocorre a partir do momento em que não é emitida uma aresta para trás do vértice 3 para o vértice 0, pois a condição apresentada na Linha 23 do Algoritmo é dada como falsa. Note ainda que o vértice 3 fica isolado quando é determinado como v_{origem} (Figura 30(d)), mas se o vértice 0 for considerado como v_{origem} , o vértice 3 não ficará perdido, pois, como pode ser visto na Figura 30(a), o vértice 3 é o único vizinho do vértice 0. Portanto, nesse caso, ocorrerá um estado de máximo local.

Por fim, são retornados os componentes conectados representados por pares $\langle v_{origem}, v_{conectado} \rangle$, em que v_{origem} representa o vértice de menor *ID* de um determinado componente conectado e $v_{conectado}$ cada vértice maior ou igual a v_{origem} .

6.2.3.1.2 Construção de Hierarquia de MR-HDBSCAN* - Abordagem Divisiva

Após detectar os subcomponentes conectados, usando *CC-MR*, a cada construção de nível hierárquico do método divisivo proposto neste trabalho de mestrado, é necessário obter algumas informações sobre cada subcomponente: 1) o número de objetos; 2) e um valor binário que indica se o subcomponente em questão faz parte ou não de uma verdadeira divisão de grupo válido no nível anterior da hierarquia, assim como descrito na Subseção 3.1.3 do Capítulo 3.

Para encontrar essas informações, uma função de *ReduceByKey* pode ser utilizada, tendo em vista que, as entradas dessa funções correspondem à pares do tipo $\langle v_{origem}, v_{conectado} \rangle$ (saídas do algoritmo *CC-MR*), em que a chave v_{origem} representa a identificação de cada subcomponente conectado. Além disso, essa função utiliza como entrada, o parâmetro m_{clSize} , a(s) aresta(s) afetada(s) pela remoção no nível corrente da hierarquia, o(s) rótulo(s) e o(s) tamanho(s) do(s) grupo(s) afetado(s) pela remoção dessas arestas. Dessa forma, a função *ReduceByKey* consegue calcular para cada subcomponente identificado como *origem*, seu tamanho ($tamanhoCC_{origem}$), e se o subcomponente foi originado a partir de um grupo que foi ou não dividido em um dois ou mais grupos válidos ($divididoCC_{origem}$).

Depois que os subcomponentes e suas respectivas informações são determinados, dois funções são necessárias uma função de *mapper* (Algoritmo 16) e outra de *ReduceByKey* (Algoritmo 17) para construir o nível hierárquico em questão.

Logo, a função *mapper* apresentada no Algoritmo 16 tem como entrada os pares $\langle v_{origem}, v_{conectado} \rangle$ dos componentes, seus respectivos tamanhos e se são parte de grupos que foram ou não verdadeiramente divididos. Além disso, uma estrutura de dados, chamada *Hierarquia* também é passada nesse algoritmo. Essa estrutura representa informações dos grupos que foram afetados pela remoção de arestas de maior peso no nível corrente da hierarquia, i.e., suas respectivas estabilidades, número de objetos, nível de aparecimento e desaparecimento na hierarquia, os grupos filhos, o grupo pai⁷, o nível em que o objeto mais denso do grupo se tornou ruído⁸, e outras informações relevantes (CAMPELLO *et al.*, 2013a). Então, nessa função, cada objeto ($v_{conectado}$) é mapeado com seu devido rótulo de grupo ou determinado como ruído, segundo as definições em (CAMPELLO *et al.*, 2013a).

Algoritmo 16 – Terceira etapa de MR-HDBSCAN* - *Mapper*: atribuição de objetos à grupos ou determinação de ruído.

Entrada: componentes representados pelos pares $\langle v_{origem}, v_{conectado} \rangle$; o número mínimo de objetos em um grupo válido m_{clSize} (opcional); hierarquia do nível corrente (*Hierarquia*); $tamanhoCC_{origem}$ o tamanho do componente; $divididoCC_{origem}$ indica se o grupo original (afetado) foi dividido ou não.

```

1: procedimento Mapper( $\langle v_{origem}, v_{conectado} \rangle$ ,  $m_{clSize}$ , Hierarquia,  $tamanhoCC_{origem}$ ,  $divididoCC_{origem}$ )
2:   se  $tamanhoCC_{origem} \geq m_{clSize}$  então
3:     se  $divididoCC_{origem} == verdadeiro$  então                                ▷ um novo grupo aparece
4:        $GInfo \leftarrow NovoGrupo(v_{origem})$ 
5:     senão                                                                    ▷ o grupo antigo é propagado
6:        $GInfo \leftarrow AtualizaGrupoAnterior(Hierarquia)$ 
7:     fim se
8:   senão                                                                    ▷ o objeto é um ruído
9:     se  $v_{conectado}$  pertencia a um grupo válido no nível anterior então
10:       $GInfo \leftarrow informacaoObjetoRuido(v_{conectado})$ 
11:    fim se
12:     $GInfo \leftarrow rudoID$ 
13:  fim se
14:   $originalID \leftarrow extraiVerticePaiID(Hierarquia, GInfo)$ 
15:  emite  $\langle originalID, \{v_{conectado}, GInfo\} \rangle$ 
16: fim procedimento

```

Para cada grupo afetado na construção de um determinado nível hierárquico, identificado pela chave *originalID*, as tuplas $\langle originalID, \{v_{conectado}, GInfo\} \rangle$ são combinadas utilizando uma função *ReduceByKey*, apresentada no Algoritmo 17. Para cada valor (*originalID*), essa função combina as informações presentes em *GInfo* de cada objeto para saber se é necessário

⁷ Subgrupos gerados a partir deste e o grupo que deu origem, respectivamente

⁸ Utilizado para estimativa de detecção de *outliers* pelo método *GLOSH*.

criar um novo grupo, aplicar uma fragmentação no grupo original, ou determinar se o grupo é um conjunto de ruídos. Neste contexto, uma hierarquia local H_{local} que contém o grupo original (afetado) é extraída a partir do nível corrente da hierarquia (*Hierarquia*) e atualizada a partir das informações obtidas na função *mapper* (Algoritmo 16). Uma importante informação a ser atualizada durante a construção de níveis hierárquicos, é a estabilidade de cada grupo formado. Pois, a estabilidade pode ser utilizada posteriormente, pelo método de extração de grupos mais significativos (*FOSC*) presentes nos dados (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Por fim, o Algoritmo 17 retorna uma hierarquia local a partir do grupo original afetado. Essa hierarquia pode ser combinada posteriormente em uma solução global para tarefas de pós-processamento de *HDBSCAN**, i.e., detecção de *outliers* (*GLOSH*) e extração de partição significativa (*FOSC*).

Algoritmo 17 – *ReduceByKey*: Construção de Hierarquias de MR-HDBSCAN*.

Entrada: as tuplas $\langle originalID, \{v_{conectado}, GInfo\} \rangle$ com a mesma chave e; a hierarquia corrente *Hierarquia*.

```

1: procedimento ReduceByKey( $\langle originalID, \{v_{conectado}, GInfo\} \rangle$ , Hierarquia)
2:   para todo  $\langle originalID, \{v_{conectado}, GInfo\} \rangle$  faça
3:      $CombinacaoGrupoInfo \leftarrow Combinacao(v_{conectado}, GInfo)$ 
4:   fim para
5:    $H_{local} \leftarrow ExtraiHierarquiaLocal(Hierarquia)$ 
6:    $H_{local} \leftarrow AtualizaInfo(H_{local}, CombinacaoGInfo)$ 
7:   emite  $\langle 1, H_{local} \rangle$ 
8: fim procedimento

```

Na Figura 31 é apresentado um exemplo de execução de *CC-MR* a partir de uma *MST* global estendida hipotética como passo para obter o primeiro nível de uma hierarquia de *MR-HDBSCAN**. Note que as arestas da *MST* estão distribuídas em três unidades de processamento independentes. Note também que a aresta de maior peso hipotético a ser removida é a que conecta os objetos 11 e 13. Então, após a remoção da aresta entre os objetos 11 e 13, *CC-MR* é executado sobre as arestas restantes da *MST*. Como resultado da execução de *CC-MR*, dois componentes conectados são obtidos, um destacado na cor azul e outro na cor vermelha. O componente de cor azul contém os objetos 1, 3, 9, 10 e 11. Já o componente na cor vermelha contém os objetos 2, 7, 8 e 13. Por fim, os vértices (objetos) 1 e 2 são os representantes de cada componente no exemplo da Figura 31, azul e vermelho, respectivamente, pois são os vértices de menores índices desses componentes.

Na Figura 32 é apresentado um exemplo de execução do algoritmo *CC-MR* e de funções *mapper* (Algoritmo 16) e *ReduceByKey* (Algoritmo 17), responsáveis pela construção de níveis hierárquicos de *MR-HDBSCAN**. Especificamente, esse exemplo mostra a construção do primeiro nível hierárquico com base na MST_{est} e nos componentes conectados hipotéticos (azul e vermelho) da Figura 31.

O processo de construção de qualquer nível hierárquico contém uma sequência de cinco

Figura 31 – Exemplo de execução de CC-MR a partir de uma MST global estendida hipotética.

passos (Figura 32). O primeiro passo (passo 1) corresponde à entrada das arestas da MST_{est} hipotética no algoritmo CC-MR para a obtenção de subcomponentes conectados. Cada aresta da MST_{est} é representada por um par do tipo $\langle v_{origem}, adj(v_{origem}) \rangle$, em que v_{origem} representa qualquer vértice (objeto) da MST_{est} e $adj(v_{origem})$ seu(s) vértice(s) (objetos) adjacente(s).

Assumindo que o tamanho ($tamanhoCC_{origem}$) e a se o subcomponente conectado seja parte ou não de uma divisão verdadeira ($divididoCC_{origem}$), essa informação e os pares de saída do algoritmo CC-MR são apresentadas no passo 2 da Figura 32. No exemplo dessa Figura existem dois subcomponentes conectados (passo 2). O primeiro contém os objetos 1, 3, 9, 10 e 11 e o segundo, os objetos 2, 7, 8 e 13. Em termos técnicos, cada objeto $v_{conectado}$ é representado por um par do tipo $\langle v_{origem}, v_{conectado} \rangle$, acrescidos das informações, $tamanhoCC_{origem}$ e $divididoCC_{origem}$ de cada subcomponente. Especificamente, o vértice (objeto) v_{origem} representa o vértice de menor índice do (sub)componente de identificação (*origem*) e $v_{conectado}$ é qualquer vértice (objeto) conectado à v_{origem} , tal que $v_{origem} \leq v_{conectado}$.

Ainda na Figura 32, os pares $\langle v_{origem}, v_{conectado} \rangle$, $tamanhoCC_{origem}$ e $divididoCC_{origem}$

gerados no passo 2, são utilizados como entradas ainda no passo 2 para a atribuição de cada objeto $v_{conectado}$ a seu grupo apropriado. Cada atribuição de objeto é feita por uma chamada de função *mapper* (Algoritmo 16). Adicionalmente, a cada chamada de função *mapper* também é passada como entrada as informações sobre o(s) grupo(s) afetado(s) pelo nível corrente da hierarquia, representados pela estrutura de dados *Hierarquia*. Essa estrutura, a princípio, é composta por apenas um grupo único (denotado como $GInfo_1$) representando informações como o número de objetos, rótulo de grupo e níveis de seu aparecimento e desaparecimento na hierarquia. Adicionalmente, os grupos contidos na estruturas *Hierarquia* são representados como pares $\langle 1, GInfo_1, \dots, GInfo_k \rangle$ ⁹.

Note que, no passo 2 da Figura 32, cada objeto ($v_{conectado}$) é atribuído ao seu devido grupo. Nesse caso, são determinadas informações iniciais sobre grupo conferido a cada objeto. Assim, no passo 4 da Figura 32 essas informações são atualizadas com uma função *ReduceByKey* com base nos pares de saídas no passo 4 das funções de *mapper*, i.e., $\langle originalID, v_{conectado}, GInfo \rangle$. Como resultado desse atualização, hierarquias locais (H_{local}) referentes aos grupos afetados no nível hierárquico em questão são criadas ou atualizadas.

⁹ A chave igual a 1 é utilizada em etapa de pós-processamento, tal como, extração de grupos mais significativos, a partir da árvore de grupos gerada.

Figura 32 – Exemplo de construção do primeiro nível de uma hierarquia de *MR-HDBSCAN**.

6.2.3.2 Construção Aglomerativa da Hierarquia

O segundo método proposto neste projeto de mestrado segue uma abordagem aglomerativa (*bottom-up*) de construção de hierarquia de *MR-HDBSCAN**. Considerando que as arestas da MST_{est} estejam distribuídas em diferentes unidades de processamento, então a proposta desse método é unir parte a parte das arestas da MST_{est} em paralelo, seguindo uma ordem crescente de seus pesos. Isso é feito como a finalidade de encontrar a cada nível hierárquico os subcomponentes conectados que formarão novos grupos ao longo da hierarquia. Para isso, o método proposto faz uso da estrutura de dados conhecida como *Disjoint Set* (SEDGEWICK; WAYNE, 2011). Ainda, *Disjoint Set* possui uma formulação teórica de como utilizar esse tipo de estrutura de dados para buscar componentes em grandes grafos distribuídos, apresentada em (MANNE; PATWARY, 2010).

6.2.3.2.1 Estrutura de Dados - Disjoint Set - Abordagem Paralela

Disjoint Set é uma estrutura de dados cujo objetivo é particionar um conjunto de elementos em um determinado número de subconjuntos não sobrepostos. Seja $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_k\}$ uma coleção de subconjuntos γ_i , em que $1 \leq i \leq k$, então γ_i e γ_j são subconjuntos não sobrepostos se $\gamma_i \cap \gamma_j = \emptyset$, para $i \neq j$ (SEDGEWICK; WAYNE, 2011). Esse tipo de estrutura de dados é utilizado na composição de algoritmos, tais como: árvore geradora mínima *MST* (e.g., algoritmo de *Kruskal* (KRUSKAL, 1956)); segmentação de imagens (FIORIO et al., 2000); e computação de componentes conectados em grafos (SEDGEWICK; WAYNE, 2011). Além disso, o algoritmo de *union-find* é utilizado para a implementação desse tipo de estrutura de dados (CORMEN et al., 2001).

O algoritmo *union-find* consiste em três operações: *make-set*; *union*; e *find*. Diante disso, considerando que a estrutura de dados para implementação desse algoritmo seja uma floresta, em que cada árvore que compõe essa floresta represente um conjunto de elementos conectados, então, a operação de *make-set* é responsável por determinar que cada elemento v tenha um ponteiro $p(v)$ inicializado com o próprio elemento v . A intuição é que cada elemento v represente ele mesmo como uma subárvore única, inicialmente (GALIL; ITALIANO, 1991). Já a operação de *find*(v) é utilizada para encontrar o elemento raiz da subárvore em que v pertence. Por fim, a operação de *union*(v, u) é utilizada para unir os elementos da subárvore de v aos elementos pertencentes à subárvore u , caso v e u não sejam de uma mesma subárvore de elementos (CORMEN et al., 2001; SEDGEWICK; WAYNE, 2011).

Dessa maneira, o processo de união de subárvores distintas é semelhante à procura por componentes em grafos (CORMEN et al., 2001). Logo, a implementação padrão de *union-find* para encontrar componentes conectados em um determinado grafo $G(V, E)$ pode ser representada pelo Algoritmo 18:

Note que o Algoritmo 18 é uma versão clássica de *union-find* que retorna um conjunto

Algoritmo 18 – Algoritmo *union-find* sequencial.

Entrada: Seja o grafo $G(V, E)$, em que $u, v \in V$ representa um conjunto de vértices, e E representa um conjunto de arestas (v, u) .

```

1:  $\Gamma \leftarrow \emptyset$                                 ▷ Conjunto vazio de componentes conectados.
2: para todo  $v \in V$  faça
3:    $p(v) \leftarrow v$ 
4: fim para
5: para todo aresta  $(v, u) \in E$  faça
6:   se  $find(v) \neq find(u)$  então                ▷ Compara raízes das subárvores de  $v$  e  $u$ .
7:      $union(v, u)$                                 ▷ Une subárvores (componentes) de  $v$  e  $u$ 
8:      $\Gamma \leftarrow \Gamma \cup \{v, u\}$             ▷ Armazena em  $\Gamma$  o componente que contém  $v$  e  $u$ .
9:   fim se
10: fim para
11: Retorna Componentes Conectados em  $\Gamma$ 

```

de componentes conectados não sobrepostos. Intuitivamente, essa versão pode ser estendida para outros cenários de aplicação (e.g., calcular árvore geradora mínima *MST*).

Existem outras maneiras mais eficientes de implementar *union-find* de forma centralizada (CORMEN *et al.*, 2001). Ainda, em um contexto de análise de grafos de larga escala que não podem ser processados por apenas uma unidade de processamento, métodos que são capazes de processá-los de maneira paralela e distribuída são bastante requisitados. Nesse contexto, Manne e Patwary (2010) propõem uma versão paralela de *union-find* aplicada a cenários de memória distribuída, i.e., em cenários que existem duas ou mais unidades de processamento independentes interconectadas por alguma topologia de redes.

Isto posto, a versão paralela de *union-find*, descrita em (MANNE; PATWARY, 2010), considera que um grafo denotado por $G(V, E)$ seja particionado em p unidades de processamento independentes. Então, dado um grafo $G(V, E)$, o conjunto de vértices $V = \{V_1, \dots, V_p\}$ e o conjunto de arestas $E = \{E_1, \dots, E_p\}$, em que cada par (V_i, E_i) seja alocado a uma unidade de processamento i , tal que $1 \leq i \leq p$. Ainda, se $v \in V_i$ (ou $(v, u) \in E_i$) em uma unidade de processamento i , então v (ou (v, u)) são locais ao processador i . Uma vez que cada subconjunto de vértices e arestas são consideradas como uma subárvore, a intuição desse método é unir localmente cada subárvore em paralelo utilizando uma versão sequencial de *union-find* em cada unidade de processamento independente. Por fim, combiná-las em uma solução global em forma de árvore geradora global ou mesmo componentes conectados (MANNE; PATWARY, 2010). A partir dessa intuição, é proposta, neste projeto de mestrado, uma adaptação do método de *union-find* paralelo apresentado em (MANNE; PATWARY, 2010) baseada em *MapReduce* com o propósito de buscar por componentes conectados em grafos (*MST* baseada em densidade) para a construção de hierarquias do *MR-HDBSCAN**.

6.2.3.2.2 Construção de Hierarquia de MR-HDBSCAN* - Abordagem Aglomerativa

A ideia do método aglomerativo de construção de hierarquia proposto neste trabalho consiste em combinar e ordenar “localmente” (em ordem crescente de peso) em paralelo as *MSTs* locais oriundas da primeira etapa do *MR-HDBSCAN**. Então, combinar as soluções parciais em uma solução global (MST_{est}). Assim, durante o processo de combinação e ordenação, os níveis hierárquicos podem ser determinados com o auxílio da estrutura de dados *union-find* como recurso para determinar os grupos de cada nível.

Em outras palavras, os componentes conectados que formarão grupos na hierarquia são encontrados pelo algoritmo *union-find* em paralelo por uma versão sequencial desse algoritmo, que é executada independentemente em cada *MST* local ordenada. Esse algoritmo inicia a busca pelos componentes a partir do menor peso das arestas da *MST* combinada e determina a união desses componentes formando um nível hierárquico de *MR-HDBSCAN**. Esse processo continua até que não exista mais arestas a serem exploradas na *MST* combinada. O retorno do procedimento de construção é uma hierarquia de grupos, como descrito no Capítulo 3.

Tanto para a abordagem divisiva, quanto para a aglomerativa, o valor do nível (escala do dendrograma) de aparecimento de quaisquer grupos é uma informação essencial para calcular/atualizar a estabilidade desses grupos a cada nível construído na hierarquia. Como descrito no Capítulo 3, a estabilidade de cada grupo é calculada e atualizada à medida (nível hierárquico) que seus objetos deixam de pertencê-lo durante a construção da hierarquia. Entretanto, é necessário aplicar a propriedade aditiva (ARITHMETIC, 2007) (Equação 6.2) à Equação 3.4 (Capítulo 3) para calcular e atualizar a estabilidade de qualquer grupo baseado em densidade no contexto aglomerativo, pois, como pode ser visto no exemplo da Figura 29, o nível de aparecimento de qualquer grupo legítimo só é obtido quando o mesmo se une a outro grupo legítimo na hierarquia, ou seja, não é possível realizar o cálculo de estabilidade por demanda se a Equação 3.4 não for desmembrada por essa propriedade.

$$\begin{aligned}
 S(C_i) &= \sum_{x_j \in C_i} \frac{1}{\varepsilon_{min}(x_j, C_i)} - \sum_{x_j \in C_i} \frac{1}{\varepsilon_{max}(C_i)} \\
 &= S_1(C_i) - S_2(C_i)
 \end{aligned} \tag{6.2}$$

Na Equação 6.2, cada termo pode ser calculado separadamente durante a construção da hierarquia. O termo $S_1(C_i)$ corresponde ao nível hierárquico em que qualquer objeto x_j deixa de pertencer a um grupo válido C_i . Já o termo $S_2(C_i)$, que representa o “aparecimento” de um grupo C_i , é calculado apenas quando esse grupo aparece de fato na hierarquia.

Outro ponto que deve ser observado na construção de hierarquia de *MR-HDBSCAN** de maneira aglomerativa é a maneira de calcular o grau de anomalia de cada objeto do conjunto completo de dados. Nessa abordagem, é preciso salvar o momento em que cada um dos objetos

passa de ruído para o primeiro grupo válido (abordagem (*bottom-up*)), ao invés de salvar o último grupo que qualquer objeto pertenceu antes de se tornar ruído (*top-down*).

Outra questão importante a ser observada é a capacidade que o método original *HDBSCAN** tem em modelar ruídos em sua hierarquia. A partir dessas questões, no início do processo de construção de hierarquia (em *bottom-up*), no primeiro nível hierárquico, todos os objetos são considerados como ruídos para $m_{pts} > 1$ por definição (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Ainda, assim como definido anteriormente, o algoritmo *union-find* parte de duas subárvores (conjuntos de elementos) por vez e então decide uni-las se as mesmas não pertencerem ao mesmo subconjunto de elementos, no contexto de *MR-HDBSCAN**, essas subárvores são as arestas das *MSTs* locais e arestas intergrupos.

No mais, haja vista todas as situações citadas para construir cada nível hierárquico de *MR-HDBSCAN**, outro ponto que deve ser observado é a utilização do parâmetro m_{clSize} , que indica o tamanho mínimo de qualquer grupo legítimo na hierarquia. Nesse contexto, ao considerar $m_{clSize} > 1$, existem quatro situações para definir que tipo de grupo está sendo formado em qualquer nível da hierarquia, por exemplo: considere que duas subárvores (*MSTs* locais) T_1 e T_2 deverão ser unidas em um nível hierárquico i , e $m_{clSize} > 1$, logo:

1. Se o número de elementos em T_1 e T_2 forem menores do que m_{clSize} , e a soma dos elementos dessas duas subárvores também for menor que m_{clSize} . Então, T_1 e T_2 são ruídos antes da união e serão automaticamente definidas como um grupo de elementos ruidosos nesse nível.
2. Se ambos os números de elementos das subárvores T_1 e T_2 são menores que m_{clSize} , mas a soma do número de elementos de ambas subárvores é igual ou maior que m_{clSize} . Então, existe uma situação que indica que os objetos que eram ruídos passaram a ser anexados ao seu primeiro grupo legítimo na hierarquia.
3. Se o número de elementos em T_1 é maior ou igual a m_{clSize} e o número de elementos em T_2 é menor que m_{clSize} , ou vice-versa. Então, existe uma situação de “desencolhimento” do grupo formado.
4. Se as subárvores T_1 e T_2 contêm individualmente um número de objetos maior do que m_{clSize} . Então, tem-se uma situação de “junção” de dois grupos legítimos em um outro grupo no nível anterior, ou seja, o grupo formado pela junção dos dois grupos será o “pai” de ambos.

No exemplo da Figura 33a, tem-se uma *MST* baseada em densidade hipotética e sem “*self-edges*” e na Figura 33b, é apresentado um dendrograma construído a partir da *MST* apresentada. Note que são apresentadas as quatro situações descritas anteriormente. Basicamente, no primeiro nível hierárquico ($\mathbf{n}_0$), os objetos são grupos únicos (ruídos, no contexto de *HDBSCAN**

Figura 33 – Exemplo das possíveis situações que podem ocorrer durante a construção de níveis hierárquicos de MR-HDBSCAN*.

(CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015)). Então, no nível n_1 , o primeiro par de grupos são unidos (3 e 4), formando o grupo $\{3, 4\}$. Nesse ponto, como $m_{pts} = m_{clSize} = 3$, a união dos grupos 3 e 4 não torna-os objetos de um grupo legítimo. Isso porque a soma do número de elementos pertencentes aos grupos 3 e 4 ainda é menor que m_{clSize} . Em sequência, o par de grupos a serem unidos são (6 e 7) no nível n_2 . Os níveis n_3 e n_4 apresentam junções entre os grupos de ruídos $\{\{3,4\}, 2\}$ e $\{\{6,7\}, 5\}$, respectivamente. Observe que acontece a segunda situação descrita no Item 2, pois cada um dos dois grupos, i.e., $\{\{3,4\}, 2\}$ e $\{\{6,7\}, 5\}$, possui um número de objetos inferior a m_{clSize} , mas, somado esses números de objetos na junção, um grupo legítimo é formado. Já nos níveis n_5 e n_6 , têm-se a situação de “desencolhimento” de grupos, ou seja, dado um grupo legítimo e outro grupo de ruídos, o que aconteceu nesse cenário foi apenas a reintegração dos objetos ruidosos ao grupo legítimo. Por fim, no nível n_7 , a junção que é feita corresponde à união de dois grupos legítimos já existentes na hierarquia, ou seja, a situação 4 ocorre neste exemplo.

Para implementar o método aglomerativo proposto neste trabalho de mestrado, é necessária uma função *ReduceByKey* (representada pelo Algoritmo 19). Essa função que tem como entrada as tuplas $\langle 1, [u, v, d_{mreach}(u, v)] \rangle$, que correspondem às arestas das *MSTs* locais juntamente com as arestas intergrupos encontradas na primeira etapa de MR-HDBSCAN*, e m_{clSize} , que denota o número mínimo de objetos de qualquer grupo legítimo (válido).

Diante disso, note que a MST_{est} será combinada diretamente pelo Algoritmo 19, portanto, o algoritmo MR-HDBSCAN* não precisa realizar a segunda etapa, se o método aglomerativo for

executado. Nesse contexto, o Algoritmo 19 agrega e ordena localmente as arestas $[v, u, d_{mreach}]$ em ordem crescente de peso, à medida que as instâncias de *ReduceByKey* são executadas.

Nesse ponto, no Algoritmo 19 começa o processo de construção de hierarquia, e, primeiramente, considera os objetos (vértices) como subárvores (subcomponentes) individuais, tal como descrito anteriormente para o contexto do método aglomerativo. Assim, para cada aresta, iniciando da aresta de menor peso e partindo iterativamente, determina os componentes conectado, usando *union-find*, no nível corrente da hierarquia e define rótulos de grupos apropriados para cada componente identificado.

Diante disso, nesse algoritmo, além determinar qual tipo de grupo será formado em cada nível hierárquico, nele também são realizados cálculos referentes a estabilidade de cada grupo, i.e., se dois componentes são potenciais ruídos e quando são unidos em um grupo C_i no nível corrente e se tornam um grupo válido, ou se dois componentes, em que um é potencial ruído e o outro não, então o termo $S_1(C_i)$. Caso, dois componentes supostamente, em relação à m_{clSize} , são grupos válidos no nível anterior, então o termo $S_2(C_i)$ e a estabilidade geral desses grupos são calculadas (6.2), pois é neste ponto que esses grupos nasceriam, segundo (CAMPELLO *et al.*, 2013a). O resultado final do Algoritmo 19 é a hierarquia composta por todos os grupos, representados pelos pares $\langle 1, Hierarquia \rangle$.

Adicionalmente, no Algoritmo 19 também é calculado a estimativa de detecção de *outliers* pelo método *GLOSH*. Para que isso ocorra, a ideia é recolher as informações necessárias para cada objeto, i.e., último nível em que cada objeto pertenceu a um grupo válido na hierarquia, à medida que a hierarquia é construída. É preciso, encontrar para cada grupo da hierarquia, qual ou quais objetos são os mais densos, ou seja, aqueles objetos que permaneceram por mais tempo em grupos considerados válidos, segundo (CAMPELLO *et al.*, 2015).

6.2.4 Grupos Significativos e Detecção de Outliers

No que se refere a extração de grupos mais significativos a partir da *Hierarquia*, tal extração é feita pela função de *ReduceByKey* descrita no Algoritmo 20. Esse algoritmo segue uma *bottom-up* aninhada a um problema de otimização (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015) (descrito em detalhes na Seção 6.2.3).

De maneira resumida, a abordagem *bottom-up* inicia o processo considerando que a coleção de grupos que pertence à *Hierarquia* (árvore de grupos), exceto pelo grupo raiz que representa o grupo único com todos os objetos, estejam na solução final. Então, pelos níveis mais profundos das árvores, ou seja, pelos grupos denotados como *nó* folhas (que não possuem subgrupos filhos) e de baixo para cima, analisa a estabilidade de cada grupo em relação aos seus grupos filhos (caso existam). Ou seja, se a estabilidade de um determinado grupo for menor do que a soma das estabilidades de seus grupos filhos, então esses grupos filhos permanecem na solução. Caso contrário, são retirados. Esse processo é feito para todos os grupos até que a

Algoritmo 19 – *ReduceByKey*: Método Aglomerativo de Construção de hierarquia de MR-HDBSCAN*

Entrada: os pares $\langle 1, \{v, u, d_{mreach}\} \rangle$ das *MSTs* locais e das arestas intergrupos originadas na primeira etapa de MR-HDBSCAN*; m_{clSize} o número mínimo de objetos de um grupo válido (opcional).

```

1:  $Arestas \leftarrow ordenaOrdemCrescente(\{v, u, d_{mreach}\})$ 
2:  $Vertices \leftarrow procuraTodosVertices(Arestas)$ 
3: para todo  $v \in Vertices$  faça
4:    $(T_v) \leftarrow CriaArvore(v)$ 
5: fim para
6:  $Hierarquia \leftarrow \emptyset$  ▷ inicializa hierarquia
7: para todo  $Arestas$  faça
8:    $\{v, u, d_{mreach}\} \leftarrow ExtraiPrimeiraAresta(Arestas)$ 
9:   se  $T_v \neq T_u$  então ▷ vértices em diferentes árvores
10:    se  $|T_v| < m_{clSize}$  então
11:      se  $|T_u| < m_{clSize}$  então
12:        se  $|T_v| + |T_u| \geq m_{clSize}$  então
13:           $C_{v,u} \leftarrow CriaGrupo(T_v, T_u)$ 
14:           $adicionaGrupo(C_{v,u}, d_{mreach}, Hierarquia)$ 
15:           $estabilidade(C_{v,u}, T_v, T_u, d_{mreach}, Hierarquia)$ 
16:           $outlierInformacao(C_{v,u}, T_v, T_u, d_{mreach}, Hierarquia)$ 
17:        fim se
18:      senão
19:         $C_u \leftarrow AdicionaObjeto(T_v, C_u)$ 
20:         $atualizaGrupo(C_u, Hierarquia)$ 
21:         $atualizaEstabilidade(C_u, T_v, d_{mreach}, Hierarquia)$ 
22:         $outlierInformacao(C_u, T_v, d_{mreach}, Hierarquia)$ 
23:      fim se
24:    senão
25:      se  $|T_u| < m_{clSize}$  então
26:         $C_v \leftarrow AdicionaObjeto(T_u, C_v)$ 
27:         $atualizaGrupo(C_v, Hierarquia)$ 
28:         $atualizaEstabilidade(C_v, T_u, d_{mreach}, Hierarquia)$ 
29:         $outlierInformacao(C_v, T_u, d_{mreach}, Hierarquia)$ 
30:      senão
31:         $C_{v,u} \leftarrow UneGrupos(C_v, C_u)$ 
32:         $adicionaObjeto(C_{v,u}, d_{mreach}, Hierarquia)$ 
33:         $calculaEstabilidadeFinal(C_v, d_{mreach}, Hierarquia)$ 
34:         $calculaEstabilidadeFinal(C_u, d_{mreach}, Hierarquia)$ 
35:         $estabilidade(C_{v,u}, T_v, T_u, d_{mreach}, Hierarquia)$ 
36:      fim se
37:    fim se
38:     $uneSubarvores(T_v, T_u, \{v, u, d_{mreach}\})$ 
39:  fim se
40: fim para
41: retorna  $\langle 1, Hierarquia \rangle$ 

```

solução com os grupos mais significativos é obtida. Por fim, o Algoritmo 20 tem como saída os *Grupos Significativos* representados por $\langle \text{nulo}, \mathbf{P} \rangle$.

Algoritmo 20 – *Reducer*: Grupos Significativos de MR-HDBSCAN*.

Input: a hierarquia $\langle l, \text{Hierarquia} \rangle$, incluindo a estabilidade dos grupos.

```

1: procedimento ReduceByKey( $\langle l, \text{Hierarquia} \rangle$ )
2:    $\mathbf{P} \leftarrow \text{ordenacaoBottomUp}(\text{ExtraiTodosOsGrupos}(\text{Hierarquia}))$ 
3:   remove  $\mathbf{C}_{raiz}$  de  $\mathbf{P}$ 
4:   para todo  $\mathbf{C}_i \in \mathbf{P}$  faça
5:     se  $\text{contemFilhos}(\mathbf{C}_i)$  então
6:       se  $S(\mathbf{C}_i) \leq S(\mathbf{C}_{il}) + S(\mathbf{C}_{ir})$  então
7:          $S(\mathbf{C}_i) \leftarrow S(\mathbf{C}_{il}) + S(\mathbf{C}_{ir})$ 
8:         remove  $\mathbf{C}_i$  de  $\mathbf{P}$ 
9:       senão
10:        remove  $\mathbf{C}_{il}$  e  $\mathbf{C}_{ir}$  de  $\mathbf{P}$ 
11:      fim se
12:    fim se
13:  fim para
14:  retorna  $\langle l, \mathbf{P} \rangle$ 
15: fim procedimento

```

No que se refere ao cálculo de graus de anomalia pelo método *GLOSH*, pode ser feito por uma função de *mapper* que recebe como entrada os pares $\langle l, \text{Hierarquia} \rangle$, que além de conter os grupos, contém todos os momentos em que qualquer objeto do conjunto de dados foi desanexado de um grupo válido para se tornar um ruído em um nível da hierarquia. Como já citado anteriormente, a *Hierarquia* contém o nível em que o objeto mais denso deixou de existir, dando fim a esse grupo. Então, com a união dessas informações e com apenas uma busca, a função *mapper* é capaz de calcular em paralelo os graus de anomalia para cada objeto utilizando a Equação 3.9.

6.2.5 Análise de Complexidade Assintótica

Uma análise de complexidade assintótica foi feita no que se refere a cada etapa de *MR-HDBSCAN**. Para fins de abstração de arquiteturas computacionais, foi considerado a existência de p unidades de processamento (processadores) disponíveis que trabalham em paralelo e detêm cargas de trabalho balanceadas em relação a quantidade de dados processada ¹⁰.

Desta maneira, seja $\mathbf{X}_i$ um subconjunto de dados composto por n_i objetos d -dimensionais. Então, na primeira etapa de *MR-HDBSCAN**, a fase de *MapPartitionsToPair* contém complexidade assintótica de tempo de execução $O(n_i^2 \times d)$ no pior caso, quando a capacidade de processamento (τ) não é excedida, ou seja, quando *MSTs* locais são construídas a partir de $\mathbf{X}_i$. Por outro lado, quando τ é excedido, i.e., quando é necessário a construção de modelos locais

¹⁰ O valor p pode ser considerado como o número de instâncias de *mappers* ou *reducers* em execução simultaneamente.

sobre m_i amostras aleatórias (ou *Data Bubbles*) extraídas de $\mathbf{X}_i$, a complexidade de tempo é determinada como $O(\frac{d \times m_i \times n_i}{p})$, no pior caso.

Ainda na primeira etapa de *MR-HDBSCAN**, para construir cada estatística suficiente para extração de um determinado *Data Bubble* ou objeto representativo por meio de uma amostra aleatória, a complexidade assintótica de tempo é $O(d \times n_{ij})$, em que n_{ij} representa o número de objetos atribuídos à j -ésima amostra aleatória mais próxima em $\mathbf{S}_i$. Dessa forma, dado que exista m_i amostras aleatórias em $\mathbf{S}_i$, a complexidade assintótica total da fase de combinação de *MR-HDBSCAN** é dada por $O(\frac{d \times m_i \times \max_i(n_{ij})}{p})$ no pior caso, em que $\max_i(n_{ij})$ representa o número máximo de objetos atribuídos à j -ésima amostra em $\mathbf{S}_i$. Já na fase de construção de modelos locais de *HDBSCAN** sobre *Data Bubbles* (ou amostras aleatórias), a complexidade assintótica de tempo é dada por $O(m_i^2 \times d)$. Por fim, a fase de indução de partição sobre um determinado subconjunto de dados pode ser dada como $O(\frac{n_i \times m_i}{p})$.

A segunda etapa de *MR-HDBSCAN**, que representa a combinação de *MSTs locais* em uma única *MST* global estendida ordenada, contém complexidade de tempo igual a $O(|E| \log |E|)$ no pior caso, em que $|E|$ representa o número de arestas a serem ordenadas.

Por fim, a terceira e última etapa de *MR-HDBSCAN**, que corresponde à construção de hierarquia baseada em densidade a partir de todos os objetos de um determinado conjunto de dados tem sua análise de complexidade de tempo descrita de duas maneiras. A primeira corresponde à utilização do método divisivo de construção de hierarquia (apresentado na Subseção 6.2.3.1.2), enquanto a segunda corresponde à utilização do método aglomerativo (Subseção 6.2.3.2.2).

A análise de complexidade assintótica referente ao tempo de execução do método divisivo de construção de hierarquias é dividida em três principais partes: a determinação de subcomponentes conectados (grupos) pelo método *CC-MR* (SEIDL *et al.*, 2012); a atribuição de objetos aos seus devidos grupos; e a atualização de árvores de grupos. Nesse contexto, os componentes conectados podem ser determinados a cada nível hierárquico em $O(c \times \frac{|U| \times (|V| + |V| \log |V|)}{p})$ no pior caso¹¹, em que c é uma constante que representa o número de iterações do algoritmo *CC-MR*, $|U|$ o número de vértices origem e $|V|$ o número de vértices adjacentes de todos os vértices origem do grafo (como descrito na Subseção 6.2.3.1.1). Já a atribuição de rótulos de grupos para cada objeto em um determinado nível hierárquico contém complexidade de tempo de $O(\frac{n}{p})$ no pior caso, em que n representa o número de objetos a ser rotulados. Por fim, para atualizar a árvore de grupos a cada nível hierárquico, a complexidade de tempo é dada por $O(\frac{k_j \times \log n_i}{p})$ no pior caso, em que k_j representa o número de grupos encontrados a cada nível da hierarquia e n_i representa o número de objetos atribuídos ao i -ésimo grupo, i.e., (C_i) .

No que se refere ao método aglomerativo de construção de hierarquias, a análise de

¹¹ Dado que, para cada vértice origem pertencente a U , seja necessário $O(|v_i|)$ para a emissão de $v_i \cup V$ novos pares de arestas e $O(|v_i| \log |v_i|)$ para ordenar v_i vértices adjacentes, em que v_i representa os vértices adjacentes de um determinado vértice origem.

complexidade de tempo é dividida em três partes: a determinação de subcomponentes conectados a cada nível hierárquico; atribuição dos objetos aos seus devidos grupos; e a atualização da árvore de grupos. Nesse contexto, para determinar os subcomponentes conectados, a complexidade de tempo pode ser dada por $O(|U| \times |V|)$ de uma estrutura de *union-find* (CORMEN *et al.*, 2001). Já a atribuição de objetos aos seus devidos grupos é feita em $O(n)$ no pior caso, em que n corresponde ao número de objetos a ser rotulado. Por fim, para atualizar a árvore de grupos a complexidade de tempo é definida por $O(k)$, em que k representa o número de grupos da árvore.

No que se refere à complexidade geral de *MR-HDBSCAN**, a mesma pode ser descrita pela soma de todas as complexidades das etapas do métodos. Nesse sentido, *MR-HDBSCAN** é um algoritmo de ordem quadrática, em relação ao número de objetos do conjunto de dados.

Adicionalmente, para a extração de grupos significativos a partir da hierarquia construída por *MR-HDBSCAN** a complexidade de tempo pode ser dada por $O(\log k)$, em que k representa o número de grupos da árvore. Já a complexidade para calcular a estimativa de *outliers* pelo método *GLOSH* pode ser determinada como $O(\frac{k \times n}{p})$.

6.3 Considerações Finais

Neste capítulo foi descrita uma nova arquitetura de implementação de *Recursive Sampling* baseada em *MapReduce*, conhecida como *MR-HDBSCAN**, proposta neste trabalho de pesquisa. Especificamente, essa arquitetura é uma importante contribuição para o cenário de agrupamento de grandes quantidades de dados distribuídos em locais distintos. Ainda, a arquitetura proposta pode servir como uma abstração da intuição de *MR-HDBSCAN** e, consequentemente, ser aplicada e implementada em outros conceitos de programação ou arquitetura computacional, por exemplo, no contexto de memória compartilhada.

EXPERIMENTOS

Neste capítulo são discutidos dois experimentos realizados com o propósito de validar as técnicas de agrupamento de dados propostas neste trabalho de mestrado. A estrutura desses dois experimentos é ilustrada na Figura 34. Assim, o primeiro experimento, apresentado na Seção 7.2, foi feito com variações de *HDBSCAN** baseadas em *MapReduce* (Capítulo 6) incorporado à plataforma *Apache Spark*. As implementações das variações utilizadas nesse primeiro experimento são consideradas como aplicações direcionadas ao cenário de computação paralela composta por memória distribuída. Já o segundo experimento, apresentado na Seção 7.3, foi realizado a partir de variações de *HDBSCAN**, apresentadas no Capítulo 5, baseadas em arquitetura de *hardware* paralela e com memória compartilhada. Além disso, as variações nesse segundo experimento foram implementadas sobre a linguagem R. Adicionalmente, cada experimento relaciona as técnicas envolvidas por meio de três etapas: (i) avaliação de qualidade de agrupamento de dados; (ii) avaliação de desempenho computacional; (iii) e avaliação de detecção de *outliers* nos conjuntos de dados investigados.

7.1 Conjunto de Dados

Para avaliar as técnicas baseadas em memória compartilhada e as baseadas em memória distribuída, descritas nos Capítulos 5 e 6, respectivamente, foram utilizados 12 conjuntos de dados, sendo 3 conjuntos artificiais e 9 conjuntos reais. Os conjuntos de dados artificiais são compostos por misturas gaussianas obtidas pelo pacote do R, conhecido como *MixSim* (MELNYKOV *et al.*, 2012). Especificamente, os 3 conjuntos de dados artificiais possuem as seguintes características descritas na Tabela 1:

Já os 9 conjuntos de dados reais são dados que representam diferentes cenários de aplicações e são disponíveis no repositório de aprendizado de máquina (UCI)¹ (BACHE; LICHMAN, 2013). Na Tabela 2 são apresentadas as características dos 9 conjuntos de dados reais utilizados

¹ <https://archive.ics.uci.edu/ml/index.php>

Figura 34 – Esquema de experimentos utilizando memória distribuída e compartilhada.

Tabela 1 – Características dos conjuntos de dados artificiais.

Conjunto de dados	número de objetos	número de atributos	número de grupos
<i>Gauss1</i>	1×10^6	10	20
<i>Gauss2</i>	3×10^6	10	30
<i>Gauss3</i>	5×10^6	10	50

nos experimentos deste capítulo. Especificamente, cada objeto de todos os conjuntos de dados analisados nos experimentos deste capítulo possui um atributo de rótulo de classe associado a ele. Cada rótulo representa um histórico ou fenômeno de interesse de um determinado objeto (TAN *et al.*, 2005).

Tabela 2 – Características dos conjuntos de dados reais.

Conjunto de dados	número de objetos	número de atributos	número de classes
<i>YearPrediction</i>	515345	90	89
<i>Poker</i>	1025010	11	10
<i>Credit</i>	30000	24	2
<i>Magic</i>	19020	11	2
<i>Shuttle</i>	58000	9	3
<i>SensorLess</i>	58509	49	11
<i>Skin</i>	245057	4	2
<i>HEPMASS</i>	10500000	28	2
<i>HT Sensor</i>	919438	11	3

- *YearPrediction*: É um conjunto de dados composto por atributos numéricos e um atributo de classe baseado em previsões do ano de lançamento de músicas a partir de recursos de

áudio. As músicas representadas neste conjunto de dados são, em sua maioria, ocidentais, faixas comerciais que variam entre os anos de 1922 a 2011, com uma elevação no ano de 2000.

- *Poker*: É um conjunto de dados de atributos numéricos proposto para predição de mãos do jogo de *poker*. Esse conjunto de dados é composto por registros que representam diferentes exemplos de mãos ou jogadas elaboradas com cinco cartas retiradas de um baralho padrão de 52 cartas no total. Cada carta é descrita por dois atributos, chamados de terno e classificação, os quais totalizam 10 atributos preditivos (representados no intervalo de $[0,9]$). Existe um atributo classe que representa o que é conhecido como “mão do *poker*”, em que a ordem das cartas são importantes para as jogadas.
- *Credit*: É um conjunto de dados numéricos utilizado para predição de padrões de pagamentos de consumidores de Taiwan por meio da comparação da precisão preditiva da probabilidade de inadimplência dentre esses consumidores. Esse conjunto de dados é dividido em duas classes: a de clientes credíveis ou não credíveis.
- *Magic*: São dados numéricos gerados para a simulação de registros de partículas de gama alta em um telescópio de gama *Cherenkov* atmosférico terrestre usando a técnica de imagem (BOCK *et al.*, 2004). Esse conjunto de dados conta com duas classes que representam um padrão denominado *shower image*.
- *Shuttle*: É um conjunto de dados composto por nove atributos numéricos. Esse conjunto de dados contém três classes preditivas, em que uma delas contém cerca de 80% dos dados. Adicionalmente, esse conjunto de dados é utilizado comumente em estudos de métodos de detecção de *outliers* não-supervisionados (ABE *et al.*, 2006).
- *SensorLess*: Conjunto de dados baseado em características extraídas de sinais de acionamento de corrente elétrica. Essas características foram geradas por diferentes mecanismos, resultando em 11 classes distintas. Cada característica representa valores numéricos medidos por uma sonda de corrente e um osciloscópio em duas fases.
- *Skin*: É um conjunto de dados de pele coletado por meio de valores de RGB de imagens de rostos de diferentes grupos etários, sendo: jovens, adultos e idosos. Nesse conjunto também estão presentes características de grupos étnicos e de gêneros obtidos a partir das bases de dados *FERET* e *PAL* (PHILLIPS *et al.*, 1998). No mais, esse conjunto de dados é dividido em conjunto de treinamento (194198 objetos) e conjunto de teste (50859). Esse tipo de divisão de dados é comum na avaliação de técnicas de aprendizado supervisionado de dados (TAN *et al.*, 2005).
- *HEPMass*: Conjunto de dados referentes à detecção de colisões desconhecidas de partículas. Tal característica é fundamental no contexto da física e possibilita que a partir do aprendizado sobre esses dados seja possível estimar ou prever a massa de partículas

desconhecidas. Esse conjunto de dados é separado em conjunto de treinamento (7 milhões de objetos) e conjunto de teste (3.5 milhões de objetos). Adicionalmente, esse conjunto é dividido em duas classes: classe (*signal*) e classe (*background*). Essas classes são utilizadas para separar colisões produtoras de partículas de um recurso de fundo.

- *HT Sensor*: Esse conjunto de dados foi construído a partir de gravações de um conjunto de sensores de gás composto por 8 sensores de gás MOX e um sensor de temperatura e umidade. Esse conjunto de sensores foi submetido a dois estímulos distintos: o de vinho e de banana. As respostas em relação aos estímulos de vinho e banana foram coletadas a partir do estímulo próximo aos sensores de gás. Cada estímulo foi registrado entre o intervalo de tempo de sete minutos a duas horas e com duração média de quarenta e dois minutos. Nesse contexto, esse conjunto de dados possui classes preditivas baseadas em três condições: vinho, banana e atividade de fundo.

7.2 Experimento utilizando Memória Distribuída

O principal objetivo do presente experimento foi comparar a técnica *MR-HDBSCAN**, proposta neste trabalho, com a técnica de divisão de dados *Random Blocks* utilizada para a paralelização exata *HDBSCAN** (SYED, 2014). A comparação está relacionada à qualidade de agrupamentos de dados, ao tempo computacional e aos graus de *outliers* reproduzidos por essas técnicas (descritas em detalhes na Subseção 7.2.1). As técnicas *MR-HDBSCAN** e *Random Blocks* foram implementadas sobre o conceito de *MapReduce* na plataforma *Apache Spark*², que é um ambiente de computação paralela e distribuída para análise massiva de dados.

7.2.1 Algoritmos

As três técnicas de agrupamento de dados utilizadas no presente experimento são derivações paralelas e distribuídas do original *framework HDBSCAN** (CAMPELLO *et al.*, 2015). A primeira se refere ao algoritmo de divisão de dados *Random Blocks*, apresentada no Capítulo 4. Esse algoritmo foi implementado sobre o conceito de *MapReduce*, como sugerido em (SYED, 2014). A segunda técnica corresponde ao algoritmo *MR-HDBSCAN**, proposto neste trabalho. Basicamente, essa técnica foi avaliada em suas duas vertentes, i.e., utilizando apenas amostras aleatórias ou *Data Bubbles* em sua composição, como descrito no Capítulo 6. Para fins de simplificação, ambas as técnicas derivadas de *MR-HDBSCAN** são denotadas neste experimento como *MR-HDBSCAN*_{sampling}* e *MR-HDBSCAN*_{bubbles}*, que são aplicadas no contexto de amostras aleatórias simples e *Data Bubbles*, respectivamente.

Os algoritmos *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*, apresentados no Capítulo 6, e a técnica *Random Blocks* (Capítulo 4) produzem *MSTs* globais estendidas baseadas em

² *Apache Spark* na versão 2.1.1 e a linguagem utilizada foi *Java 7*.

densidade de $HDBSCAN^*$ a partir de conjuntos completos de dados. À vista disso, foram utilizados dois métodos de construção de hierarquias de grupos baseados em densidade, o método divisivo (método_{divisivo}) e o método aglomerativo (método_{aglomerativo}), ambos propostos neste trabalho de mestrado para determinar hierarquias a partir de quaisquer $MSTs$ globais estendidas adquiridas por meio de quaisquer algoritmos investigados neste experimento.

7.2.2 Configurações de Algoritmos

Os algoritmos *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ possuem alguns parâmetros em comum que são descritos na Tabela 3. O parâmetro m_{pts} correspondente as três técnicas é utilizado para indicar o número mínimo de objetos pertencente a uma determinada vizinhança, como definido em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). O parâmetro m_{clSize} (Tabela 3) é utilizado nos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ para definir o tamanho mínimo de qualquer grupo válido nos modelos locais de $HDBSCAN^*$ construídos durante execuções desses algoritmos.

Adicionalmente, os parâmetros m_{pts} e m_{clSize} nos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ são considerados iguais para cada conjunto de dados, i.e., $m_{pts} = m_{clSize}$, como sugerido em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Ainda, os valores de m_{pts} e m_{clSize} foram ajustados empiricamente para cada conjunto de dados avaliado de acordo com a qualidade dos agrupamentos feitos pelos algoritmos *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$.

O parâmetro τ dos algoritmos *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ foi determinado empiricamente como $\tau = 10000$ para todos os experimentos de acordo com a capacidade de processamento de cada unidade de processamento utilizada neste trabalho de mestrado. Esse valor de τ indica o número de objetos que podem ser processados localmente por qualquer uma das unidades utilizadas.

Tabela 3 – Conjunto de parâmetros utilizados pelos algoritmos *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$.

Conjuntos Dados	Random Blocks		$MR-HDBSCAN^*_{sampling}$			$MR-HDBSCAN^*_{bubbles}$		
*	s	m_{pts}	m_{clSize}	m_{pts}	m (%)	m_{clSize}	m_{pts}	m (%)
<i>YearPrediction</i>	2783	30	30	30	1.9	30	30	1.9
<i>Poker</i>	10251	50	50	50	0.9	50	50	0.9
<i>HT Sensor</i>	5506	30	30	30	1	30	30	1
<i>Skin</i>	1961	40	40	40	4	40	40	4
<i>Gauss1</i>	10000	50	50	50	0.9	50	50	0.9
<i>Gauss2</i>	30000	50	50	50	0.3	50	50	0.3
<i>Gauss3</i>	50000	50	50	50	0.2	50	50	0.2
<i>HEPMASS</i>	105000	50	50	50	0.09	50	50	0.09

O parâmetro s do algoritmo *Random Blocks* indica o número de subconjuntos de dados em

que um determinado conjunto de dados será dividido em diferentes unidades de processamento independentes. Como descrito no Capítulo 4, o número de subconjuntos s é utilizado para criar $\binom{s}{2 \times m_{pts}}$ *Data Blocks* não sobrepostos para gerar *MSTs* locais baseadas em densidade. Além disso, para calcular o número de subconjuntos s sobre cada conjunto de dados, foi necessário determinar primeiramente o número de objetos n_s que compunha cada subconjunto s pela Equação 7.1 e, então, utilizar a Equação 7.2 para determinar s :

$$n_s = \frac{\tau}{2 \times m_{pts}} \quad (7.1)$$

$$s = \frac{n}{n_s} \quad (7.2)$$

em que n indica o número total de objetos em qualquer conjunto de dados. A ideia por trás das Equações 7.1 e 7.2 é garantir que cada *Data Block* construído a partir de cada subconjunto s (descrito no Capítulo 4) seja inferior ou igual a τ em número de objetos.

Já o parâmetro m representa a porcentagem do número de objetos amostrados de cada subconjunto de dados para a construção de modelos locais de *HDBSCAN** por *MR-HDBSCAN** (em *Data Bubbles* ou amostras aleatórias), assim como descrito no Capítulo 6. Um exemplo seria considerar $m = 10$, isso significa que 10% de objetos serão amostrados a partir de um determinado (sub)conjunto de dados. Nesse contexto, os valores de m , dos algoritmos *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*, apresentados na Tabela 3, foram determinados com base na capacidade de processamento de cada unidade de processamento utilizada e no tamanho de cada conjunto de dados avaliado nos experimentos. Em outras palavras, a porcentagem de objetos a serem amostrados (m) de um determinado subconjunto de dados, durante iterações dos algoritmos *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*, deve ser menor ou igual ao número de objetos que podem ser processados localmente (τ) pelas unidades de processamento.

Para a construção de hierarquias a partir de cada *MST* global estendida obtida em cada execução dos algoritmos *Random Blocks*, *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*, foi utilizado o método divisivo proposto e apresentado na Seção 6.2.3.1 deste trabalho. Em resumo, a ideia desse método é construir, em paralelo, níveis hierárquicos a partir de remoções iterativas de arestas de maior peso de qualquer *MST* global estendida baseada em densidade. Adicionalmente, o único parâmetro utilizado pelo método divisivo é o m_{clSize} , que corresponde ao número mínimo de objetos que um grupo válido pode conter em uma hierarquia baseada em densidade referente ao conjunto completo dos dados. Logo, para cada conjunto de dados, seguem os seus respectivos valores de m_{clSize} na Tabela 4:

7.2.3 Configurações de Hardware

Em termos de *hardware*, foram utilizados um conjunto de 10 computadores (unidade de processamento independentes) interligados por uma topologia de rede local. Cada computador

Tabela 4 – Parâmetro m_{clSize} utilizado pelo método divisivo de construção de hierarquias baseadas em densidade.

Conjuntos Dados	m_{clSize}
<i>YearPrediction</i>	30
<i>Poker</i>	50
<i>HT Sensor</i>	30
<i>Skin</i>	40
<i>Gauss1</i>	50
<i>Gauss2</i>	50
<i>Gauss3</i>	50
<i>HEPMASS</i>	50

foi configurado com a plataforma *Apache Spark* em conjunto com o sistema de arquivo *HDFS*. Especificamente, cada um deles é composto por 8 GB de *RAM*, processadores *AMD FX(tm)-6100 six-core*, 6 núcleos e um *terabyte* (1T) de memória secundária.

7.2.4 Avaliação de Qualidade de Agrupamento de Dados

O objetivo dos experimentos apresentados nesta seção foi de mostrar a qualidade de agrupamentos de dados feitos pelos algoritmos *Random Blocks*, *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* sobre vários conjuntos de dados reais e artificiais. Diante disso, para fins de simplicidade, na Figura 35 é apresentada a forma com que um experimento específico foi conduzido sobre um determinado conjunto de dados. Note que o conjunto de dados avaliado possui uma partição de referência, que representa os rótulos dos objetos do conjunto de dados. No mais, essa regra se aplica aos outros conjuntos de dados avaliados neste experimento.

Ainda como pode ser visto na Figura 35, cada algoritmo, i.e., *Random Blocks*, *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* é executado sobre um mesmo conjunto de dados. Cada algoritmo produz uma *MST* global estendida sobre os dados. A partir de cada *MST* global estendida, uma hierarquia de grupos baseados em densidade é construída pelo método *divisivo* (apresentado na Seção 6.2.3). Assim, uma partição composta pelos grupos mais significativos dos dados é obtida a partir de cada hierarquia, como descrito na Subseção 3.2.2. Então, cada partição adquirida é comparada com a partição de referência do conjunto de dados em questão usando o método descrito na Subseção 7.2.4.1.

7.2.4.1 Critério de Validação

A validação de agrupamentos de dados realizados pelos algoritmos *Random Blocks*, *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* foi baseada nas partições extraídas (como descrito na 3.2.2) das hierarquias construídas a partir de cada *MST* global estendida obtida por esses algoritmos. De maneira geral, para comparar as partições obtidas por esses algoritmos, foram utilizadas as partições de referência dos conjuntos de dados avaliados neste experimento.

Figura 35 – Esquema de condução de experimento utilizando os algoritmos *Random Blocks*, *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* sobre um determinado conjunto de dados.

Especificamente, o objetivo da comparação de partições neste experimento foi mostrar que versões aproximadas de *HDBSCAN^{*}*, i.e., *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}*, podem retornar resultados próximos aos resultados exatos fornecidos pela técnica *Random Blocks* com base no *HDBSCAN^{*}*, em termos de qualidade de agrupamentos particionais de dados.

Neste contexto, o critério de validação externo *Rand Ajustado* (ou *Adjusted Rand Index* (*ARI*)) foi utilizado para validar as partições obtidas a cada execução dos algoritmos *Random Blocks*, *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* neste experimento. Esse critério (ou índice) *ARI* é um aprimoramento dos índices *Rand* (RAND, 1971) e *Jaccard* (JACCARD, 1901). Diferentes desses índices, *ARI* tem como característica principal a correção por mudança, i.e., a partir de duas partições completamente aleatórias, o valor de *ARI* esperado é próximo de zero (XU; WUNSCH, 2009). Nessa perspectiva, o índice *Rand Ajustado* (*ARI* sigla em inglês) resolve essa limitação normalizando o valor esperado para 0 quando as partições são selecionadas aleatoriamente (por mudança) ou para 1 quando existe uma perfeita concordância entre as partições comparadas (HUBERT; ARABIE, 1985; VENDRAMIN *et al.*, 2010). *ARI* é apresentado na Equação 7.3:

$$ARI = \frac{a - \frac{(a+c)(a+b)}{M}}{\frac{(a+c)(a+b)}{M} - \frac{(a+c)(a+b)}{M}} \quad (7.3)$$

em que $M = n(n-1)/2 = a + b + c + d$, i.e., corresponde à quantidade de pares possíveis de

objetos.

Assim como descrito no Capítulo 6, as partições obtidas por qualquer um dos algoritmos avaliados neste experimento são compostas por grupos mais significativos e objetos ruídos segundo as definições de Campello *et al.* (2013a), Campello *et al.* (2015). Contudo, o critério *ARI* é definido para comparar apenas duas partições livres de dados ruidosos.

Diante disso, com base na metodologia apresentada por Campello *et al.* (2013a), Campello *et al.* (2015), foi realizada uma adaptação na validação das partições obtidas por *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ para que os objetos tratados como ruídos no modelo fossem considerados como grupos únicos nas comparações com as partições de referência dos conjuntos de dados analisados.

7.2.4.2 Resultados e Discussão

A realização do presente experimento foi baseada em uma execução do algoritmo *Random Blocks* e 45 execuções de cada algoritmo ($MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$) sobre cada conjunto de dados mostrado na Tabela 3 e detalhado na Seção 7.1. Cada execução de qualquer algoritmo avaliado foi estabelecida a partir dos parâmetros indicados na Tabela 3. Ainda, cada execução dos algoritmos sobre cada conjunto de dados corresponde ao esquema de condução de experimento apresentado na Figura 35. Assim, os resultados obtidos em relação a agrupamentos de dados particionais feitos pelos algoritmos *Random Blocks*, $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ em conjunto com o método divisivo de construção de hierarquias de grupos baseados em densidade (método_{divisivo}) são apresentados na Tabela 5.

O algoritmo *Random Blocks* foi executado apenas uma vez para cada conjunto de dados devido à sua característica de obter sempre uma *MST* exata baseada em densidade, independente do número de execução, a partir de um mesmo conjunto de parâmetros, i.e., m_{pts} , que indica o número mínimo de objetos em uma determinada vizinhança, e s , que representa o número de vezes que um determinado conjunto de dados será dividido em subconjuntos menores (SYED, 2014). Já os algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ foram executados 45 vezes sobre cada conjunto de dados, pois ambos os algoritmos podem produzir resultados diferentes para a mesma configuração de seus parâmetros.

Os valores de *ARI* absoluto, mostrados na Tabela 5, indicam os resultados obtidos por *Random Blocks*. Observe que todos os valores de *ARI* absoluto são destacados em negrito para indicar a superioridade em relação aos resultados (*ARI médios*) obtidos pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$. Já os valores de *ARI* médio e *ARI* desvio padrão obtidos pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ foram calculados com base em 45 execuções de cada um desses algoritmos sobre cada conjunto de dados apresentado na Tabela 5.

Os resultados apresentados na Tabela 5 demonstram a superioridade do algoritmo *Ran-*

Tabela 5 – Agrupamentos particionais utilizando *ARI*

Conjuntos Dados	Random Blocks	MRHDBSCAN* _{sampling}		MRHDBSCAN* _{bubbles}	
*	<i>ARI</i> absoluto	<i>ARI</i> médio	<i>ARI</i> desvio padrão	<i>ARI</i> médio	<i>ARI</i> desvio padrão
<i>YearPrediction</i>	0.403	0.30176	0.00523	0.38833	0.00439
<i>Poker</i>	0.310	0.19601	0.00535	0.29774	0.00119
<i>HT Sensor</i>	0.359	0.28702	0.00161	0.33056	0.00085
<i>Skin</i>	0.441	0.36039	0.00465	0.42520	0.00211
<i>Gauss1</i>	0.881	0.69032	0.00190	0.86419	0.00087
<i>Gauss2</i>	0.820	0.58821	0.00136	0.75945	0.00212
<i>Gauss3</i>	0.801	0.60229	0.00650	0.77750	0.00275
<i>HEPMASS</i>	0.546	0.40855	0.00525	0.52949	0.00030

dom Blocks em relação aos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ em termos de qualidade de agrupamentos de dados. Tal fato já era esperado, uma vez que os algoritmos que utilizam alguma técnica de sumarização de dados, i.e., amostras aleatórias ou *Data Bubbles*, produzem agrupamentos aproximados dos dados, assim como descrito no Capítulo 6. Já *Random Blocks* pode fornecer *MSTs* globais estendidas exatas baseadas em densidade de $HDBSCAN^*$ e invariantes ao número de execuções com a mesma parametrização (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Repare também que os valores médios *ARI* obtidos pelo algoritmo $MR-HDBSCAN^*_{bubbles}$ estiveram próximos aos de *Random Blocks*. Devido à maneira sofisticada de sumarização de dados realizadas por *Data Bubbles*, as estruturas dos grupos originais, ou seja, do conjunto completo de dados, não se perdem tanto durante o agrupamento. Por outro lado, os agrupamentos feitos pelo $MR-HDBSCAN^*_{sampling}$, como pode ser visto na Tabela 5, surtiram a perda de estrutura dos grupos para a maioria dos conjuntos de dados em relação aos agrupamentos feitos por *Random Blocks*. Como discutido no Capítulo 6, a utilização de amostras simples como técnicas de sumarização de dados pode afetar drasticamente o desempenho de técnicas de agrupamento hierárquicos de dados, i.e., $HDBSCAN^*$, pois a distância utilizada para comparar pares de amostras representativas pode não se aproximar da distância real entre seus objetos representados.

É interessante notar também que os resultados obtidos pelo algoritmo $MR-HDBSCAN^*_{bubbles}$ obtiveram uma variação menor (pelo desvio padrão de cada *ARI* calculado) em termos de qualidade de agrupamentos (*ARI*) do que os resultados obtidos por $MR-HDBSCAN^*_{sampling}$ para a maioria dos conjuntos de dados avaliados, exceto para o conjunto de dados *Gauss2*. A menor variação dos resultados de $MR-HDBSCAN^*_{bubbles}$ é justificada pela capacidade dessa técnica de representar melhor um determinado conjunto de dados pela sumarização por *Data Bubbles*, mesmo que cada execução desse algoritmo seja baseada em seleções distintas de amostras aleatórias para a construção desses *Data Bubbles*.

7.2.4.3 Teste de Significância Estatística

Com o intuito de testar a confiança dos resultados obtidos no presente experimento, foi utilizado o teste t pareado unicaudal. Esse teste é capaz de inferir estatísticas sobre uma população a partir de um conjunto amostral de dados utilizando hipóteses a serem testadas (PANIK, 2012). Nesse contexto, as hipóteses levantadas para o teste realizado neste experimento abordam a qualidade dos agrupamentos de dados adquiridos sobre os conjuntos de dados descritos na Seção 7.1 pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ em ambiente distribuído.

Os agrupamentos por ambas as técnicas são avaliados com base em melhores partições extraídas a partir das hierarquias construídas. Assim, a qualidade de cada partição é obtida pelo critério externo de validação *rand* ajustado (*ARI*). Diante disso, as hipóteses de teste (nula H_0 e alternativa H_1) são descritas como:

H_0 : “Não é possível detectar diferença significativa entre os resultados do $MR-HDBSCAN^*_{bubbles}$ e do $MR-HDBSCAN^*_{sampling}$ em termos de qualidade de agrupamentos de dados (particional pelo critério *ARI*).”

H_1 : “Foi detectada diferença significativa entre os resultados dos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ em termos de qualidade de agrupamento de dados (de acordo com o critério *ARI*).”

Em termos estatísticos do teste t pareado, o $MR-HDBSCAN^*_{bubbles}$ só apresentará resultados significativamente melhores em relação ao algoritmo $MR-HDBSCAN^*_{sampling}$ se a hipótese nula (H_0) for rejeitada. Especificamente, a hipótese nula é rejeitada em casos em que o valor p é menor que 0.05. Desse modo, como pode ser visto nos valores de p destacados em negrito na Tabela 6, para todos os conjuntos de dados testados neste experimento, esse valor para cada conjunto de dados foi indicado como menor que 0.05. Portanto, $MR-HDBSCAN^*_{bubbles}$ apresentou resultados significativamente melhores do que $MR-HDBSCAN^*_{sampling}$ em termos de qualidade de agrupamentos.

7.2.5 Avaliação de Desempenho Computacional

No presente experimento também foi realizada uma avaliação de desempenho computacional dos três algoritmos investigados, i.e., *Random Blocks*, $MR-HDBSCAN^*_{sampling}$ e $MR-HDBSCAN^*_{bubbles}$. Nesse contexto, o objetivo dessa avaliação foi medir o tempo despendido por esses algoritmos para a construção de *MSTs* globais estendidas baseadas em densidade de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

7.2.5.1 Resultados e Discussão

A condução dos experimentos relatados nesta subseção tem como característica uma execução de *Random Blocks*, 45 repetições de $MR-HDBSCAN^*_{sampling}$ e 45 repetições de $MR-HDB$

Tabela 6 – Testes t pareado em relação aos resultados de agrupamentos particionais feitos pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ em ambiente distribuído.

Conjunto de Dados	valor t	valor p	DM	GL	NC (%)
<i>YearPrediction</i>	74.555	2.95006322942526e-48	0.086	44	95
<i>Poker</i>	129.054	1.08139683928633e-58	0.101	44	95
<i>HT Sensor</i>	158.730	1.22256534392544e-62	0.043	44	95
<i>Skin</i>	87.015	3.43781470843531e-51	0.064	44	95
<i>Gauss1</i>	538.298	5.83777083228703e-86	0.173	44	95
<i>Gauss2</i>	427.958	1.40850563232542e-81	0.171	44	95
<i>Gauss3</i>	169.006	7.77308455179169e-64	0.175	44	95
<i>HEPMASS</i>	151.791	8.7084790244053e-62	0.120	44	95

DM = diferença entre médias dos resultados de $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$; GL = Grau de Liberdade; NC = Nível de Confiança.

$SCAN^*_{bubbles}$ sobre cada conjunto de dados indicados na Tabela 7 e detalhados na Seção 7.1. Adicionalmente, os parâmetros desses algoritmos e configurações de *hardware* são descritos nas Subseções 7.2.2 e 7.3.4, respectivamente.

Ante o exposto, na Tabela 7 são apresentados os resultados obtidos em relação aos algoritmos *Random Blocks*, $MR-HDBSCAN^*_{sampling}$ e $MR-HDBSCAN^*_{bubbles}$. Note que, para todos os conjuntos de dados, $MR-HDBSCAN^*_{sampling}$ garante o menor tempo médio em minutos em relação a $MR-HDBSCAN^*_{bubbles}$. Ainda é possível constatar que o algoritmo *Random Blocks* detém o maior tempo de execução para todos os conjuntos de dados. Este último ocorre devido ao alto número de replicação dos dados em diferentes *data blocks* por esse algoritmo (assim como descrito no Capítulo 4) e, conseqüentemente, isso aumenta o tempo de execução de maneira drástica.

Tabela 7 – Desempenho computacional (tempo em minutos de execução) em ambiente paralelo com memória distribuída.

Conjuntos Dados	Random Blocks	MR-HDBSCAN [*] _{sampling}		MR-HDBSCAN [*] _{bubbles}	
*	tempo absoluto	média	desvio padrão	média	desvio padrão
<i>YearPrediction</i>	11622.61	106.57	17.62	109.89	21.30
<i>Poker</i>	28955.89	52.81	5.84	97.06	35.32
<i>HT Sensor</i>	31450.89	42.54	4.74	82.07	25.89
<i>Skin</i>	1743.93	21.14	4.99	60.19	26.00
<i>Gauss1</i>	13312.24	67.35	18.11	82.75	31.48
<i>Gauss2</i>	28393.65	162.07	30.50	225.40	23.45
<i>Gauss3</i>	48980.04	115.39	0.076431	182.05	0.077955
<i>HEPMASS</i>	79092.60	385.67	26.62	695.29	28.01

7.2.5.2 Teste de Significância Estatística

Foi realizado o teste t pareado unicaudal sobre cada resultado obtido a partir dos conjuntos de dados avaliados, cujo objetivo constituiu em avaliar a confiança desses resultados em termos de tempos computacionais despendidos pelos algoritmos $MRHDBSCAN^*_{sampling}$ e $MRHDBSCAN^*_{bubbles}$ para a construção de $MSTs$ globais estendidas baseadas em densidade. Diante disso, como formulação de hipótese nula H_0 e hipótese alternativa H_1 , tem a seguinte característica:

H_0 : “Os algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ possuem o mesmo desempenho em termos de desempenho computacional medido em minutos.”

H_1 : “O algoritmo $MR-HDBSCAN^*_{bubbles}$ despende maior tempo computacional medido em minutos em relação ao algoritmo $MR-HDBSCAN^*_{sampling}$.”

No que se refere à hipótese alternativa H_1 , inferir que o desempenho computacional de $MR-HDBSCAN^*_{bubbles}$ é inferior que o de $MR-HDBSCAN^*_{sampling}$ é afirmar que a construção de *Data Bubbles* por $MR-HDBSCAN^*_{bubbles}$, a cada iteração em um determinado conjunto de dados, aumenta o tempo de execução em minutos se comparado com a sumarização simples realizada pelo algoritmo $MR-HDBSCAN^*_{sampling}$.

No ponto de vista estatístico do teste t pareado unicaudal aplicado aos resultados, o algoritmo $MR-HDBSCAN^*_{bubbles}$ é denominado inferior a $MR-HDBSCAN^*_{sampling}$ em termos de desempenho computacional quando a hipótese alternativa H_1 é verdadeira. Para que H_1 seja considerada como verdadeira, o valor de p precisa ser menor que 0.05. Nesse contexto, na Tabela 8 são apresentados os resultados do teste estatístico para cada conjunto de dados avaliado neste experimento.

Tabela 8 – Teste t pareado em relação aos tempos de execução para construção de $MSTs$ globais estendidas pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$.

Conjunto de Dados	valor t	valor p	DM	GL	NC (%)
<i>YearPrediction</i>	0.797	0.214755022372993	3.323	44	95
<i>Poker</i>	8.311	7.23411251388389e-11	44.250	44	95
<i>HT Sensor</i>	10.501	7.23252277827985e-14	39.527	44	95
<i>Skin</i>	9.406	2.13009078506688e-12	39.054	44	95
<i>Gauss1</i>	2.758	0.00421600782894639	15.405	44	95
<i>Gauss2</i>	9.757	7.09176488613118e-13	53.016	44	95
<i>Gauss3</i>	9.824	5.75043291609756e-13	63.334	44	95
<i>HEPMASS</i>	49.347	1.82694444898058e-40	309.616	44	95

DM = diferença entre médias dos resultados de $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$; GL = Grau de Liberdade; NC = Nível de Confiança.

Como pode ser visto na Tabela 8, o valor de p indica a rejeição da hipótese nula H_0 para a maioria dos conjuntos de dados, exceto para o conjunto *YearPrediction*. Tal fato indica que, em termos de tempo de execução, o algoritmo $MR-HDBSCAN^*_{bubbles}$ possui um desempenho inferior a $MR-HDBSCAN^*_{sampling}$ para a maioria dos conjuntos de dados avaliados.

7.2.6 Avaliação de Métodos de Construção de Hierarquias

Foram propostos, neste trabalho de mestrado, dois métodos de construção de hierarquias baseadas em densidade (ver Seção 6.2.3) sobre o conceito de *MapReduce*, quais sejam: o método divisivo (método_{divisivo}) e o método aglomerativo (método_{aglomerativo}).

O método_{divisivo} consiste em uma abordagem divisiva de construção em que os vértices de uma determinada *MST* baseada em densidade são partes de um único grupo, inicialmente, e, então, iterativamente, a partir da remoção de arestas da *MST* em ordem decrescente de peso, subcomponentes conectados são encontrados e denominados grupos na hierarquia, de acordo com as definições apresentadas em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Já o método_{aglomerativo} utiliza uma abordagem aglomerativa para construir hierarquias baseadas em densidade. Esse método considera que todos os vértices (objetos) da *MST* são objetos (grupos) ruídos ou grupos únicos, com relação à m_{clSize} , e, então, pares de grupos são unidos por ordem crescente de peso até que formem um único grupo contendo todos os objetos.

Assim, para avaliar o desempenho computacional desses dois métodos de construção de hierarquias, foram realizados experimentos sobre os conjuntos de dados apresentados na Seção 7.1³. Especificamente, esses experimentos foram baseados em 45 *MSTs* estendidas construídas pelo método $MR-HDBSCAN^*_{bubbles}$ sobre cada conjunto de dados avaliado. A proposta foi de aplicar tanto o método divisivo, quanto o método aglomerativo sobre cada *MST* estendida e extrair médias e desvios padrão do tempo de execução (em minutos) desses métodos, a partir das repetições sobre as *MSTs* de cada conjunto de dados. Os resultados podem ser vistos na Tabela 9.

Note que, em média, método_{aglomerativo} retornou melhores resultados em relação a método_{divisivo} para todos os conjuntos de dados analisados. Isso acontece porque, diferente de método_{divisivo}, método_{aglomerativo} não necessita realizar um determinado número de iterações até a convergência para encontrar componentes conectados a cada construção de nível (como descrito no Capítulo 6). Caso a convergência para encontrar os componentes demore, o tempo de execução de método_{divisivo} pode aumentar consideravelmente.

Especificamente, o método aglomerativo (método_{aglomerativo}), ao invés de executar diversas iterações para encontrar componentes em um determinado nível hierárquico, esse método une os componentes em uma única iteração por nível hierárquico, utilizando estratégias eficientes de

³ Todas as *MSTs* utilizadas neste experimento, para a construção de hierarquia por ambos os métodos, foram geradas nos experimentos descritos na Subseção 7.2.5.1. Portanto, não foi necessário realizar um novo conjunto de experimentos completos, i.e., desde a construção de *MSTs* globais até a construção de hierarquias.

Tabela 9 – Avaliação de desempenho computacional de partições obtidas a partir de hierarquias de grupos baseados em densidade construídas por método_{divisivo} e método_{aglomerativo}.

Conjuntos Dados	método _{divisivo}		método _{aglomerativo}	
	tempo médio (minutos)	desvio padrão	tempo médio (minutos)	desvio padrão
* <i>YearPrediction</i>	106.59	5.68562	67.65	4.52118
<i>Poker</i>	212.72	11.35723	135.37	12.94467
<i>HT Sensor</i>	188.90	5.80578	124.82	8.61783
<i>Skin</i>	51.21	2.80014	32.95	2.11927
<i>Gauss1</i>	206.87	28.570	136.38	10.59621
<i>Gauss2</i>	356.00	21.53010	266.95	2.36464
<i>Gauss3</i>	520.17	17.52785	402.20	46.51758
<i>HEPMASS</i>	692.87	27.18023	532.63	8.73798

união e procura (*union-find*), discutidas no Capítulo 6. Esse fato pode tornar método_{aglomerativo} mais rápido do que método_{divisivo}.

7.2.7 Avaliação de Detecção de Outliers usando o método GLOSH

Assim como o original *framework HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), os métodos de construção de hierarquias baseadas em densidade propostos neste trabalho (Capítulo 6), i.e., método_{divisivo} ou método_{aglomerativo}, também são capazes de determinar uma estimativa (*score*) de *outliers* para cada objeto de um determinado conjunto de dados pelo método GLOSH (CAMPELLO *et al.*, 2015), a partir de qualquer *MST* baseada em densidade de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Nesse contexto, nesta subseção é apresentado um novo conjunto de experimentos com o objetivo de avaliar a estimativa de detecção de *outliers* de GLOSH calculada pelo método divisivo de construção de hierarquias baseadas em densidade proposto neste projeto de pesquisa a partir de *MSTs* formadas pelos algoritmos *MR-HDBSCAN**_{bubbles}, *MR-HDBSCAN**_{sampling} e *Random Blocks*. De maneira ilustrativa, o esquema de cada experimento pode ser descrito pela Figura 36.

7.2.7.1 Critério de Validação

Para a validação dos resultados referentes à estimativa de detecção de *outliers* pelo método GLOSH obtidos a partir do método divisivo de construção de hierarquias condicionado as *MSTs* fornecidas pelos algoritmos *MR-HDBSCAN**_{bubbles}, *MR-HDBSCAN**_{sampling} e *Random Blocks*, foi utilizada uma medida baseada na curva *Receiver Operating Characteristic (ROC)* (KRZANOWSKI; HAND, 2009).

A curva *ROC* é representada por um gráfico de teste de sensibilidade (taxa de verdadeiros positivos) *versus* especificidade (taxa de falsos positivos). Ainda, a curva *ROC* pode ser utilizada para representar a área sobre a curva (*Area Under the Curve (AUC)*) por meio de uma única

Figura 36 – Esquema de condução de experimento utilizando os algoritmos *Random Blocks*, *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}* sobre um determinado conjunto de dados para a avaliação de detecção de *outliers*.

medição. Dessa forma, uma única representação gráfica de *ROC* utilizando *AUC* pode ilustrar uma gama de resultados numéricos.

A curva *ROC AUC* pode ser representada graficamente no intervalo de 0 a 1. Para um contexto *ranking* aleatório que descreve as taxas de verdadeiros positivos e falsos positivos, ambos irão aumentar suas taxas e resultarão em uma área que, aproximadamente, preenche metade do espaço (aproximadamente 0.5). Um perfeito resultado de *ROC AUC* retorna um valor igual a 1, o que corresponde a recuperação de todos os *outliers* anteriormente ao primeiro *inlier* recuperado. De maneira geral, a medida *ROC AUC*, no contexto de detecção de *outliers*, pode ser interpretada como a probabilidade de que um par de objetos escolhidos aleatoriamente seja composto por um exemplo positivo (*outlier*) e um exemplo negativo (*inlier*) em uma ordenação.

Formalmente, seja O o conjunto de objetos o considerados como *outliers* e I o conjunto de objetos i denominados como *inliers*, então, a medida *ROC AUC* pode ser calculada pela Equação 7.4:

$$ROCAUC := media_{i \in I, o \in O} = \begin{cases} 1 & \text{se } score(o) > score(i) \\ 1/2 & \text{se } score(o) = score(i) \\ 0 & \text{se } score(o) < score(i) \end{cases} \quad (7.4)$$

em que $score(.)$ representa o cálculo de *GLOSH* para qualquer objeto de um determinado conjunto de dados.

7.2.7.2 Configurações

Foram considerados os conjuntos de dados *YearPrediction*, *Credit*, *Magic*, *Shuttle*, *SensorLess* e *Skin* (Tabela 2) no presente experimento. Então, para avaliar os *scores* de *GLOSH* pela *ROC (AUC)* sobre cada um desses conjuntos de dados, foi utilizada uma adaptação de abordagens comuns praticadas por (ABE *et al.*, 2006; YANG *et al.*, 2008; KRIEGEL *et al.*, 2008; ZHANG *et al.*, 2009; KELLER *et al.*, 2012; ZIMEK *et al.*, 2013).

Nessas abordagens é considerada que uma pequena porcentagem dos objetos seja determinada como *outliers* e os demais objetos como *inliers* a partir de um determinado conjunto de dados. Além disso, os dados de treinamento e teste foram combinados para cada conjunto de dados, com o intuito de desempenhar a tarefa de detecção de *outliers*, como realizado por Campello *et al.* (2015).

Então, para o conjunto de dados *YearPrediction*, todos os objetos contidos nas classes 1922 à 1954 foram considerados como *outliers*, correspondendo em 0.25% do total de objetos desse conjunto de dados. Já para os conjuntos de dados *Credit*, *Magic*, *SensorLess* e *Skin*, foram considerados 5% dos objetos como *outliers* de suas respectivas menores classes: classe 0, *hadron*, classe 11 e classe 1, respectivamente. O restante (95%) dos objetos desses conjuntos foram considerados como *inliers*. Por fim, utilizando uma adaptação de Abe *et al.* (2006), todos os objetos das classes 2,3, 6 e 7, que são as menores classes do conjunto de dados *Shuttle*, foram considerados como *outliers*.

Para a avaliação de detecção de *outliers* pelos algoritmos $MR-HDBSCAN^*_{bubbles}$, $MR-HDBSCAN^*_{sampling}$ e *Random Blocks*, foi considerada uma nova série de experimentos com configurações variadas dos parâmetros de cada algoritmo avaliado. Desse modo, no que se refere aos parâmetros m_{pts} e m_{clSize} desses três algoritmos, foi considerado um intervalo de 10 valores da seguinte maneira: $m_{pts} = m_{clSize} = \{4, 8, 12, 20, 30, 40, 50, 60, 70\}$, para cada conjunto de dados avaliado neste experimento. Já o parâmetro que indica a capacidade de processamento de cada unidade de processamento foi determinado em todos experimentos como $\tau = 10000$. Adicionalmente, todos esses parâmetros foram escolhidos de maneira empírica.

Configurações do parâmetro m dos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ são apresentadas na Tabela 10. Para cada conjunto de dados avaliado neste experimento é considerada uma determinada porcentagem (representada por m) do número de objetos a serem amostrados a cada iteração de ambos os algoritmos para a construção de modelos locais de $HDBSCAN^*$ sobre *Data Bubbles* (ou amostras aleatórias).

Assim como descrito anteriormente, cada valor de m foi determinado empiricamente com base na capacidade de processamento de cada unidade de processamento (τ) utilizada e

Tabela 10 – Conjunto de parâmetros utilizados pelos algoritmos $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$.

Conjuntos Dados	$MR-HDBSCAN^*_{sampling}$	$MR-HDBSCAN^*_{bubbles}$
*	m (%)	m (%)
<i>YearPrediction</i>	1.9	1.9
<i>Credit</i>	10	10
<i>Magic</i>	10	10
<i>Shuttle</i>	10	10
<i>SensorLess</i>	10	10
<i>Skin</i>	4	4

no tamanho de cada conjunto de dados avaliados neste experimento de maneira que o conjunto de amostras selecionado a cada iteração de $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ não exceda a capacidade de processamento τ de cada uma dessas unidades.

Tabela 11 – Conjunto de parâmetros utilizados pelo algoritmo *Random Blocks*.

Conjuntos Dados		Parâmetros										
<i>YearPrediction</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 413	825	1237	1650	2062	3093	4123	5154	6185	7215	
<i>Credit</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 24	48	72	96	120	180	240	300	360	420	
<i>Magic</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 16	31	46	61	77	115	153	191	229	267	
<i>Shuttle</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 47	93	140	186	232	348	464	580	696	812	
<i>SensorLess</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 47	94	141	188	235	352	469	586	703	820	
<i>Skin</i>	$m_{pts} = m_{clSize}$	4	8	12	16	20	30	40	50	60	70	
		s 197	393	589	785	981	1471	1961	2451	2941	3431	

O parâmetro s de *Random Blocks*, que representa o número de subconjuntos de dados gerados a partir de um determinado conjunto completo de dados avaliado por esse algoritmo, foi determinado de acordo com a variação de cada valor de m_{pts} apresentado anteriormente para cada conjunto de dados avaliado no presente experimento (apresentados na Tabela 11).

7.2.7.3 Resultados e Discussão

Para cada conjunto de dados, foram realizadas 25 repetições de $MR-HDBSCAN^*_{bubbles}$, 25 repetições de $MR-HDBSCAN^*_{sampling}$ e uma de *Random Blocks* a partir da combinação de parâmetros desses algoritmos, descritos na Subseção 7.2.7.2. Cada repetição por quaisquer desses algoritmos gera uma *MST* global estendida e, a partir dessa *MST*, é construída uma hierarquia de grupos baseados em densidade pelo método *divisivo*. Como descrito no Capítulo 6, durante

a construção de qualquer hierarquia por método *divisivo*, para cada objeto de um determinado conjunto de dados, é possível guardar informações para o cálculo de *GLOSH*. Essas informações são relacionadas ao nível em que um determinado objeto deixou de pertencer a um grupo válido e se tornou ruído e o nível em que o objeto mais denso de qualquer grupo na hierarquia se tornou ruído.

Uma vez calculado o *score GLOSH* dos objetos para cada repetição de *MR-HDBSCAN^{*}_{bubbles}*, *MR-HDBSCAN^{*}_{sampling}* e *Random Blocks* a partir dos conjuntos de dados, de acordo com as configurações de m_{pts} e m_{clSize} (Subseção 7.2.7.2), foi calculada a medida *ROC AUC*, apresentada na Subseção 7.2.7.1. Dessa maneira, 25 amostras de *ROC AUC* foram obtidas para cada configuração de m_{pts} e m_{clSize} em cada conjunto de dados descrita na Subseção 7.2.7.2, no que se refere aos algoritmos *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}*. Já para *Random Blocks*, foram obtidas apenas uma medida de *ROC AUC* para cada configuração de m_{pts} e m_{clSize} em cada conjunto de dados.

Para avaliar as amostras geradas pelo conjunto de repetições de *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* a partir de cada configuração de m_{pts} e m_{clSize} sobre cada conjunto de dados, foi utilizado o cálculo de Intervalo de Confiança (IC) de médias (OTT; LONGNECKER, 2006). O IC é uma medida utilizada para inserir credibilidade estatística a experimentos, em que não é possível saber a média populacional desses mesmos (JOHNSON; COUTINHO, 2000). Ou seja, o IC é aplicado a um conjunto de amostras de uma determinada população para estimar parâmetros populacionais (OTT; LONGNECKER, 2006).

Nesse contexto, os resultados apresentados nas Figuras reffig:sensor1, 38, 39, 40, 41 e 42 mostram as médias e os cálculos de IC sobre as amostras de *ROC AUC* obtidas pelos algoritmos *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* para cada configuração de m_{pts} e m_{clSize} em todos os conjuntos de dados avaliados. Adicionalmente, o valor absoluto de *ROC AUC* obtido pelo algoritmo *Random Block* para cada configuração de m_{pts} e m_{clSize} também é apresentado nas Figuras 37, 38, 39, 40, 41 e 42 em todos os conjuntos de dados.

Como pode ser visto nas Figuras 37, 38, 39, 40, 41 e 42, o algoritmo *Random Blocks* apresentou melhores resultados em relação aos algoritmos *MR-HDBSCAN^{*}_{bubbles}* e *MR-HDBSCAN^{*}_{sampling}* sobre todos os conjuntos de dados avaliados. Entretanto, *MR-HDBSCAN^{*}_{bubbles}* obteve resultados próximos ao de *Random Blocks* na maioria dos conjuntos de dados, exceto para o conjunto de dados *Credit* para os valores de $m_{pts} = m_{clSize} = \{20, 30\}$. O que pode ter ocorrido nesses casos é a má seleção de amostras aleatórias iniciais para a construção de *Data Bubbles* na maioria das execução de *MR-HDBSCAN^{*}_{bubbles}*, resultando em hierarquias distorcidas.

Já o algoritmo *MR-HDBSCAN^{*}_{sampling}* obteve resultados inferiores em relação aos algoritmos *Random Blocks* e *MR-HDBSCAN^{*}_{sampling}* para todos os conjuntos de dados avaliados neste experimento. Esse fato pode ser justificado pela má seleção de amostras aleatórias em cada etapa de construção de modelos locais de *HDBSCAN^{*}* sobre as mesmas.

Figura 37 – Conjunto de dados - SensorLess

Figura 38 – Conjunto de dados - Credit

Figura 39 – Conjunto de dados - Magic

Figura 40 – Conjunto de dados - Shuttle

No mais, como pode ser visto nas Figuras 37, 39, 40, 41 e 42, $MR-HDBSCAN^*_{sampling}$ obteve maiores variações de intervalos de confiança em relação ao algoritmo $MR-HDBSCAN^*_{bubbles}$ para a maioria das configurações de parâmetros de m_{pts} e m_{clSize} . Esse fato é característico, pois, a cada execução de $MR-HDBSCAN^*_{sampling}$ a partir da mesma configuração de parâmetros e conjunto de dados, diferentes seleções de amostras aleatórias são realizadas, resultando, assim, em diferentes hierarquias.

Figura 41 – Conjunto de dados - Skin

Figura 42 – Conjunto de dados - YearPrediction

7.2.8 Conclusões

Foram analisadas três variações do original algoritmo *HDBSCAN**, i.e., *Random Blocks*, *MR-HDBSCAN*_{sampling}* e *MR-HDBSCAN*_{bubbles}*. A primeira corresponde a uma variação paralela exata de *HDBSCAN**, já as outras duas são versões paralelas estimadas. À vista disso, a proposta do presente experimento foi mostrar o desempenho das três variações em diferentes aspectos: a qualidade de agrupamentos; o desempenho computacional em termos de tempo de execução; e a estimativa de detecção de *outliers*.

Como pode ser visto nas Subseções 7.2.4 e 7.2.7, o algoritmo *Random Blocks* apresentou melhores resultados do que as variações *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}* em termos de qualidade de agrupamento e detecção de *outliers* para todos os conjuntos de dados avaliados, respectivamente.

Entretanto, o desempenho computacional referente ao tempo de execução de *Random Blocks* supera amplamente os tempos dos algoritmos *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*, demorando até semanas para processar um conjunto de dados composto por mais de um milhão de objetos, enquanto os outros algoritmos processam o mesmo conjunto de dados em apenas algumas horas.

Também é possível notar a qualidade de agrupamentos e detecção de *outliers* feitos a partir dos algoritmos aproximados propostos, i.e., *MR-HDBSCAN*_{bubbles}* e *MR-HDBSCAN*_{sampling}*. Para esses algoritmos, em especial, o *MR-HDBSCAN*_{bubbles}*, foi possível observar pouca perda de informações em relação aos agrupamentos e estimativas de *GLOSH* obtidas sobre a maioria dos dados (Subseções 7.2.4 e 7.2.7). Por esse motivo, *MR-HDBSCAN*_{bubbles}* manteve-se próximo aos resultados obtidos pela versão paralela e exata de *HDBSCAN**, i.e., *Random Blocks*. Tal

fato demonstra que a técnica de sumarização *Data Bubbles* aninhada ao algoritmo *MR-HDBSCAN** pode ser uma boa alternativa tanto em termos de velocidade de processamento de grandes quantidades de dados quanto em termos de qualidade de agrupamentos e detecção de *outliers*.

No que se refere ao algoritmo *MR-HDBSCAN**_{sampling}, se comparado ao algoritmo *MR-HDBSCAN**_{bubbles}, foi possível notar uma perda maior de informações relacionadas aos agrupamentos feitos e às estimativas de detecção de *outliers* calculadas sobre todos os conjuntos de dados avaliados no presente experimento. Essa perda de informações pode ser explicada pelo má representação das estruturas dos dados por amostras aleatórias simples a cada construção de modelos locais de *HDBSCAN** por *MR-HDBSCAN**_{sampling}.

Por outro lado, em comparação com *MR-HDBSCAN**_{bubbles} e *Random Blocks*, *MR-HDBSCAN**_{sampling} demonstrou ser mais rápido em termos de tempo execução para todos os conjuntos de dados analisados. Neste contexto, em aplicações em que o tempo despendido for mais importante do que a qualidade dos agrupamentos ou estimativas de detecção de *outliers* realizados, *MR-HDBSCAN**_{sampling} pode ser uma alternativa mais interessante para determinados cenários.

7.2.8.1 Limitações

Durante o desenvolvimento do presente experimento, foi identificada uma limitação relacionada à sensibilidade de parâmetros m_{pts} e s do algoritmo de divisão de dados *Random Blocks*. Em resumo, o parâmetro m_{pts} representa o número mínimo de objetos em uma determinada vizinhança e s o número de divisões de um conjunto de dados em subconjuntos de dados menores não sobrepostos. Nesse contexto, à medida que os valores desses parâmetros aumentam linearmente, o número de replicações dos dados em *data blocks* aumenta exponencialmente, como descrito por Syed (2014). Por tal motivo, o agrupamento em paralelo de conjuntos de dados compostos por alguns milhões de objetos pode se tornar proibitivo devido o tempo de execução despendido por *Random Blocks*.

7.3 Experimento utilizando Memória Compartilhada

O objetivo do presente experimento foi comparar variações de *HDBSCAN** propostas para o contexto de computação paralela em ambiente composto por memória compartilhada, com a versão original centralizada apresentada em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Foram utilizadas como métricas de avaliação à qualidade de agrupamentos, o desempenho computacional e a estimativa de detecção de *outliers* produzida pelos algoritmos testados. Especificamente, as implementações dos algoritmos foram realizadas sobre a linguagem R (R Core Team, 2014).

7.3.1 Algoritmos

Os algoritmos utilizados no presente experimento são o original *HDBSCAN**, proposto em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), e duas de suas variações para o contexto paralelo de agrupamento dos dados investigadas no presente trabalho, quais sejam: *Recursive Sampling* baseado em amostras aleatórias; e em *Data Bubbles*. Para fins de simplicidade, as variações de *HDBSCAN** são denominadas neste experimento como $RS_{sampling}$ e $RS_{bubbles}$, para lidar com amostras aleatórias e *Data Bubbles*, respectivamente.

Assim como descrito no Capítulo 5, a partir de um conjunto de dados qualquer, *Recursive Sampling* produz uma *MST* global estendida baseada em densidade. Logo, a partir de qualquer *MST* global estendida produzida por $RS_{sampling}$ e $RS_{bubbles}$, foi utilizado o conceito de construção de hierarquias de grupos baseados em densidade, como descrito em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), cuja implementação é realizada de maneira centralizada.

7.3.2 Critério de Validação

A validação de agrupamentos de dados realizados pelos algoritmos *HDBSCAN**, $RS_{sampling}$ e $RS_{bubbles}$ foi baseada nas partições extraídas (como descrito na 3.2.2) das hierarquias construídas sobre cada *MST* baseada em densidade obtida por esses algoritmos. Adicionalmente, os conjuntos de dados utilizados no presente experimento são todos compostos por partições de referência em que todos os objetos possuem rótulos. Nesse contexto, o critério *ARI* (descrito na Subseção 7.2.4.1) foi utilizado para validar as partições obtidas nos experimentos por esses algoritmos.

Assim como descrito no Capítulo 6, as partições obtidas por qualquer um dos algoritmos avaliados neste experimento são compostas por grupos mais significativos e objetos ruidosos (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Entretanto, o critério *ARI* é definido para comparar apenas duas partições livres de dados ruidosos. Diante disso, com base na metodologia utilizada por Campello *et al.* (2013a), Campello *et al.* (2015), foi realizada uma adaptação na validação das partições obtidas por *HDBSCAN**, $RS_{bubbles}$ e $RS_{sampling}$ para que os objetos tratados como ruídos no modelo fossem considerados como grupos únicos nas comparações com as partições de referência dos conjuntos de dados analisados.

7.3.3 Configurações de Algoritmos

Os algoritmos *HDBSCAN**, $RS_{bubbles}$ e $RS_{sampling}$ têm m_{pts} e m_{clSize} , em que m_{pts} indica o número mínimo de objetos de uma determinada vizinhança e m_{clSize} o número mínimo de objetos em qualquer grupo válido (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). Desse modo, esses parâmetros foram selecionados empiricamente para cada conjunto de dados avaliado no presente experimento. O parâmetro m , que indica a porcentagem do número de objetos a serem amostrados a cada iteração das técnicas $RS_{bubbles}$ e $RS_{sampling}$, também foi escolhido

empiricamente para cada conjunto de dados, de acordo com o número de objetos de cada conjunto e a capacidade de processamento do *hardware* utilizado neste experimento. Por fim, o parâmetro τ , que é indicador da capacidade de processamento (em número de objetos) de qualquer unidade de processamento, foi determinado empiricamente como $\tau = 30000$ para todos os experimentos. Detalhes de todos os parâmetros dos algoritmos *HDBSCAN**, *RS_{bubbles}* e *RS_{sampling}* são mostrados na Tabela 12:

Tabela 12 – Conjunto de parâmetros utilizados pelos algoritmos *HDBSCAN**, *RS_{bubbles}* e *RS_{sampling}*.

Conjuntos Dados	HDBSCAN*		RS _{sampling}			RS _{bubbles}		
*	m_{pts}	m_{clSize}	m_{pts}	m (%)	m_{clSize}	m_{pts}	m (%)	m_{clSize}
<i>YearPrediction</i>	30	30	30	1.9	30	30	1.9	30
<i>Poker</i>	50	50	50	0.9	50	50	0.9	50
<i>HT Sensor</i>	30	30	30	1	30	30	1	30
<i>Skin</i>	40	40	40	4	40	40	4	40
<i>Gauss1</i>	50	50	50	0.9	50	50	0.9	50
<i>Gauss2</i>	50	50	50	0.3	50	50	0.3	50
<i>Gauss3</i>	50	50	50	0.2	50	50	0.2	50

7.3.4 Configurações de Hardware

Em termos de *hardware*, foi utilizado um computador composto por 32 GB de *RAM*, um processador *AMD FX(tm)-6100 six-core* com 6 núcleos de processamento e 500 *gigabytes* de armazenamento em disco.

7.3.5 Avaliação de Qualidade de Agrupamento de Dados

As avaliações de qualidade de agrupamentos de dados produzidas neste experimento pelos algoritmos *HDBSCAN**, *RS_{sampling}* e *RS_{bubbles}* são baseadas em partições compostas pelos grupos mais significativos dos dados extraídos de hierarquias baseadas em densidade, assim como descrito em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Dito isso, grupos significativos são aqueles que demonstraram maior estabilidade em termos de “sobrevivência” entre os níveis de uma determinada hierarquia baseada em densidade (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015). O método de extração desses grupos significativos, conhecido como *FOSC*, é baseado em um problema de otimização (CAMPELLO *et al.*, 2013b). De modo complementar, na Figura 43 é apresentado o esquema de execução de cada algoritmo avaliado sobre um determinado conjunto de dados.

Como pode ser visto na Figura 43, dado um conjunto de dados, é obtida uma *MST* exata estendida de *HDBSCAN** e *MSTs* globais estendidas pelos algoritmos *RS_{sampling}* e *RS_{bubbles}*. A construção de hierarquias a partir de cada *MST* obtida pelos algoritmos é realizada pelo método proposto em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015), cuja ideia é construir

Figura 43 – Esquema de condução de experimento utilizando os algoritmos *HDBSCAN**, *RS_{bubbles}* e *RS_{sampling}* sobre um determinado conjunto de dados.

hierarquias que maneira centralizada. Assim, uma partição composta pelos melhores grupos é extraída de cada hierarquia construída. Posteriormente, cada partição extraída é comparada com a partição de referência dos dados em questão.

Por definição, tanto as partições resultantes do algoritmo *HDBSCAN** quanto as partições retornadas por *RS_{sampling}* e *RS_{bubbles}* consistem em objetos pertencentes a grupos válidos com adição de objetos considerados como ruído pelos modelos. Entretanto, o critério de validação externo de validação de agrupamentos particionais de dados (*ARI*) consiste apenas em avaliar partições que não contêm dados ruidosos. Nesse contexto, cada objeto considerado ruidoso foi determinado como grupo único em cada partição resultante pelos algoritmos investigados, como feito em (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

7.3.5.1 Resultados e Discussão

Os resultados em termos de qualidade de agrupamento de dados obtidos no presente experimento são apresentados na Tabela 13. Especificamente, foi realizada uma execução de *HDBSCAN** sobre cada conjunto de dados descrito na Tabela 13. Também foram realizadas 45 repetições de *RS_{bubbles}* e 45 de *RS_{sampling}* a partir dos mesmos conjuntos de dados experimentados por *HDBSCAN**.

Como pode ser visto na Tabela 13, o original *HDBSCAN** foi superior em relação aos

resultados médios de qualidade medido por *ARI* dos algoritmos $RS_{bubbles}$ e $RS_{sampling}$ para todos os conjuntos de dados avaliados. Ademais, o algoritmo $RS_{bubbles}$ mostrou resultados próximos aos obtidos por $HDBSCAN^*$ para todos os conjuntos de dados. Tal característica foi possível devido ao uso da técnica de sumarização de dados (*Data Bubbles*) para construir modelos locais de $HDBSCAN^*$ a cada iteração de $RS_{bubbles}$. Em outras palavras, a utilização de *Data Bubbles* por $RS_{bubbles}$ permitiu que a estrutura real de um determinado subconjunto de dados fosse melhor representada por essa técnica.

No que se refere ao algoritmo $RS_{sampling}$, os resultados tiveram maior perda de informação em termos de qualidade de agrupamentos se comparado aos algoritmos $HDBSCAN^*$ e $RS_{bubbles}$ (Tabela 13). Tal fato pode ser explicado pela utilização de amostras aleatórias simples por $RS_{sampling}$ para a construção local de modelos de $HDBSCAN^*$, como descrito no Capítulo 5. Especificamente, em cada modelo local de $HDBSCAN^*$ construído por $RS_{sampling}$ a partir de amostras aleatórias simples, o que representa todos os objetos de um determinado subconjunto de dados é apenas o conjunto de objetos amostrados. A questão é que nem sempre os objetos amostrados são bons representantes da estrutura real dos dados, fazendo com que os modelos locais sejam distorcidos.

Tabela 13 – Avaliação de qualidade de agrupamentos particionais utilizando *ARI*

Conjuntos Dados	$HDBSCAN^*$			$RS_{sampling}$		$RS_{bubbles}$
	<i>ARI</i> absoluto	<i>ARI</i> médio	<i>ARI</i> desvio padrão	<i>ARI</i> médio	<i>ARI</i> desvio padrão	
* <i>YearPrediction</i>	0.413	0.300	0.00584	0.388	0.00413	
<i>Poker</i>	0.321	0.193	0.00731	0.297	0.00109	
<i>HT Sensor</i>	0.361	0.286	0.00168	0.330	0.00079	
<i>Skin</i>	0.442	0.359	0.00523	0.425	0.00222	
<i>Gauss1</i>	0.880	0.689	0.00173	0.864	0.00078	
<i>Gauss2</i>	0.812	0.588	0.00142	0.760	0.00235	
<i>Gauss3</i>	0.819	0.601	0.00676	0.777	0.00264	

Como pode ser visto ainda na Tabela 13, as variações (desvios padrão) em relação aos critérios *ARI* calculados para a maioria dos conjuntos adquiridos por $RS_{bubbles}$ foram menores que as variações obtidas por $RS_{sampling}$, exceto para o conjunto de dados *Gauss2*. Tal fato pode ser explicado pela melhor representação dos dados pela técnica *Data Bubbles* implementada sobre o algoritmo $RS_{bubbles}$. Nesse conjuntura, os *Data Bubbles* criados durante a execução de $RS_{bubbles}$ garantiram que a estrutura real dos dados fosse melhor representada e mantida sem muita perda estrutural dos mesmos.

7.3.5.2 Teste de Significância Estatística

Para a avaliação da confiança dos resultados obtidos, foram realizados testes *t* pareados para cada conjunto de dados avaliado no presente experimento. Nesse contexto, as hipóteses levantadas para esses testes foram definidas como:

H_0 : “Não é possível detectar diferença significativa entre os resultados do $RS_{bubbles}$ e do $RS_{sampling}$ em termos de qualidade de agrupamentos de dados (particional pelo critério ARI).”

H_1 : “Foi detectada diferença significativa entre os resultados dos algoritmos $RS_{bubbles}$ e $RS_{sampling}$ em termos de qualidade de agrupamento de dados (de acordo com o critério ARI).”

Estatisticamente, H_0 é rejeitada quando o valor de $p < 0.05$. Logo, os resultados apresentados na Tabela 14 indicam que para todos os conjuntos de dados avaliados no presente experimento foi encontrada uma diferença significativa entre os resultados de $RS_{bubbles}$ e $RS_{sampling}$. Tal fato pode ser expresso justamente pelos valores de p que são menores que 0.05, e, consequentemente, a hipótese H_1 é dada como verdadeira para todos os conjuntos de dados apresentados na Tabela 14.

Tabela 14 – Testes t pareado em relação aos resultados de agrupamentos particionais feitos pelos algoritmos $RS_{bubbles}$ e $RS_{sampling}$ em ambiente de memória compartilhada.

Conjunto de Dados	valor t	valor p	DM	GL	NC(%)
<i>Gauss1</i>	606.172	3.14280753641522e-88	0.174	44	95
<i>Gauss2</i>	458.227	6.9688412162031e-83	0.171	44	95
<i>Gauss3</i>	171.809	3.77334098644707e-64	0.176	44	95
<i>HT Sensor</i>	152.491	7.1156772027603e-62	0.043	44	95
<i>Poker</i>	91.977	3.03523357167001e-52	0.104	44	95
<i>Skin</i>	79.616	1.67459012754949e-49	0.066	44	95
<i>YearPrediction</i>	83.075	2.60928170642947e-50	0.088	44	95

DM = diferença entre médias dos resultados de $RS_{bubbles}$ e $RS_{sampling}$; GL = Grau de Liberdade; NC = Nível de Confiança.

7.3.6 Avaliação de Desempenho Computacional

Foram realizadas avaliações de desempenho computacional em relação ao tempo de execução despendido pelos três algoritmos avaliados no presente experimento. Especificamente, foram realizadas 45 repetições de $RS_{bubbles}$, 45 repetições de $RS_{sampling}$ e uma execução de $HDBSCAN^*$ para cada conjunto de dados avaliado.

7.3.6.1 Resultados e Discussão

Os resultados obtidos no presente experimento são apresentados na Tabela 15. Especificamente, nessa tabela são apresentados os resultados em tempo de execução em minutos de cada algoritmo. Repare que os menores tempos médios referentes aos conjuntos de dados avaliados são obtidos pelo algoritmo $RS_{sampling}$ (destaque em negrito). Por outro lado, os maiores tempos, i.e., tempos em minutos de uma única execução para cada conjunto de dados, são obtidos pelo

algoritmo *HDBSCAN**. Esse último pode ser justificado pela característica de processamento centralizado de dados realizada pelo original *HDBSCAN**, acarretando em um maior tempo em relação aos outros algoritmos avaliados para agrupar os dados.

Tabela 15 – Desempenho computacional em ambiente paralelo com memória compartilhada.

Conjuntos Dados	HDBSCAN*	RS _{sampling}		RS _{bubbles}	
*	tempo absoluto	média	desvio padrão	média	desvio padrão
YearPrediction	1132.35	795.13	11.89	834.66	12.92
Poker	4463.10	450.22	36.24	450.62	24.53
HT Sensor	3598.97	317.26	3.50	345.82	3.25
Skin	366.55	180.11	18.17	195.61	20.17
Gauss1	4233.34	266.21	20.38	267.46	31.55
Gauss2	17002.02	367.06	16.40	396.05	20.90
Gauss3	34666.70	612.55	8.37	615.66	25.39

7.3.6.2 Teste de Significância Estatística

Para avaliar a confiança dos resultados obtidos em relação aos tempos de execução dos algoritmos *RS_{sampling}* e *RS_{bubbles}*, foram realizados testes *t* pareados unicaudais para inferir respostas estatísticas em relação às amostras de tempo coletadas a partir de cada conjunto de dados avaliado no presente experimento. Além disso, foram utilizadas como hipóteses de teste as seguintes condições:

H₀: “Não foi encontrada diferença significativa entre os *RS_{bubbles}* e *RS_{sampling}* em termos de tempo de execução medido em minutos para a mesma tarefa de agrupamento de dados.”

H₁: “Foi encontrada diferença significativa nos resultados de *RS_{bubbles}* e *RS_{sampling}* em termos de tempo de execução medido em minutos para a mesma tarefa de agrupamento de dados.”

Nesse contexto, são apresentados na Tabela 16 os resultados referentes a testes *t* pareados realizados para as amostras de cada conjunto de dados, amostras essas que representam o tempo de execução despendido pelos algoritmos *RS_{bubbles}* e *RS_{sampling}*. Note que a hipótese nula **H₀** é aceita (valor de $p > 0.05$) para os conjuntos de dados *Poker*, *Gauss1*, *Gauss2* e *Gauss3*. Tal fato indica que os algoritmos *RS_{sampling}* e *RS_{bubbles}* não obtiveram desempenhos computacionais distintos em relação aos conjuntos de amostras de tempos adquiridas por esses algoritmos. Por outro lado, para os conjuntos de dados *YearPrediction*, *Skin* e *HT Sensor* a hipótese nula **H₀** é rejeitada, pois os valores de p obtidos são menores que 0.05. Nessa perspectiva, houve diferença significativa de desempenho entre *RS_{bubbles}* e *RS_{sampling}*.

Tabela 16 – Teste t pareado em relação aos tempos de execução para construção de hierarquias de grupos baseados em densidade $RS_{bubbles}$ e $RS_{sampling}$.

Conjunto de Dados	valor t	valor p	DM	GL	NC(%)
<i>YearPrediction</i>	20.529	1.82066044257738e-24	39.533	44	95
<i>Poker</i>	0.062	0.475200162714662	0.396	44	95
<i>Skin</i>	3.695	0.000302532883984124	15.498	44	95
<i>HT Sensor</i>	44.370	1.79386376763748e-38	28.554	44	95
<i>Gauss1</i>	-7.804	0.99999999961425	-28.987	44	95
<i>Gauss2</i>	-0.218	0.586063327336381	-1.247	44	95
<i>Gauss3</i>	0.800	0.213761554046954	3.110	44	95

DM = diferença entre médias dos resultados de $RS_{bubbles}$ e $RS_{sampling}$; GL = Grau de Liberdade; NC = Nível de Confiança.

7.3.7 Avaliação de Detecção de Outliers usando o método GLOSH

O objetivo do presente experimento foi avaliar a estimativa de detecção de *outliers* de GLOSH calculada durante a construção de hierarquias do original *HDBSCAN** (CAMPELLO *et al.*, 2015) e suas variações, $RS_{bubbles}$ e $RS_{sampling}$. O esquema de execução de cada experimento desta subseção pode ser expresso pela Figura 44.

Figura 44 – Esquema de condução de experimento utilizando os algoritmos *HDBSCAN**, $RS_{bubbles}$ e $RS_{sampling}$ sobre um determinado conjunto de dados para a avaliação de detecção de *outliers* pelo método GLOSH.

7.3.7.1 Configurações

Os conjuntos de dados *YearPrediction*, *Credit*, *Magic*, *Shuttle*, *SensorLess* e *Skin* (Tabela 2) foram utilizados para a condução do presente experimento. Para avaliar os *scores* de *GLOSH* pela *ROC* (*AUC*) sobre cada um desses conjuntos de dados, foi utilizada uma adaptação de abordagens comuns praticadas por (ABE *et al.*, 2006; YANG *et al.*, 2008; KRIEGEL *et al.*, 2008; ZHANG *et al.*, 2009; KELLER *et al.*, 2012; ZIMEK *et al.*, 2013).

Nessas abordagens é considerada que uma pequena porcentagem dos objetos seja determinada como *outliers* e os demais objetos como *inliers* a partir de um determinado conjunto de dados. Ainda com o intuito de desempenhar a tarefa de detecção de *outliers*, os dados de treinamento e teste foram combinados para cada conjunto de dados, como realizado por Campello *et al.* (2015).

Então, para o conjunto de dados *YearPrediction*, todos os objetos contidos nas classes 1922 à 1954 foram considerados como *outliers*, correspondendo a 0.25% do total de objetos desse conjunto de dados. Já para os conjuntos de dados *Credit*, *Magic*, *SensorLess* e *Skin*, foram considerados 5% dos objetos como *outliers* de suas respectivas menores classes, classe 0, *hadron*, classe 11 e classe 1, respectivamente. O restante (95%) dos objetos desses conjuntos foram considerados como *inliers*. Por fim, utilizando uma adaptação de Abe *et al.* (2006), todos os objetos das classes 2,3, 6 e 7, que são as menores classes do conjunto de dados *Shuttle*, foram considerados como *outliers*.

No que se refere aos parâmetros m_{pts} e m_{clSize} dos algoritmos $RS_{bubbles}$ e $RS_{sampling}$ e $HDBSCAN^*$, foi considerado empiricamente um intervalo de 10 valores, tais como: $m_{pts} = m_{clSize} = \{4, 8, 12, 20, 30, 40, 50, 60, 70\}$, para cada conjunto de dados analisado. Já o parâmetro τ , dos algoritmos $RS_{bubbles}$ e $RS_{sampling}$, que representa a capacidade de processamento de cada unidade de processamento, foi considerado em todos experimentos como: $\tau = 30000$.

São apresentados na Tabela 17 os parâmetros individuais dos algoritmos $RS_{bubbles}$ e $RS_{sampling}$ para cada conjunto de dados avaliado neste experimento.

Tabela 17 – Conjunto de parâmetros utilizados pelos algoritmos $RS_{bubbles}$ e $RS_{sampling}$.

Conjuntos Dados	$RS_{sampling}$	$RS_{bubbles}$
*	m (%)	m (%)
<i>YearPrediction</i>	1.9	1.9
<i>Credit</i>	10	10
<i>Magic</i>	10	10
<i>Shuttle</i>	10	10
<i>SensorLess</i>	10	10
<i>Skin</i>	4	4

7.3.7.2 Resultados e Discussão

Foram realizadas 25 repetições de $RS_{bubbles}$, 25 repetições de $RS_{sampling}$ e uma de $HDBSCAN^*$ para cada configuração de parâmetros m_{pts} e m_{clSize} sobre cada conjunto de dados citado na Subseção 7.3.7.1. Assim como descrito no Capítulo 5, o algoritmo *Recursive Sampling* ($RS_{bubbles}$ e $RS_{sampling}$) produz apenas uma *MST* global estendida baseada em densidade. Então, como pode ser visto na Figura 44, a hierarquia e, conseqüentemente, o cálculo de detecção de *outliers* pelo método *GLOSH*, de cada execução de $RS_{bubbles}$ e $RS_{sampling}$ foi construída de maneira centralizada, assim como é feita pelo original $HDBSCAN^*$ (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015).

Foram calculados intervalos de 95% de confiança referentes a cada configuração de m_{pts} e m_{clSize} no intervalo de $m_{pts} = m_{clSize} = \{4, 8, 12, 20, 30, 40, 50, 60, 70\}$ a partir de cada experimento feito pelos algoritmos $RS_{bubbles}$ e $RS_{sampling}$. O intervalo de confiança é utilizado para estimar parâmetros em relação a uma população inteira, a partir de uma amostra dessa população (OTT; LONGNECKER, 2006). Os resultados obtidos são apresentados nas Figuras 45, 46, 47, 48, 49 e 50, nas quais são destacados os intervalos de confiança calculados para cada configuração de m_{pts} e m_{clSize} . Os resultados de $HDBSCAN^*$ também são apresentados nas Figuras 37, 38, 39, 40, 41 e 42, mas por apenas um valor de *ROC AUC* para cada configuração de parâmetros sobre cada conjunto de dados.

Figura 45 – Conjunto de dados - SensorLess

Figura 46 – Conjunto de dados - Credit

Como pode ser visto nas Figuras 45, 46, 47, 48, 49 e 50, o algoritmo $HDBSCAN^*$ obteve melhores resultados sobre todo o conjunto de dados. Entretanto, o algoritmo $RS_{bubbles}$ (baseado em *Data Bubbles*) apresentou resultados próximos aos de $HDBSCAN^*$ para todos os conjuntos de dados avaliados, exceto para algumas configurações de m_{pts} e m_{clSize} , i.e., para o conjunto

Figura 47 – Conjunto de dados - Magic

Figura 48 – Conjunto de dados - Shuttle

Figura 49 – Conjunto de dados - Skin

Figura 50 – Conjunto de dados - YearPrediction

de dados *Credit* (Figura 46) com os parâmetros $m_{pts} = m_{clSize} = 30$ e para o conjunto de dados *Skin* (Figura 49) com $m_{pts} = m_{clSize} = 60$.

Por outro lado, os resultados de $RS_{sampling}$ foram inferiores aos de $RS_{bubbles}$ e $HDBSCAN^*$ para todos os conjuntos de dados avaliados. Especificamente, houve uma variação maior de intervalo de confiança do que os intervalos encontrados por $RS_{bubbles}$ na maioria dos resultados apresentados nas Figuras 45, 46, 47, 48, 49 e 50. Esse fato pode ser explicado devido às hierarquias de baixa qualidade, em certo aspecto, produzidas pelo algoritmo $RS_{sampling}$.

7.3.8 Conclusões

Como pode ser visto na Subseção 7.3.5.1, *HDBSCAN** em sua versão centralizada apresentou melhores resultados em relação a suas variantes, $RS_{sampling}$ e $RS_{bubbles}$, em termos de qualidade de agrupamentos de dados para todos os conjuntos de dados avaliados. Esse resultado era esperado, pois as variantes são aproximações de *HDBSCAN**. Entretanto, mesmo com alguma perda de informação nos resultados dos agrupamentos de dados, as variantes $RS_{sampling}$ e $RS_{bubbles}$ mostraram-se próximas da versão exata de *HDBSCAN**.

A perda de informações nos resultados dos agrupamentos dos dados pelos algoritmos $RS_{sampling}$ e $RS_{bubbles}$ pode ser explicada por dois principais fatores: o primeiro corresponde a aleatoriedade de amostras selecionadas para a construção direta de modelos locais de *HDBSCAN** por $RS_{sampling}$ ou pela construção indireta (via *Data Bubbles*) pelo algoritmo $RS_{bubbles}$. Quando a amostra dos dados não é representativa o suficiente, os modelos criados por esses algoritmos podem ser distorcidos, afetando o resultado final do agrupamento.

Já o segundo fator implica diretamente no algoritmo $RS_{sampling}$, pois o mesmo é baseado na construção direta de modelos locais de *HDBSCAN** a partir das amostras aleatórias simples. O problema nessa abordagem é que as medidas de proximidade utilizadas para mensurar a relação entre essas amostras não são suficientes para representar a real relação entre os objetos que são representados por essas amostras na construção de qualquer modelo local de *HDBSCAN**.

No que refere ao $RS_{bubbles}$, esse fator pode ser minimizado devido à utilização de *Data Bubbles* e medidas apropriadas para mensurar a proximidade entre os objetos do conjunto de dados que são representados por esses *Data Bubbles* na construção de modelos locais de *HDBSCAN**.

No que se refere ao desempenho computacional em tempo de execução em minutos, os algoritmos $RS_{sampling}$ e $RS_{bubbles}$, se comparados à *HDBSCAN**, despenderam menores tempos para realizar a mesma tarefa de agrupamento sobre todos os conjuntos de dados. O que garantiu melhores tempos de execução desses algoritmos foi o fato de eles distribuírem as cargas de processamento dos dados e processá-las de maneira paralela. No mais, $RS_{bubbles}$, mesmo tendo tempos de execução menores que *HDBSCAN**, foi inferior ao algoritmo $RS_{sampling}$. Tal fato é explicado pelo esforço adicional de $RS_{bubbles}$ para construir *Data Bubbles* em quase toda iteração para serem agrupados localmente por *HDBSCAN**.

Para a detecção de *outliers* pelo método *GLOSH*, *HDBSCAN** mostrou melhores resultados para todos os conjuntos de dados analisados se comparados aos algoritmos $RS_{bubbles}$ e $RS_{sampling}$. Entretanto, $RS_{bubbles}$ e $RS_{sampling}$ apresentaram resultados próximos aos de *HDBSCAN**, mesmo com algumas variações de resultados para determinadas configurações de parâmetros desses algoritmos.

Com a realização do presente experimento, foi possível identificar alguns pontos positivos e negativos das técnicas variantes de *HDBSCAN** propostas neste trabalho de mestrado.

Os pontos positivos estão relacionados à rapidez e à escalabilidade das técnicas propostas para o agrupamento e detecção de *outliers* em grandes quantidades de dados. Já os pontos negativos das técnicas $RS_{bubbles}$ e $RS_{sampling}$ estão relacionados à qualidade um pouco inferior dos agrupamentos e detecção de *outliers*, quando comparadas a $HDBSCAN^*$.

Portanto, quando a qualidade dos resultados não é fator crucial para uma determinada aplicação, e sim o tempo para obter esses resultados, as técnicas $RS_{bubbles}$ e $RS_{sampling}$ podem garantir um bom suporte para esse tipo de aplicação.

7.4 Considerações Finais

Os experimentos apresentados neste capítulo são parte fundamental das contribuições deste trabalho de mestrado, pois agregam experimentações inéditas para o contexto de agrupamento de dados em dois cenários distintos: o de técnicas de agrupamento de dados projetadas para o cenário de memória distribuída e o de técnicas de agrupamento de dados baseadas em memória compartilhada de recursos.

Especificamente, $HDBSCAN^*$ proposto por [Campello et al. \(2013a\)](#), [Campello et al. \(2015\)](#) foi o principal objeto de estudo neste trabalho de mestrado. Diante disso, foram propostas, neste trabalho, variantes paralelas e distribuídas (*MapReduce*) de $HDBSCAN^*$ com o objetivo de aumentar a escalabilidade do processo de agrupamento de grandes quantidades de dados.

As variantes de $HDBSCAN^*$ propostas neste trabalho para o cenário de memória distribuída (*MapReduce*), chamadas de $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ (Capítulo 6), foram comparadas no primeiro experimento (Seção 7.2) com uma abordagem paralela e exata de $HDBSCAN^*$ proposta por [Syed \(2014\)](#), chamada *Random Blocks*. Os resultados dos experimentos realizados com essas técnicas mostraram que $MR-HDBSCAN^*_{bubbles}$ e $MR-HDBSCAN^*_{sampling}$ obtiveram resultados próximos aos obtidos por *Random Blocks* em termos de qualidade de agrupamento e detecção de *outliers*, além de mostrar maior rapidez e escalabilidade no processamento de grandes quantidades de dados se comparadas a *Random Blocks*.

Já no contexto de memória compartilhada, as variantes de $HDBSCAN^*$, chamadas $RS_{sampling}$ e $RS_{bubbles}$ e também propostas neste trabalho de mestrado, foram comparadas com a versão original de $HDBSCAN^*$ ([CAMPELLO et al., 2013a](#); [CAMPELLO et al., 2015](#)) em um segundo experimento apresentado na Seção 7.3. Os resultados da comparação mostraram que $RS_{sampling}$ e $RS_{bubbles}$, mesmo sendo aproximações de $HDBSCAN^*$, apresentaram resultados próximos à versão exata do método em termos de qualidade de agrupamento e detecção de *outliers*. Adicionalmente, foi possível observar a escalabilidade e rapidez das técnicas $RS_{sampling}$ e $RS_{bubbles}$ para processar de maneira paralela grandes conjuntos de dados.

CONCLUSÕES E TRABALHOS FUTUROS

No presente trabalho de mestrado foram propostas novas arquiteturas de paralelização de *HDBSCAN** capazes de processar grandes conjuntos de dados. As arquiteturas apresentadas são direcionadas para dois contextos distintos: o de memória compartilhada e o de memória distribuída. Especificamente, a metodologia de paralelização de *HDBSCAN** com base no cenário de memória compartilhada foi implementada utilizando a linguagem R. Já a arquitetura baseada no cenário de memória distribuída foi desenvolvida sobre o conceito de programação *MapReduce*.

A partir das arquiteturas propostas neste trabalho, tanto para o cenário de memória compartilhada como distribuída, foram realizados diversos experimentos sobre conjuntos de dados artificiais e reais. Diante disso, foi possível analisar o impacto na utilização dessas arquiteturas em relação às versões exatas de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015; SYED, 2014). Esse impacto corresponde tanto à qualidade de agrupamentos e detecção de *outliers*, quanto o desempenho computacional dos métodos estudados.

Especificamente, os resultados apresentados no Capítulo 7, mostram que as versões propostas neste trabalho de mestrado, são escaláveis e mais rápidas para o processamento de grandes quantidades de dados, e ainda garantem uma qualidade de agrupamento e detecção *outliers* próxima às versões em seu estado da arte (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015; SYED, 2014). Esses resultados também mostram indícios de que as versões exatas de *HDBSCAN** (CAMPELLO *et al.*, 2013a; CAMPELLO *et al.*, 2015; SYED, 2014) podem ser extremamente lentas a medida que os conjunto de dados aumentam em número de objetos, tornando-as impraticáveis em cenários reais de aplicação. Fato este que pode ser amenizado com as versões propostas neste trabalho.

No decorrer da presente pesquisa de mestrado, foi possível identificar dois possíveis trabalhos futuros para fins de aprimoramento dos métodos propostos, tanto para o cenário distribuído quanto para o cenário paralelo. O primeiro trabalho corresponde ao uso de técnicas de

sumarização de dados que permitam a utilização de diversas medidas de proximidade, i.e. *Data Bubbles* aplicados em dados não vetoriais (ZHOU; SANDER, 2003). Dessa forma, é possível ampliar a aplicação dos métodos propostos neste trabalho para o agrupamento de dados textuais, por exemplo. Já o segundo trabalho futuro está relacionado à adaptação das arquiteturas propostas para o cenário de análise semi-supervisionada de dados, assim como é feito pelo estado da arte apresentado em (CAMPELLO *et al.*, 2015).

REFERÊNCIAS

- ABE, N.; ZADROZNY, B.; LANGFORD, J. Outlier detection by active learning. In: **Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2006. (KDD '06), p. 504–509. ISBN 1-59593-339-5. Disponível em: <<http://doi.acm.org/10.1145/1150402.1150459>>. Citado nas páginas 141, 155 e 168.
- AGGARWAL, C. C. Outlier analysis. In: _____. **Data Mining: The Textbook**. Cham: Springer International Publishing, 2015. p. 237–263. ISBN 978-3-319-14142-8. Disponível em: <https://doi.org/10.1007/978-3-319-14142-8_8>. Citado nas páginas 36 e 69.
- AGGARWAL, C. C.; REDDY, C. K. **Data Clustering: Algorithms and Applications**. 1st. ed. Chapman & Hall/CRC, 2013. ISBN 1466558210, 9781466558212. Disponível em: <<https://dl.acm.org/citation.cfm?id=2535015>>. Citado nas páginas 37, 39, 40, 42, 43, 44 e 51.
- AGGARWAL, C. C.; ZHAI, C. X. **Mining Text Data**. Springer Publishing Company, Incorporated, 2012. ISBN 1461432227, 9781461432227. Disponível em: <<https://dl.acm.org/citation.cfm?id=2331533>>. Citado na página 35.
- ANANDAN, A. **Efficient Algorithms for Hierarchical Agglomerative Clustering**. Dissertação (Mestrado) — University of Alberta, Canada, 2015. Citado na página 73.
- ANDERBERG, M. R. **Cluster Analysis for Applications**. 1st. ed. Academic Press, 1973. Disponível em: <<https://www.elsevier.com/books/cluster-analysis-for-applications/anderberg/978-0-12-057650-0>>. Citado na página 37.
- ANDERSON, I. Graphs, an introductory approach, by wilson r. j. and watkins j. j. . pp 340. £11.95. 1990. isbn 0-471-51340-7 (john wiley). **The Mathematical Gazette**, Cambridge University Press, v. 75, n. 471, p. 119–120, 1991. Citado na página 75.
- ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: Ordering points to identify the clustering structure. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 28, n. 2, p. 49–60, jun. 1999. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/304181.304187>>. Citado nas páginas 13, 47, 49, 50, 51, 56, 57, 74, 75, 87, 88, 90 e 93.
- ANKERST, M.; KASTENMÜLLER, G.; KRIEGEL, H.-P.; SEIDL, T. 3d shape histograms for similarity search and classification in spatial databases. In: **Proceedings of the 6th International Symposium on Advances in Spatial Databases**. London, UK, UK: Springer-Verlag, 1999. (SSD '99), p. 207–226. ISBN 3-540-66247-2. Disponível em: <<http://dl.acm.org/citation.cfm?id=647226.719092>>. Citado na página 49.
- ANTONIE, M.-L.; ZAIANE, O. R.; COMAN, A. Application of data mining techniques for medical image classification. In: **Proceedings of the Second International Conference on Multimedia Data Mining**. London, UK, UK: Springer-Verlag, 2001. (MDMKDD'01), p. 94–101. Disponível em: <<http://dl.acm.org/citation.cfm?id=3012377.3012388>>. Citado na página 35.

ARITHMETIC. In: HANDBOOK of Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–46. ISBN 978-3-540-72122-2. Disponível em: <https://doi.org/10.1007/978-3-540-72122-2_1>. Citado na página 131.

ARNOB, R. I. Advanced Agglomerative Clustering Technique for Phylogenetic Classification Using Manhattan Distance. BIOCAMP'17, p. 9–13, 2016. Citado na página 29.

BACHE, K.; LICHMAN, M. **UCI Machine Learning Repository**. 2013. Disponível em: <<http://archive.ics.uci.edu/ml>>. Citado na página 139.

BEZDEK, J. C. **Pattern Recognition with Fuzzy Objective Function Algorithms**. Norwell, MA, USA: Kluwer Academic Publishers, 1981. ISBN 0306406713. Citado na página 38.

BOCK, R.; CHILINGARIAN, A.; GAUG, M.; HAKL, F.; HENGSTEBECK, T.; JIŘINA, M.; KLASCHKA, J.; KOTRČ, E.; SAVICKÝ, P.; TOWERS, S.; VAICIULIS, A.; WITTEK, W. Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope. **Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment**, v. 516, n. 2, p. 511 – 528, 2004. ISSN 0168-9002. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0168900203025051>>. Citado na página 141.

BREUNIG, M. M.; KRIEGEL, H.-P.; KRÖGER, P.; SANDER, J. Data bubbles: Quality preserving performance boosting for hierarchical clustering. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 30, n. 2, p. 79–90, maio 2001. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/376284.375672>>. Citado nas páginas 31, 32, 33, 38, 87, 88, 89, 90, 91 e 92.

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. Lof: Identifying density-based local outliers. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 29, n. 2, p. 93–104, maio 2000. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/335191.335388>>. Citado na página 69.

CAMPELLO, R. J. G. B.; MOULAVI, D.; SANDER, J. Density-based clustering based on hierarchical density estimates. In: PEI, J.; TSENG, V. S.; CAO, L.; MOTODA, H.; XU, G. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 160–172. ISBN 978-3-642-37456-2. Citado nas páginas 30, 31, 39, 43, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 77, 79, 80, 82, 83, 84, 93, 100, 101, 105, 106, 113, 116, 117, 118, 124, 125, 132, 133, 134, 143, 147, 148, 149, 152, 153, 160, 161, 162, 163, 169, 172 e 173.

CAMPELLO, R. J. G. B.; MOULAVI, D.; ZIMEK, A.; SANDER, J. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. **DataMinKnowl-Disc**, v. 27, n. 3, p. 344–371, 2013. Citado nas páginas 29, 39 e 162.

CAMPELLO, R. J. G. B.; MOULAVI, D.; ZIMEK, A.; SANDER, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. **ACM Trans. Knowl. Discov. Data**, ACM, New York, NY, USA, v. 10, n. 1, p. 5:1–5:51, jul 2015. ISSN 1556-4681. Disponível em: <<http://doi.acm.org/10.1145/2733381>>. Citado nas páginas 13, 30, 31, 32, 39, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 77, 79, 80, 82, 83, 84, 90, 100, 101, 105, 106, 113, 116, 117, 118, 125, 132, 133, 134, 142, 143, 147, 148, 149, 152, 153, 155, 160, 161, 162, 163, 167, 168, 169, 172, 173 e 174.

CASTRO, L. N. d.; FERRARI, D. G. **Introdução à Mineração de Dados: conceitos básicos, algoritmos e aplicações**. São Paulo: Saraiva, 2016. ISBN 8547200983, 9788547200985. Disponível em: <<https://www.saraiva.com.br/introducao-a-mineracao-de-dados-9319913.html>>. Citado nas páginas 13, 36 e 37.

CHAKRABORTY, S.; CHAKI, N. A survey on the semi-structured data models. In: _____. **Computer Information Systems – Analysis and Technologies: 10th International Conference, CISIM 2011, Kolkata, India, December 14-16, 2011. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 257–266. ISBN 978-3-642-27245-5. Disponível em: <https://doi.org/10.1007/978-3-642-27245-5_31>. Citado na página 35.

CHANDRA, R.; DAGUM, L.; KOHR, D.; MAYDAN, D.; MCDONALD, J.; MENON, R. **Parallel Programming in OpenMP**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1-55860-671-8, 9781558606715. Citado na página 102.

CHONG, K. W.; HAN, Y.; IGARASHI, Y.; LAM, T. W. Improving the efficiency of parallel minimum spanning tree algorithms. **Discrete Applied Mathematics**, v. 126, n. 1, p. 33 – 54, 2003. ISSN 0166-218X. 5th Annual International Computing and combinatorics Conference. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0166218X02005607>>. Citado na página 75.

COHEN, J. Graph twiddling in a mapreduce world. **Computing in Science and Engg.**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 11, n. 4, p. 29–41, jul. 2009. ISSN 1521-9615. Disponível em: <<http://dx.doi.org/10.1109/MCSE.2009.120>>. Citado na página 119.

CORMEN, T. H.; STEIN, C.; RIVEST, R. L.; LEISERSON, C. E. **Introduction to Algorithms**. 2nd. ed. McGraw-Hill Higher Education, 2001. ISBN 0070131511. Disponível em: <<https://dl.acm.org/citation.cfm?id=580470>>. Citado nas páginas 102, 129, 130 e 138.

DAI, Y.-s. **Advanced Parallel And Distributed Computing: Evaluation, Improvement And Practice**. Nova Science Publication, Incorporation, 2006. ISBN 1600212026. Disponível em: <<https://dl.acm.org/citation.cfm?id=1213192>>. Citado na página 31.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. **Commun. ACM**, ACM, New York, NY, USA, v. 51, n. 1, p. 107–113, jan. 2008. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/1327452.1327492>>. Citado nas páginas 31, 101, 102, 105 e 108.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the EM algorithm. **JRStatistSocB**, v. 39, n. 1, p. 1–31, 1977. Citado na página 39.

DENOEUDE, L.; GARRETA, H.; GUENOCHÉ, A. Comparison of distance indices between partitions. In **Proceedings 11th Conference of the Applied Stochastic Models and Data Analysis (ASMDA), Brest, France, 2005**. Citado na página 52.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification (2Nd Edition)**. Wiley-Interscience, 2000. ISBN 0471056693. Disponível em: <<https://dl.acm.org/citation.cfm?id=954544>>. Citado na página 40.

EL-REWINI, H.; ABD-EL-BARR, M. Shared memory architecture. In: _____. **Advanced Computer Architecture and Parallel Processing**. John Wiley and Sons, Inc., 2005. p. 77–102.

ISBN 9780471478386. Disponível em: <<http://dx.doi.org/10.1002/0471478385.ch4>>. Citado na página 98.

ERL, T.; PUTTINI, R.; MAHMOOD, Z. **Cloud Computing: Concepts, Technology & Architecture**. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2013. ISBN 0133387526, 9780133387520. Citado na página 104.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. AAAI Press, 1996. (KDD'96), p. 226–231. Disponível em: <<http://dl.acm.org/citation.cfm?id=3001460.3001507>>. Citado nas páginas 30, 39, 47, 48, 49, 57, 62 e 63.

EVERITT, B. S.; LANDAU, S.; LEESE, M. **Cluster Analysis**. 4th. ed. Wiley Publishing, 2009. ISBN 0340761199, 9780340761199. Disponível em: <<https://dl.acm.org/citation.cfm?id=1538772>>. Citado nas páginas 29, 38 e 39.

FIORIO, C.; GUSTEDT, J.; LANGE, T. Union-Find Volume Segmentation. In: MALGOUYRES, R. (Ed.). **7th International Workshop on Combinatorial Image Analysis - IWCIA'2000**. Caen, France: [s.n.], 2000. p. 181–197. Colloque avec actes et comité de lecture. internationale. Disponível em: <<https://hal.inria.fr/inria-00099034>>. Citado na página 129.

FLOREK, K.; LUKASZEWICZ, J.; PERKAL, J.; STEINHAUS, H.; ZUBRZYCKI, S. **Sur la Liaison et la Division des Points d'un Ensemble Fini**. 1951. 282–285 p. Citado nas páginas 29, 30 e 43.

GALIL, Z.; ITALIANO, G. F. Data structures and algorithms for disjoint set union problems. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 23, n. 3, p. 319–344, set. 1991. ISSN 0360-0300. Disponível em: <<http://doi.acm.org/10.1145/116873.116878>>. Citado na página 129.

GODSIL, C.; ROYLE, G. F. **Algebraic graph theory**. Springer Science & Business Media, 2013. v. 207. ISBN 0387952209, 978-0-387-95220-8. Disponível em: <<https://www.amazon.co.uk/Algebraic-Graph-Theory-Graduate-Mathematics/dp/0387952209>>. Citado nas páginas 86 e 119.

GRAHAM, P. **On LISP: Advanced Techniques for Common LISP**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. ISBN 0130305529. Citado na página 102.

GROSS, J. L.; YELLEN, J. **Graph Theory and Its Applications (Discrete Mathematics and Its Applications)**. 2nd. ed. Chapman & Hall/CRC, 2005. ISBN 158488505X. Disponível em: <<https://dl.acm.org/citation.cfm?id=1212553>>. Citado na página 59.

GUHA, S.; RASTOGI, R.; SHIM, K. Cure: An efficient clustering algorithm for large databases. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 27, n. 2, p. 73–84, jun. 1998. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/276305.276312>>. Citado na página 45.

GUPTA, G.; LIU, A.; GHOSH, J. Automated hierarchical density shaving: A robust automated clustering and visualization framework for large biological data sets. **IEEE TCBB**, v. 7, n. 2, p. 223–237, 2010. Citado na página 56.

HAN, J. **Data Mining: Concepts and Techniques**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 1558609016. Citado nas páginas 37, 38, 41, 42 e 51.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. ISBN 1-55860-489-8. Citado nas páginas 41, 42 e 51.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791. Citado nas páginas 35, 38, 41, 42, 43 e 51.

HAND, D. J. Data clustering: Theory, algorithms, and applications by guojun gan, chaoqun ma, jianhong wu. **International Statistical Review**, Blackwell Publishing Ltd, v. 76, n. 1, p. 141–141, 2008. ISSN 1751-5823. Citado nas páginas 29, 35, 36, 38, 42 e 90.

HARTIGAN, J. A. **Clustering Algorithms**. New York, London, Sydney, Toronto: WILEY, 1975. Citado nas páginas 38, 40, 55, 57, 61 e 65.

HENDRIX, W.; Ali Patwary, M.; AGRAWAL, A.; LIAO, W.; CHOUDHARY, A. Parallel hierarchical clustering on shared memory platforms. In: _____. **2012 19th International Conference on High Performance Computing, HiPC 2012**. [s.n.], 2012. ISBN 9781467323703. Disponível em: <<http://ieeexplore.ieee.org/document/6507511/>>. Citado na página 31.

HOLMES, A. **Hadoop in Practice**. Greenwich, CT, USA: Manning Publications Co., 2012. ISBN 1617290238, 9781617290237. Citado nas páginas 103 e 105.

HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of Classification**, Springer-Verlag, v. 2, n. 1, p. 193–218, 1985. ISSN 0176-4268. Citado nas páginas 53 e 146.

JACCARD, P. Distribution de la florine alpine dans la Bassin de Dranses et dans quelques regions voisines. **Bulletin de la Societe Vaudoise des Sciences Naturelles**, v. 37, p. 241–272, 1901. Citado nas páginas 52 e 146.

JAIN, A. K. Data clustering: 50 years beyond k-means. v. 31, p. 651–666, 2010. Citado na página 29.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. Englewood Cliffs: Prentice Hall, 1988. Citado nas páginas 29, 30, 31, 38, 39, 40, 42, 43, 44, 45, 46, 49, 51, 52, 56, 58, 59, 60, 63, 64 e 73.

JIN, C.; AGRAWAL, A.; HENDRIX, W.; LIAO, W.-k.; CHOUDHARY, A. DiSC: A Distributed Single-Linkage Hierarchical Clustering Algorithm using MapReduce. 2013. Citado nas páginas 13, 73 e 74.

JIN, C.; PATWARY, M. M. A.; AGRAWAL, A.; HENDRIX, W.; LIAO, W.-k.; CHOUDHARY, A. Disc: A distributed single-linkage hierarchical clustering algorithm using mapreduce. in **Proceedings of the 4th International SC Workshop on Data Intensive Computing in the Clouds**, 2013. Citado na página 31.

JOHNSON, S. Hierarchical clustering schemes. **Psychometrika**, Springer-Verlag, v. 32, n. 3, p. 241–254, 1967. ISSN 0033-3123. Citado na página 43.

JOHNSON, T. de Melo e S. M.; COUTINHO, M. **Avaliação de Desempenho de Sistemas Computacionais**. Grupo Gen - LTC, 2000. ISBN 9788521618973. Disponível em: <<https://books.google.com.br/books?id=upwunQAACAAJ>>. Citado na página 157.

JOLLIFFE, I. T. **Principal Component Analysis**. 2nd. ed. Springer, 2002. ISBN 978-0-387-95442-4, 978-1-4419-2999-0. Disponível em: <<http://www.springer.com/us/book/9780387954424>>. Citado na página 38.

JR., J. H. W. Hierarchical grouping to optimize an objective function. **Journal of the American Statistical Association**, v. 58, n. 301, p. 236–244, 1963. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845>>. Citado nas páginas 29, 30, 43 e 44.

KANG, U.; TSOURAKAKIS, C. E.; FALOUTSOS, C. Pegasus: A peta-scale graph mining system implementation and observations. In: **Proceedings of the 2009 Ninth IEEE International Conference on Data Mining**. Washington, DC, USA: IEEE Computer Society, 2009. (ICDM '09), p. 229–238. ISBN 978-0-7695-3895-2. Disponível em: <<http://dx.doi.org/10.1109/ICDM.2009.14>>. Citado na página 119.

KARAU, H.; KONWINSKI, A.; WENDELL, P.; ZAHARIA, M. **Learning Spark: Lightning-Fast Big Data Analytics**. 1st. ed. O'Reilly Media, Inc., 2015. ISBN 1449358624, 9781449358624. Disponível em: <<https://dl.acm.org/citation.cfm?id=2717070>>. Citado nas páginas 102, 105 e 112.

KAUFMAN, L.; ROUSSEEUW, P. J. Clustering by means of medoids. p. 405–416, 1987. Citado na página 39.

KAUFMAN, L.; ROUSSEEUW, P. J. **Finding groups in data: an introduction to cluster analysis**. [S.l.]: John Wiley & Sons, 2009. v. 344. Citado nas páginas 35, 39 e 46.

KELLER, F.; MULLER, E.; BOHM, K. Hics: High contrast subspaces for density-based outlier ranking. In: **Proceedings of the 2012 IEEE 28th International Conference on Data Engineering**. Washington, DC, USA: IEEE Computer Society, 2012. (ICDE '12), p. 1037–1048. ISBN 978-0-7695-4747-3. Disponível em: <<http://dx.doi.org/10.1109/ICDE.2012.88>>. Citado nas páginas 155 e 168.

KIM, W. Parallel clustering algorithms: survey. **Parallel Algorithms, Spring**, 2009. Citado na página 98.

KING, R. S. **Cluster Analysis and Data Mining: An Introduction**. USA: Mercury Learning & Information, 2014. ISBN 1938549384, 9781938549380. Citado nas páginas 29, 36, 38 e 39.

KRIEGEL, H.-P.; KRÖGER, P.; SANDER, J.; ZIMEK, A. Density-based clustering. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 1, n. 3, p. 231–240, 2011. Citado nas páginas 30, 39, 47 e 49.

KRIEGEL, H.-P.; RENZ, M.; SCHUBERT, M.; ZÜFLE, A. Statistical density prediction in traffic networks. In: **SDM08**. [S.l.: s.n.], 2008. p. 692–703. Citado nas páginas 155 e 168.

KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. **Proceedings of the American Mathematical Society**, American Mathematical Society, v. 7, n. 1, p. 48–50, 1956. ISSN 00029939, 10886826. Disponível em: <<http://www.jstor.org/stable/2033241>>. Citado na página 129.

KRZANOWSKI, W. J.; HAND, D. J. **ROC Curves for Continuous Data**. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2009. ISBN 1439800219, 9781439800218. Citado na página 153.

LAM, C. **Hadoop in Action**. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2010. ISBN 1935182196, 9781935182191. Citado na página 102.

LANE, S. M. Functions, transformations, and groups. In: _____. **Mathematics Form and Function**. New York, NY: Springer New York, 1986. p. 123–149. ISBN 978-1-4612-4872-9. Disponível em: <http://dx.doi.org/10.1007/978-1-4612-4872-9_6>. Citado na página 102.

LAROSE, D. T. **Discovering Knowledge in Data: An Introduction to Data Mining**. 2nd. ed. Wiley Publishing, 2014. ISBN 0470908742, 9780470908747. Disponível em: <<https://dl.acm.org/citation.cfm?id=2683925>>. Citado nas páginas 39 e 42.

LELIS, L.; SANDER, J. Semi-supervised density-based clustering. In: IEEE. **Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on**. [S.l.], 2009. p. 842–847. Citado na página 58.

LEOPOLD, C. **Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches**. New York, NY, USA: John Wiley and Sons, Inc., 2001. ISBN 0471358312. Citado na página 31.

LI, R.; HU, H.; LI, H.; WU, Y.; YANG, J. Mapreduce parallel programming model: A state-of-the-art survey. **International Journal of Parallel Programming**, v. 44, n. 4, p. 832–866, 2016. ISSN 1573-7640. Disponível em: <<http://dx.doi.org/10.1007/s10766-015-0395-0>>. Citado na página 103.

LIN, J.; DYER, C. **Data-Intensive Text Processing with MapReduce**. [S.l.]: Morgan and Claypool Publishers, 2010. ISBN 1608453421, 9781608453429. Citado na página 102.

MACNAUGHTON-SMITH, P.; WILLIAMS, W.; DALE, M.; MOCKETT, L. Dissimilarity analysis: a new technique of hierarchical sub-division. Nature Publishing Group, 1964. Citado na página 46.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: **5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics**. [S.l.: s.n.], 1967. v. 1, p. 281–297. Citado nas páginas 29, 39, 45 e 88.

MANNE, F.; PATWARY, M. M. A. A scalable parallel union-find algorithm for distributed memory computers. In: **Proceedings of the 8th International Conference on Parallel Processing and Applied Mathematics: Part I**. Berlin, Heidelberg: Springer-Verlag, 2010. (PPAM'09), p. 186–195. ISBN 3-642-14389-X, 978-3-642-14389-2. Disponível em: <<http://dl.acm.org/citation.cfm?id=1882792.1882815>>. Citado nas páginas 129 e 130.

MELNYKOV, V.; CHEN, W.-C.; MAITRA, R. MixSim: An R package for simulating data to study performance of clustering algorithms. **Journal of Statistical Software**, v. 51, n. 12, p. 1–25, 2012. Disponível em: <<http://www.jstatsoft.org/v51/i12/>>. Citado na página 139.

MELOUN, M.; MILITKY, J. **Statistical Data Analysis: A Practical Guide**. Woodhead Publishing, Limited, 2011. ISBN 0857091093, 9780857091093. Disponível em: <<https://dl.acm.org/citation.cfm?id=2031572>>. Citado na página 35.

MICHAELSON, G. **An Introduction to Functional Programming Through Lambda Calculus**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201178125. Citado na página 102.

- MINER, D.; SHOOK, A. **MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems**. 1st. ed. O'Reilly Media, Inc., 2012. ISBN 1449327176, 9781449327170. Disponível em: <<https://dl.acm.org/citation.cfm?id=2632825>>. Citado na página 102.
- MOHAMMAD, S. M.; KIRITCHENKO, S.; ZHU, X. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. **CoRR**, abs/1308.6242, 2013. Disponível em: <<http://arxiv.org/abs/1308.6242>>. Citado na página 35.
- MUKAKA, M. M. Statistics Corner: A guide to appropriate use of Correlation coefficient in medical research. v. 24, n. September, p. 69–71, 2012. Citado na página 40.
- MULLER, D. W.; SAWITZKI, G. Excess mass estimates and tests for multimodality. **JASA**, v. 86, n. 415, p. 738–746, 1991. Citado na página 65.
- NOBARI, S.; CAO, T.-T.; KARRAS, P.; BRESSAN, S. Scalable parallel minimum spanning forest computation. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 47, n. 8, p. 205–214, fev. 2012. ISSN 0362-1340. Disponível em: <<http://doi.acm.org/10.1145/2370036.2145842>>. Citado na página 75.
- ODERSKY, M.; SPOON, L.; VENNERS, B. **Programming in Scala: A Comprehensive Step-by-Step Guide, 2Nd Edition**. 2nd. ed. USA: Artima Incorporation, 2011. ISBN 0981531644, 9780981531649. Citado na página 102.
- OLMAN, V.; MAO, F.; WU, H.; XU, Y. Parallel clustering algorithm for large data sets with applications in bioinformatics. **IEEE/ACM Trans. Comput. Biol. Bioinformatics**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 6, n. 2, p. 344–352, abr. 2009. ISSN 1545-5963. Disponível em: <<http://dx.doi.org/10.1109/TCBB.2007.70272>>. Citado nas páginas 73 e 75.
- OTT, R. L.; LONGNECKER, M. T. **Introduction to Statistical Methods and Data Analysis**. [S.l.]: Duxbury Press, 2006. ISBN 0495017582. Citado nas páginas 157 e 169.
- PACHECO, P. S. **Parallel Programming with MPI**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. ISBN 1-55860-339-5. Citado na página 102.
- PANIK, M. J. Testing statistical hypotheses. In: _____. **Statistical Inference**. John Wiley and Sons, Inc., 2012. p. 184–216. ISBN 9781118309773. Disponível em: <<http://dx.doi.org/10.1002/9781118309773.ch10>>. Citado na página 149.
- PAPADIMITRIOU, S.; KITAGAWA, H.; GIBBONS, P.; FALOUTSOS, C. LOCI: Fast outlier detection using the local correlation integral. In: **ICDE03**. [S.l.: s.n.], 2003. p. 315–326. Citado na página 70.
- PATWARY, M. A.; PALSETIA, D.; AGRAWAL, A.; LIAO, W.-k.; MANNE, F.; CHOUDHARY, A. Scalable parallel optics data clustering using graph algorithmic techniques. In: **Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis**. New York, NY, USA: ACM, 2013. (SC '13), p. 49:1–49:12. ISBN 978-1-4503-2378-9. Disponível em: <<http://doi.acm.org/10.1145/2503210.2503255>>. Citado nas páginas 13, 73, 74 e 75.
- PEDRYCZ, W. **Knowledge-Based Clustering: From Data to Information Granules**. Wiley-Interscience, 2005. ISBN 0471469661. Disponível em: <<https://dl.acm.org/citation.cfm?id=1044924>>. Citado na página 38.

- PHILLIPS, P.; WECHSLER, H.; HUANG, J.; RAUSS, P. J. The feret database and evaluation procedure for face-recognition algorithms. **Image and Vision Computing**, v. 16, n. 5, p. 295 – 306, 1998. ISSN 0262-8856. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S026288569700070X>>. Citado na página 141.
- PRIM, R. C. Shortest connection networks and some generalizations. **The Bell System Technical Journal**, v. 36, n. 6, p. 1389–1401, Nov 1957. ISSN 0005-8580. Citado nas páginas 63, 74, 77 e 83.
- R Core Team. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2014. Disponível em: <<http://www.R-project.org/>>. Citado na página 160.
- RAMASWAMY, S.; RASTOGI, R.; SHIM, K. Efficient algorithms for mining outliers from large data sets. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 29, n. 2, p. 427–438, may 2000. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/335191.335437>>. Citado na página 70.
- RAND, W. M. Objective criteria for the evaluation of clustering methods. **JASA**, v. 66, n. 336, p. 846–850, 1971. Citado nas páginas 52 e 146.
- SEDGEWICK, R.; WAYNE, K. **Algorithms**. 4th. ed. [S.l.]: Addison-Wesley Professional, 2011. 992 p. ISBN 032157351X, 9780321573513. Citado na página 129.
- SEIDL, T.; BODEN, B.; FRIES, S. Cc-mr – finding connected components in huge graphs with mapreduce. In: FLACH, P. A.; BIE, T. D.; CRISTIANINI, N. (Ed.). **Machine Learning and Knowledge Discovery in Databases**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 458–473. ISBN 978-3-642-33460-3. Citado nas páginas 119, 120 e 137.
- SHAW, S.; VERMEULEN, A. F.; GUPTA, A.; KJERRUMGAARD, D. Querying semi-structured data. In: **Practical Hive**. [S.l.]: Springer, 2016. p. 115–131. ISBN 978-1-4842-0272-2, 978-1-4842-0271-5. Citado na página 35.
- SIMA, D.; KACSUK, P. **Advanced Computer Architectures**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN 0201422913. Citado na página 31.
- SNEATH, P. H. A. The application of computers to taxonomy. **JGenMicrobiol**, v. 17, p. 201–226, 1957. Citado nas páginas 30 e 43.
- SYED, T. I. **Parallelization of Hierarchical Density-Based Clustering using MapReduce**. 114 p. Dissertação (Mestrado) — University of Alberta, 2014. Citado nas páginas 13, 31, 32, 33, 73, 76, 77, 79, 80, 84, 85, 86, 87, 93, 98, 101, 105, 142, 147, 160, 172 e 173.
- TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. 1. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367. Citado nas páginas 35, 36, 37, 38, 41, 73, 87, 88, 140 e 141.
- VALDURIEZ, P. Shared-memory architecture. In: _____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 2638–2638. ISBN 978-0-387-39940-9. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_1082>. Citado na página 98.
- VENDRAMIN, L.; CAMPELLO, R. J. G. B.; HRUSCHKA, E. R. Relative clustering validity criteria: A comparative overview. **SAM**, v. 3, n. 4, p. 209–235, 2010. Citado nas páginas 52, 53 e 146.

VENNER, J. **Pro Hadoop**. 1st. ed. Berkely, CA, USA: Apress, 2009. ISBN 1430219424, 9781430219422. Citado nas páginas 103, 104 e 105.

WHITE, T. **Hadoop: The Definitive Guide**. 1st. ed. O'Reilly Media, Inc., 2009. ISBN 0596521979, 9780596521974. Disponível em: <<https://dl.acm.org/citation.cfm?id=1717298>>. Citado na página 102.

WHITE, T. **Hadoop: The Definitive Guide**. O'Reilly Media, Inc., 2012. ISBN 1449311520, 9781449311520. Disponível em: <<https://dl.acm.org/citation.cfm?id=2285539>>. Citado nas páginas 102, 103 e 105.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques**. 2nd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 0120884070. Disponível em: <<https://dl.acm.org/citation.cfm?id=1205860>>. Citado na página 37.

XU, R.; WUNSCH, D. Survey of clustering algorithms. **IEEE Transactions on Neural Networks**, v. 16, n. 3, p. 645–678, May 2005. ISSN 1045-9227. Citado nas páginas 13, 35, 36, 37, 38, 43, 46 e 51.

XU, R.; WUNSCH, D. **Clustering**. Wiley-IEEE Press, 2009. ISBN 9780470276808. Disponível em: <<https://dl.acm.org/citation.cfm?id=1483087>>. Citado nas páginas 13, 35, 36, 37, 39, 40, 41, 42, 44, 46, 47, 51, 52, 53 e 146.

XU, X.; JÄGER, J.; KRIEGEL, H.-P. A fast parallel clustering algorithm for large spatial databases. **DataMinKnowlDisc**, v. 3, n. 3, p. 263–290, 1999. Citado na página 73.

YANG, J.; ZHONG, N.; YAO, Y.; WANG, J. Local peculiarity factor and its application in outlier detection. In: **Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2008. (KDD '08), p. 776–784. ISBN 978-1-60558-193-4. Disponível em: <<http://doi.acm.org/10.1145/1401890.1401983>>. Citado nas páginas 155 e 168.

ZAHARIA, M.; CHOWDHURY, M.; DAS, T.; DAVE, A.; MA, J.; MCCAULEY, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: **Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation**. Berkeley, CA, USA: USENIX Association, 2012. (NSDI'12), p. 2–2. Disponível em: <<http://dl.acm.org/citation.cfm?id=2228298.2228301>>. Citado na página 105.

ZHANG, K.; HUTTER, M.; JIN, H. A new local distance-based outlier detection approach for scattered real-world data. In: **Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining**. Berlin, Heidelberg: Springer-Verlag, 2009. (PAKDD '09), p. 813–822. ISBN 978-3-642-01306-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-01307-2_84>. Citado nas páginas 155 e 168.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: An efficient data clustering method for very large databases. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 25, n. 2, p. 103–114, jun 1996. ISSN 0163-5808. Disponível em: <<http://doi.acm.org/10.1145/235968.233324>>. Citado nas páginas 31, 32, 33, 43, 45, 88, 89 e 112.

ZHAO, Y.; KARYPIS, G.; FAYYAD, U. Hierarchical clustering algorithms for document datasets. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 10, n. 2, p. 141–168, mar. 2005. ISSN 1384-5810. Disponível em: <http://dx.doi.org/10.1007/s10618-005-0361-3>. Citado na página 29.

ZHOU, J.; SANDER, J. Data bubbles for non-vector data: Speeding-up hierarchical clustering in arbitrary metric spaces. In: . [S.l.: s.n.], 2003. p. 452–463. Citado na página 174.

ZIMEK, A.; GAUDET, M.; CAMPELLO, R. J.; SANDER, J. Subsampling for efficient and effective unsupervised outlier detection ensembles. In: **Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2013. (KDD '13), p. 428–436. ISBN 978-1-4503-2174-7. Disponível em: <http://doi.acm.org/10.1145/2487575.2487676>. Citado nas páginas 155 e 168.