UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Aruanda Simões Gonçalves Meiguins

Programação Automática Aplicada à Geração de Algoritmos de Agrupamento

Belém

2019

Aruanda Simões Gonçalves Meiguins

Programação Automática Aplicada à Geração de Algoritmos de Agrupamento

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas e Naturais como requisito parcial para obtenção do título de Doutorado.

Universidade Federal do Pará

Orientador: Dr. Jefferson Magalhães de Morais

Coorientador: Dr. Bianchi Serique Meiguins

Belém

2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

M512p Meiguins, Aruanda Simões Gonçalves
Programação automática aplicada à geração de algoritmos de agrupamento / Aruanda Simões Gonçalves Meiguins. — 2019.
88 f. : il. color.

Orientador(a): Prof. Dr. Jefferson Magalhães de Morais
Coorientador(a): Prof. Dr. Bianchi Serique Meiguins
Tese (Doutorado) - Programa de Pós-Graduação em Ciência da Computação, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2019.

1. Programação automática. 2. Algoritmos de agrupamento baseado em densidade. 3. Algoritmos baseados em estimativa de distribuição. 4. Mineração de dados. I. Título.

CDD 006.3

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARUANDA SIMÕES GONÇALVES MEIGUINS

PROGRAMAÇÃO AUTOMÁTICA APLICADA À GERAÇÃO DE ALGORITMOS DE AGRUPAMENTO

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará como requisito para obtenção do título de Doutor em Ciência da Computação, defendida e aprovada em 13/12/2019, pela banca examinadora constituída pelos seguintes membros:

Prof. Dr. Jefferson Magalhães de Morais
Orientador – PPGCC/UFPA

Prof. Dr. Bianchi Serique Meiguins
Co-orientador – PPGCC/UFPA

Prof. Dr. Nelson Cruz Sampaio Neto
Membro Interno – PPGCC/UFPA

Prof. Dr. Rommel Thiago Jucá Ramos
Membro Interno – PPGCC/UFPA

Prof. Dr. Lídio Mauro Lima de Campos
Membro Externo – FACOMP/UFPA

Prof. Dr. Gustavo Augusto Lima de Campos
Membro Externo – UECE

Visto:
Prof. Dr. Nelson Cruz Sampaio Neto
Coordenador do PPGCC/UFPA

Prof. Dr. Nelson Cruz Sampaio Neto
Coordenador do PPGCC/UFPA
SIAPE: 2659210

Ao meu pai, Artur Pojo,

a quem dediquei este trabalho antes mesmo de começar a escrevê-lo.

Agradecimentos

Ao meu orientador, Professor Dr. Jefferson Magalhães de Morais eu agradeço imensamente pela confiança que depositou em mim.

Agradeço ao Professor Dr. Bianchi Serique Meiguins, que conheci como pesquisador há muitos anos, e desde então alcançamos tanto juntos. Este trabalho passa a ser mais uma conquista que dividimos com orgulho, assim como nossa família, nossa casa e nossa história.

Aos membros da banca, Prof. Dr. Gustavo Campos, Prof. Dr. Lidio Campos, Prof. Dr. Rommel Ramos e Prof. Dr. Nelson Sampaio, eu agradeço pelas valiosas contribuições, desde a etapa de qualificação, que realmente transformaram este trabalho em sua versão final.

Um agradecimento especial ao Professor Dr. Alex Freitas, pela inspiração, apoio e todo o conhecimento compartilhado ao longo dos anos.

Foram muitas as pessoas que trabalharam comigo nesta área de pesquisa, contribuindo de diversas formas para que este objetivo fosse alcançado. Gostaria de citar em especial os colegas Samuel Felix, Yuri Nassar, Paulo Igor Godinho, Cesar Oliveira, Rafael Veras e Michel Montenegro, bem como o Prof. Dr. Roberto Limão.

À minha família, alegria de todos os meus dias... Meus pais Rosangela e Artur, que me prepararam para a vida com amor, exemplo, proteção e incentivo; meus filhos Bruno e Barbara, que nem em sonho eu poderia imaginar tão perfeitos; meus irmãos Ariana, Artur e Amanda, que só me dão orgulho e carinho, tornando tudo mais fácil; minha avó Laura, fonte de inspiração. Que sorte que eu tenho de ter nascido, crescido e casado sempre rodeada do amor de tantos familiares e amigos.

Agradeço ainda o apoio do LabVIS e da Gol Software.

"Darwinism is not a theory of random chance."
(Richard Dawkins)

Resumo

A programação automática estuda mecanismos para geração automatizada de programas de computador. Este trabalho busca produzir, automaticamente, algoritmos de mineração de dados, em particular os algoritmos de agrupamento (*clustering*). Foram estudados algoritmos de agrupamento baseados em densidade, uma classe de algoritmos de agrupamento que suporta a identificação de grupos de forma arbitrária, no lugar do formato tipicamente esférico de outras abordagens. Para a geração dos algoritmos, usou-se a técnica de Computação Evolutiva chamada "Algoritmos de Estimativa de Distribuição" (EDA, do inglês *Estimation of Distribution Algorithms*). De forma a garantir a geração de algoritmos válidos de agrupamento, definiu-se um grafo acíclico direcionado, onde cada nó representa um bloco de construção – um procedimento representativo de um algoritmo de agrupamento; e cada aresta representa uma possível sequência de execução entre dois nós. O grafo especifica o alfabeto do EDA, ou seja, cada caminho do nodo inicial ao nodo terminal representa um algoritmo de agrupamento. Os algoritmos produzidos são avaliados de forma objetiva com uso do método *Clest*, originalmente proposto para determinar o número ideal de grupos para uma base de dados, neste trabalho adaptado para determinar a qualidade dos algoritmos de agrupamento. Os resultados apresentados avaliam os algoritmos de agrupamento automaticamente produzidos para sete bases de dados de domínio público, com apoio de técnicas de visualização. Para todas as bases analisadas, o EDA produziu algoritmos originais, ou seja, formados por uma combinação de procedimentos que não corresponde a nenhum algoritmo previamente existente. Para atingir esse objetivo, o EDA analisa mais de um bilhão de combinações possíveis, considerando os caminhos do grafo, as formas de funcionamento de cada procedimento, os critérios de distância e os valores de parâmetros.

Palavras-chave: programação automática. agrupamento baseado em densidade. algoritmos baseados em estimativa de distribuição. mineração de dados.

Abstract

Automatic programming studies mechanisms for the automatic generation of computer programs. The goal of this work is to automatically build data mining algorithms, particularly clustering algorithms. The focus of the research is on density-based clustering algorithms, a class of clustering algorithms that supports the identification of arbitrary-shape clusters, instead of the typically spherical clusters produced by other clustering approaches. We propose the use of Estimation of Distribution Algorithms for the artificial generation of density-based clustering algorithms. In order to guarantee the generation of valid algorithms, a directed acyclic graph (DAG) was defined where each node represents a procedure (building block) and each edge represents a possible execution sequence between two nodes. The Building Blocks DAG specifies the alphabet of the EDA, that is, each different path from the initial node to a terminal node represents a possibly generated algorithm. The *fitness* of the EDA objectively evaluates the quality of the clustering algorithms. To this end, we propose an adaptation of the Clest method, which was originally proposed to determine the ideal number of clusters for a given dataset. We analyze the performance of the clustering algorithms artificially generated for seven public-domain datasets with the help of visualization techniques. The EDA generated original clustering algorithms for each of the analyzed datasets - a sequence of procedures that did not correspond to any previously existing algorithm. There are more than 200 million possible combinations when all the factors are considered - the DAG path, the distance metrics and the parameter values.

Keywords: automatic programming. density-based clustering. estimation of distribution algorithms. data mining.

Lista de ilustrações

Sumário

1 Introdução

Algoritmos para mineração de dados (em inglês, *Data Mining*) têm aplicações de destaque nas áreas acadêmica e comercial. Da análise de sequências de proteínas para determinação de sua função no sistema nervoso (WANG et al., 2005) à descoberta de um padrão de consumo relacionando produtos de um supermercado (WESTPHAL, 1998), as aplicações de mineração de dados proporcionam o melhor aproveitamento das volumosas informações armazenadas nos diversos bancos de dados das organizações.

Neste contexto, destacam-se os algoritmos de Agrupamento (*clustering*), uma área da mineração de dados que analisa bases de dados em busca de padrões comuns a grupos de itens, com o objetivo de dividir a base original em subconjuntos com características em comum (BERRY GORDON, 2004).

Originalmente, duas classes de algoritmos de agrupamento se desenvolveram de forma consolidada: os algoritmos de particionamento, baseados na ideia de dividir, sucessivamente, a base de dados original para definir os grupos ideais; e os algoritmos hierárquicos, capazes de determinar uma organização em forma de níveis e sub-níveis para os grupos, identificando uma hierarquia para os dados. Uma limitação dessas abordagens, no entanto, é a dificuldade de tratar grupos com distribuição irregular de dados, ou seja, grupos de formato arbitrário.

A abordagem baseada em densidade (HAN; KAMBER; PEI, 2012), por outro lado, identifica os grupos de acordo com a densidade de itens de dados em cada região analisada. Cada algoritmo tem uma maneira específica de medir a densidade dos itens, mas todos identificam um grupo pela frequência relativa de pontos em uma determinada área do espaço de dados.

A diversidade das estratégias para especificação de algoritmos de agrupamento provoca um fenômeno comum a diversas classes de algoritmos: uma proliferação de extensões e variações de um conjunto de algoritmos básicos, dificultando a tarefa de seleção do algoritmo mais adequado. A seleção do algoritmo deve levar em consideração características específicas do domínio do problema e da base de dados em particular. Torna-se inviável analisar manualmente todas as opções, e com isso, muitas vezes, deixa-se de usar o algoritmo mais apropriado para um determinado projeto.

Uma área de pesquisa que atende bem este tipo de problema é a programação automática, que busca projetar uma solução específica a um determinado problema, desenvolvendo um novo algoritmo de forma automática - um novo programa que não foi produzido de forma manual, por um ser humano, mas projetado por outro programa de computador.

A área de Inteligência Computacional mais usada para a programação automática é a Programação Genética, com primitivas baseadas na teoria evolucionista de Darwin como troca de material genético (ou cruzamento, do inglês *crossover*), mutação e seleção do mais apto.

Outra área da Computação Evolutiva, os Algoritmos para Estimativa de Distribuição (EDA – *Estimation of Distribution Algorithms*) eliminam a necessidade de operadores de cruzamento e mutação, baseando-se na evolução de modelos probabilísticos da população de possíveis soluções para o problema abordado. A eficiência desta classe de algoritmos evolutivos em problemas anteriormente abordados indica seu grande potencial para a evolução de programas (LARRAñAGA; LOZANO, 2002; KREJCA; WITT, 2020).

Considerando-se a dificuldade da seleção e parametrização de um algoritmo de agrupamento para uma base de dados específica, a hipótese levantada por esta pesquisa é que a programação automática é eficaz na produção de novos algoritmos para o agrupamento de dados, evoluindo o algoritmo e os valores de parâmetros mais adequados para uma base de dados em particular.

É importante garantir que os algoritmos produzidos sejam algoritmos de agrupamento válidos, isto é, gerem como resultado agrupamentos consistentes com a base de dados analisada. Para tanto, a abordagem evolutiva deve se basear em um processo de geração automática direcionada, mantendo o resultado compatível com algoritmos de agrupamento previamente propostos.

O método usado para o desenvolvimento da solução proposta se baseou na pesquisa de algoritmos de agrupamento baseados em densidade, com a identificação das ideias centrais e procedimentos significativos desses algoritmos, de forma a guiar a geração automática de algoritmos de agrupamento. Uma estrutura auxiliar adequada foi proposta para que os algoritmos produzidos sejam soluções válidas para o agrupamento de dados.

1.1 Justificativa

A programação automática é uma área com o objetivo ambicioso de construir programas capazes de desenvolver, automaticamente, outros programas de computador. Como destacado em (O'NEILL; SPECTOR, 2019), os objetivos da programação automática evoluíram ao longo dos anos. Décadas atrás se considerava um compilador uma forma de programação automática. Desde então, a programação automática vem sendo aplicada a diversas áreas e conseguiu atingir resultados melhores que os programas manualmente desenvolvidos em vários cenários, como por exemplo a substancial redução em número de linhas de código do serviço Google Translate (GOTTSCHLICH, 2019).

Em uma área complexa como a Mineração de Dados, em especial quando se

considera a subjetiva tarefa de Agrupamento, observa-se o grande potencial de aplicação da programação automática.

Para um usuário com a tarefa de analisar e agrupar dados de uma determinada base de dados, a escolha do algoritmo de agrupamento mais adequado é um fator determinante de sucesso. Infelizmente, o número de opções à disposição é tão grande que a busca exaustiva se torna inviável. Neste contexto, a simples seleção do algoritmo de agrupamento mais adequado poderia ser uma significativa contribuição.

No entanto, ao se explorar o universo de algoritmos previamente elaborados para o problema específico de agrupamento, percebe-se que nem todas as possibilidades foram exploradas pela comunidade científica de Mineração de Dados. A programação automática pode ser usada para ultrapassar essa limitação, buscando de forma muito mais livre a melhor solução para o problema específico.

Por outro lado, apesar da relevância de uma busca mais aberta por possíveis soluções, é importante garantir que a busca seja direcionada a soluções válidas de agrupamento. É portanto, relevante a especificação de uma estrutura de dados auxiliar que sirva como guia durante a geração dos algoritmos, garantindo a produção de algoritmos de agrupamento funcionais. A estrutura proposta neste trabalho, um grafo de procedimentos, permite orientar a busca por soluções na forma de caminhos possíveis em um grafo acíclico direcionado. Os procedimentos que fazem parte do grafo são procedimentos previamente propostos em algoritmos de agrupamento existentes, e os caminhos do grafo determinam as possíveis sequencias de execução.

Desta forma, espera-se explorar as soluções possíveis de forma a maximizar as possibilidades avaliadas, sem sobrecarregar o processo com a avaliação um grande número de alternativas inviáveis.

1.2 Objetivos

Com base nas ideias centrais dos algoritmos de agrupamento baseado em densidade já propostos, o objetivo geral deste trabalho é produzir, automaticamente, com apoio de Algoritmos para Estimativa de Distribuição, algoritmos de mineração de dados para agrupamento.

Como objetivos específicos, espera-se:

- Identificar ideias centrais de algoritmos de agrupamento, destacando os procedimentos básicos que fazem parte de tais algoritmos;
- Organizar os procedimentos destacados em uma estrutura de dados que possa mapear as sequências válidas de execução dos procedimentos, de forma a produzir algoritmos

válidos de agrupamento.

- Especificar uma função de avaliação objetiva que possa comparar e selecionar o melhor algoritmo de agrupamento para uma determinada base de dados.
- Desenvolver um algoritmo de estimativa de distribuição que construa, automaticamente, um algoritmo de agrupamento para uma base de dados em particular, a partir da estrutura de dados auxiliar elaborada.
- Analisar os resultados de testes do EDA proposto com uso de bases de dados de domínio público.

1.3 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos:

1. Introdução: apresenta de forma geral o escopo da pesquisa, justificando a relevância do trabalho e apresentando seus objetivos.
2. Computação Evolutiva: apresenta as diferentes técnicas evolutivas, com destaque para a adotada neste projeto: os algoritmos de estimativa de distribuição.
3. Mineração de Dados: introduz conceitos relacionados ao processo de descoberta de conhecimento em bases de dados e diferentes tarefas de mineração de dados.
4. Agrupamento de Dados: apresenta a área de aplicação do trabalho, em especial as abordagens existentes para agrupamento.
5. Trabalhos Relacionados: comenta trabalhos relacionados nas áreas de programação automática e algoritmos de estimativa de distribuição, destacando a contribuição do presente trabalho.
6. Um Algoritmo para Geração Automática de Algoritmos de Agrupamento: descreve o projeto Autoclustering, critérios de avaliação e organização dos algoritmos, listando os algoritmos básicos de agrupamento selecionados e seus blocos de construção.
7. Experimentos e Resultados: discute os resultados obtidos após as execuções do Autoclustering com uso de bases de dados de domínio público.
8. Conclusões e trabalhos futuros: neste capítulo são apresentadas as considerações finais, assim como sugestões de trabalhos futuros.

2 Computação Evolutiva

Este capítulo descreve conceitos da área de Computação Evolutiva, apresentando as principais abordagens de algoritmos evolutivos e discutindo sua aplicação para a área de mineração de dados.

A Computação Evolutiva é inspirada no processo de seleção natural da Teoria da Evolução de Darwin. Utilizando um conjunto de indivíduos, algumas vezes chamados "cromossomos", o algoritmo evolucionário busca melhorar a cada geração o conjunto de indivíduos da geração anterior. Ao contrário da maioria dos mecanismos de busca que, em geral, utilizam um único ponto para varrer uma superfície de busca, algoritmos evolucionários utilizam-se de uma população de indivíduos pontuais que passa por um processo iterativo de evolução, aumentando assim a possibilidade de que uma solução satisfatória seja encontrada (FOGEL, 1998).

2.1 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) popularizaram-se na comunidade científica com a publicação do livro "*Adaptation in Natural and Artificial Systems*" por (HOLLAND, 1975), apesar das primeiras pesquisas datarem da década de 50. Inúmeras implementações de Algoritmos Genéticos seguiram-se, com especial foco em problemas de otimização.

Os AGs se inspiram na teoria da evolução das espécies de Charles Darwin. A teoria afirma que, em um determinado ambiente, os indivíduos que tendem a sobreviver são os que conseguem extrair recursos essenciais para a sobrevivência com maior êxito. A evolução se dá através da combinação genética entre indivíduos de uma população para a produção de gerações sucessivas, sendo que os mais aptos têm uma tendência maior de passar o material genético para seus descendentes diretos, o que acontece com uma menor frequência com os indivíduos menos adaptados.

Um Algoritmo Genético básico, segundo (FOGEL, 1998), é descrito pelos seguintes passos:

1. Define-se uma função objetivo que determina a qualidade de qualquer solução em potencial para um determinado problema.

2. Cria-se uma população de soluções candidatas que são tipicamente codificadas como um vetor de caracteres (cromossomos). Os valores possíveis que compõem um cromossomo devem ser capazes de representar todas as possíveis soluções para

o problema em questão. Um exemplo de representação dos cromossomos pode ser através de valores binários (0 e 1).

3. Decodificam-se os cromossomos de forma apropriada para a etapa de avaliação. De acordo com a função objetivo criada no passo 1, atribui-se a cada cromossomo um valor que representa a qualidade da solução correspondente.

4. Selecionam-se os cromossomos de acordo com uma entre várias técnicas de seleção existentes. Uma das técnicas mais comuns é usar uma roleta, na qual cada cromossomo corresponde a uma fatia. Quanto maior for a aptidão do cromossomo, maior será sua fatia na roleta. Assim, a probabilidade de um cromossomo ser selecionado para as operações genéticas é proporcional ao seu valor de aptidão.

5. Os cromossomos selecionados participam das operações genéticas de cruzamento e mutação. Como resultado, uma nova população de indivíduos é criada, dando início a uma nova geração.

6. O algoritmo volta ao passo 3 se uma solução satisfatória para o problema não for encontrada; ou enquanto um determinado número de gerações não for atingido.

2.2 Programação Genética

A Programação Genética (PG) pode ser considerada como uma evolução dos Algoritmos Genéticos, pois utiliza vários conceitos do algoritmo original desenvolvido por (HOLLAND, 1975).

O surgimento da Programação Genética, proposta por (KOZA, 1990), foi consequência, principalmente, da necessidade crescente de se utilizar estruturas de dados mais poderosas, como estruturas baseadas em árvores, para representar soluções candidatas de problemas mais complexos, como tarefas de aprendizagem.

(KOZA, 1990) buscou uma solução para a programação automática que consistia em evoluir programas de computador representados por estruturas baseadas em árvore, cuja forma, tamanho e complexidade pudessem ser alterados no decorrer do processo.

A PG é uma técnica usada para fazer com que máquinas possam criar automaticamente um programa de computador para resolver um determinado problema de uma forma singular, a partir de uma descrição em alto nível das necessidades do problema em questão.

O processo pode ser resumido como uma busca por um programa de computador que consiga resolver ou se aproximar de uma solução satisfatória para um dado problema. Essa busca se dá dentro de uma população de programas que evoluem a cada geração através da aplicação dos operadores genéticos de cruzamento e mutação.

2.3 Algoritmos de Estimativa de Distribuição

Os Algoritmos para Estimativa de Distribuição (EDA - *Estimation of Distribution Algorithms*) são uma abordagem mais recente da Computação Evolucionária. Generalizando o uso de algoritmos genéticos, esta técnica elimina o uso dos operadores de cruzamento e mutação.

A cada geração do EDA uma nova população é produzida a partir da distribuição probabilística da avaliação dos indivíduos mais aptos. O modelo probabilístico é formado a cada geração, a partir da avaliação da geração anterior. A principal diferença entre esta abordagem e outros algoritmos evolucionários é, portanto, o caráter explícito em que se expressam as inter-relações entre as variáveis que formam os indivíduos.

A execução de um EDA é formada por quatro passos principais (LARRAñAGA; LOZANO, 2002):

1. Inicialmente, uma população de N indivíduos é produzida, geralmente assumindo-se uma distribuição uniforme das variáveis.
2. Um número $m < N$ de indivíduos é então selecionado da população por qualquer dos métodos de seleção usados em algoritmos evolucionários, como torneio ou truncamento.
3. A partir da avaliação dos indivíduos selecionados, monta-se um melhor modelo probabilístico para as variáveis.
4. Com base na distribuição obtida no passo anterior, produz-se uma nova população de N indivíduos.

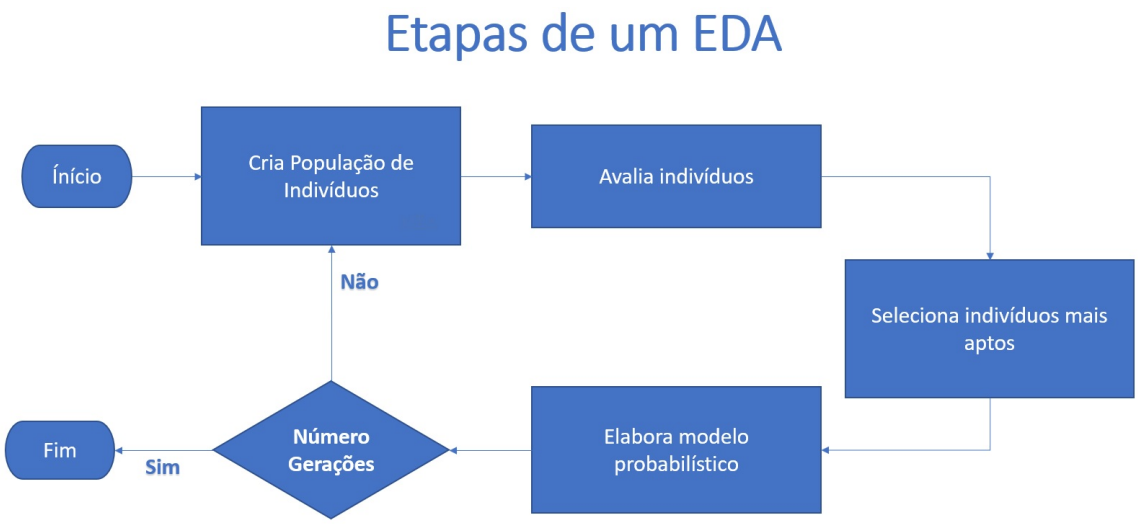
O algoritmo repete esses passos até que um critério de saída previamente estabelecido seja atendido. Por exemplo, o critério pode ser um número máximo de gerações, como ilustrado na Figura 1.

Na computação evolutiva, as soluções candidatas são geralmente produzidas pela combinação e modificação das soluções existentes, de forma estocástica. Cruzamento e mutação são em geral os operadores responsáveis pela exploração do espaço de busca, mas a distribuição de probabilidade das novas soluções neste espaço não fica explícita durante o processo.

Um EDA atua de forma mais explícita, obtendo uma distribuição probabilística da população e usando esta distribuição para gerar novas soluções candidatas, por amostragem.

Além da vantagem de sua representação mais simples e compacta, uma solução baseada em EDA apresenta maior robustez quanto à convergência prematura, um problema frequente em soluções fortemente baseadas em cruzamento (ESHELMAN; SCHAFFER,

Figura 1. Etapas de um EDA



Fonte: Autora, 2019

1991)(LARRAñAGA; LOZANO, 2002)(ZHANG; MüHLENBEIN, 2004). Segundo (SHAN et al., 2007), a substituição dos operadores genéticos por um modelo simples, mas ainda assim poderoso, facilita muito a compreensão do comportamento do algoritmo.

3 Mineração de Dados

Este capítulo define o processo de Descoberta de Conhecimento em Bases de Dados e suas etapas mais importantes, apresentando algumas técnicas mais utilizadas de mineração de dados.

3.1 Descoberta de Conhecimento em Bases de Dados

Com o aumento da capacidade de armazenamento e processamento dos sistemas computacionais, multiplicaram-se as possibilidades de aplicações a serem oferecidas para um número crescente de usuários de tecnologia.

Nas últimas décadas, o foco dos sistemas de informação passou pelos processos que precisavam ser formalizados computacionalmente, pelos dados que precisavam ser armazenados em meio digital, até se concentrar, atualmente, no negócio do cliente, que precisa ser apoiado de forma mais inteligente pelo computador. Dependendo do tipo de organização, o negócio ou objetivo em questão pode ser lucro, melhoria do atendimento, disseminação facilitada de informação ou desenvolvimento tecnológico, para citar os focos mais comuns.

A típica organização atual possui sistemas de informação para o controle de grande parte de suas informações. E atualmente já não é exceção a organização que usa esses dados de forma inteligente para atingir mais rápida ou facilmente seus objetivos.

O objetivo da área de Descoberta de Conhecimento em Banco de Dados (ou KDD, do inglês *"Knowledge Discovery in Databases"*) é produzir ferramentas computacionais capazes de extrair conhecimento útil e novo do grande volume de dados transacionais armazenados nos sistemas computacionais, suportando decisões estratégicas da organização.

> Descoberta de conhecimento em bases de dados é o processo não trivial de identificação de padrões válidos, inéditos, potencialmente úteis e essencialmente compreensíveis (FAYYAD, 1996).

O processo KDD deve ser realizado com apoio de diversas ferramentas computacionais, em todas as etapas, que envolvem preparação dos dados, aplicação de um algoritmo de mineração de dados, análise dos resultados e, finalmente, consolidação do conhecimento, quando as novas informações descobertas efetivamente provocam mudanças nos processos da organização.

Um processo KDD agrega à execução do algoritmo de mineração de dados importantes etapas de pré e pós-processamento. Antes da execução do algoritmo, é preciso

filtrar e tratar as informações que serão consideradas no processo. É também durante o pré-processamento que se determina a tarefa de mineração de dados a ser empregada, bem como o algoritmo a ser utilizado para a tarefa em questão. Após a execução do algoritmo, atividades de validação e consolidação das informações descobertas garantem que o processo foi bem-sucedido, e aplicado aos problemas da organização.

Mineração de dados é considerada, portanto, uma parte de um processo maior, que busca a descoberta de conhecimento em uma base de dados. Apesar de ser uma parte essencial deste processo, o uso isolado de um algoritmo de mineração de dados, sem as devidas fases de pré e pós-processamento, tende a produzir resultados irrelevantes ou mesmo inválidos. Em geral, processos KDD conduzidos de forma descompromissada não provocam impacto significativo na organização, sendo frequentemente descontinuados antes que resultados relevantes sejam atingidos.

3.2 Etapas do processo KDD

Em um processo que busca descobrir informações interessantes sobre um determinado domínio, o sucesso pode ser determinado pelo critério com que os dados disponíveis são organizados, assim como pela precisão com que os padrões são extraídos e pela atenção às informações inicialmente obtidas.

Um processo KDD pode precisar de várias iterações para atingir seus objetivos, dependendo da estrutura inicial dos dados analisados, do nível de conhecimento prévio sobre o domínio do problema e do algoritmo utilizado para a extração de padrões.

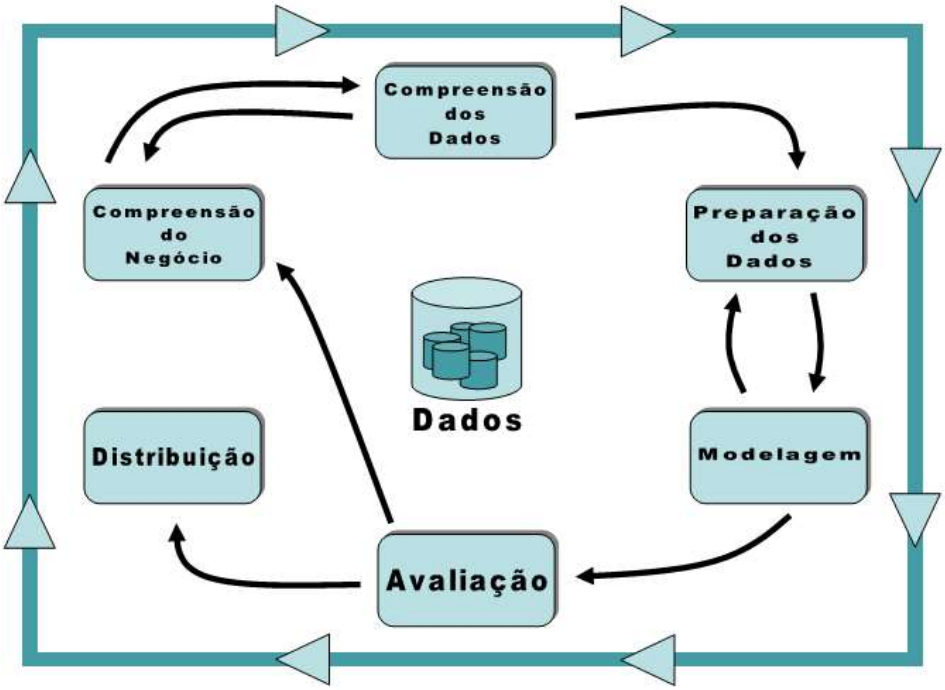
A definição das etapas necessárias para o sucesso de um processo KDD, assim como a ordem em que devem ser executadas, depende de características específicas de cada base de dados e do objetivo do projeto.

A busca de um modelo genérico para o processo, no entanto, é uma preocupação para profissionais da área. Uma boa iniciativa nesse sentido partiu da indústria de software. O consórcio CRISP-DM (*Cross-Industry Standard Process Model for Data Mining* – Modelo Multi-plataforma Padrão para mineração de dados) foi criado em 1996, com participação da DaimlerChrisler, indústria pioneira no investimento em mineração de dados, e da SPSS, uma empresa que lançou uma das primeiras ferramentas comerciais da área, tendo sido posteriormente adquirida pela IBM. Em agosto de 2000, foi lançado o CRISP-DM 1.0, que é o modelo atualmente mais bem aceito para a padronização de processos para descoberta de conhecimento em bases de dados (PIATESKY, 2014).

A Figura 2 apresenta o ciclo típico do modelo CRISP-DM e como as etapas apresentadas são conectadas. O processo KDD só se encerra após a efetiva utilização dos resultados para o fim proposto. Espera-se que os conhecimentos descobertos pelo processo

influenciem a tomada de decisão da organização, modificando, de alguma forma, o seu modo de operação.

Figura 2. Processo CRISP-DM.



Fonte: (PIATESKY, 2014)

As etapas que formam o ciclo do modelo de referência CRISP-DM são descritas, de forma resumida, nas próximas subseções.

3.2.1 Compreensão do Negócio

O foco desta etapa está no entendimento dos objetivos e requisitos do projeto. Este conhecimento deve ser adaptado na forma de uma definição para o problema de mineração de dados, a partir da qual será projetado um plano preliminar para atingir os objetivos definidos.

3.2.2 Preparação dos Dados

Esta etapa envolve todas as atividades necessárias à construção do conjunto final de dados, aquele que será submetido ao algoritmo de mineração de dados. Essas atividades, normalmente, repetem-se diversas vezes, incluindo a seleção de tabelas, registros e atributos, assim como a transformação e limpeza dos dados.

Muitas vezes referenciada como fase de pré-processamento do processo KDD, esta etapa é de crítica importância para o sucesso de qualquer projeto de descoberta de conhecimento.

As informações utilizadas no processo são, frequentemente, oriundas de diversos sistemas da mesma organização, e em alguns casos também de sistemas externos, como de fornecedores, clientes e parceiros. Mesmo em se tratando de sistemas da mesma organização, é comum que os dados não estejam armazenados de forma integrada. É importante, portanto, nesta etapa, usar ferramentas que suportem diversas fontes de dados e preparadas para trabalhar com dados heterogêneos.

Uma das atividades de pré-processamento envolve a escolha das informações da base de dados a serem levadas em consideração no restante do processo. Em geral, há uma diversidade de informações disponíveis na base de dados que será submetida ao processo KDD. Algumas informações são diretamente relacionadas ao problema que se está abordando, outras são totalmente irrelevantes. No entanto, a divisão destes dois grupos, geralmente, não é muito clara. Alguns atributos aparentemente irrelevantes podem ter alguma relação com o problema. Neste caso, um padrão envolvendo um subconjunto de atributos seria particularmente interessante pelo grau de surpresa associado. Da mesma forma, alguns atributos, intuitivamente relacionados ao problema, podem apenas introduzir ruído na base de dados a ser minerada, prejudicando o processo KDD.

3.2.3 Modelagem

A fase de modelagem inclui a seleção da tarefa de mineração que especifica o objetivo do processo. Nesta etapa também é selecionada a técnica mais adequada para a tarefa especificada.

Existem diversas tarefas de mineração de dados. A tarefa a ser trabalhada deve ser selecionada de acordo com a compreensão do negócio e dos dados do projeto. Cada uma exige uma diferente entrada de dados, e apresenta os padrões descobertos em um diferente formato. As tarefas de Classificação, Agrupamento e Regras de Associação, descritas mais adiante neste capítulo, são as mais comumente utilizadas.

Há tipicamente diversas técnicas de mineração de dados disponíveis para o mesmo problema. Nesta fase, vários algoritmos de mineração de dados são selecionados e aplicados, até que seus parâmetros estejam ajustados de forma otimizada. O objetivo deste projeto é facilitar esta etapa do projeto, permitindo a geração automática de um algoritmo adequado à base de dados selecionada.

3.2.4 Avaliação

Após a produção de um ou mais modelos de qualidade para os dados selecionados, é importante avaliar se esses resultados realmente atingem os objetivos inicialmente estipulados. Nesta fase, avalia-se a adequação dos resultados da fase anterior.

A fase de avaliação determina, em especial, se os padrões encontrados correspondem às expectativas do usuário. Em alguns casos, problemas nos dados de origem, decisões equivocadas no início do projeto ou utilização inadequada das ferramentas computacionais podem prejudicar os resultados. Na maioria dos projetos, o ciclo completo precisa ser repetido algumas vezes até que resultados realmente interessantes possam ser obtidos. Nesta etapa, é comum que a equipe opte por uma mudança de ferramenta, de técnica e até mesmo da tarefa especificada.

3.2.5 Distribuição

É preciso organizar o novo conhecimento obtido e apresentá-lo de forma adequada ao usuário. Dependendo dos requisitos iniciais, esta fase pode resumir-se à geração de um relatório, ou pode exigir a implementação de um processo contínuo de mineração de dados e re-avaliação.

A fase de distribuição se encarrega de divulgar o conhecimento descoberto de forma apropriada ao projeto. O projeto só é considerado concluído quando provocou algum efeito para o usuário, seja uma mudança nas regras da empresa, uma nova estratégia de marketing, ou muitas outras possíveis manifestações que dependem tanto do tipo de projeto quanto do perfil do usuário.

3.3 Tarefas de Mineração de Dados

As tarefas de mineração de dados estão associadas aos diferentes possíveis objetivos de um processo KDD. Para a mesma base de dados, várias tarefas de mineração de dados podem ser usadas com objetivos distintos e resultados igualmente relevantes (WESTPHAL, 1998).

Por exemplo, uma empresa no ramo de varejo pode usar sua base de dados de transações de vendas tanto para determinar o perfil de compras dos clientes (tarefa de agrupamento) quanto para determinar que produtos podem ser vendidos, promocionalmente, em conjunto (tarefa de associação). Da mesma forma, um gerente da área da saúde pode usar uma base de dados com informações médicas para identificar grupos de pacientes com propensão a determinadas doenças (tarefa de classificação) ou para descobrir que doenças típicas da infância estão associadas a doenças da meia idade (tarefa de padrões sequenciais).

3.3.1 Classificação

A tarefa de classificação objetiva prever o valor de um atributo específico a partir dos valores de outros atributos de uma base de dados. É uma tarefa muito comumente usada porque está relacionada a uma atividade intuitivamente realizada pelos seres humanos para analisar qualquer ambiente ou contexto. Classificar itens de acordo com diversos critérios é uma forma eficaz de facilitar a compreensão do objeto em análise.

No contexto de mineração de dados, classificação é uma função de aprendizado que mapeia dados de entrada em um número finito de categorias, chamadas classes. A partir da análise de várias instâncias (registros de uma base de dados), cada uma pertencente a uma classe especificada, a tarefa de classificação busca relacionamentos entre os atributos e uma das classes. Os relacionamentos descobertos podem ser usados, então, para predizer a classe de uma instância nova e desconhecida (REZENDE, 2003).

Várias técnicas podem ser usadas para classificação de dados, como redes neurais, máquina de vetores de suporte (SVM, do inglês *"Support Vector Machihe"*) e árvores de decisão (NAYAK; NAIK; BEHERA, 2015; HAN; KAMBER; PEI, 2012; NIKAM, 2015). Um método comumente usado para a tarefa de classificação se baseia na indução de regras, que será o foco da presente seção.

No caso da classificação por indução de regras, os relacionamentos podem ser especificados por regras no formato SE [uma condição é verdadeira] ENTAO [a classe do objeto é X], onde a condição pode envolver quaisquer atributos da base de dados, mas X é sempre um dos valores do atributo selecionado como meta da classificação. Por exemplo, uma regra de classificação poderia ser:

SE Sexo= 'F' E FaixaEtaria = 'Idoso' ENTAO Credito = 'Aprovado'

É importante ressaltar que a tarefa de classificação exige que um conjunto completo de regras de classificação seja especificado. Ou seja, no exemplo de uma base de dados para análise de crédito, seriam necessárias regras que determinassem, corretamente, a classe (aprovado ou reprovado) para todas as instâncias da base de dados.

Além disso, é importante estimar se as regras terão uma alta precisão preditiva em dados futuros. A técnica de validação cruzada é a mais usada para este fim. Para se aplicar validação cruzada em uma base de dados, separa-se a base de dados em N partes, executando-se o algoritmo de classificação N vezes. Em cada passo, $[N-1]$ seções da base de dados são usadas para treinamento do algoritmo, isto é, para descoberta das regras de classificação. Em seguida, a seção não usada no treinamento é usada para teste, ou seja, para estimar a precisão preditiva das regras descobertas. Em cada execução, uma parte diferente da base de dados é usada para teste e as demais são usadas para treinamento. Desta forma, é possível avaliar o desempenho do algoritmo classificador em dados não conhecidos durante o treinamento, o que mede sua adequação para dados futuros (BERRY

GORDON, 2004).

3.3.2 Regras de Associação

A tarefa de descoberta de regras de associação seleciona conjuntos de itens que co-ocorrem frequentemente em uma base de dados de transações. As regras descobertas são do formato $X \rightarrow Y$, onde X e Y são conjuntos não vazios de itens e $X \cap Y = \varnothing$.

Para medir a qualidade da regra, dois critérios são utilizados: o suporte e a confiança. Diz-se que uma regra $X \rightarrow Y$ tem confiança C se C % as transações da base de dados que contêm X também contêm Y. O suporte da regra é S se S% das transações contêm tanto X quanto Y. Por exemplo, após a análise de registros de empréstimo em uma biblioteca pode-se descobrir uma regra que indique que 80% dos usuários que emprestam o livro "Inteligência de Negócios" também emprestam o livro "Sistemas Inteligentes", sendo que 3% dos empréstimos envolvem ambos os itens. Neste caso, a confiança seria 80% e o suporte 3%.

Um algoritmo para descoberta de regras de associação é executado para uma base de transações quaisquer com a especificação de parâmetros para confiança mínima e suporte mínimo. O algoritmo sempre retorna todas as regras de associação entre itens que atendem os requisitos de suporte e confiança especificados. O algoritmo mais utilizado para esta tarefa é o Apriori (AGRAWAL, 1993).

Esta é uma das tarefas mais difundidas no setor de varejo, normalmente aplicada à análise de produtos comercializados com frequência na mesma transação de compra. Atualmente, há amplo uso desta tarefa para recomendações em comércio eletrônico, onde a página de um produto em geral inclui uma seção com o título "quem viu este produto, viu também:", seguida de produtos associados recomendados.

3.3.3 Agrupamento

Esta tarefa, também conhecida como *clustering*, busca agrupar itens da base de dados que tenham características em comum. O processo pode ser hierárquico ou não, dependendo da organização e da semântica dos dados. Existem diversas abordagens para a identificação dos grupos mais significativos em uma base de dados, e essas abordagens serão mais amplamente discutidas no próximo capítulo.

4 Agrupamento de Dados

Este capítulo descreve a tarefa de mineração de dados responsável pelo Agrupamento de Dados, apresentando as classes de algoritmos mais utilizadas para este fim.

Agrupamento (*clustering*) é o processo de agrupar um conjunto de objetos físicos ou abstratos em grupos de objetos similares. O objetivo do agrupamento é que objetos dentro de um grupo sejam similares ou relacionados entre si, ao mesmo tempo em que são diferentes ou não relacionados a objetos de outros grupos. Quanto maior a similaridade dentro de um grupo, e quanto maior a diferença entre os grupos, melhores são considerados os grupos (MIRKIN, 2005).

O Agrupamento é uma atividade de grande importância, que está presente nas atividades do dia-a-dia da humanidade. Desde cedo, as crianças aprendem a distinguir entre animais e plantas, ou separar figuras geométricas de acordo com formas, cores e tamanhos. As pessoas naturalmente criam grupos de amizades segundo o grau de afinidade, nível de parentesco e pela proximidade. Além das inúmeras tarefas instintivas observadas no comportamento humano, o Agrupamento está presente em diversas outras atividades e é aplicado a outras áreas do conhecimento, como: psicologia, biologia, estatística, marketing, reconhecimento de padrões, aprendizagem de máquina, mineração de dados, entre outras.

Na área de negócios, a tarefa de Agrupamento pode ser útil para uma empresa descobrir grupos distintos de clientes levando em consideração seus padrões de compra (BRITO et al., 2015). Na biologia, os cientistas podem utilizá-la para criar taxonomias (classificações hierárquicas) de seres vivos. Ainda nesta área, em estudos com DNA, a técnica é usada para categorização biológica (HENEGAR et al., 2006). Uma outra aplicação de Agrupamento é o seu uso para auxiliar usuários para uma melhor exploração dos resultados de consultas feitas na Internet, que podem resultar em milhares de páginas retornadas. Nesse caso, um algoritmo de agrupamento hierárquico pode ser utilizado para agrupar os resultados em grupos e sub-grupos separados por categorias (ALAM; SADAF, 2013).

Adicionalmente, a tarefa de Agrupamento pode ser utilizada para identificar anomalias em bases de dados (*outliers*). Neste caso, o objetivo é identificar objetos que estejam significativamente distantes de qualquer grupo. Aplicações nesse sentido incluem detecção de fraudes em cartões de crédito e monitoramento de atividades suspeitas no comércio eletrônico.

A tarefa de Agrupamento é aplicada também com frequência como uma etapa de pré-processamento para outros algoritmos de mineração de dados, como algoritmos de classificação. Neste cenário, os grupos identificados pelo algoritmo de agrupamento são considerados sugestões de classes, para que sejam mapeadas pelo algoritmo de classificação.

4.1 Requisitos de um Algoritmo de Agrupamento

Agrupamento é uma tarefa com aplicação multidisciplinar e que possui requisitos típicos para a sua utilização. A seguir, alguns desses requisitos são listados como sendo desejáveis para boas implementações de algoritmos de agrupamento (HAN; KAMBER; PEI, 2012; BERKHIN, 2002).

- **Escalabilidade**: algoritmos de agrupamento devem funcionar, adequadamente, em bases de dados de qualquer tamanho.
- **Habilidade para lidar com diferentes tipos de atributos:** bases de dados podem conter atributos numéricos, categóricos, binários, reais, entre outros.
- **Descoberta de grupos de formatos variados:** alguns algoritmos de agrupamento utilizam medidas de similaridade, como a distância euclidiana, para encontrar grupos. Esses métodos tendem a encontrar grupos esféricos. É desejável que um algoritmo consiga detectar grupos de qualquer formato, ou seja, o formato de um grupo deveria ser determinado pelos dados, em vez de ser artificialmente imposto pelo algoritmo de agrupamento.
- **Número mínimo de parâmetros de entrada:** alguns algoritmos são iniciados com parâmetros de entrada, como o número de grupos desejados. Os resultados dos algoritmos são sensíveis a esses parâmetros, o que torna complexa a tarefa de especificar valores adequados a cada parâmetro. Quanto menor o número de parâmetros, portanto, mais fácil será a obtenção dos melhores resultados possíveis do algoritmo.
- **Habilidade para lidar com dados contendo ruídos e outros defeitos:** bases de dados podem conter valores discrepantes, dados desconhecidos, errados ou incompletos. Algoritmos de agrupamento devem saber lidar com todo tipo de defeito nos dados.
- **Agrupamento incremental:** É desejável que um algoritmo possa incorporar novos dados a um Agrupamento já processado, para que a busca por novos grupos prossiga analisando os novos dados inseridos. É importante também que o bom funcionamento dos algoritmos não seja comprometido pela ordem de inserção dos novos pontos.
- **Habilidade para lidar com bases de dados com muitos atributos:** um dos maiores desafios para um algoritmo de agrupamento é que ele consiga encontrar grupos de pontos em bases de dados com grande número de dimensões, que possuam centenas ou milhares de atributos.

- **Agrupamento baseado em restrições:** aplicações do mundo real exigem que algumas restrições sejam satisfeitas para a identificação dos grupos. Por exemplo, pode ser necessário que um dos grupos inclua pelo menos a metade da base de dados.
- **Usabilidade:** os resultados de algoritmos de agrupamento devem ser compreensíveis, interpretáveis e úteis.

Entre as demais propriedades desejáveis, (BERKHIN, 2002) ainda cita a habilidade de executar em espaço de memória pré-especificado; de pausar e reiniciar; e de apresentar resultados intermediários.

4.2 Métodos de Agrupamento

A grande quantidade e diversidade de algoritmos de agrupamento existentes dificulta sua classificação, ou seja, a separação dos algoritmos quanto a uma determinada heurística empregada não é trivial. Alguns algoritmos são mistos, possuindo aspectos de abordagens distintas, o que dificulta ainda mais essa classificação.

Segundo (HAN; KAMBER; PEI, 2012), a maioria dos algoritmos de agrupamento pode ser classificada em um dos seguintes métodos: métodos de particionamento; métodos hierárquicos; métodos baseados em densidade; métodos baseados em grade; e métodos baseados em modelos.

(MIRKIN, 2005) classifica as abordagens usadas em algoritmos de agrupamento, de forma mais abrangente, em: extensões do algoritmo K-Means; algoritmos baseados em grafos; e algoritmos para descrição conceitual de clusters.

Nesta seção, descrevem-se os principais métodos identificados.

4.2.1 Métodos de Particionamento

Algoritmos de Agrupamento por particionamento buscam critérios para a divisão dos dados em grupos adequados.

Dada uma base de dados de tamanho n, um método de particionamento procura separar a base de dados em k partições, onde cada partição representa um grupo e $k \leq n$. Os dados são classificados em k grupos, que devem satisfazer os seguintes requisitos:

- Cada grupo deve conter pelo menos um ponto.
- Cada ponto deve pertencer a um único grupo.

A técnica de particionamento baseada em lógica *fuzzy* é uma exceção à segunda regra, pois neste caso um ponto pode pertencer a um grupo com maior intensidade, e a outro grupo com menor intensidade (MIRKIN, 2005).

Um método de particionamento, tipicamente, cria k partições e através da técnica de relocação iterativa vai obtendo melhores partições pela movimentação de pontos de um grupo para outro.

Este método apresenta duas principais heurísticas para a separação dos dados em partições, abordadas nas subseções a seguir.

4.2.1.1 Particionamento baseado em centróide

Algoritmos de agrupamento baseados em centróides são bastante populares, e apresentam boas soluções para bases de dados numéricas.

A técnica baseada em centroides utiliza um parâmetro de entrada k para determinar o número de grupos em que a base de dados será particionada. Os grupos obtidos ao final do processo possuem alta similaridade intra-grupo e baixa similaridade intergrupos. A medida de similaridade se baseia no valor médio dos pontos de um grupo, que se torna o centróide do grupo.

Um dos algoritmos de particionamento mais usados é o K-Means (MCQUEEN, 1967), que executa os seguintes passos:

1. Estipula-se uma quantidade de grupos que se deseja encontrar a partir de um conjunto de pontos;
2. Atribui-se, aleatoriamente, cada ponto a um determinado grupo;
3. Enquanto não for obtida a menor soma da distância entre os pontos dentro de um grupo;
    a) Calcula-se o vetor médio para todos os pontos em cada grupo;
    b) Reagrupa-se cada ponto para o grupo com centro mais próximo;

4.2.1.2 Particionamento baseado em objeto representativo

Algoritmos baseados em objetos representativos (medóide) apresentam como vantagens a simplicidade, a habilidade para tratar atributos numéricos e a resistência a ruídos. O algoritmo K-Means é sensível a ruídos porque um ponto com um valor discrepante pode distorcer a distribuição dos dados devido ao cálculo de médias entre os valores dos pontos. Uma forma de diminuir essa sensibilidade é selecionar para cada grupo um ponto representativo, no lugar de se basear nos valores médios do grupo. Cada ponto restante (não-representativo) é associado ao grupo que contém o ponto representativo mais próximo.

O método de particionamento baseia-se no princípio de minimização da soma das dissimilaridades entre cada ponto e seu ponto de referência. O processo iterativo ocorre pela análise da substituição de pontos representativos por novos pontos.

O algoritmo PAM (*Partitioning Around Medoids*) (KAUFMAN, 1990) foi um dos primeiros algoritmos que utilizou essa heurística e é descrito a seguir:

1. Especifica-se o número k de grupos que se deseja obter.
2. Aleatoriamente, escolhem-se k pontos da base de dados como pontos representativos iniciais
3. Atribui-se cada ponto restante ao grupo com o ponto representativo mais próximo
4. Seleciona-se, aleatoriamente, um ponto o_k
5. Calcula-se o custo total, S, da substituição dos pontos representativos o_j por um ponto o_k
6. $S < 0$ então o ponto representativo o_j é trocado pelo ponto o_k para formar o novo conjunto de k pontos representativos
7. Retorna-se ao passo 3 até que nenhuma alteração seja feita no conjunto de pontos representativos.

4.2.2 Métodos Hierárquicos

Os métodos hierárquicos de agrupamento organizam os pontos da base de dados em uma árvore de grupos. Esses métodos são divididos em :

- **Método hierárquico aglomerativo:** Utiliza uma estratégia *bottom-up*. Inicia criando grupos atômicos, onde cada grupo contém um único ponto. Um processo iterativo então une os grupos em grupos cada vez maiores de acordo com algum critério, até que todos os pontos encontrem-se em um único grupo.
- **Método hierárquico divisivo:** Utiliza um estratégia *top-down*. O algoritmo inicia com todos pontos em um mesmo grupo e, iterativamente, subdivide o grupo em partes cada vez menores, até que cada ponto forme seu próprio grupo.

Em ambas as abordagens, o número de grupos pode ser especificado pelo usuário como critério de término dos algoritmos. Na abordagem aglomerativa, um possível critério de união de grupos é a distância euclidiana mínima. Por exemplo, caso um ponto pertencente a um grupo A_1 e um ponto de um grupo A_2 atinjam distância mínima entre pontos de grupos diferentes, os grupos em questão devem ser unidos. No caso da estratégia divisiva,

um critério para separação dos grupos pode ser a máxima distância euclidiana entre os pontos vizinhos de um grupo.

4.2.3 Métodos Baseados em Grade

Agrupamento baseado em grade utiliza uma estrutura de dados para dividir pontos em um número finito de células que formam uma estrutura de grade na qual todas as operações de agrupamento são realizadas.

A principal vantagem do método é que as operações independem do número de itens da base de dados, dependendo, apenas, do número de células da estrutura de grade. Isto melhora, consideravelmente, a performance dos algoritmos baseados nessa heurística. Por esta razão, a técnica é usada de forma complementar em versões adaptadas de algoritmos de método particionado, hierárquico ou baseado em densidade (BUNGKOMKHUN; AUWATANAMONGKOL, 2009).

O algoritmo STING (*STatistical INformation Grid*), proposto por (WEI J. YANG, 1997) é um exemplo de algoritmo baseado em grade. Ele divide um conjunto de dados espaciais em células retangulares. Geralmente, há inúmeros níveis de células retangulares que formam uma estrutura hierárquica. Cada célula de um nível mais alto é particionada para formar o número de células do próximo nível mais baixo. Informações estatísticas que dizem respeito aos atributos em cada célula da grade, tais como: quantidade de pontos, média, valores máximos e mínimos e tipos de distribuição dos atributos, são calculadas previamente e armazenadas. Essas informações estatísticas são utilizadas no processo de consultas.

Quando os dados são carregados da base de dados, diversos parâmetros são calculados diretamente a partir dos dados, incluindo a quantidade de pontos e os valores mínimo, médio e máximo das células de mais baixo nível. O processo de consulta e resposta inicia-se em uma das camadas de células da estrutura hierárquica. Para cada célula da camada atual é calculado um intervalo de confiança que reflete as células relevantes para a consulta atual. As células irrelevantes vão sendo removidas, até que se chegue no nível mais baixo da hierarquia, onde as células que satisfazem a consulta são retornadas.

4.2.4 Métodos Baseados em Densidade

Os métodos baseados em densidade permitem descobrir grupos com qualquer tipo de formato. Esses métodos consideram grupos como regiões densas de pontos no espaço de dados, separados por regiões de baixa densidade, que geralmente representam ruídos.

Neste projeto adotou-se a abordagem baseada em densidade para especificação dos algoritmos de agrupamento possivelmente gerados. A seguir, são apresentados os

algoritmos de agrupamento baseados em densidade considerados algoritmos básicos neste trabalho.

4.2.4.1 DBSCAN: Um algoritmo baseado em densidade para a descoberta de agrupamentos em grandes bases de dados espaciais com ruído.

O algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) (ESTER et al., 1996) obtém grupos de acordo com a análise de conectividade baseada em densidade. Trata-se de um algoritmo de agrupamento baseado em densidade capaz de descobrir grupos de formatos arbitrários em bases de dados espaciais e com ruídos. Para o DBSCAN, um grupo é definido como um conjunto máximo de pontos densamente conectados.

As definições a seguir são importantes para a compreensão do algoritmo.

- A vizinhança dentro de um raio ϵ de um dado ponto é chamada de ϵ-vizinhança do ponto.
- Se a ϵ-vizinhança de um ponto contém pelo menos um número mínimo de pontos, MinPts, então o ponto é chamado de ponto núcleo.
- Dado um conjunto de dados D, diz-se que um ponto p é diretamente alcançável por densidade a partir de um ponto q, se p está dentro da ϵ-vizinhança de q e q é um ponto núcleo.
- Um ponto p é alcançável por densidade a partir de um ponto q considerando os parâmetros ϵ e MinPts e um conjunto de dados D, se existe uma cadeia de pontos $p_1, \ldots, pn$ onde $p_1 = q$ e $p_n = p$, e p_{i+1} é diretamente alcançável por densidade a partir de p_i com relação aos parâmetros ϵ e MinPts, dados $1 \leq i \leq n$ e $p_i \in D$.
- Um ponto p é conectado por densidade a um ponto q se existir um ponto $o \in D$, os quais tanto p e q são alcançáveis por densidade.

O algoritmo DBSCAN utiliza o conceito de acessibilidade de densidade e conectividade por densidade para criar grupos baseados nessa técnica.

Um grupo para o DBSCAN é um conjunto máximo de pontos densamente conectados com respeito ao conceito de acessibilidade de densidade. Todos os pontos não contidos em nenhum grupo são desconsiderados e chamados de ruídos.

4.2.4.2 Denclue: Uma proposta eficiente para o agrupamento de grandes bases de dados multimídia com ruído

O algoritmo proposto por (HINNEBURG; KEIM, 2003) tem um passo de pré-processamento em que constrói um mapa da parte relevante dos dados. Para isso, os dados

são divididos em hiper-cubos multidimensionais, sendo que apenas usam-se os hipercubos que realmente contêm dados. Estes hipercubos são ordenados de acordo com sua posição relativa, podendo desta forma ser mapeados de forma unidimensional em uma lista, e assim armazenados em uma árvore de busca, junto a informações como o número de pontos no cubo e ponteiros para estes pontos.

Em seguida, na etapa de agrupamento, o algoritmo considera apenas hipercubos muito populados ou hipercubos conectados a hipercubos muito populados. O número de pontos para que um hipercubo seja considerado muito populado é um parâmetro do algoritmo, que não deve ter valor muito baixo, de forma a evitar que em áreas pouco densas sejam formados grupos com apenas um item.

Com estes dados de entrada, o algoritmo calcula a função de densidade local de cada ponto e o gradiente local. Um procedimento de melhora iterativa (*hill-climbing*) determina para cada ponto o ponto máximo local de acordo com a função de densidade, chamado atrator de densidade. Um grupo é, então, definido para cada ponto atrator de densidade e os pontos por ele atraídos.

4.2.4.3 DBCLASD: Um algoritmo de agrupamento baseado em distribuição para mineração de grandes bases de dados espaciais

O algoritmo proposto por (XU et al., 1998) incrementa um grupo inicial enquanto a distância interna é compatível com a distribuição de distância esperada.

Pontos que ainda não pertencem ao grupo corrente são avaliados como candidatos por pesquisas regionais usando SAM (*spacial access methods*), como uma árvore R^* (XU et al., 1998).

A pesquisa retorna todos os pontos em um determinado raio de distância, não tão grande que possa prejudicar a distribuição do grupo. Caso o candidato mantenha essa distribuição ao ser incluído no grupo, este é selecionado. No entanto, candidatos não selecionados podem ser testados novamente ao final do processo. Da mesma forma, pontos atribuídos a um grupo podem mudar de grupo em um passo seguinte. Essa medida evita que a ordem de processamento dos candidatos interfira demais no resultado.

4.2.4.4 CLIQUE: agrupamento automático de sub-espaços para aplicações de mineração de dados multidimensionais

O algoritmo proposto por (AGRAWAL et al., 1993) divide-se em três etapas: identificação dos subespaços que contêm grupos; identificação dos grupos; e geração de descrições mínimas dos grupos.

Na primeira etapa, o algoritmo procura as unidades densas usando uma estratégia *bottom-up* (de baixo para cima). Inicialmente, percorre a base de dados completa, identi-

ficando unidades densas unidimensionais. Em seguida, para cada conjunto de unidades densas de dimensão $k-1$, o algoritmo gera as unidades candidatas de dimensão k.

No segundo passo, o algoritmo recebe como entrada um conjunto D de unidades densas de mesma dimensão k e produz partições de D, sendo que todas as unidades na mesma partição são conectadas, mas nenhuma unidade de uma partição é conectada a outra partição. Cada partição é considerada um grupo. Nesta etapa, o algoritmo considera as unidades densas como vértices de um grafo, e usa uma busca em profundidade para localizar os componentes conectados do grafo.

Na terceira etapa, o algoritmo produz uma cobertura para cada grupo, indicando um conjunto de regiões do sub-espaço contidas no grupo, sendo que todas as unidades do grupo estão contidas nessas regiões. O processo começa com uma busca gulosa (*greedy*) para identificar um número máximo de retângulos cobrindo o grupo e, em seguida, outra busca gulosa indica a cobertura mínima.

4.2.4.5 DHC: um método de agrupamento hierárquico baseado em densidade para séries temporais de dados de expressões genéticas

Proposto por (JIANG; PEI; ZHANG, 2003), o algoritmo DHC, em um primeiro passo, produz uma árvore de atração em que cada ponto da base de dados está conectado ao seu atrator (ponto com maior atração). Atração é definida de acordo com a Equação 4.1.

$$\frac{density(O_1)density(O_2)}{d(O_1,O_2)^{k-1}} \tag{4.1}$$

onde *density* indica a densidade do ponto, d indica a distância entre dois pontos e k indica a dimensionalidade dos pontos.

No segundo passo, a árvore de atração é sumarizada em uma árvore de densidade, onde são identificados grupos (folhas) e áreas densas (nós internos).

4.2.4.6 DESCRY: um algoritmo de agrupamento baseado em densidade para grandes bases de dados

O algoritmo proposto por (ANGIULLI; PIZZUTI; RUFFOLO., 2004) combina os métodos de partição, hierárquico e baseado em densidade, dividindo-se em quatro etapas. Na primeira etapa, retira uma amostra da base de dados usando o algoritmo Z (VITTER, 1985), que permite o tratamento eficiente de grandes bases de dados e realiza um filtro preliminar dos dados.

Na segunda etapa, o algoritmo particiona a base de dados, produzindo uma árvore do tipo $k-d$ adaptável, ou seja, uma árvore binária que representa uma subdivisão recursiva do espaço ocupado pela base de dados em sub-espaços (SAMET, 1988). Ao final

do particionamento, cada uma das M folhas da árvore está associada a, aproximadamente, o mesmo número F de pontos. O centro de gravidade dos pontos associados a uma folha chama-se meta-ponto. Os meta-pontos podem coincidir ou não com o centro geométrico da região.

A terceira etapa recebe como entrada os meta-pontos identificados na etapa anterior e executa um agrupamento hierárquico aglomerativo, para agrupar esses meta-pontos em K grupos parciais, onde K é um parâmetro do algoritmo. Os autores usaram o algoritmo single-linkage (ANGIULLI; PIZZUTI; RUFFOLO., 2004) para esta etapa e a distância euclidiana como critério de similaridade, mas ambas as escolhas são parametrizadas.

Na quarta etapa, cada ponto da base de dados original recebe um rótulo indicando o grupo associado ao meta-ponto mais próximo.

4.2.4.7 SUDEPHIC: agrupamento auto-ajustável, de particionamento e hierárquico

O algoritmo proposto por (ZHOU et al., 2004) tem dois passos principais: DEP e DEH. DEP particiona a base de dados em grades de mesmo tamanho e organiza estas grades em ordem descendente por densidade. Em seguida, cria os grupos básicos, selecionando pontos de forma ordenada, das regiões mais densas para as menos densas.

Na segunda etapa, o algoritmo DEH realiza a fusão hierárquica dos grupos básicos, criando MC (*merging clusters* – grupos intercalados). A similaridade entre os grupos é determinada pela densidade dos pontos na sobreposição. O algoritmo decide não fundir dois grupos se a diferença entre a soma das densidades dos pontos na sobreposição e cada soma de densidade dos grupos for menor que um parâmetro de similaridade γ. DEH finaliza quando todos os MC tornam-se máximos (MMC), ou seja, quando a fusão com outros grupos não tem mais como melhorar a densidade dos pontos na sobreposição.

O algoritmo pode realizar o auto-ajuste de parâmetros de duas formas. Quando o número de grupos é informado, o parâmetro γ é ajustado com base em seu histórico e número correspondente de grupos. Caso o número COUNT de grupos não seja informado, o SUDEPHIC leva em consideração a relação entre COUNT e γ para ajustar este parâmetro.

4.2.4.8 AMR: um algoritmo de agrupamento baseado em grade usando refinamento de malha adaptativo

AMR é uma técnica que começa com uma grade uniforme cobrindo todo o espaço dos dados e automaticamente refina certas regiões da grade, criando sub-grades de forma recursiva. Este processo produz uma árvore hierárquica que representa os dados como um conjunto de grades aninhadas.

O algoritmo, proposto por (LIAO; LIU; CHOUDHARY, 2004), tem duas etapas. Na primeira, constrói a árvore AMR de cima para baixo, a partir do nó que cobre o volume

completo dos dados. O algoritmo identifica a célula adequada para cada ponto, atualizando a densidade da célula. As células com densidade acima de um limite são marcadas para refinamento.

Cria-se então uma nova sub-grade a partir de todas as células contíguas marcadas. O algoritmo continua dividindo as grades filhas recursivamente e construindo a árvore correspondente até que tenha atingido a profundidade máxima da árvore ou que não existam mais células marcadas.

Em um segundo passo, o algoritmo executa o processo de agrupamento na árvore, considerando inicialmente cada folha um grupo. Em seguida analisa os nós pais, atribuindo cada ponto existente ao grupo (sub-grade) mais próximo.

O algoritmo permite ainda que se estabeleça um nível inicial para o processo de agrupamento. Neste caso, todos os nós daquele nível são considerados folhas, e os pontos abaixo do nível especificado são considerados parte do grupo correspondente ao seu nó pai naquele nível.

4.2.4.9 SNN Density - agrupamento baseado em densidade e vizinhos próximos compartilhados

Uma abordagem tradicional de agrupamento é baseada em selecionar os K vizinhos mais próximos de cada item da base de dados (ORTEGA et al., 2019). No entanto, dependendo da distribuição dos itens da base de dados, a distância entre um ponto e seu K vizinho mais próximo pode ser alta. O algoritmo proposto por (YE; LV; SUN, 2016) traz uma melhoria para a técnica tradicional, acrescentando o critério de densidade.

O algoritmo SNN Density seleciona os K vizinhos mais próximos de cada instância da base de dados. Em seguida, redefine a similaridade entre pares de objetos, considerando o número de vizinhos próximos que os dois objetos compartilham. Com base neste novo critério de similaridade, o algoritmo identifica os pontos principais da base de dados e cria grupos em volta destes pontos.

4.2.4.10 CDP - agrupamento pela descoberta de picos de densidade baseado na desigualdade de Chebyshev

O algoritmo CDP (DING et al., 2016) define centros de grupos como os elementos com densidade local superior a de seus vizinhos; e a uma maior distância a outros elementos com alta densidade local.

Inicialmente, o algoritmo normaliza as medidas de distância e densidade dos elementos. Em seguida, avalia os itens da base de dados de acordo com um índice Z_i, associado ao menor valor entre as duas medidas normalizadas.

Para identificar o limite superior para o índice, usa a desigualdade de Chebyshev, selecionando como centros de grupo os elementos $Z_i > \mu + 3 * \sigma$, onde μ é o valor esperado

de Z_i e σ é a variância.

4.3 Avaliação do número ideal de grupos

Para identificar o melhor número k de grupos em que se deve dividir uma base de dados, algumas estratégias já foram propostas, como os métodos Consensus (MONTI et al., 2003), Gap (TIBSHIRANI; WALTHER; HASTIE, 2000) e Clest (DUDOIT; FRIDLYAND, 2002).

O método CLEST, detalhado a seguir, é uma generalização de um trabalho anterior, chamado análise de replicação (BRECKENRIDGE, 1989).

O processo repete os seguintes passos para cada número possível de grupos k:

1. A base de dados é dividida em dois conjuntos: um conjunto de treinamento e um conjunto de teste.
2. O algoritmo de agrupamento é aplicado ao conjunto de treinamento, obtendo uma partição P. A cada ponto é atribuído um rótulo (classe) que indica o seu grupo.
3. Um classificador C é aplicado ao conjunto de treinamento, usando como atributo de classificação os rótulos dos grupos. Ou seja, o algoritmo de classificação é usado para gerar um modelo que identifique o grupo correto de cada ponto.
4. O algoritmo de agrupamento é então aplicado ao conjunto de teste, gerando um grupo (classe) para cada ponto.
5. Em seguida, o classificador produzido com base no conjunto de treinamento é aplicado ao conjunto de teste, para também identificar uma classe para cada ponto.
6. Os dois conjuntos de pontos rotulados obtidos por ambos são então comparados.
7. Um valor t_k de similaridade estatística para o agrupamento com k grupos é então calculado.
8. O número de grupos é estimado pela comparação de t_k com seu valor esperado segundo uma distribuição nula com $k = 1$.

5 Trabalhos Relacionados

Neste capítulo serão apresentados trabalhos relacionados, de forma a explicitar a efetiva contribuição da presente tese.

Este trabalho se baseia nas áreas de mineração de dados e computação evolutiva. Apesar do grande número de trabalhos de pesquisa baseados nestas duas áreas, é importante destacar a diferença na abordagem deste trabalho em particular.

5.1 Aplicações dos Algoritmos para Estimativa de Distribuição

Existem aplicações de EDA para diversas áreas (PELIKAN; HAUSCHILD; LOBO, 2015), como a bioinformática (ARMAÑANZAS et al., 2008), engenharia de software (SAGARNA; LOZANO, 2005), reconhecimento de imagens (BENGOETXEA et al., 2008) e problemas clássicos de otimização, como de escala de trabalho (AICKELIN, 2007), detecção de intrusão (MAZA; TOUAHRIA, 2019) e do caixeiro viajante (ROBLES P. MIGUEL, 2002). A seguir, comentam-se algumas aplicações específicas para a área de mineração de dados.

(INZA et al., 2001) aplicaram um EDA para a tarefa de seleção de atributos. Neste trabalho, a distribuição de probabilidades é modelada por redes bayesianas ou modelos probabilísticos mais simples, dependendo do número de dimensões da base de dados tratada. Nos testes comparativos, a solução baseada em EDA obteve melhores resultados, com menor custo de processamento.

(SIERRA et al., 2002) avalia a abordagem baseada em EDA para a tarefa de indução de regras. Três modelos probabilísticos de diferentes complexidades foram comparados a dois algoritmos clássicos de indução de regras, com melhores resultados em algumas das bases de dados utilizadas.

Em (OLIVEIRA et al., 2007), um EDA usa uma abordagem baseada em densidade para a tarefa de agrupamento, atingindo resultados positivos nos testes comparativos com o algoritmo DBSCAN.

(LIU et al., 2006) desenvolveram um algoritmo para agrupar dados de expressões de genéticas baseado em EDA e lógica difusa. Os resultados mostraram-se competitivos em termos de precisão, mas foram atingidos mais rapidamente pela solução proposta.

(AYODELE; MCCALL; REGNIER-COUDERT, 2017) propõem um EDA para escalonamento de projetos com restrições de recursos e múltiplos modos de processamento (MRCPSP), com bom desempenho em especial no caso de bases de maior volume.

5.2 Uso de Computação Evolutiva para Programação Automática

As aplicações citadas na seção anterior são soluções baseadas em EDA, desenvolvidas manualmente por um programador. Uma área relacionada, geralmente chamada EDP – *Estimation of Distribution Programming*, estuda o uso de EDA para geração automática de programas (YANAI; IBA, 2007).

Em geral, as soluções baseadas em EDA/EDP para programação automática modelam a estrutura do programa em forma de uma árvore explícita, e usam um modelo probabilístico como rede bayesiana para acompanhar durante o processo evolutivo a distribuição de probabilidade relacionada.

O uso de programação genética para geração automática de programas de computador foi consideravelmente mais explorado (RAMESH, 2001), por se tratar da primeira abordagem evolutiva com este foco.

Em especial, a geração automática de algoritmos de mineração de dados com uso de programação genética foi originalmente proposta por (PAPPA; FREITAS, 2004), quando os autores usaram programação genética baseada em gramática para evoluir automaticamente algoritmos de indução de regras.

5.3 Seleção automática de algoritmos de Mineração de Dados

Existem trabalhos recentes concentrados em facilitar a tarefa do usuário responsável por um processo de descoberta de conhecimento em bases de dados. Neste sentido, algumas ferramentas foram disponibilizadas para agilizar o processo de seleção do algoritmo mais adequado.

O software STATISTICA oferece a opção chamada "Data Mining Recipe"que possibilita inclui um passo a passo, em forma de receita, que guia o usuário em etapas do processo KDD, como preparação de dados e seleção do modelo. A seleção do modelo com menor taxa de erro é realizada de forma automática (NISBET; MINER; YALE, 2017).

A ferramenta AUTOWEKA usa otimização bayesiana para buscar a solução com maior precisão para a base de dados submetida, avaliando todo o conjunto de opções de algoritmos e parâmetros da amplamente utilizada ferramenta WEKA. Atualmente, AUTOWEKA suporta as tarefas de classificação e seleção de atributos (THORNTON et al., 2013).

A plataforma de serviços em nuvem Google Cloud inclui o serviço AutoML Tables, que suporta a preparação de dados, seleção de modelo, critérios de avaliação e apoio aos testes (BISONG, 2019). O serviço contempla as tarefas de classificação e regressão.

5.4 Contribuição

O presente trabalho propõe o uso de um EDA chamado Autoclustering para geração automática de algoritmos de agrupamento, usando um grafo como estrutura de dados auxiliar para garantir a produção de algoritmos de agrupamento válidos.

É importante destacar a diferença da presente proposta em relação aos trabalhos citados de computação evolutiva aplicada ao agrupamento de dados (FREITAS, 2007) (FREITAS, 2002) (SAMMUT; WEBB, 2017).

Nos demais trabalhos, a saída do algoritmo é um conjunto de grupos, ou seja, um agrupamento proposto para a base de dados submetida.

No caso do Autoclustering, a saída do algoritmo é um novo algoritmo de agrupamento, projetado especificamente para a base de dados submetida, incluindo a especificação dos blocos de código a serem executados, a medida de distância adotada, e todos os valores de parâmetros relacionados.

Da mesma forma, o projeto Autoclustering se diferencia das ferramentas para geração automática de modelos como AutoWeka e Data Mining Recipe por dois motivos. Do ponto de vista do objetivo do usuário final, as ferramentas citadas não contemplam a tarefa de mineração de dados de agrupamento, que é o foco do presente trabalho. Do ponto de vista da abordagem utilizada, as outras ferramentas buscam a melhor opção entre algoritmos existentes, não produzindo algoritmos potencialmente originais como é o caso do Autoclustering.

6 Um Algoritmo para Geração Automática de Algoritmos de Agrupamento

Este capítulo descreve o algoritmo Autoclustering, um EDA para geração automática de algoritmos de agrupamento.

Para atingir o objetivo de construir automaticamente um algoritmo de mineração de dados aplicado à tarefa de agrupamento, foi desenvolvido um algoritmo evolutivo, usando a abordagem de Estimativa de Distribuição.

Para simplificar o EDA e aumentar as chances de que o algoritmo de agrupamento produzido seja eficaz, o alfabeto do EDA é formado por blocos de construção (*building blocks*) tipicamente existentes em algoritmos de agrupamento baseado em densidade. Esses blocos são combinados de diversas formas pelo EDA, produzindo novos algoritmos de agrupamento – potencialmente livres das limitações associadas aos algoritmos atuais de agrupamento, os quais foram manualmente projetados.

Adicionalmente, para garantir que estes blocos sejam conectados em uma sequência adequada, que realmente constitua um algoritmo de agrupamento, o EDA tem uma estrutura auxiliar que determina apenas sequências validadas de blocos de código.

A estrutura auxiliar utilizada é um grafo acíclico direcionado. O grafo especifica a estrutura genérica dos algoritmos de agrupamento. Assim, após a seleção de cada nodo do grafo, o EDA tem um conjunto limitado de opções a selecionar em seguida - apenas as opções válidas.

Para definir os diferentes blocos que podem fazer parte de um algoritmo de agrupamento, foi realizada uma ampla revisão dos algoritmos existentes, especificamente algoritmos de agrupamento baseados em densidade. No entanto, muitos dos algoritmos recentemente propostos apresentam apenas pequenas contribuições em relação à algoritmos anteriormente apresentados.

Foi necessário, portanto, identificar, entre os diversos algoritmos de agrupamento baseados em densidade, um pequeno conjunto de algoritmos básicos. Os algoritmos básicos identificados, listados na subseção 4.2.4, são aqueles que realmente apresentam uma abordagem diferenciada para definir os agrupamentos baseados em densidade. A primeira versão do Autoclustering contava com oito algoritmos básicos (MEIGUINS et al., 2012). Posteriormente, os algoritmos CDP (DING Z. CHEN; ZHAN, 2016) e SNN (YE; LV; SUN, 2016) foram adicionados.

O processo de seleção dos algoritmos se baseou em pesquisa pelos termos *"Density-*

based Clustering", "Density-based Algorithm" em repositórios da ACM e IEEE. Após a revisão de mais de mil artigos que continham termos relacionados a agrupamento baseado em densidade, foi possível perceber que muitos artigos relatavam uma extensão ou melhoria de um algoritmo anteriormente proposto. Por exemplo, os algoritmos OPTICS (ANKERST et al., 1999) e GDBSCAN (SANDER; ESTER; KRIEGER, 2020) são extensões do algoritmo DBSCAN. Apesar de serem contribuições relevantes, não concretizam uma ideia significativamente diferenciada de como se obter agrupamentos por densidade. Por esta razão, foi reduzido o número de algoritmos com real potencial para fazer parte do projeto.

Após a seleção dos algoritmos básicos, estes passaram por detalhada análise para que os blocos de construção fossem extraídos e generalizados. O objetivo deste passo foi determinar os componentes principais dos algoritmos de agrupamento selecionados. Observou-se, como esperado, que os algoritmos básicos tinham alguns procedimentos comuns.

Finalmente, estes blocos de construção foram ligados em todas as sequências válidas possíveis. A estrutura formada, representada na forma de um grafo, é a estrutura auxiliar usada para guiar o EDA na construção das populações temporárias. O grafo de procedimentos produzido é apresentado na Figura 3, da seção 6.3.

É importante destacar que, apesar da pesquisa realizada priorizar algoritmos baseados em densidade, as combinações possíveis de blocos de construção permitem a geração de algoritmos de agrupamento com abordagens diferentes, por exemplo, um algoritmo de agrupamento baseado em grade. Todos os algoritmos produzidos, no entanto, são algoritmos de agrupamento válidos.

Nas próximas seções serão apresentados os blocos de construção selecionados a partir dos algoritmos básicos de agrupamento baseado em densidade (subseção 4.2.4); o grafo montado como estrutura auxiliar do EDA; e, finalmente, a especificação do EDA para geração automática de algoritmos de agrupamento.

6.1 Seleção dos Blocos de Construção

A partir da análise do conjunto de algoritmos básicos para agrupamento baseado em densidade (subseção 4.2.4), foi possível identificar um conjunto de blocos de construção. Os blocos de construção selecionados são procedimentos contidos nos algoritmos básicos, que podem ser utilizados de forma independente, antes ou após outros procedimentos do mesmo ou de outros algoritmos. Tais blocos de construção são, portanto, primitivas básicas de algoritmos de agrupamento baseado em densidade existentes.

Para determinar o valor dos parâmetros em cada indivíduo, foi necessário, para

cada procedimento, especificar os valores máximo e mínimo, de forma que o Autoclustering possa escolher valores de parâmetros de forma automática, dentro da faixa de valores pertinentes a cada parâmetro necessário. O objetivo é que o usuário final não precise informar, manualmente, valores de parâmetros, pois são avaliados de forma automática e evoluídos em conjunto com a especificação do algoritmo.

A faixa de valores é configurada em um arquivo XML que serve como entrada do algoritmo. Em alguns casos, a faixa de valores dos parâmetros adotada durante os testes foi sugerida pelos autores originais. Em outros casos, os valores foram aplicados após um período de testes de funcionamento dos algoritmos, sempre incluindo, no intervalo selecionado, os valores explicitamente citados pelos autores originais.

O algoritmo divide o intervalo definido de valores, para cada parâmetro, em um determinado número de fatias. Nos testes, adotou-se sempre 10 fatias. A cada vez que um procedimento é selecionado, aleatoriamente, pelo algoritmo, também seleciona-se, aleatoriamente, os valores de parâmetros necessários.

Por exemplo, durante os testes executados, o parâmetro NUMPTS (número de pontos) do algoritmo DBSCAN permitiu valores em um intervalo de 1 a 20. Com uso de 10 fatias, foram avaliados os valores 1, 3, 5, 7, 9, 11, 13, 15, 16 e 19 para este parâmetro.

A seguir, descrevem-se os blocos de construção que foram destacados dos algoritmos anteriormente apresentados. A lista inclui, para cada procedimento, uma breve descrição, as entradas e saída previstas, bem como os parâmetros necessários.

6.1.1 CandidatesByDistance - candidatos por distância

Procedimento que identifica grupos candidatos cuja distância ao centro do grupo é menor ou igual a uma distância máxima.

- Entrada: um conjunto de pontos, o tipo de distância, o número mínimo de pontos (valores de 1 a 20), a distância máxima (valores de 1 a 50)
- Saída: um conjunto de grupos candidatos
- Assinatura do método:

```
private List<Group> candidatesByDistance (
Instances instances, int min_pts, double maxDist,
DistanceType distance)
```

6.1.2 ClustersByConnectiveness - agrupamento por conectividade

Verifica se todos os pontos do grupo estão conectados por densidade, como no algoritmo DBSCAN (ESTER et al., 1996). Caso o grupo seja válido, adiciona o grupo à

lista de grupos.

- Entrada: um conjunto de grupos, o limiar de distância (valores de 1 a 50)
- Saída: um conjunto de grupos.
- Assinatura do método:

```
private List<Group> clustersByConnectiveness (double limiar,
List<Group> groups, DistanceType distance)
```

6.1.3 CandidatesByNPts - seleciona candidatos por número de pontos

Procedimento que identifica grupos candidatos. Se o ponto ainda não faz parte de um grupo, cria um grupo com este ponto e um número determinado de pontos mais próximos. Em seguida, cria uma lista de candidatos a entrar no grupo, com pontos próximos aos já selecionados, com raio de busca calculado como no algoritmo DBCLASD (XU et al., 1998).

- Entrada: a matriz de distância entre os pontos, o número de pontos (valores de 1 a 20)
- Saída: um conjunto de grupos.
- Assinatura do método:

```
private List<Group> candidatesByNpts (Map<Instance,
SortedList<Distance<Instance>>> distances, int pts)
```

6.1.4 ClustersByDistribution - agrupamento por distribuição

Para cada ponto candidato, adiciona o ponto ao grupo e verifica se a distribuição de densidade no grupo é aceitável, como no algoritmo DBCLASD (XU et al., 1998). Se a distribuição é aceitável, o ponto é mantido no grupo, caso contrário é removido.

- Entrada: um conjunto de grupos, os limites, um conjunto de pontos candidatos, o tipo de distância.
- Saída: um conjunto de grupos
- Assinatura do método:

```
private List<Group> clustersByDistribution
(List<Group> groups, double[] lowerBounds,
Instances instances, DistanceType distance)
```

6.1.5 AttractionTree - cria árvore de atração

Cria uma árvore conectando cada ponto a seu atrator, com a definição de atração como no algoritmo DHC (JIANG; PEI; ZHANG, 2003).

- Entrada: conjunto de pontos, limite de similaridade (valores de 1 a 15)
- Saída: árvore de atração
- Assinatura do método:

```
private List<NodoAtractor> attractionTree
(Instances instances, double sigma)
```

6.1.6 DensityTree - cria árvore de densidade

Transforma a árvore de atração em uma árvore de densidade (JIANG; PEI; ZHANG, 2003), em que cada folha é um grupo. Armazena os grupos identificados na lista de grupos.

- Entrada: árvore de atração, número de pontos (valores de 5 a 20), limite de similaridade (valores de 1 a 15)
- Saída: lista de grupos.
- Assinatura do método:

```
private List<Group> densityTree(List<NodoAtractor> ns,
int minPts,double sigma)
```

6.1.7 ASH - cria histogramas ASH

Constrói histogramas ASH (average shifted histograms) para identificar áreas densas nos dados, como no algoritmo DENCLUE (HINNEBURG; KEIM, 2003).

- Entrada: conjunto de pontos, limites de valores superiores,limites de valores inferiores, sigma (valores de 1 a 60), epsilon (valores de 1 a 20), tipo de distância
- Saída: um hiper-espaço contendo os pontos das áreas densas.
- Assinatura do método:

```
public HyperSpace(Instances instances,
double[] upper_bounds, double[] lower_bounds,
double sigma, double epsilon, DistanceType distance)
```

6.1.8 ClustersByAttractor - agrupamento por atratores

Identifica o ponto atrator de densidade, usando um procedimento *hill-climbing* guiado pelo gradiente da função de densidade como no algoritmo DENCLUE (HINNEBURG; KEIM, 2003). Acrescenta à lista de grupos cada conjunto formado por um ponto-atrator e seus pontos relacionados.

- Entrada: conjunto de instâncias, sigma (valores de 1 a 60), epsilon (valores de 1 a 20), tipo de distância
- Saída: lista de grupos.
- Assinatura do método:

```
private List<Group> clustersByAtrractor(List<Instance>
spatialInstances, double sigma, double epsilon,
DistanceType distance)
```

6.1.9 DenseAreas - seleciona áreas densas

Identifica unidades densas, iterativamente, a partir de unidades unidimensionais, no estilo do algoritmo Apriori, como especificado no algoritmo CLIQUE (AGRAWAL et al., 1993).

- Entrada: conjunto de pontos, atributo (dimensão), número de partições (valores de 3 a 40), limite (valores de 0,0000001 a 0,3)
- Saída: conjunto de unidades densas
- Assinatura do método:

```
private Subspace identifyDenseUnits(Instances inst,
Attribute att, int partitions, float threshold)
```

6.1.10 ClustersByPartition - agrupamento por partições

Produz partições conectadas a partir das unidades densas, sendo que todas as unidades na mesma partição são conectadas, mas nenhuma unidade de uma partição é conectada a outra partição (AGRAWAL et al., 1993).

- Entrada: conjunto de unidades densas, número de indivíduos, limite (valores de 0,0000001 a 0,3), número de atributos)
- Saída: conjunto de grupos

- Assinatura do método:

```
private List<Group> clusterByPartition(Subspace[] subspaces,
int population, float threshold, int dimensions)
```

6.1.11 EquallySizedGrid - cria grade de tamanho fixo

Produz uma divisão da base de dados em uma grade com células de tamanho fixo parametrizado (LIAO; LIU; CHOUDHARY, 2004).

- Entrada: conjunto de pontos, limite superior, limite inferior, slices (valores de 25 a 30)
- Saída: grade de células, cada célula com informação sobre sua densidade.
- Assinatura do método:

```
private Grid equallySizedGrid (Instances instances,
double[] lowerBounds, double[] upperBounds, float radius)
```

6.1.12 AdaptableKDTree - criação de árvore KD Adaptável

Produz uma divisão da base de dados em uma árvore binária em que cada folha está associada a, aproximadamente, o mesmo número de pontos (ANGIULLI; PIZZUTI; RUFFOLO., 2004).

- Entrada: conjunto de pontos, parâmetro de densidade (valores de 1 a 60), parâmetro de profundidade (valores de 2 a 20) .
- Saída: nó inicial da árvore KD adaptável.
- Assinatura do método:

```
public Node kdtree(List<Instance> instances, int depth,
int density)
```

6.1.13 AMRTree - criação de árvore AMR

Cria uma árvore produzindo, recursivamente, subgrades a partir de cada conjunto de células contíguas com densidade acima de um valor parametrizado, como no algoritmo AMR (LIAO; LIU; CHOUDHARY, 2004).

- Entrada: conjunto de pontos, nodo inicial da árvore, o parâmetro slices (valores de 5 a 45), o parâmetro de densidade (valores de 1 a 20)

- Saída: árvore AMR
- Assinatura do método:

```
public Tree amrTree(Instances instances,Node node,
int slices, float density)
```

6.1.14 ClustersAMR - agrupamento AMR

Considera cada folha da árvore AMR de entrada um grupo e analisa os nós, atribuindo cada ponto no nodo pai ao grupo mais próximo (LIAO; LIU; CHOUDHARY, 2004).

- Entrada: árvore AMR, parâmetros density (valores de 1 a 20), parâmetro lambda (valores de 1 a 25)
- Saída: lista de grupos
- Assinatura do método:

```
public List<Group>  amrCluster(Node node,
DistanceType distance, int density, int lambda)
```

6.1.15 MergeByOverlap - aglomera grupos por sobreposição

Realiza a fusão hierárquica, determinando a similaridade entre os grupos pela densidade dos pontos na sobreposição, como no algoritmo DEH (ZHOU et al., 2004).

- Entrada: lista de grupos, grade de células, parâmetro raio (valores de 2 a 50)
- Saída: lista de grupos
- Assinatura do método:

```
public List<Group> merge (List<Group> result)
```

6.1.16 MergeByDistance - aglomera grupos mais próximos

Realiza a fusão dos grupos mais próximos, até atingir o número parametrizado K de grupos (ANGIULLI; PIZZUTI; RUFFOLO, 2004).

- Entrada: lista de grupos, parâmetro K (valores de 2 a 20)
- Saída: lista de K grupos.

- Assinatura do método:

```
public List<Group> merge (List<Group> result)
```

6.1.17 CandidatesBySNN - Candidatos por vizinhos compartilhados

Produz uma matriz de similaridade para todos os vizinhos compartilhados de cada ponto, mantendo na matriz apenas os K vizinhos mais similares em cada caso (YE; LV; SUN, 2016).

- Entrada: pontos da base de dados, parâmetro threshold
- Saída: matriz de similaridade.
- Parâmetro: SNN_THRESHOLD (min = 10, max = 50)
- Assinatura do método:

```
public CRSMatrix candidatesBySNN (Instances
instances, int threshold)
```

6.1.18 SNNbyConnectiveness - Agrupa por conectividade dos vizinhos compartilhados

Identifica grupos a partir da matriz de similaridade, usando como critério de conectividade a densidade SNN definida em (YE; LV; SUN, 2016). Identifica pontos como núcleo de grupos e descarta pontos classificados como ruído.

- Entrada: pontos da base, matriz de similaridade, parâmetro threshold (valores de 10 a 50), parâmetro distancia (valores de 1 a 100), parâmetro minPts (valores de 2 a 30)
- Saída: grupos
- Parâmetro: SNN_THRESHOLD (min = 10, max = 50)
- Assinatura do método:

```
private List<Group> snnbyConnectiveness(Instances instances,
CRSMatrix sharedNeighbors, int threshold, float distancia,
int minPts)
```

6.1.19 DistanceInformation - Informação normalizada sobre distância e densidade

Calcula e normaliza a distância entre os pontos e a densidade local (DING et al., 2016).

- Entrada: pontos da base de dados, parâmetro percent (valores de 1 a 99).
- Saída: matriz de distância e densidade
- Assinatura do método:

```
private DistanceMatrix normalize (Instances instances,
int percent)
```

6.1.20 ClusteringByDensityPeak - Agrupamento por Picos de Densidade

Calcula o limite superior e identifica os centros de cada grupo. Associa em seguida cada um dos elementos remanescentes aos grupos (DING et al., 2016).

- Entrada: matriz de distância e densidade.
- Saída: grupos
- Assinatura do método:

```
private List<Group> clusteringByDensityPeak (DistanceMatrix
nearestneighbor)
```

6.2 Codificação do Indivíduo

Um indivíduo para o EDA proposto é um algoritmo de agrupamento. O algoritmo deve ser formado pelos blocos de construção identificados, em uma sequência que seja válida.

Os blocos de construção relacionados na seção anterior conectam-se para formar um conjunto de sequências válidas de procedimentos para algoritmos de agrupamento .

Como destacado na introdução deste capítulo, apesar da pesquisa ter priorizado algoritmos de agrupamento baseado em densidade, existem possíveis combinações de procedimentos que não envolveriam a abordagem baseada em densidade. Por exemplo, um algoritmo que inicia criando uma grade de tamanho fixo (procedimento equallySizedGrid) e em seguida realiza a intercalação por distância (procedimento mergeByDistance) é um algoritmo de agrupamento válido, mas é baseado em grade, não em densidade. Partes de

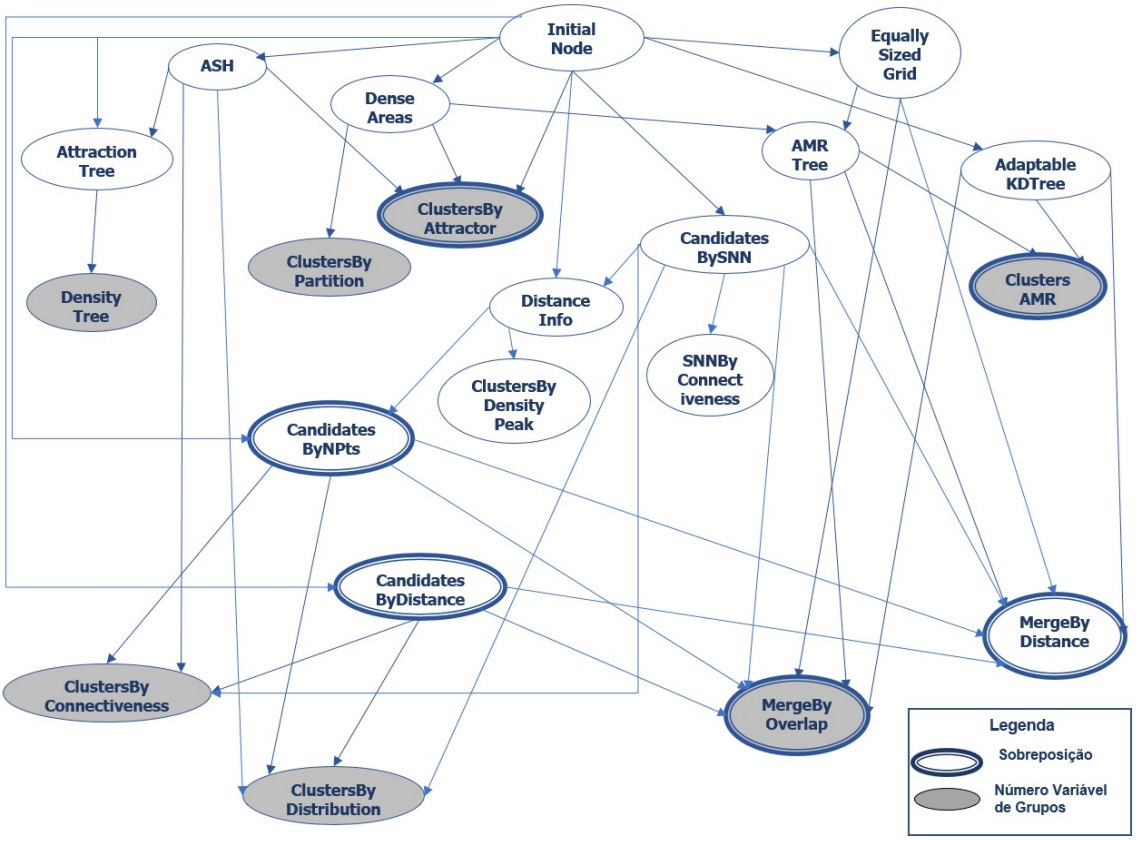
algoritmos baseados em densidade, ao serem combinadas podem produzir algoritmos que não se caracterizam pela abordagem de densidade.

Para especificar quais sequências de procedimentos são válidas, o EDA usa uma estrutura de dados auxiliar: um grafo acíclico direcionado (DAG), em que cada nodo está associado a um dos procedimento principais. O grafo auxiliar é especificado na próxima seção.

6.3 Grafo de Procedimentos Básicos de Agrupamento

O grafo acíclico direcionado da Figura 3 representa as sequências válidas de blocos de construção, servindo como estrutura auxiliar do EDA na criação de indivíduos.

Figura 3. Grafo de Procedimentos.



Fonte: Autora, 2019

Cada nodo representa um procedimento básico para construção de algoritmos de agrupamento. Cada aresta direcionada representa uma conexão sequencial entre dois blocos de construção, ou seja, o procedimento para o qual a aresta aponta deve ser executado logo após o procedimento na origem da aresta.

Cada aresta tem um peso, que representa a probabilidade do procedimento para o qual a aresta aponta ser escolhido durante a construção de um algoritmo (indivíduo).

A soma dos pesos (probabilidades) das arestas saindo de um nodo é sempre igual a 1 (um), mas as probabilidades individuais das arestas são atualizadas a cada geração do EDA, de acordo com a avaliação dos indivíduos contendo aquela aresta.

Cada indivíduo da população é um algoritmo de agrupamento. Um indivíduo é representado por um caminho no grafo de procedimentos do nodo Inicio até um nodo terminal no grafo, ou seja, um nodo de onde aresta alguma sai.

Mais precisamente, um caminho no grafo de procedimentos consiste de uma sequência de nodos n_1 ,..., n_k, onde n_1 é o nodo Inicio; n_i, $i = 2,\ldots,k$ é um nodo sucessor do nodo n_{i-1} (ou seja, há uma aresta apontando de n_{i-1} para n_i), e n_k é um nodo terminal.

A Figura 3 apresenta o grafo de procedimentos especificado com legendas para representar suas diferentes formas de execução. Os blocos sombreados representam procedimentos que podem suportar número opcional de grupos. Isto quer dizer que há duas versões de procedimento para cada bloco sombreado, sendo que um exige um parâmetro com um número específico de grupos a ser produzido, e outro não exige este parâmetro, selecionando o número de grupos de forma automática.

Os blocos com bordas duplas representam procedimentos que podem permitir a geração de sobreposição entre os grupos. Neste caso, uma versão dos procedimentos permite que um item faça parte de mais de um grupo, e outra versão trata as sobreposições forçando cada item a participar de apenas um grupo.

Todos os procedimentos previstos podem funcionar, além disso, de oito formas diferentes dependendo da medida de distância adotada para comparação entre itens de dado. As medidas de distância suportadas são (HAN; KAMBER; PEI, 2012): Euclidiana, Manhattan, Chebychev, Minkowski, Canberra, Braycurtis, Correlation Coefficient e Angular Separation.

O grafo permite a produção, portanto, de 31 caminhos diferentes, formando 1590 possíveis combinações de bloco, ao considerarmos como blocos distintos os casos em que estes podem suportar ou não um número opcional de grupos ou sobreposição.

Quando se leva em consideração as diferentes medidas de distância, há 466.632 possibilidades de algoritmos diferentes.

E quando consideramos todas as possíveis variações de sequencia de procedimentos e seus parâmetros automaticamente selecionados, há 1.235.578.026 possíveis soluções diferentes que podem ser exploradas pelo gerador de programas para cada base de dados.

Um aspecto adicional a considerar é que a estrutura de dados usada em um procedimento nem sempre corresponde de forma exata à estrutura necessária para o

próximo procedimento representado no grafo. Foi necessário, portanto, desenvolver versões adequadas de alguns procedimentos para permitir sua compatibilidade com um maior número de procedimentos complementares. Por exemplo, o procedimento MergeByDistance pode ser precedido de diversos outros procedimentos, que possuem parâmetros de saída diferentes, como uma lista de candidatos no caso do candidatesByDistance ou uma árvore como o AMRTree. Portanto, o procedimento mergeByDistance foi codificado com três variações, prevendo uma árvore no formato do procedimento AdaptableKDTree, uma árvore no formato do procedimento AMRTree, ou uma lista de grupos como nos demais casos.

6.4 Avaliação de um Indivíduo

Cada indivíduo produzido pelo EDA é um algoritmo de agrupamento. A função de avaliação (*fitness*) de um indivíduo deve usar, portanto, um método de avaliação de algoritmos de agrupamento.

Este tipo de avaliação não é trivial, pois a tarefa de agrupamento, de caráter não supervisionado, produz uma descrição ou organização da base de dados, e apesar das diversas medidas objetivas de avaliação propostas (MIRKIN, 2005), ainda há alto grau de subjetividade na avaliação dos resultados.

Algoritmos de classificação, por outro lado, têm uma avaliação objetiva eficiente pelo método de validação cruzada, que separa a base de dados em N partes, considerando em cada passo uma das partições como conjunto de teste e as demais como conjunto de treinamento. Assim, quanto mais adequadas para o conjunto de teste forem as regras identificadas no conjunto de treinamento, melhor é considerado o desempenho do algoritmo para aquela base de dados.

O resultado de um algoritmo de agrupamento é uma lista de pontos em que cada ponto é associado a um grupo ou, em alguns casos, nenhum (quando o ponto é considerado ruído). Este resultado é mais dificilmente validado que um modelo de classificação e, normalmente, um usuário com domínio do problema é o responsável por determinar se uma solução em particular é melhor que qualquer outra.

Para avaliar de forma objetiva os algoritmos de agrupamento produzidos pelo EDA, o que se propõe neste trabalho é usar o processo de avaliação dos algoritmos de classificação.

O método Clest, descrito na seção 4.3, foi definido como uma estratégia para definição do número de grupos em uma base de dados. Neste trabalho, no entanto, foi adotada uma adaptação do método Clest para avaliação dos grupos identificados, no lugar da avaliação do melhor número de grupos. Para este objetivo, que não depende

do número de grupos produzidos, os últimos dois passos do método são desnecessários. Portanto, usa-se apenas uma iteração do método, obtendo-se pela acurácia do algoritmo de classificação uma medida de qualidade do agrupamento produzido.

O método Clest adaptado para o Autoclustering baseia-se nos seguintes passos.

1. O algoritmo de agrupamento é aplicado à base de dados. A cada ponto é atribuído um rótulo (classe) que indica o seu grupo.
2. O classificador J48 da ferramenta WEKA (WITTEN et al., 2016) é aplicado à base de dados, usando como atributo de classificação os rótulos dos grupos. Ou seja, o algoritmo de classificação é usado para gerar um modelo que identifique o grupo correto de cada ponto.
3. Usando o método de validação cruzada, mede-se a raiz do erro quadrático médio (RMSE, do inglês *root mean square error*) do modelo de classificação produzido pelo algoritmo J48.
4. A medida de avaliação do algoritmo de agrupamento é dada por $(1 - RMSE) * 100$.

Desta forma, é possível medir, de forma objetiva, a qualidade do resultado de um algoritmo de agrupamento para uma base de dados. Considera-se como medida de qualidade do algoritmo de agrupamento o percentual de acertos no conjunto de teste do classificador produzido com base no conjunto de treinamento, usando-se o algoritmo de agrupamento avaliado para gerar os rótulos de grupo, de forma isolada, para cada um dos conjuntos.

A função de avaliação do EDA atribui, portanto, uma medida de desempenho para o algoritmo de agrupamento de acordo com o resultado da aplicação do método Clest.

6.5 Especificação do EDA - algoritmo Autoclustering

Um importante aspecto da especificação de um EDA é a forma pela qual a população será atualizada a cada geração. O método PBIL (*Population Based Incremental Learning*), proposto por (BALUJA; CARUANA, 1995), representa a população por um vetor de probabilidades(Equação 6.1).

$$p_l(x) = (p_l(x_1), ..., p_l(x_i), ..., p_l(x_n)) \tag{6.1}$$

onde $p_l(x_i)$ é a probabilidade de se obter um valor 1 no i-ésimo componente da população de indivíduos na população l.

A cada geração, usando o vetor de probabilidades, o algoritmo gera M indivíduos e os avalia, selecionando então os N melhores ($N < M$). Os indivíduos selecionados são usados para atualizar o vetor de probabilidades com a Equação 6.2. A nova probabilidade $p_i(x)$ de um item do vetor é dada pela probabilidade anterior $p_i(x)'$, pela nova probabilidade $p_i(x)''$ e por um parâmetro α. Este parâmetro varia de 0 a 1 e determina quanto o novo valor de probabilidade será influenciado pelo valor anterior. Nos testes, adotou-se o valor 0.5.

$$p_i(x) = (1 - \alpha)p_i(x)' + \alpha p_i(x)'' \tag{6.2}$$

Em um EDA, ao se adotar o método PBIL, usa-se, normalmente, um vetor de probabilidades. Para atender as restrições quanto à geração de algoritmos válidos, no entanto, a estrutura do EDA precisou ser adaptada para usar, no lugar de um vetor de probabilidades, um grafo de probabilidades - o DAG de procedimentos da Figura 3.

Com a evolução do EDA, as probabilidades associadas a cada aresta do grafo são atualizadas de acordo com a avaliação dos indivíduos contendo esta aresta. O pseudo-código do procedimento que atualiza as probabilidades associadas a cada aresta do grafo de procedimentos é apresentado na Figura 4.

Usando o procedimento da Figura 4, a evolução do EDA é guiada pelo grafo de procedimentos. Quanto melhores os indivíduos usando uma determinada aresta, ou seja, quanto melhor o desempenho dos algoritmos usando o procedimento associado, maiores serão as chances desta aresta ser selecionada.

É importante ressaltar que, para se usar a abordagem padrão em EDA, a probabilidade de cada aresta seria atualizada de acordo com a proporção de indivíduos selecionados que a utilizassem. Este EDA, como especificado, tem uma abordagem diferente. A probabilidade associada à aresta depende da avaliação (*fitness*) dos indivíduos selecionados usando esta aresta. Este critério reforça a opção pela qualidade dos indivíduos produzidos, no lugar da simples probabilidade de seleção.

No restante dos passos, o EDA proposto segue a execução padrão para algoritmos baseados em estimativa de distribuição. Em resumo, os seguintes passos formam a execução do EDA para geração de algoritmos de agrupamento, como ilustrado na Figura 5.

1. Produz uma população contendo N algoritmos de agrupamento. Cada algoritmo (indivíduo) representa um caminho possível no grafo de procedimentos.

2. Usando a adaptação proposta do método CLEST para avaliação de algoritmos de agrupamento, atribui a cada indivíduo uma medida de avaliação adequada (*fitness*).

3. Seleciona por torneio um número $m < N$ de indivíduos.

Figura 4. Procedimento que atualiza as probabilidades do grafo

Pseudo-código – atualizaGrafoProbabilidades

Parâmetros

nodo – Um nodo em um grafo acíclico direcionado de procedimentos básicos de algoritmos baseados em densidade.

individuosSelecionados– um subconjunto de indivíduos da população corrente selecionado por torneio

atualizaGrafoProbabilidades (nodo, individuosSelecionados)

```
1.   Para cada aresta saindo do nodo corrente
2.   {
3.     probAresta = α ( avaliação média dos indivíduos selecionados
       que usam esta aresta)  +  (1 – α) probAresta;
4.     atualizaGrafoProbabilidades(nodo alvo da aresta corrente,
       individuosSelecionados);
5.   }
6.   totalProb = soma de probAresta para todas as arestas saindo do
     nodo corrente;
7.   Para cada aresta saindo do nodo corrente
8.   {
9.     probAresta = probAresta / totalProb;
10.  }
```

Fonte: Autora, 2019

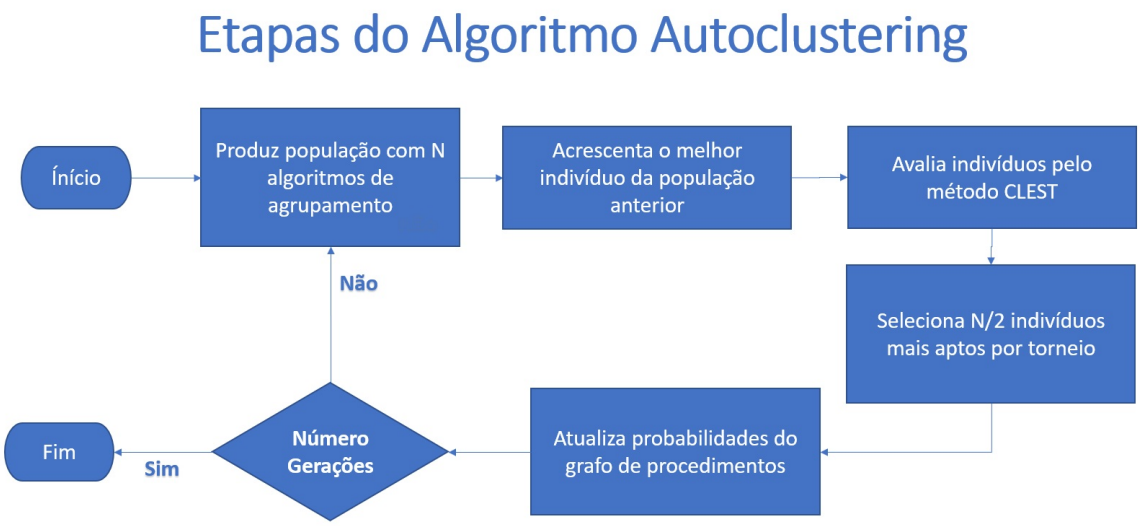
4. Com base na avaliação dos indivíduos, o procedimento da Figura 4 recebe como parâmetro o nodo raiz do grafo de procedimentos e os indivíduos selecionados no passo anterior e atualiza as probabilidades de cada aresta do grafo de procedimentos.

5. Usando o grafo de procedimentos atualizado, produz uma nova população com N indivíduos

6. Aplicando a técnica de elitismo, acrescenta o melhor indivíduo da geração anterior à nova população, caso seja superior aos novos indivíduos produzidos.

7. Retorna ao Item 2

Figura 5. Etapas do Autoclustering



Fonte: Autora, 2019

7 Experimentos e Resultados

Neste capítulo serão discutidos os resultados obtidos após a aplicação dos algoritmos de agrupamento construídos, automaticamente, pelo Autoclustering a um conjunto de bases de dados selecionadas.

De forma geral, analisar resultados de algoritmos evolutivos é uma tarefa complexa. Durante o projeto, buscou-se avaliar diferentes técnicas visuais que facilitassem este processo de análise (MEIGUINS et al., 2019; SANTOS et al., 2019).

7.1 Descrição do experimento

O algoritmo Autoclustering foi aplicado a sete bases de dados de domínio público do repositório UCI (DUA; GRAFF, 2017). As bases utilizadas têm as características em destaque no Quadro 1 (CORTEZ et al., 2009). Em geral, as bases de dados são voltadas à tarefa de classificação, mas o atributo classe, quando existente, foi eliminado das bases antes de qualquer processamento do Autoclustering.

Quadro 1. As bases de dados utilizadas

Bases	**Descrição**	**Atributos**	**Instâncias**
Cleveland	Diagnóstico de doenças cardíacas	14	303
Glass	Identificação de tipos de vidro	10	214
Pima	Diagnóstico de Diabetes	8	768
Bupa	Diagnóstico de doenças no fígado	7	345
Wine	Amostras de vinhos tintos portugueses	12	1600
Abalone	Características de um tipo de molusco	8	4177
Yeast	Localização de proteínas nas células	8	1484

Fonte: (DUA; GRAFF, 2017)

Para cada base de dados, o algoritmo foi executado 30 vezes, com 500 gerações a cada execução, evoluindo uma população formada por 50 indivíduos.

Nas seções a seguir, serão realizadas diversas análises dos algoritmos produzidos pelo Autoclustering. Nestas análises, foram consideradas apenas os blocos distintos, não sendo feita a diferenciação pelas opções e parâmetros disponíveis, como o critério de distância e o uso de sobreposição de grupos.

O procedimento clustersByAttractors, por exemplo, pode representar 4 caminhos diferentes no grafo, pois permite funcionamento com ou sem sobreposição, bem como com número fixo ou opcional de grupos. Cada um desses procedimentos pode ainda funcionar com oito medidas diferentes de distância e 10 valores possíveis para cada parâmetro:

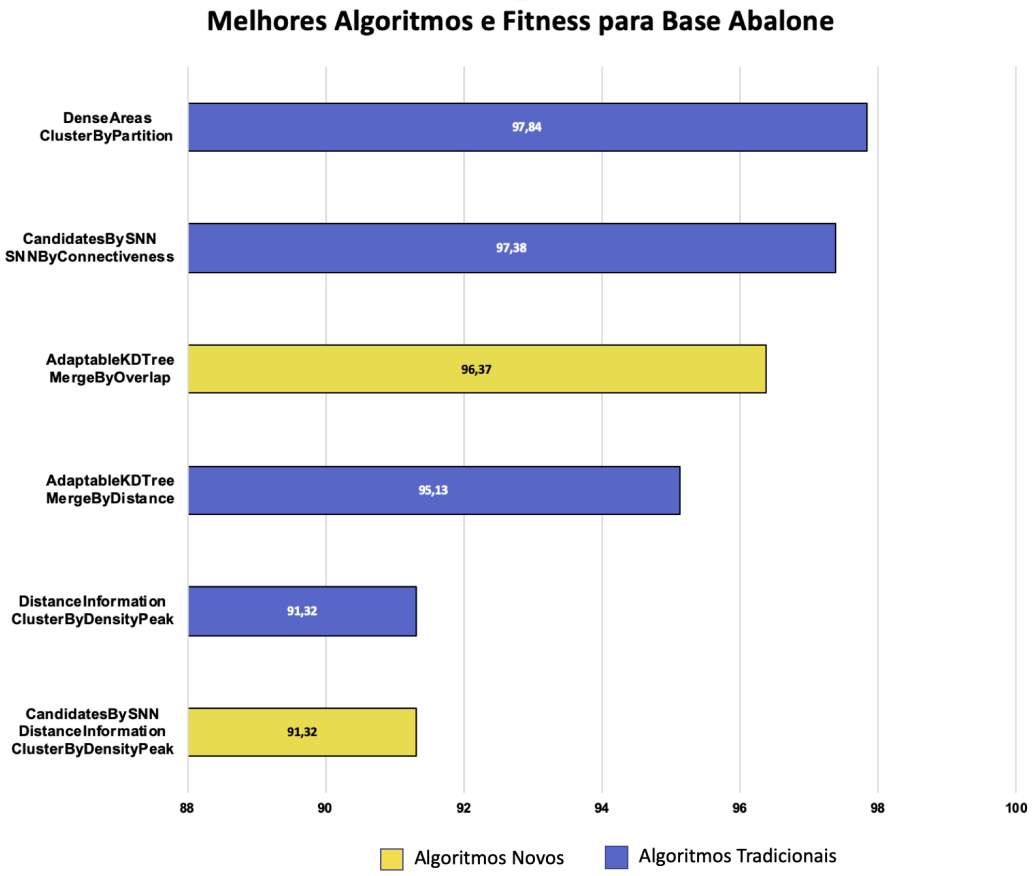
SIGMA e *EPSILON*. No entanto, para fins das análises a seguir, essas variações não foram consideradas.

7.2 Melhores Algoritmos por Base de Dados

Para cada base de dados, as figuras a seguir apresentam os seis indivíduos com melhor avaliação na última geração, ou seja, ao final de uma execução completa do Autoclustering.

Para a base Abalone, os melhores algoritmos produzidos são apresentados na Figura 6.

Figura 6. Melhores algoritmos para a base Abalone



Fonte: Autora, 2019

Entre os seis melhores algoritmos, quatro sequências de blocos correspondem a algoritmos previamente existentes (CLIQUE, SNN, DESCRY e CDP). Estes algoritmos estão representados em azul na figura, como algoritmos Tradicionais.

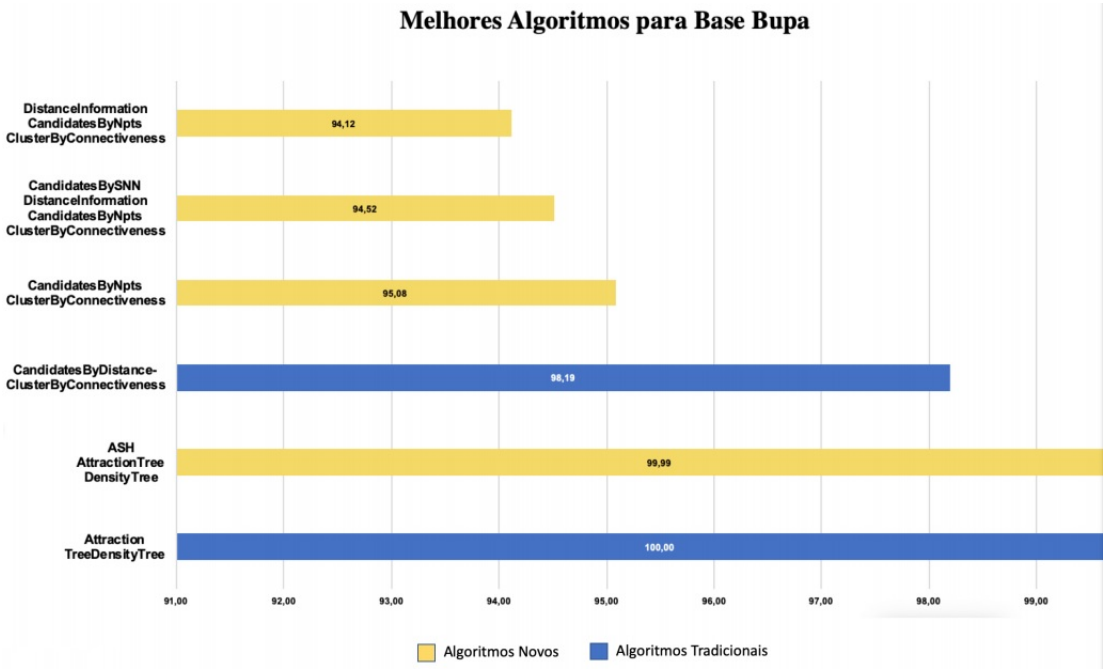
Já os algoritmos representados em amarelo correspondem a novos algoritmos,

produzidos a partir de combinações originais de procedimentos identificados nos algoritmos básicos. Estes novos algoritmos foram automaticamente produzidos pelo Autoclustering para esta base em particular.

- O primeiro algoritmo novo construído inicia com o procedimento adaptableKDTree (do algoritmo DESCRY) e em seguida obtém os grupos finais por aglomeração com o procedimento mergeByOverlap (do algoritmo SUDEPHIC).
- O segundo algoritmo combina os procedimentos candidatesBySNN (SNN), distanceInformation (CDP) e clustersByDensityPeak(CDP).

No caso da base Bupa, entre os seis melhores algoritmos apresentados na Figura 7, foram selecionados dois algoritmos tradicionais : DHC e DBSCAN. O DHC obteve a *fitness* máxima para esta base de dados.

Figura 7. Melhores algoritmos para a base Bupa



Fonte: Autora, 2019

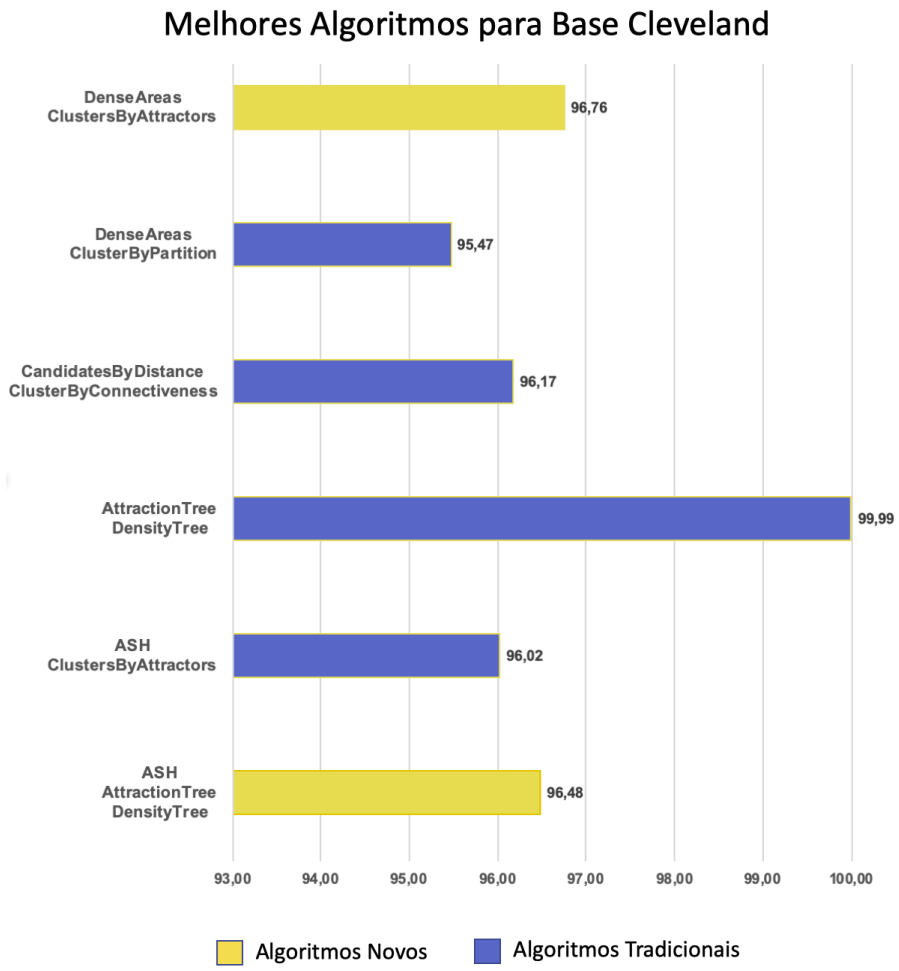
Os quatro algoritmos novos construídos para a base Bupa foram:

- A combinação entre o procedimento ASH(do algoritmo DENCLUE) e os procedimentos attractionTree e densityTree (ambos do algoritmo DHC);
- O procedimento candidatesByNPts (DBCLASD) e o procedimento clusterByConnectiveness (DBSCAN);

- Um algoritmo com quatro procedimentos, iniciando com o candidatesBySNN (SNN), seguido do distanceInformation(CDP) e do candidatesByNPts(DBCLASD) e finalizando com o clustersByConnectiveness (DBSCAN);
- Um algoritmo formado por três procedimentos, iniciando com o distanceInformation (CDP), seguido pelo candidatesByNPts (DBCLASD) e finalizando com clustersBy-Connectiveness (DBSCAN).

Para a base Cleveland, com resultados apresentados na Figura 8, quatro algoritmos tradicionais foram selecionados: o DHC, com melhor *fitness*, o DBSCAN, o DENCLUE e o CLIQUE.

Figura 8. Melhores algoritmos para a base Cleveland



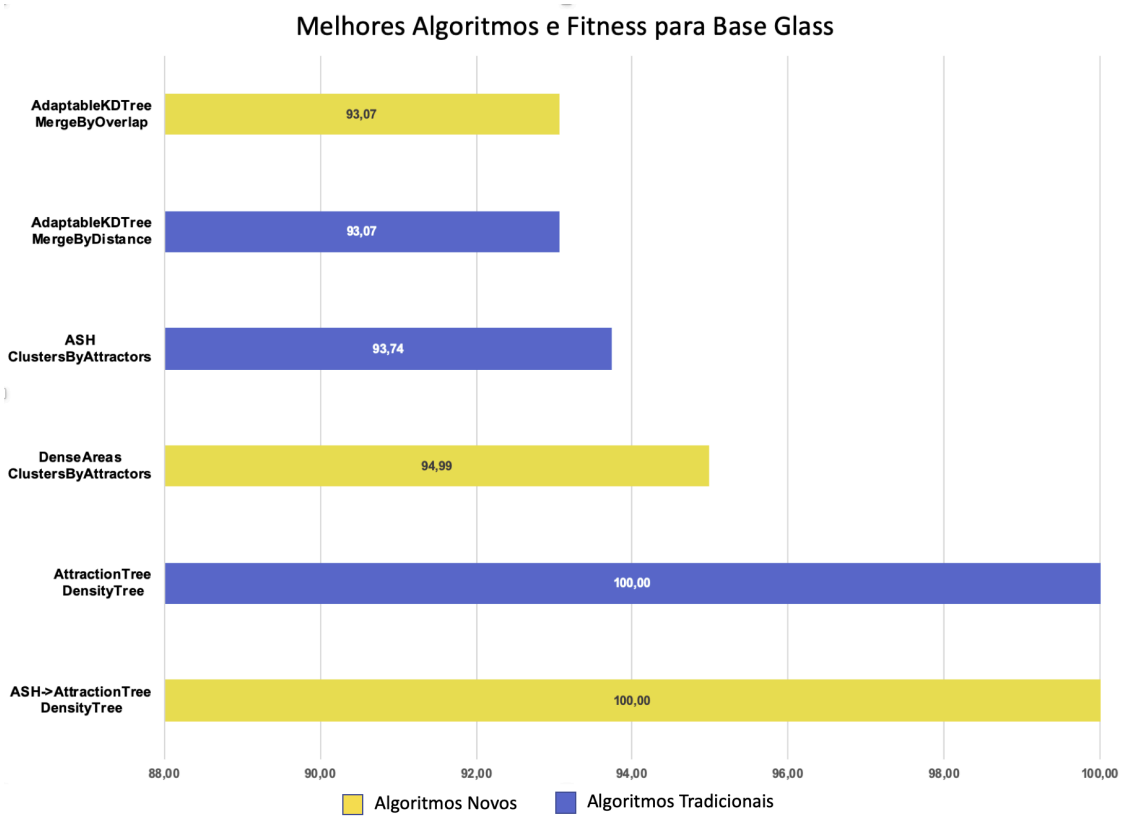
Fonte: Autora, 2019

Entre os algoritmos novos construídos, estão:

- Um algoritmo com os procedimentos denseAreas (CLIQUE) e clustersByAttractors (DENCLUE);
- Um algoritmo baseado nos procedimentos ASH (DENCLUE), attractionTree (DHC) e densityTree (DHC);

Para a base de dados Glass, como apresentado na Figura 9, o algoritmo DHC e uma variação do algoritmo DHC começando com o procedimento ASH do algoritmo DENCLUE alcançaram igualmente o máximo valor de *fitness* possível.

Figura 9. Melhores algoritmos para a base Glass



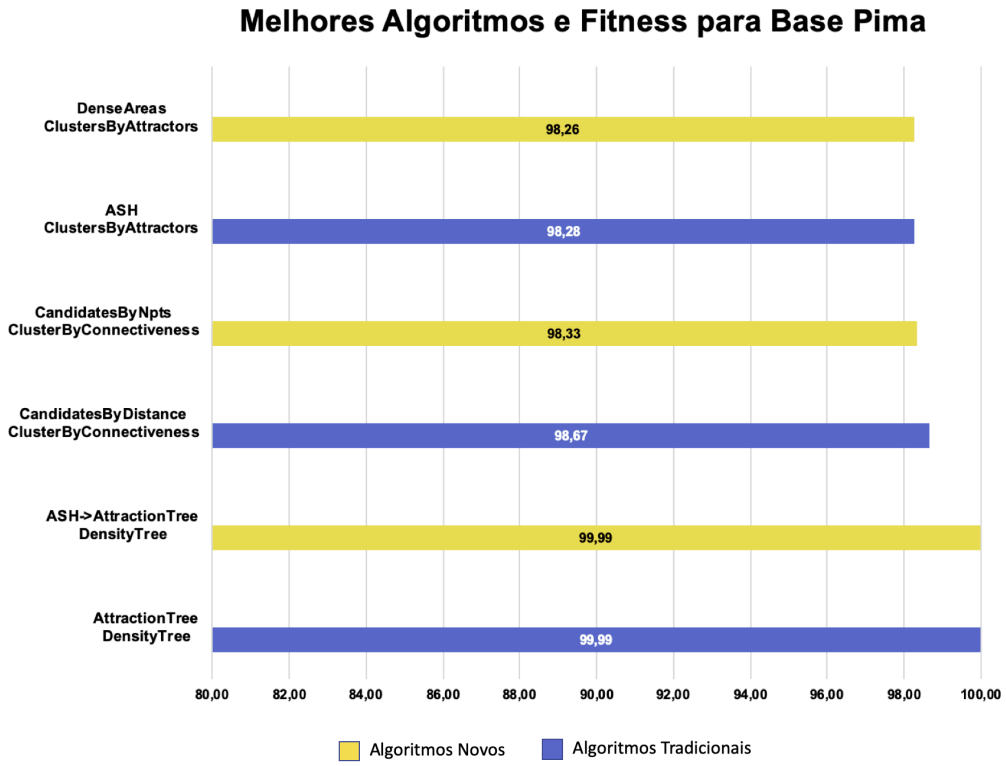
Fonte: Autora, 2019

Entre os seis melhores algoritmos também foram selecionados os algoritmos tradicionais DENCLUE e DESCRY, além dos seguintes novos algoritmos:

- o procedimento denseAreas (CLIQUE) e clustersByAttractors (DENCLUE)
- uma opção combinando o procedimento adaptableKDTree (DESCRY) e procedimento mergeByOverlap (SUDEPHIC).

No caso da base Pima, com resultados apresentados na Figura 10, a *fitness* dos melhores algoritmos teve pouca variação, de 99,9 a 98,28, ou seja, todos são boas opções para a base de dados.

Figura 10. Melhores algoritmos para a base Pima



Fonte: Autora, 2019

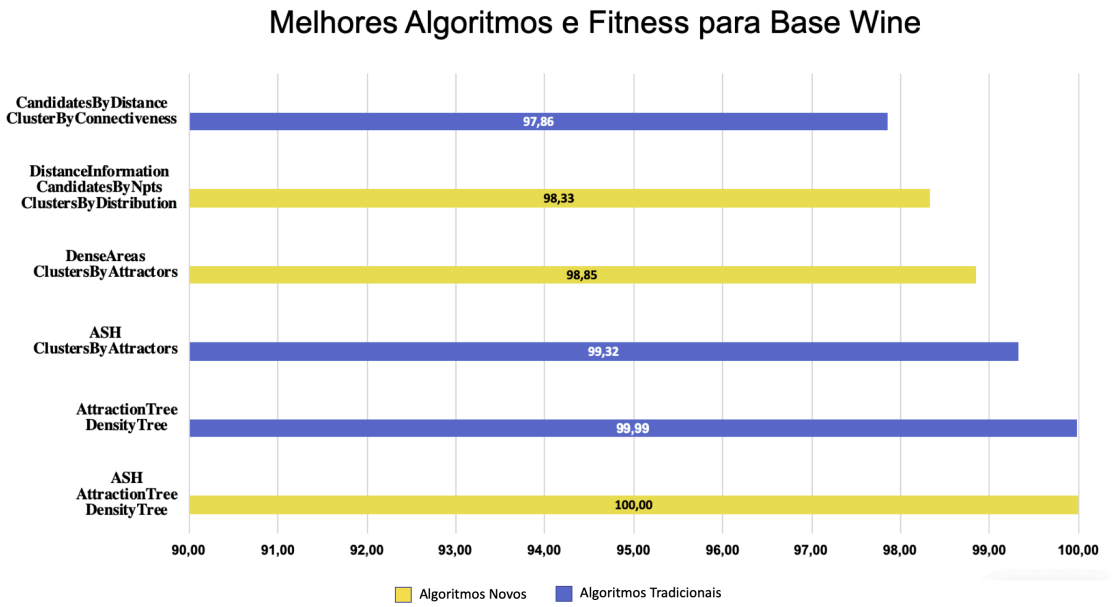
Foram selecionados DHC, DBSCAN e DENCLUE entre os algoritmos tradicionais. Os seguintes algoritmos novos foram propostos:

- Empatado com DHC em primeiro lugar, está o algoritmo com procedimentos ASH (DENCLUE), attractionTree (DHC) e densityTree (DHC).
- Outro algoritmo proposto é formado pelos procedimentos candidatesByNpts (DBCLASD) e clustersByConnectiveness (DBSCAN).

Na Figura 11, são apresentados os melhores algoritmos construídos para a base Wine.

Assim como no caso da base Pima, percebe-se pequena variação de *fitness* entre os melhores algoritmos. Neste caso, o melhor resultado, mesmo que com pequena diferença, foi obtido por um algoritmo novo.

Figura 11. Melhores algoritmos para a base Wine



Fonte: Autora, 2019

Os algoritmos tradicionais selecionados foram o DHC, o DENCLUE e o DBSCAN.

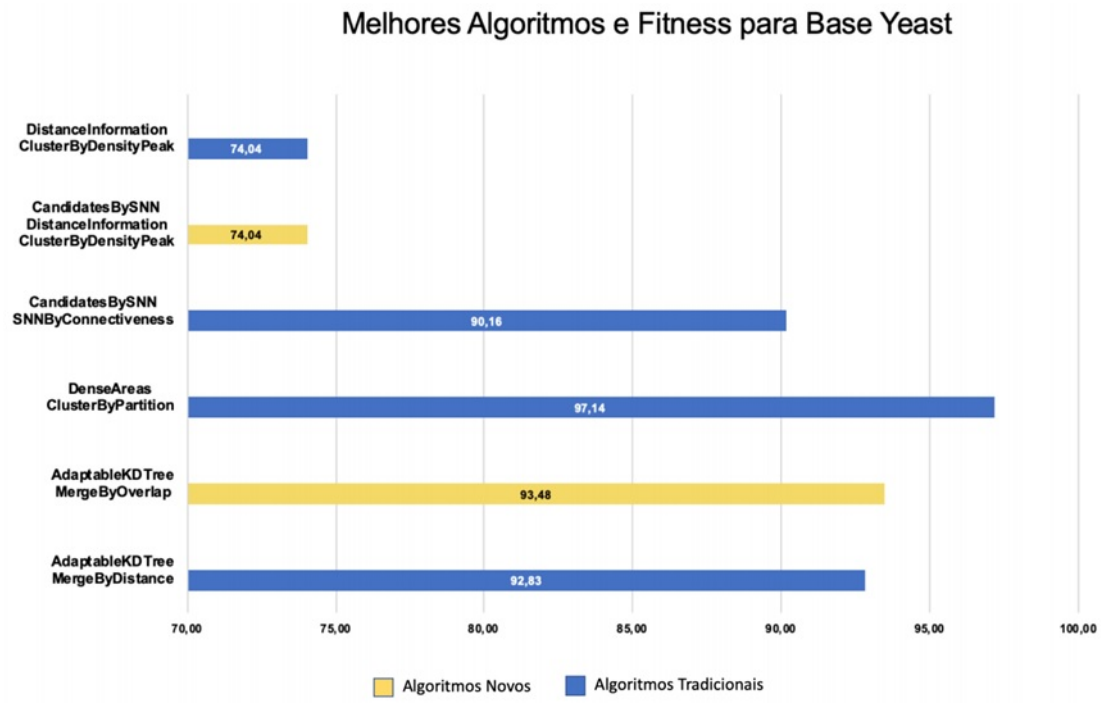
Os algoritmos novos propostos foram:

- Um algoritmo com os blocos ASH (DENCLUE), attractionTree (DHC) e densityTree (DHC). Este algoritmo, que costuma ser selecionado com frequência nos experimentos realizados, é formado pelo bloco inicial do algoritmo DENCLUE, seguido do algoritmo DHC completo. O DHC é o segundo melhor algoritmo selecionado. No caso da base Wine, o DHC executado com o procedimento ASH inicial tem um resultado melhor que o DHC executado de forma isolada.
- Um algoritmo com os procedimentos denseAreas (CLIQUE) e clustersByAttractors (DENCLUE);
- UM algoritmo com três blocos, que inicia com o procedimento de normalização distanceInformation (do algoritmo CDP) para em seguida executar candidatesByNPts e clustersByDistribution (os dois passos do algoritmo DBCLASD).

Para a base Yeast, a Figura 12 apresenta entre os resultados quatro algoritmos tradicionais (CLIQUE, DESCRY, SNN e CDP).

Os outros três algoritmos construídos são novos, incluindo as seguintes combinações:

Figura 12. Melhores algoritmos para a base Yeast



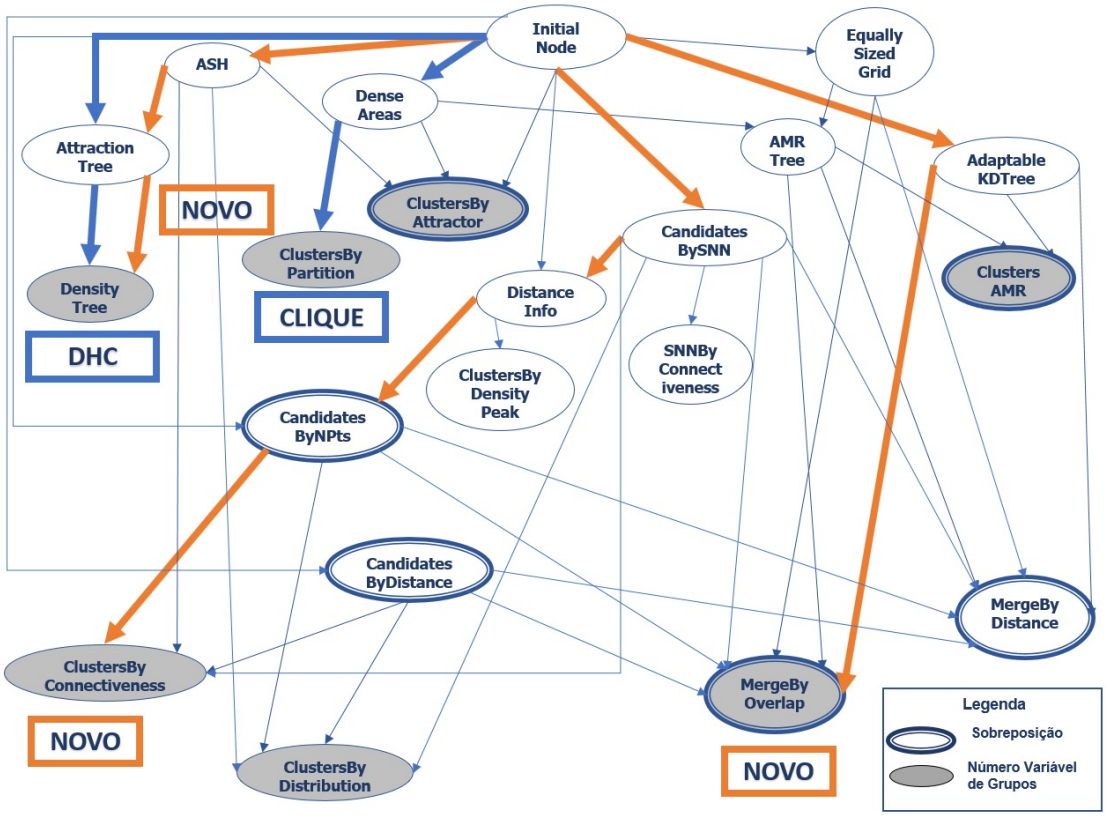
Fonte: Autora, 2019

- O procedimento adaptableKDTree (DESCRY) e mergeByOverlap (SUDEPHIC)
- O procedimento candidatesBySNN (SNN) seguido dos dois procedimentos do algoritmo CDP: distanceInformation e clustersByDensityPeak.

Importante destacar que para todas as sete bases analisadas foram produzidos algoritmos novos entre os melhores resultados. Ou seja, o Autoclustering produziu sequencias de procedimentos que não correspondem a nenhum dos algoritmos manualmente projetados entre os melhores indivíduos. Esses algoritmos são soluções originais, mesmo desconsiderando critérios de distância, opção pelo número de grupos, sobreposição, e parâmetros em geral, e analisando apenas a sequencia de blocos executados.

A Figura 13 destaca no grafo de procedimentos os melhores algoritmos produzidos. Em azul, são representados os algoritmos que correspondem a um dos algoritmos tradicionais selecionados. As sequências de procedimentos destacadas em laranja correspondem a algoritmos novos produzidos pelo Autoclustering.

Figura 13. Melhores Algoritmos Produzidos



Fonte: Autora, 2019

7.3 Frequência de Procedimentos por Base de Dados

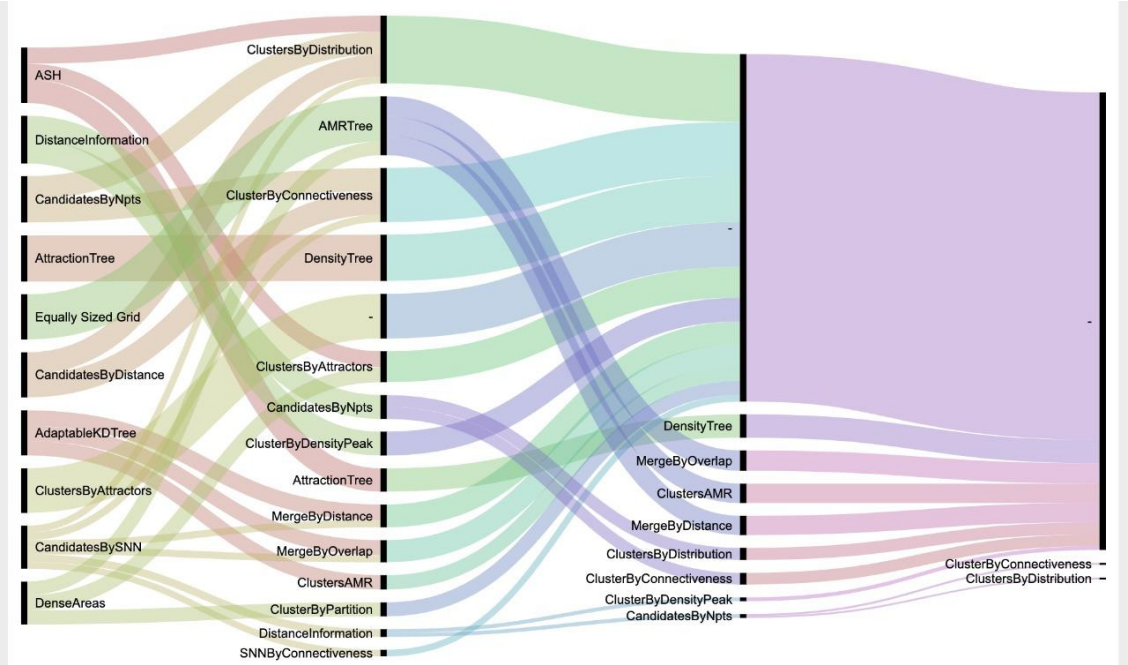
Nesta seção, analisam-se os algoritmos resultantes em maior detalhe, considerando quais blocos de construção foram selecionados, e como se conectam uns aos outros, ao final de uma execução completa do Autoclustering.

A Figura 14 apresenta na forma de uma visualização aluvial (Riehmann; Hanfler; Froehlich, 2005) a frequência de ocorrência por procedimento (*building block*) em uma execução da base de dados Cleveland Heart Diseases. A coluna à esquerda apresenta os procedimentos selecionados como bloco inicial do algoritmo, a coluna central apresenta os procedimentos selecionados como segundo bloco, e a coluna à direita os procedimentos selecionados como terceiro bloco, quando ocorreram.

O bloco na parte superior da terceira coluna representa os casos em que não houve terceiro procedimento.

Os procedimentos em cada coluna são ordenados de cima para baixo do mais frequente ao menos frequente.

Figura 14. Gráfico aluvial para a frequência de ocorrencia de cada procedimento para a base Cleveland



Fonte: Autora, 2019

As cores não tem um significado específico, servindo para facilitar a visualização das ligações entre os eixos. Já a espessura tem relevância, sendo o caminho mais frequente representado com espessura maior.

Observa-se uma distribuição uniforme dos procedimentos iniciais, mas como segundo bloco prevalecem ClustersByDistribution, AMRTree, ClustersByConnectiveness e DensityTree, além do caso em que não há um segundo bloco, já que o procedimento clustersByAttractor pode funcionar de forma isolada como um algoritmo completo.

A visualização aluvial favorece a análise da frequência dos caminhos do grafo de probabilidades.

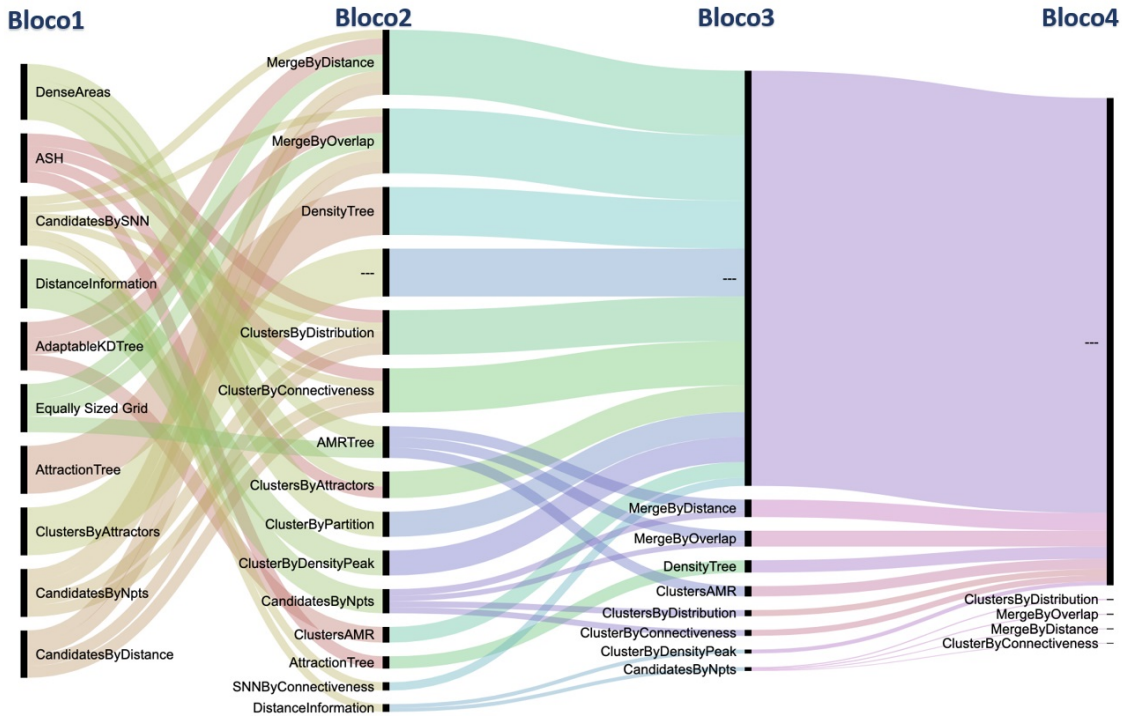
A Figura 15 apresenta uma visualização similar para os resultados de uma execução completa para a base Yeast.

Observa-se que na base Yeast prevalece como bloco inicial o procedimento DenseAreas, que era o menos frequente na execução para a base Cleveland.

Ao analisar uma execução completa, como é o caso da Figura 14 e da Figura 15, percebe-se a relativa uniformidade na seleção de procedimentos, indicando que houve ampla avaliação das possibilidades de combinação de procedimentos para a produção dos algoritmos de agrupamento.

Outra análise interessante é a comparação da frequência de procedimentos no início

Figura 15. Ocorrências de Procedimentos em uma Execução Completa - Base Yeast



Fonte: Autora, 2019

e ao final de uma mesma execução do Autoclustering.

Na Figura 16, é possível observar a frequência de procedimentos usados na primeira geração de uma execução da base Yeast.

Ao final da mesma execução, representada na Figura 17, verifica-se que ao longo do processo evolutivo, de acordo com características da base de dados, alguns procedimentos passam a ser selecionados com maior ou menor frequência.

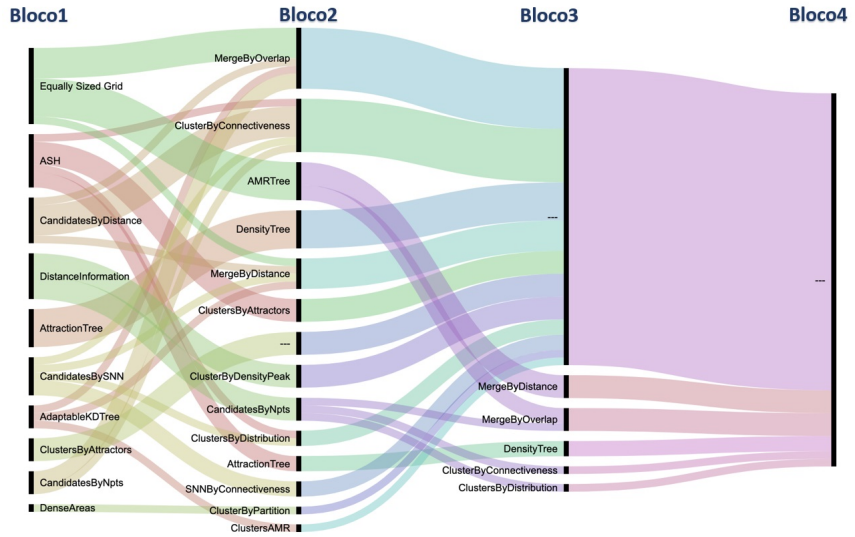
No caso do bloco AdaptableKDTree, por exemplo, a frequência de seleção foi muito maior na última geração que na primeira. Por outro lado, o procedimento EquallySizedGrid tinha maior frequência na primeira geração que na última.

7.4 Frequência e *Fitness* de Algoritmos

Uma outra forma de avaliar a frequência é mais geral, por algoritmo, no lugar de se avaliar a ocorrência individual dos blocos de construção. A Figura 18 apresenta uma visualização *Treemap* que permite analisar frequência e *fitness* de algoritmos para a base Pima.

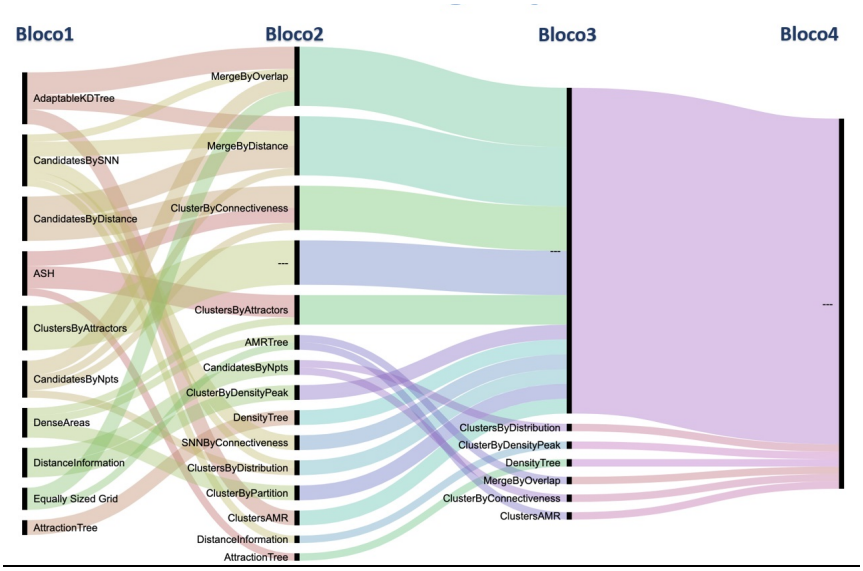
Cada item no *Treemap* inclui um rótulo com informação sobre a sequência de

Figura 16. Ocorrências de Procedimentos na Primeira Geração - Base Yeast



Fonte: Autora, 2019

Figura 17. Ocorrências de Procedimentos na Última Geração - Base Yeast



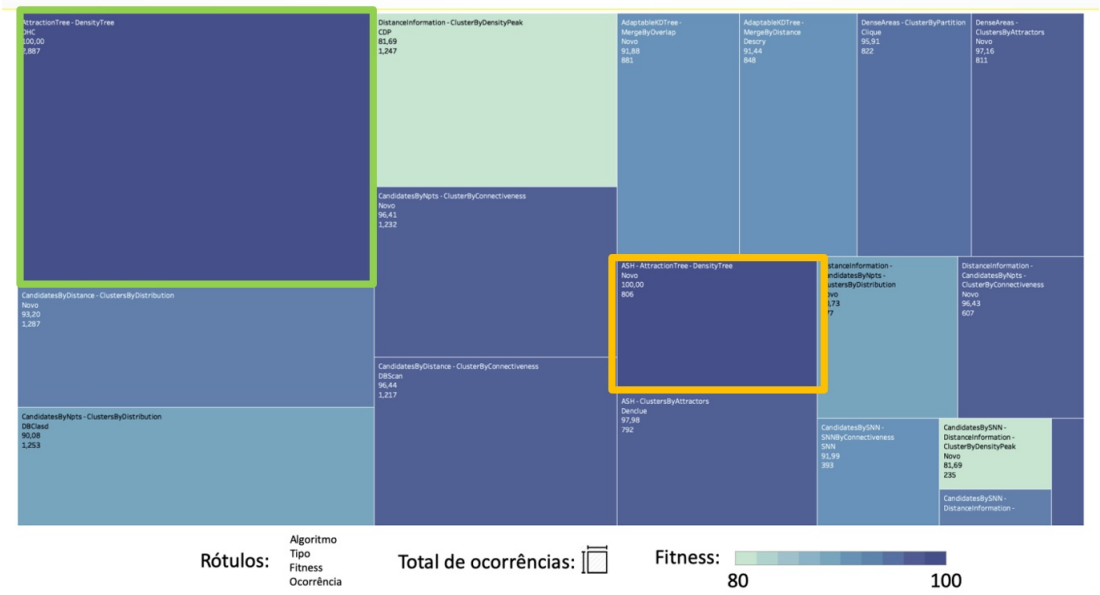
Fonte: Autora, 2019

procedimentos do indivíduo representado, sua *fitness*, número de ocorrências e um tipo, indicando se é um algoritmo tradicional ou algoritmo novo.

O tamanho de cada item é proporcional ao número de ocorrências. Observa-se o destaque em verde do algoritmo tradicional com maior número de ocorrências, o DHC.

Já a cor indica o valor da *fitness* dos indivíduos, em degradê, sendo mais escura

Figura 18. Frequência e *Fitness* de Algoritmos para a base Pima



Fonte: Autora, 2019

para valores mais altos. O DHC também está entre os itens com maior *fitness*. Em laranja, o destaque para o algoritmo novo produzido pelo Autoclustering com maior valor de *fitness*.

Em outra visualização, a Figura 19 apresenta em um histograma de ocorrência a frequência de cada algoritmo de agrupamento ao longo das gerações de uma execução completa para a base Yeast.

Observa-se que o algoritmo $CandidatesByDistance \rightarrow ClusterByConnectiveness$, que corresponde ao algoritmo tradicional DBSCAN, ocorre com frequência significativamente superior, por exemplo, ao algoritmo $candidatesByNPts \rightarrow clusterByDistribution$, que corresponde ao algoritmo tradicional DBCLASD.

7.5 *Fitness* Máxima

Nesta seção, analisa-se a *fitness* máxima geral, entre os indivíduos produzidos ao longo das trinta execuções do Autoclustering para uma base de dados.

Analisando-se a base de dados Cleveland, a Figura 20 apresenta a sequência de procedimentos com maior valor de *fitness*, considerando as 30 execuções do Autoclustering.

Os procedimentos $attractionTree \rightarrow densityTree$, que formam o algoritmo com melhor *fitness*, correspondem ao algoritmo básico DHC.

Verifica-se que, apesar de ser o algoritmo com maior valor de *fitness* em uma análise

Figura 19. Ocorrências de Algoritmos - Base Yeast

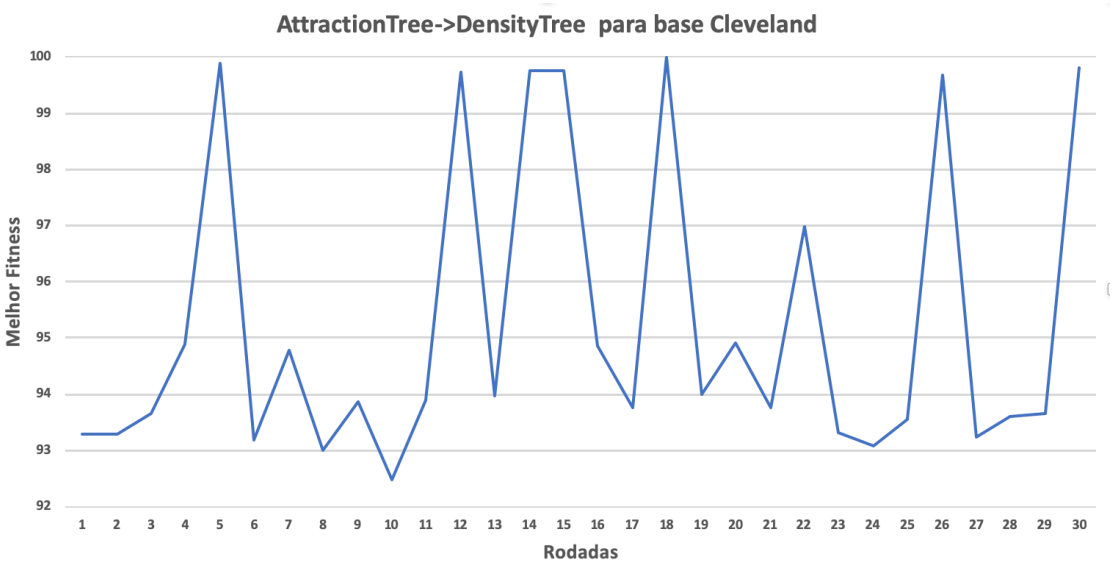
Fonte: Autora, 2019

geral, há execuções em particular que seu desempenho foi baixo. Em diferentes execuções, o algoritmo pode ter sido produzido com opções de funcionamento e parâmetros inadequados à base de dados, impactando em sua avaliação.

A Figura 21 apresenta a evolução da *fitness* máxima durante cinco rodadas (execuções completas) do Autoclustering para a base Abalone.

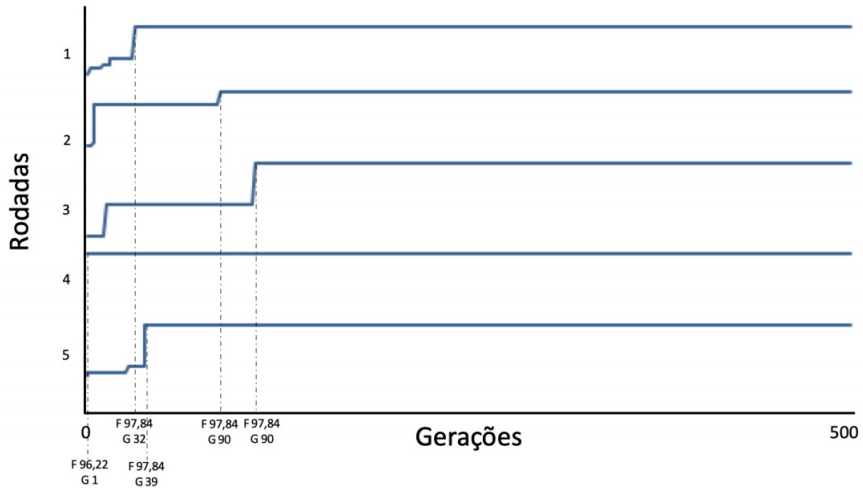
Neste caso, observa-se que a *fitness* máxima é atingida, relativamente, cedo durante as execuções, indicando a viabilidade de se reduzir o número de gerações em futuras execuções.

Figura 20. Algoritmo com *fitness* Máxima para a base Cleveland ao longo das 30 Execuções



Fonte: Autora, 2019

Figura 21. Evolução da *fitness* máxima durante execução para a base Abalone



Fonte: Autora, 2019

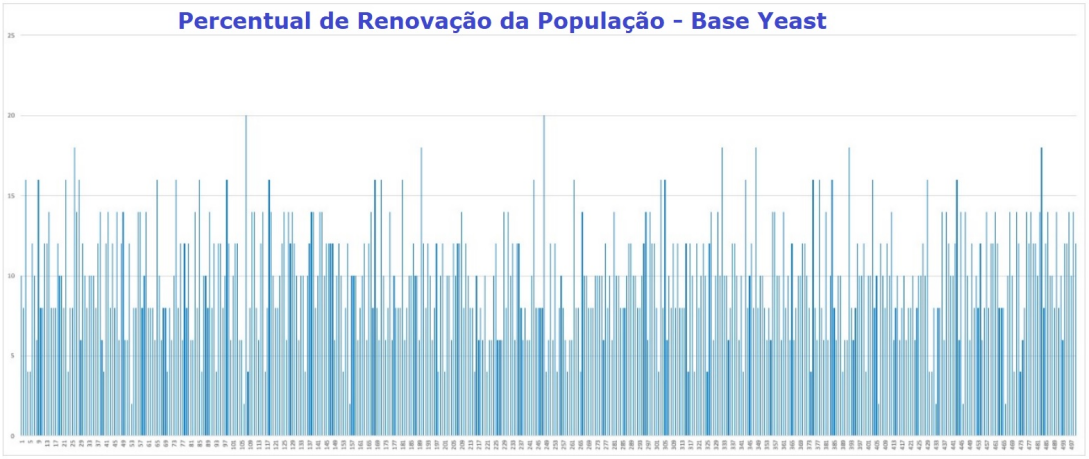
7.6 Renovação da população

Um aspecto importante a analisar é a renovação da população em geral durante o processo evolutivo. A renovação é medida pelo percentual de indivíduos presentes em uma geração, que não faziam parte da geração anterior.

A Figura 22 e a Figura 23 apresentam o percentual de renovação da população

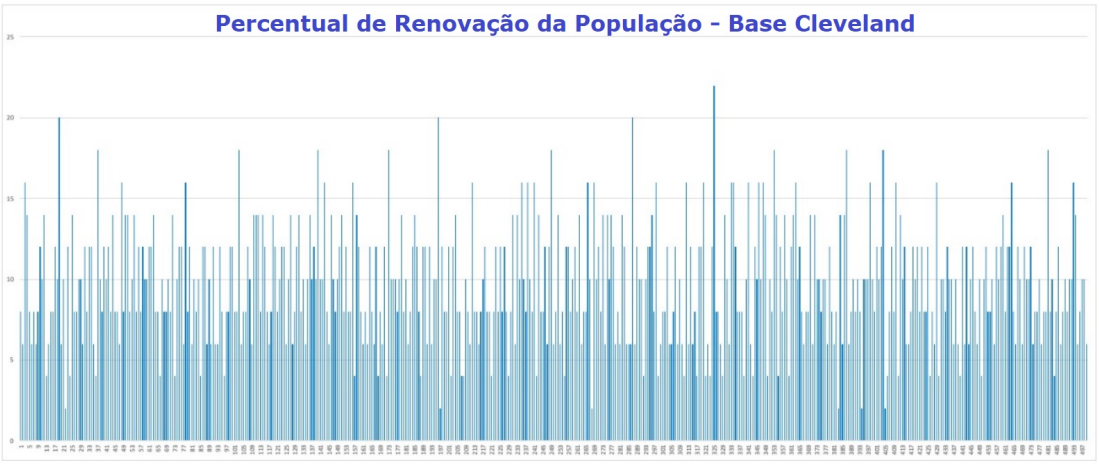
durante uma execução do Autoclustering para a base Yeast e para a base Cleveland, respectivamente.

Figura 22. Renovação da População durante execução para a base Yeast



Fonte: Autora, 2019

Figura 23. Renovação da População durante execução para a base Cleveland



Fonte: Autora, 2019

7.7 Tempo de Execução

Para analisar os caminhos possíveis do grafo de probabilidades, evoluindo um algoritmo personalizado para a base de dados, o Autoclustering exige um elevado tempo de processamento.

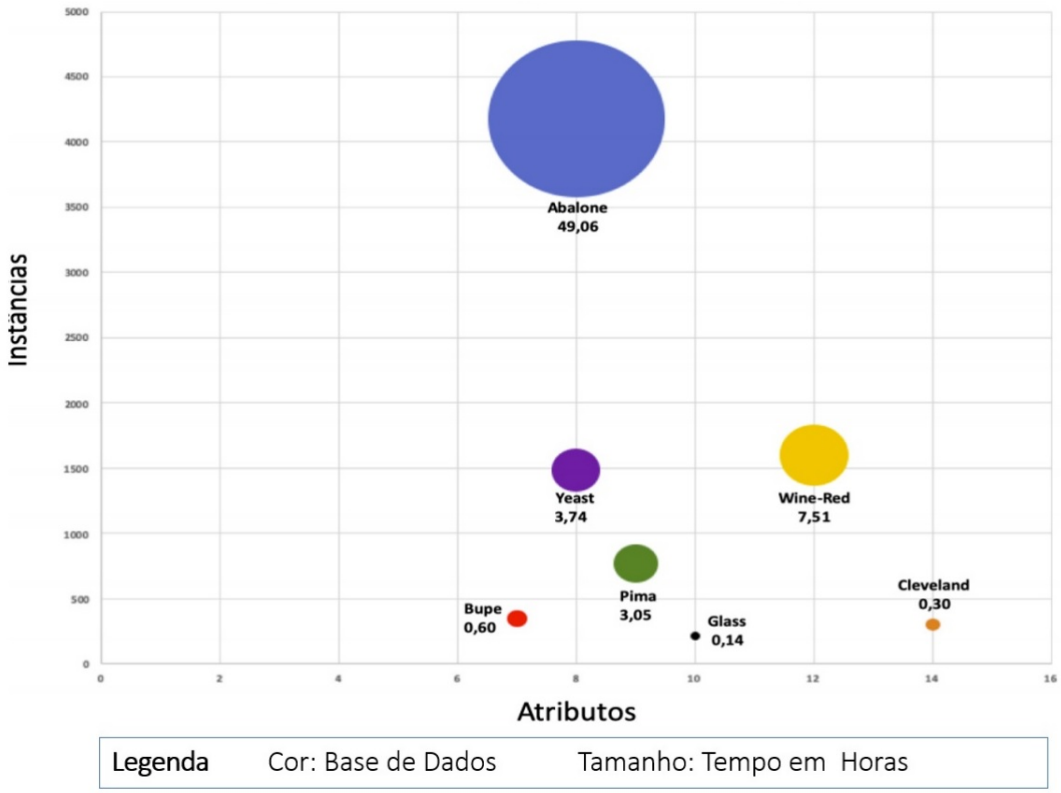
A avaliação dos indivíduos usando a adaptação do método Clest, incluindo validação cruzada em múltiplas partições de treino e teste da base de dados, foi processada a cada uma das 500 gerações, para cada um dos 50 indivíduos, durante trinta execuções.

Esses parâmetros podem ser reavaliados para redução do tempo de processamento, mas foram mantidos iguais para todas as bases, de forma a preservar a uniformidade dos experimentos.

O tamanho da base de dados, naturalmente, influencia o tempo de execução, sendo recomendável o uso de amostras no lugar do processamento completo para o caso de bases de grande porte.

A Figura 24 apresenta uma visualização do tempo de execução para as bases de dados analisadas.

Figura 24. Tempo médio de execução para as bases de dados



Fonte: Autora, 2019

Cada círculo representa uma base de dados. O posicionamento do item indica o tamanho da base de dados, sendo o eixo X correspondente ao número de atributos e o eixo Y correspondente ao número de instâncias. O tamanho do item indica o tempo de execução. Observa-se que o número de instâncias tem maior impacto no tempo de execução

que o número de atributos.

7.8 Análise das Variações dos Algoritmos

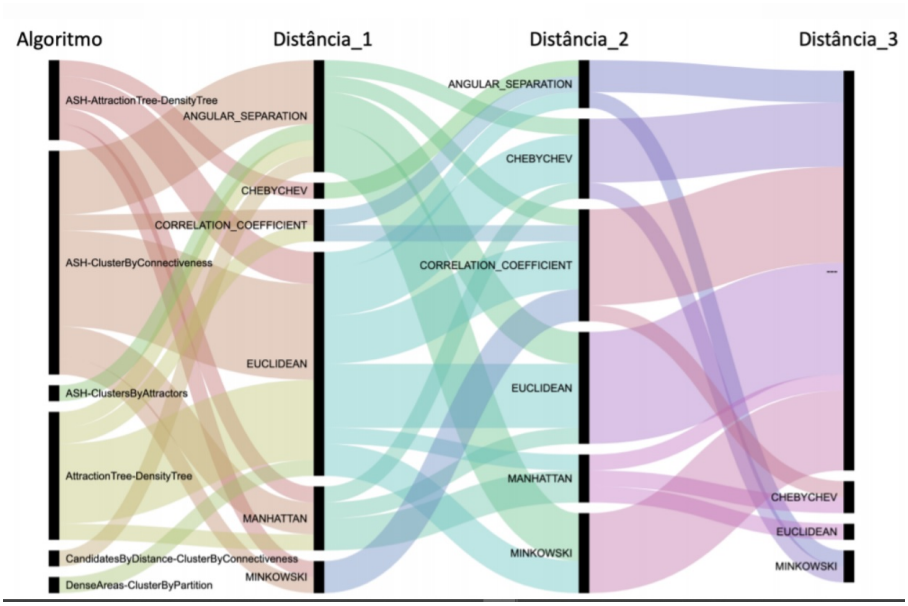
Nas seções anteriores, os resultados foram analisados levando em consideração os diferentes procedimentos selecionados, de forma a analisar cada algoritmo produzido de forma genérica, pela sequência de procedimentos que o formou.

No entanto, como destacado na seção 6.3, ao produzir um algoritmo para determinada base de dados, o Autoclustering seleciona, também dinamicamente, o critério de distância adotado, a sobreposição de grupos e os parâmetros específicos de cada procedimento.

7.8.1 Critérios de Distância

Na Figura 25, que consolida 30 execuções do Autoclustering para a base Cleveland, verifica-se que para a mesma sequência de procedimentos, por exemplo ASH → AttractionTree → DensityTree, são produzidos algoritmos com diferentes critérios de distância.

Figura 25. Análise de Critérios de Distância na Base Cleveland



Fonte: Autora, 2019

O critério de distância adotado é uma importante decisão do projeto de um algoritmo de agrupamento. Na especificação manual de um algoritmo, é comum que se

faça a verificação inicial de um método adequado de distância. Mas não é usual que se avalie como calcular a distância entre itens de forma diferenciada para cada base de dados.

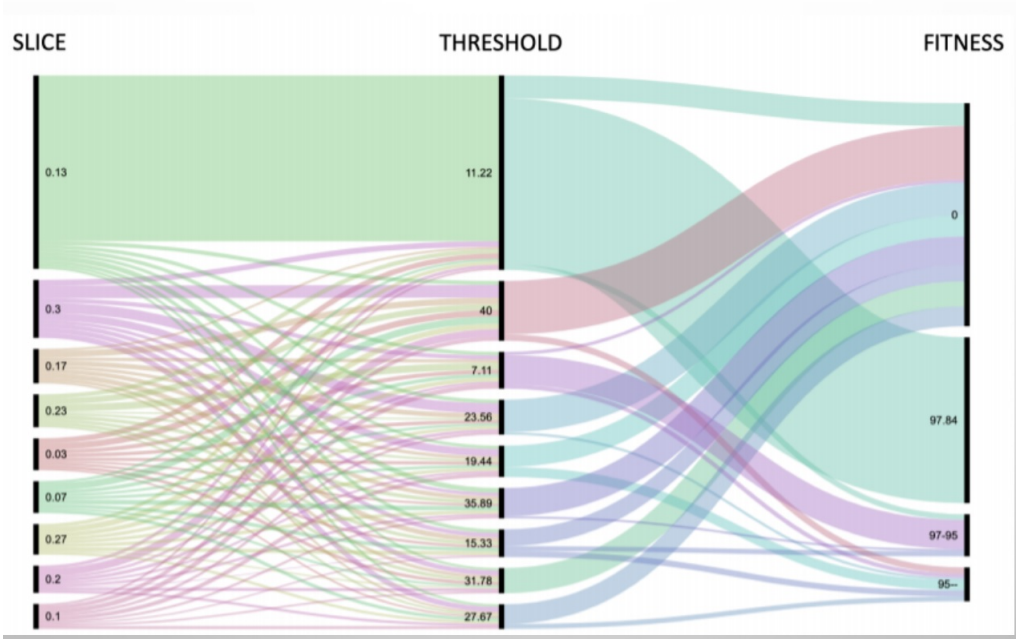
Mas no caso do Autoclustering, em que este critério é dinamicamente selecionado, observa-se que é comum usar diferentes critérios de distância em etapas diferentes do mesmo algoritmo, por exemplo iniciando com o bloco ASH com distância CHEBYCHEV, em seguida usando o critério ANGULAR SEPARATION para o procedimento *AttractionTree* e finalizando com o critério de distância MINKOVSKY para o procedimento *DensityTree*. O resultado do agrupamento é, potencialmente, diverso do resultado da mesma sequência de procedimentos com diferentes critérios de distância.

O Autoclustering suporta oito diferentes critérios de distância: Euclidiana, Manhattan, Chebychev, Minkowski, Canberra, Braycurtis, Correlation Coefficient e Angular Separation. Cada procedimento pode usar um critério diferente, possibilitando a criação de algoritmos originais mesmo com a sequência de procedimentos de um algoritmo tradicional.

7.8.2 Valores de Parâmetros

Da mesma forma, os valores de parâmetros selecionados muito influenciam o resultado final do agrupamento, e por consequência a *fitness*. Na Figura 26, é possível observar os diversos valores de parâmetros avaliados pelo Autoclustering para a mesma sequência de procedimentos que constituiu a melhor solução para a base Abalone, ou seja, os procedimentos DenseAreas → ClustersByPartition, do algoritmo tradicional CLIQUE.

Figura 26. Valores de Parâmetros para a base Abalone



Fonte: Autora, 2019

8 Conclusão

A contribuição mais direta deste trabalho é o desenvolvimento de um EDA para geração automática de algoritmos de agrupamento.

No entanto, a ideia central do trabalho, descrita de forma mais ampla, é extrair os aspectos mais representativos dos algoritmos de uma determinada área de conhecimento, explorar as conexões possíveis entre os blocos de construção identificados e usar um gerador de programas para pesquisar soluções personalizadas a cada instância de um problema. Este processo pode ser reproduzido em diferentes contextos, com a elaboração de um grafo de probabilidades específico para diferentes classes de algoritmos.

O uso de um grafo de procedimentos com probabilidades associadas, no lugar do vetor de probabilidades normalmente associado ao EDA, também pode ser considerado uma contribuição do trabalho. Esta estratégia favorece o uso de EDA para programação automática.

Outro diferencial na abordagem adotada é a atualização das probabilidades das arestas do grafo pelo critério de *fitness* dos indivíduos contendo a aresta, no lugar da abordagem padrão, que se basearia na frequência de ocorrência da aresta entre os indivíduos da população.

Diversos algoritmos de agrupamento existentes foram revisados para a identificação dos blocos de construção apropriados. A especificação do grafo de procedimentos determinou as possíveis sequências de execução dos algoritmos evoluídos de forma automática.

Um fator a contribuir para a complexidade da proposta, impactando no tempo de execução, é a necessidade de se executar, para a avaliação de cada indivíduo processado pelo EDA, o algoritmo de agrupamento correspondente e um algoritmo de classificação, conforme a adaptação do método Clest. De fato, uma contribuição relevante do trabalho é a adaptação do método Clest, originalmente usado para determinar o número ideal de grupos, mas usado aqui como medida da qualidade do algoritmo de agrupamento.

Cada procedimento básico é uma parte significativa de um dos algoritmos de agrupamento baseados em densidade selecionados. A tarefa de codificação dos procedimentos identificados como blocos de construção é mais complexa do que a simples programação dos próprios algoritmos básicos, pois foi necessário generalizar a entrada e saída de cada procedimento, tornando-o compatível com outros procedimentos baseados em algoritmos totalmente diferentes.

Os testes foram realizados com uso de bases de domínio público. Um aspecto importante entre os resultados foi a produção de algoritmos novos para todas as sete

bases de dados analisadas. Estes algoritmos foram elaborados a partir de ideias centrais de algoritmos manualmente projetados, mas a combinação de procedimentos selecionada não faz parte de um algoritmo existente.

A seleção de algoritmos novos e a diversidade de algoritmos elaborados confirmam a hipótese de que é válido elaborar, dinamicamente, algoritmos que atendam de forma mais particular cada base de dados e suas características. Essa abordagem tem a significativa vantagem de não exigir do usuário conhecimento detalhado dos algoritmos de agrupamento disponíveis, deixando a cargo do EDA elaborar o algoritmo mais adequado, ao mesmo tempo que otimiza os valores de parâmetros.

Ao construir uma solução automaticamente produzida para uma base de dados específica, o Autoclustering analisa mais de um bilhão de possíveis combinações, considerando os caminhos do grafo, as formas de funcionamento de cada bloco, os critérios de distâncias suportados e os valores possíveis de parâmetros específicos para cada bloco.

Percebe-se como fica prejudicada a tarefa manual de um analista de dados que faz testes empíricos para escolher o algoritmo de agrupamento mais adequado. Além do inviável tempo necessário para a ampla busca da solução, o universo de possibilidades permitido com o uso do Autoclustering não estaria disponível, de qualquer forma, para o analista de dados, pois as combinações possíveis incluem algoritmos originais, não disponíveis em ferramenta ou pesquisa científica anteriormente disponibilizada.

8.1 Trabalhos Futuros

Existem diversos desdobramentos possíveis deste trabalho, dependendo da perspectiva com que se analisa o algoritmo desenvolvido e os resultados obtidos.

A mais direta forma de extensão do projeto é a inclusão de algoritmos de agrupamento adicionais. Destacam-se entre os candidatos os algortimos NaNDP (CHENG et al., 2016), ADD (ANGELOV et al., 2016) e DDCAR (HYDE; ANGELOV, 2014).

Outra sugestão para trabalho futuro é a utilização de mais de um classificador para avaliação dos algoritmos de agrupamento. A *fitness* do EDA continuaria baseada no método CLEST, mas apuraria a média de acurácia de três classificadores distintos de forma a evitar possível viés pela adoção do classificador J48. Por exemplo, poderiam ser acrescentados os classificadores KNN e MLP (WU et al., 2007) (WITTEN et al., 2016).

Sugere-se ainda a adaptação dos blocos de construção para agrupar dados categóricos, temporais, espaciais ou dados não estruturados. Com a necessária adaptação dos blocos, também seria possível aplicar o projeto a bases de dados de contextos específicos, como bioinformática ou Internet das Coisas (IOT).

Um caminho de trabalho relevante, que é um desdobramento natural do presente

trabalho, tem como objetivo contemplar todas as etapas do processo KDD, descritas em seção 3.2. De forma similar às ferramentas citadas na Capítulo 5, a proposta é disponibilizar um ambiente completo que facilite o uso do algoritmo Autoclustering para o usuário final. O Autoclustering apoia apenas a etapa de modelagem (subseção 3.2.3), mas poderia ser usado como parte de um ambiente que incorporasse ferramentas de apoio às demais etapas. Uma ferramenta de apoio relevante é a de seleção automática de atributos, que permitirá identificar entre os atributos da base de dados quais são mais relevantes para o processo de agrupamento.

É possível ainda generalizar o projeto atual em um *framework* para programação automática, usando as classes do núcleo do projeto de forma a produzir, automaticamente, algoritmos em outras áreas de domínio.

Referências

AGRAWAL, R. Mining association rules between sets of items in massive databases. *Proceedings of the ACM-SIGMOD*, Washington, EUA, 1993. Citado na página 27.

AGRAWAL, R. et al. Automatic subspace clustering of high dimensional data for data mining applications. *ACM-SIGMOD MANAGEMENT OF DATA*, Washington, EUA, 1993. Citado 2 vezes nas páginas 35 e 48.

AICKELIN, J. L. U. An estimation of distribution algorithm for nurse scheduling. *Annals of Operations Research*, Holanda, v. 155, n. 1, 2007. Citado na página 40.

ALAM, M.; SADAF, K. A review on clustering of web search result. In: MEGHANATHAN, N.; NAGAMALAI, D.; CHAKI, N. (Ed.). *Advances in Computing and Information Technology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 153–159. ISBN 978-3-642-31552-7. Citado na página 28.

ANGELOV, P. et al. Autonomous data density based clustering method. In: . [S.l.: s.n.], 2016. p. 2405–2413. Citado na página 81.

ANGIULLI, F.; PIZZUTI, C.; RUFFOLO., M. Descry: A density based clustering algorithm for very large data sets. *INTELLIGENT DATA ENGINEERING AND AUTOMATED LEARNING – IDEAL 2004*, Springer Berlin, 2004. Citado 3 vezes nas páginas 36, 37 e 49.

ANGIULLI, F.; PIZZUTI, C.; RUFFOLO, M. Descry: a density based clustering algorithm for very large data sets. In: SPRINGER. *International Conference on Intelligent Data Engineering and Automated Learning*. [S.l.], 2004. p. 203–210. Citado na página 50.

ANKERST, M. et al. Optics: Ordering points to identify the clustering structure. In: . [S.l.: s.n.], 1999. v. 28, p. 49–60. Citado na página 44.

ARMAÑANZAS, R. et al. A review of estimation of distribution algorithms in bioinformatics. *BioData mining*, BioMed Central, v. 1, n. 1, p. 6, 2008. Citado na página 40.

AYODELE, M.; MCCALL, J. A. W.; REGNIER-COUDERT, O. Estimation of distribution algorithms for the multi-mode resource constrained project scheduling problem. *2017 IEEE Congress on Evolutionary Computation (CEC)*, p. 1579–1586, 2017. Citado na página 40.

BALUJA, S.; CARUANA, R. Removing the genetics from the standard genetic algorithm. In: MORGAN KAUFMANN. *PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MACHINE LEARNING*. [S.l.], 1995. p. 38–46. Citado na página 56.

BENGOETXEA, E. et al. Image recognition with graph matching using estimation of distribution algorithms. 07 2008. Citado na página 40.

BERKHIN, P. A survey of clustering data mining techniques. In: KOGAN, J.; NICHOLAS, C.; TEBOULLE, M. (Ed.). *Grouping Multidimensional Data: Recent Advances in*

Clustering. [S.l.]: Springer Berlin Heidelberg, 2002. ISBN 978-3-540-28349-2. Citado 2 vezes nas páginas 29 e 30.

BERRY GORDON, S. L. M. J. A. *Data Mining Techniques*. 2. ed. [S.l.]: Indianapolis, 2004. Citado 2 vezes nas páginas 13 e 27.

BISONG, E. An overview of google cloud platform services. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. [S.l.]: Springer, 2019. p. 7–10. Citado na página 41.

BRECKENRIDGE, J. N. Replicating cluster analysis: Method, consistency, and validity. *Multivariate Behavioral Research*, Taylor & Francis, v. 24, n. 2, p. 147–161, 1989. Citado na página 39.

BRITO, P. Q. et al. Customer segmentation in a large database of an online customized fashion business. *Robotics and Computer-Integrated Manufacturing*, v. 36, p. 93 – 100, 2015. ISSN 0736-5845. Sustaining Resilience in Today's Demanding Environments. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736584515000125>. Citado na página 28.

BUNGKOMKHUN, P.; AUWATANAMONGKOL, S. Grid-based supervised clustering - gbsc. *World Academy of Science, Engineering and Technology*, v. 60, 12 2009. Citado na página 33.

CHENG, D. et al. Natural neighbor-based clustering algorithm with density peeks. In: . [S.l.: s.n.], 2016. p. 92–98. Citado na página 81.

CORTEZ, P. et al. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, v. 47, p. 547–553, 2009. Citado na página 60.

DING, J. et al. Clustering by finding density peaks based on chebyshev's inequality. In: IEEE. *Control Conference (CCC), 2016 35th Chinese*. [S.l.], 2016. p. 7169–7172. Citado 2 vezes nas páginas 38 e 52.

DING Z. CHEN, X. H. J.; ZHAN, Y. Clustering by finding density peaks based on chebyshevs inequality. *Chinese Control Conference*, IEEE, Chengdu, China, n. 35, 2016. Citado na página 43.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>. Citado na página 60.

DUDOIT, S.; FRIDLYAND, J. A prediction-based resampling method for estimating the number of clusters in a data set. *Genome biology*, v. 3, p. RESEARCH0036, 07 2002. Citado na página 39.

ESHELMAN, L. J.; SCHAFFER, J. D. Preventing premature convergence in genetic algorithms by preventing incest. In: *ICGA*. [S.l.: s.n.], 1991. Citado na página 20.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231. Citado 2 vezes nas páginas 34 e 45.

FAYYAD, U. *Advances in Knowledge Discovery and Data Mining*. 1. ed. Cambridge: The Mit Press, 1996. Citado na página 21.

FOGEL, D. *Evolutionary Computation:Toward a New Philosophy of Machine Intelligence*. 2. ed. [S.l.]: IEEE Press, 1998. Citado na página 17.

FREITAS, A. *Data mining and knowledge discovery with evolutionary algorithms*. [S.l.]: Berlin: Springer, 2002. 264 p. Citado na página 42.

FREITAS, A. A. A review of evolutionary algorithms for data mining. In: MAIMON, O.; ROKACH, L. (Ed.). *Soft Computing for Knowledge Discovery and Data Mining*. Springer, 2007. p. 61–93. Disponível em: <https://kar.kent.ac.uk/23996/>. Citado na página 42.

GOTTSCHLICH, J. *Why More Software Development Needs to Go to the Machines*. 2019. Intel. Disponível em: <https://newsroom.intel.com/news/why-more-software-development-needs-go-machines/#gs.k1avm2>. Acesso em: 04 dec 2019. Citado na página 14.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. 703 p. ISBN 978-0-12-381479-1. Citado 5 vezes nas páginas 13, 26, 29, 30 e 54.

HENEGAR, C. et al. Clustering biological annotations and gene expression data to identify putatively co-regulated biological processes. *Journal of bioinformatics and computational biology*, v. 4 4, p. 833–52, 2006. Citado na página 28.

HINNEBURG, A.; KEIM, D. A. A general approach to clustering in large databases with noise. *Knowledge and Information Systems*, Springer, v. 5, n. 4, p. 387–415, 2003. Citado 3 vezes nas páginas 34, 47 e 48.

HOLLAND, J. Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*, University of Michigan Press, 1975. Citado 2 vezes nas páginas 17 e 18.

HYDE, R.; ANGELOV, P. A fully autonomous data density based clustering technique. In: . [S.l.: s.n.], 2014. Citado na página 81.

INZA, I. et al. Feature subset selection by estimation of distribution algorithms. *Artificial Intelligence in Medicine*, v. 23, p. 187–205, 10 2001. Citado na página 40.

JIANG, D.; PEI, J.; ZHANG, A. Dhc: a density-based hierarchical clustering method for time series gene expression data. In: IEEE. *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*. [S.l.], 2003. p. 393–400. Citado 2 vezes nas páginas 36 e 47.

KAUFMAN, P. J. R. L. *Finding Groups in Data: An Introduction to Cluster Analysis*. 1. ed. [S.l.]: Wiley, 1990. Citado na página 32.

KOZA, J. R. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. [S.l.]: Stanford University, Department of Computer Science Stanford, CA, 1990. v. 34. Citado na página 18.

KREJCA, M. S.; WITT, C. Theory of estimation-of-distribution algorithms. In: *Theory of Evolutionary Computation*. [S.l.]: Springer, 2020. p. 405–442. Citado na página 14.

LARRAñAGA, P.; LOZANO, J. A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN 0792374665. Citado 3 vezes nas páginas 14, 19 e 20.

LIAO, W.-k.; LIU, Y.; CHOUDHARY, A. A grid-based clustering algorithm using adaptive mesh refinement. In: *7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining*. [S.l.: s.n.], 2004. v. 22, p. 61–69. Citado 3 vezes nas páginas 37, 49 e 50.

LIU, F. et al. Estimation distribution of algorithm for fuzzy clustering gene expression data. In: . [S.l.: s.n.], 2006. p. 328–335. Citado na página 40.

MAZA, S.; TOUAHRIA, M. Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms. *Applied Intelligence*, Springer, p. 1–21, 2019. Citado na página 40.

MCQUEEN, J. Some methods for classification and analysis of multivariate observations. *Computer and Chemistry*, v. 4, p. 257–272, 01 1967. Citado na página 31.

MEIGUINS, A. et al. Visual analysis scenarios for understanding evolutionary computational techniques' behavior. *Information*, v. 10, p. 88, 02 2019. Citado na página 60.

MEIGUINS, A. S. G. et al. Autoclustering: An estimation of distribution algorithm for the automatic generation of clustering algorithms. In: *2012 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2012. p. 1–7. Citado na página 43.

MIRKIN, B. *Clustering for Data Mining: A Data Recovery Approach*. [S.l.]: Chapman & Hall/CRC, 2005. Citado 4 vezes nas páginas 28, 30, 31 e 55.

MONTI, S. et al. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, Springer, v. 52, n. 1-2, p. 91–118, 2003. Citado na página 39.

NAYAK, J.; NAIK, B.; BEHERA, H. A comprehensive survey on support vector machine in data mining tasks: applications & challenges. *International Journal of Database Theory and Application*, v. 8, n. 1, p. 169–186, 2015. Citado na página 26.

NIKAM, S. S. A comparative study of classification techniques in data mining algorithms. *Oriental journal of computer science & technology*, v. 8, n. 1, p. 13–19, 2015. Citado na página 26.

NISBET, R.; MINER, G.; YALE, K. *Handbook of Statistical Analysis and Data Mining Applications*. Elsevier Science, 2017. ISBN 9780124166455. Disponível em: <https://books.google.com.br/books?id=QVgXAwAAQBAJ>. Citado na página 41.

OLIVEIRA, C. et al. Edacluster: an evolutionary density and grid-based clustering algorithm. In: . [S.l.: s.n.], 2007. p. 143 – 150. ISBN 978-0-7695-2976-9. Citado na página 40.

ORTEGA, J. et al. The k-means algorithm evolution. In: *Clustering*. [S.l.: s.n.], 2019. Citado na página 38.

O'NEILL, M.; SPECTOR, L. Automatic programming: The open issue? *Genetic Programming and Evolvable Machines*, v. 20, p. 1–12, 12 2019. Citado na página 14.

PAPPA, G. L.; FREITAS, A. A. Towards a genetic programming algorithm for automatically evolving rule induction algorithms. In: *Proc. ECML/PKDD-2004 Workshop on Advances in Inductive Learning*. [S.l.: s.n.], 2004. p. 93–108. Citado na página 41.

PELIKAN, M.; HAUSCHILD, M.; LOBO, F. G. Estimation of distribution algorithms. In: *Handbook of Computational Intelligence*. [S.l.: s.n.], 2015. Citado na página 40.

PIATESKY, G. *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*. 2014. KdNuggets. Disponível em: <https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html>. Acesso em: 03 aug 2019. Citado 2 vezes nas páginas 22 e 23.

RAMESH, B. Gpr: A data mining tool using genetic programming. *Communications of the Association for Information Systems*, v. 5, 01 2001. Citado na página 41.

REZENDE, R. P. S. O. *Sistemas Inteligentes – Fundamentos e Aplicações.* [S.l.]: Editora Manole, 2003. Citado na página 26.

Riehmann, P.; Hanfler, M.; Froehlich, B. Interactive sankey diagrams. In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* [S.l.: s.n.], 2005. p. 233–240. ISSN 1522-404X. Citado na página 68.

ROBLES P. MIGUEL, P. L. V. *Solving the traveling salesman problem with EDAs.* [S.l.]: Kluwer Academic Publishers, 2002. Citado na página 40.

SAGARNA, R.; LOZANO, J. A. On the performance of estimation of distribution algorithms applied to software testing. *Applied Artificial Intelligence*, v. 19, p. 457–489, 04 2005. Citado na página 40.

SAMET, H. Hierarchical representations of collections of small rectangles. *ACM Comput. Surv.*, v. 20, p. 271–309, 12 1988. Citado na página 36.

SAMMUT, C.; WEBB, G. I. *Encyclopedia of Machine Learning and Data Mining.* 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2017. ISBN 148997685X, 9781489976857. Citado na página 42.

SANDER, J.; ESTER, M.; KRIEGER, H. Density based clustering in spatial databases. 01 2020. Citado na página 44.

SANTOS, Y. et al. Data visualization scenarios for the analysis of computational evolutionary techniques. In: . [S.l.: s.n.], 2019. p. 292–299. Citado na página 60.

SHAN, Y. et al. A survey of probabilistic model building genetic programming. In: *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. [S.l.: s.n.], 2007. v. 33, p. 121–160. Citado na página 20.

SIERRA, B. et al. Rule induction by estimation of distribution algorithms. 01 2002. Citado na página 40.

THORNTON, C. et al. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: *Proc. of KDD-2013*. [S.l.: s.n.], 2013. p. 847–855. Citado na página 41.

TIBSHIRANI, R.; WALTHER, G.; HASTIE, T. Estimating the number of clusters via the gap statistic. *Journal of Royal Statistical Society*, p. 411–423, 2000. Citado na página 39.

VITTER, J. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, v. 11, p. 37–57, 03 1985. Citado na página 36.

WANG, J. et al. Introduction to data mining in bioinformatics. 01 2005. Citado na página 13.

WEI J. YANG, R. M. W. Sting: A statistical information grid approach to spatial data mining. In: *Proceedings of the 23rd IEEE INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES*. [S.l.: s.n.], 1997. Citado na página 33.

WESTPHAL, T. B. C. *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*. 1. ed. [S.l.]: Wiley, 1998. Citado 2 vezes nas páginas 13 e 25.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016. Citado 2 vezes nas páginas 56 e 81.

WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, v. 14, 12 2007. Citado na página 81.

XU, X. et al. A distribution-based clustering algorithm for mining in large spatial databases. In: IEEE. *Data Engineering, 1998. Proceedings., 14th International Conference on*. [S.l.], 1998. p. 324–331. Citado 2 vezes nas páginas 35 e 46.

YANAI, K.; IBA, H. Estimation of distribution programming: Eda-based approach to program generation. In: *Studies in Fuzziness and Soft Computing*. [S.l.: s.n.], 2007. v. 192, p. 103–122. Citado na página 41.

YE, H.; LV, H.; SUN, Q. An improved clustering algorithm based on density and shared nearest neighbor. In: IEEE. *Information Technology, Networking, Electronic and Automation Control Conference, IEEE*. [S.l.], 2016. p. 37–40. Citado 3 vezes nas páginas 38, 43 e 51.

ZHANG, Q.; MüHLENBEIN, H. On the convergence of a class of estimation of distribution algorithms. *Evolutionary Computation, IEEE Transactions on*, v. 8, p. 127 – 136, 05 2004. Citado na página 20.

ZHOU, D. et al. Sudephic: Self-tuning density-based partitioning and hierarchical clustering. In: SPRINGER. *International Conference on Database Systems for Advanced Applications*. [S.l.], 2004. p. 554–567. Citado 2 vezes nas páginas 37 e 50.