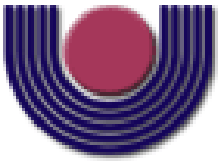

UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Formas de Mapeamento do Problema *CASH* para Agrupamento de Dados

André Afonso Knob

CASCAVEL

2015

ANDRÉ AFONSO KNOB

FORMAS DE MAPEAMENTO DO PROBLEMA *CASH* PARA AGRUPAMENTO DE DADOS

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Clodis Boscarioli

Co-Orientadora: Profª. Drª. Rosangela Villwock

CASCAVEL

2015

ANDRÉ AFONSO KNOB

FORMAS DE MAPEAMENTO DO PROBLEMA *CASH* PARA AGRUPAMENTO DE DADOS

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Ciência da Computação*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Clodis Boscarioli (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Profª. Drª. Rosangela Villwock (Co-Orientadora)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Jefferson Gustavo Martins
Colegiado de Tecnologia em Sistemas para Internet, UTFPR

Cascavel, 18 de fevereiro de 2016.

DEDICATÓRIA

À minha família, que com muita vontade sempre me guiou pelos caminhos corretos, me fornecendo todos os meios para que eu pudesse passar pelas dificuldades que surgiram. Aos meus amigos mais próximos, que sempre me fazem rir e esquecer qualquer problema.

AGRADECIMENTOS

Aos meus pais, Inacio e Regina, por terem me dado uma excelente criação, sempre me fornecendo tudo que precisei. Agradeço por me apoiarem quando falhei, dando motivos para que eu continuasse tentando.

Aos meus irmãos, Jean, Adriana e Andréia, que sempre estiveram presentes em minha vida, tornando tudo mais fácil. Agradeço por me darem os bons momentos em família que sempre temos em nossos reencontros.

Aos meus amigos mais próximos, Lucas Lira, Lucas Silvestri, Marina Giusti, Laura Fontana, Nathália Dambros, Diego Oliveira e William Clos. Agradeço por sempre fazerem meus finais de semana muito alegres, isso me deu ânimo e forças para continuar.

Aos professores Clodis e Rosangela, que me ofereceram muita ajuda durante a graduação, sem suas orientações eu não seria capaz de concluir essa etapa de minha vida. Agradeço por toda a atenção que me deram, foi realmente essencial.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

KDD	*Knowledge Discovery in Databases*
DM	*Data Mining*
YADMT	*Yet Another Data Mining Tool*
CASH	*Combined Algorithm Selection and Hyperparameter Optimization*
SOM	*Self-Organizing Maps*
CH	Calinski-Harabasz
WGSS	*Within-Group Sum of Squares*
BGSS	*Between-Group Sum of Squares*
SGBD	*Sistema Gerenciador de Banco de Dados*

LISTA DE SÍMBOLOS

V	Variância Intra-Grupos
F	Medida F
R	Índice Aleatório
D	Índice Dunn

SUMÁRIO

RESUMO

O grande armazenamento de dados por parte das empresas impulsionou a área de mineração de dados, útil para descobrir conhecimento oculto em meio a uma quantidade de dados que inviabilizaria uma análise manual. Esta área possui diversos algoritmos de aprendizado de máquina, sendo que cada um contém parâmetros de entrada diferentes. Configurar manualmente os algoritmos é um grande desafio, sendo uma tarefa incerta, empírica e trabalhosa. Este trabalho aborda essa dificuldade, com estudo e implementação dos métodos de busca aleatória e algoritmos genéticos, para automatizar o processo de seleção de parâmetros em algoritmos de agrupamento de dados da ferramenta YADMT. Com isso, é possível melhorar os resultados do processo de descoberta de conhecimento em bases de dados, por meio da geração de grupos bem divididos segundo o critério Calinski-Harabasz. Além de melhorar os resultados, também se consegue facilitar o processo, com a possibilidade de salvar e carregar as melhores configurações para um algoritmo *A* e uma base de dados *D*, além de fornecer uma visualização ao usuário, por meio de gráficos exibindo os resultados da seleção automática. Os resultados obtidos nos testes empíricos são formas de validar as implementações do trabalho, de modo que é possível observar os benefícios que os métodos adotados trazem.

Palavras-chave: Seleção de Parâmetros, Mineração de Dados, Análise de Agrupamentos.

Capítulo 1

Introdução

Armazenar dados se tornou um ato comum, seja por um usuário final gravando seus dados de maneira desorganizada ou uma empresa armazenando dados relativos as suas transações. Nesse último caso, por mais que os dados estejam bem organizados, pode-se questionar se esta fonte será aproveitada de modo que algum benefício para tomada de decisão seja obtido.

No âmbito deste questionamento surgiu a área de descoberta de conhecimento em base de dados (KDD, do inglês *Knowledge Discovery in Databases*) e também uma de suas etapas, a mineração de dados (DM, do inglês *Data Mining*). Segundo (FAYYAD *et al.*, 1996), KDD é "um processo não trivial de identificação de novos padrões válidos, úteis e compreensíveis". Esse processo compreende as etapas ilustradas na Figura 1.1.

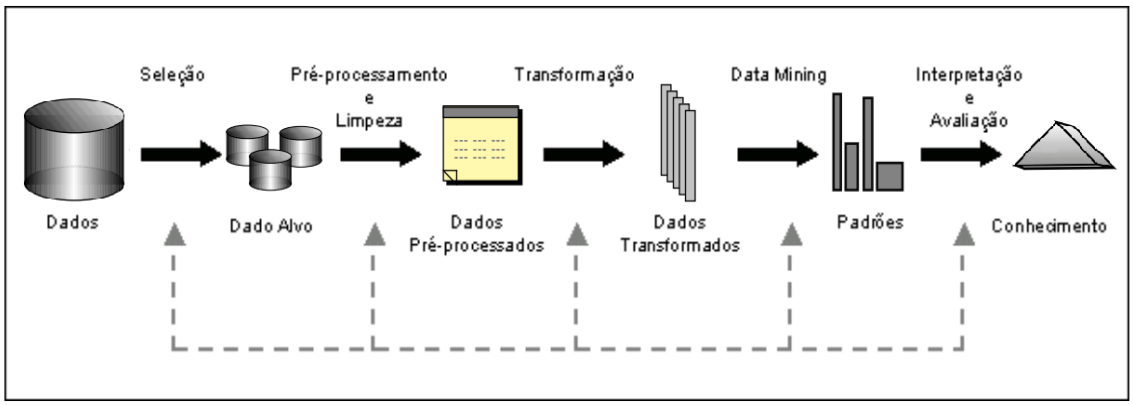

Figura 1.1: Etapas do processo de KDD

Fonte: Adaptada de (FAYYAD *et al.*, 1996)

A seleção é a etapa inicial do processo, sendo necessária para definir o conjunto de dados que será utilizado nas etapas seguintes. O pré-processamento de dados se concentra em corrigir as inconsistências da base de dados, com objetivo de aumentar a eficiência dos algoritmos de mineração. “Pode-se também estar interessado em aprender mais a respeito dos dados, o que pode ser feito, por exemplo, por meio de visualizações” (BATISTA; MONARD,

2003). Ainda, segundo esses autores, também pode ser desejável alterar o grau de granularidade dos dados, modificando a quantidade de detalhes a respeito dos dados e variando a carga de processamento necessária na mineração. Diversos estudos, a exemplo de (BATISTA; MONARD, 2003), (SCHMITT; ANDRADE; BARBETTA, 2005) e (ALBERTO; ALMEIDA, 2012), focam em técnicas do pré-processamento, pois esta é uma fase importante, que influenciará diretamente na mineração de dados.

A etapa de transformação prepara os dados para a seguinte, sendo muitas vezes necessária a aplicação de técnicas de transformação de tipo de dados, para garantir que o repositório seja uma entrada compatível com o algoritmo de mineração de dados escolhido.

Mineração de dados se refere ao processo de extrair conhecimento novo e útil de uma vasta quantidade de dados (HAN; KAMBER, 2000), por meios automáticos ou semiautomáticos. Este conhecimento adquirido é interpretado como tendências ou padrões que foram encontrados nas bases de dados. Entretanto, é um erro conceitual crer que estes conhecimentos podem ser identificados automaticamente a partir de grandes bases de dados sem qualquer intervenção humana (HAN; KAMBER, 2000). Nesta parte do processo os algoritmos irão descobrir padrões para resolver tarefas focadas, por exemplo, em criação de grupos (Agrupamento de Dados), atribuição a classes (Classificação) ou descoberta de associações entre dados (Regras de Associação).

Na etapa de interpretação e avaliação, é realizada uma organização dos resultados, com a intenção de melhorar a precisão da tomada de decisão e/ou visualizar com maior eficiência os resultados obtidos. Nesta parte do processo é necessária uma capacidade de interpretação adequada por parte dos usuários, sendo esta muitas vezes encarregada ao analista (especialista do domínio dos dados), (POZO, 2002).

Como definido, DM pode se aplicar a algumas tarefas, podendo estas ter aprendizado supervisionado ou não supervisionado. No aprendizado supervisionado, todos os exemplos de treinamento (as tuplas da base de dados) são rotulados, ou seja, cada tupla contém uma coluna que define a classe à qual aquele conjunto de dados pertence. No aprendizado não supervisionado, as tuplas não possuem um atributo definindo a classe, sendo que as similaridades encontradas têm base apenas nos padrões de entrada. Portanto, também é de interesse da DM realizar o treinamento não supervisionado para separar dados em diferentes grupos, por meio da geração de um novo atributo, que dita a qual grupo cada padrão pertence.

O processo de KDD é de interesse comum a diversas disciplinas, com pesquisadores de

áreas como aprendizado de máquina, banco de dados, matemática, estatística, inteligência artificial e visualização de dados. Portanto, é natural o surgimento de uma quantidade considerável de subáreas e diferentes focos de KDD.

1.1 Motivação

Existem muitos algoritmos de aprendizado de máquina; e, considerando os parâmetros de todos os algoritmos, existe um grande número de alternativas possíveis (THORNTON *et al.*, 2012). É interessante então, definir para uma dada base de dados, qual é o algoritmo ótimo (ou sub-ótimo) para determinada tarefa e quais são seus melhores parâmetros. Esta é uma questão ainda pouco explorada pela literatura e foi intitulada por (THORNTON *et al*., 2012) como *Combined Algorithm Selection and Hyperparameter Optimization Problem* (ou simplesmente *CASH*). Estes autores demonstram que uma pesquisa automatizada no espaço combinatorial dos algoritmos e parâmetros produzem melhores resultados que simplesmente fazer a escolha padrão (configuração *default* por parte do sistema).

Uma explicação oferecida por (THORNTON *et al*., 2012) para a baixa disponibilidade de conteúdo a respeito, é que o espaço combinatório dos algoritmos de aprendizado e seus parâmetros são itens difíceis de pesquisa: as funções de resposta contêm ruídos e os espaços de busca geralmente são de alta dimensionalidade, envolvendo valores categóricos e contínuos, assim como dependências hierárquicas (por exemplo, os parâmetros de um determinado algoritmo só fazem sentido se aquele algoritmo for escolhido). Há ainda nessa questão os parâmetros condicionais, que são utilizados ou não dependendo do valor atual de outros parâmetros. Por exemplo, o uso do parâmetro que define a porcentagem da base de dados que será utilizada para inicialização de centroides só faz sentido se o valor atual do parâmetro *sementes aleatórias* for igual a falso.

O Auto-WEKA (THORNTON *et al*., 2012) resolve o problema *CASH* para os algoritmos de classificação implementados na ferramenta WEKA (WEKA, 2015), mas até a presente data não disponibiliza solução para algoritmos de agrupamento de dados. Porém, é possível mapear seu funcionamento para entender como o problema *CASH* pode ser simplificado: um problema de seleção de melhor parâmetro, considerando a própria seleção do algoritmo como uma seleção de parâmetro em nível mais extremo. Os parâmetros seguintes serão condicionais ao algoritmo selecionado. Após esse mapeamento, é possível desenvolver uma aplicação voltada às tarefas de agrupamento.

1.2 Objetivos

Estudar formas de mapeamento do problema *CASH* para a tarefa de agrupamento de dados, com consequente elaboração e implementação de aprimoramentos para a ferramenta YADMT - *Yet Another Data Mining Tool* (BENFATTI *et al*., 2010), desenvolvida na Unioeste no campus de Cascavel, que possui para tarefa de agrupamento de dados os algoritmos *K-means*, *Self-Organizing Maps* (SOM) e Colônia de Formigas, além de um conjunto de algoritmos tidos como Métodos Hierárquicos. Mais especificamente, o objetivo do trabalho pode ser dividido em:

- Determinar a necessidade de se implementar um módulo capaz de selecionar o algoritmo de agrupamento de dados mais adequado para determinada base de dados;
- Elaborar um módulo capaz de configurar automaticamente valores de parâmetros relevantes dos algoritmos de agrupamento de dados, no intuito de gerar grupos com maior distinção do que aqueles gerados ao executar os algoritmos com os valores paramétricos na configuração *default* (padrão) e/ou alterados de maneira intuitiva;
- Implementar um algoritmo genético e um método de busca aleatória para realizar a configuração dos parâmetros; e
- Realizar experimentos para avaliar os aprimoramentos, utilizando a métrica Calinski-Harabasz (CALINSKI; HARABASZ, 1974) estudada e implementada neste trabalho para a comparação dos grupos gerados.

1.3 Organização do Documento

Além do Capítulo 1 que introduziu o tema e os conceitos básicos do assunto, este documento se divide da seguinte forma:

- O Capítulo 2 aborda a tarefa de Agrupamento de dados, citando os algoritmos desta tarefa encontrados na YADMT. Neste capítulo é dada uma explicação dos critérios de avaliação já existentes na ferramenta, além de um novo critério implementado para as comparações deste trabalho;
- O Capítulo 3 cita trabalhos correlatos, mostrando possíveis soluções para o problema proposto. Contempla a explicação do método de busca aleatória e do algoritmo genético utilizados neste trabalho.

- O Capítulo 4 apresenta a implementação da ferramenta YADMT, exibindo quais vantagens são adquiridas com os métodos de busca estudados;
- O Capítulo 5 descreve os testes realizados para avaliar os métodos implementados, bem como apresenta todos os resultados obtidos, para comparar os algoritmos de agrupamento de dados e métodos de busca;
- O Capítulo 6 faz uma breve consideração sobre o trabalho, com possíveis trabalhos futuros.

Capítulo 2

Agrupamento de Dados

De acordo com Fayyad *et al*. (1996), a mineração de dados é composta por três tarefas principais: Classificação; Regras de associação; e Agrupamento de dados. O Agrupamento de dados, ou *Clustering*, que é a tarefa objetivo deste estudo, caracteriza-se por ser de aprendizado não supervisionado. Portanto, não exige conhecimento *a priori* a respeito de classes ou rótulos em uma base de dados, assim como não existem agentes com a função de supervisionar o aprendizado durante o processo (BOSCARIOLI, 2008).

Essa tarefa busca em suas técnicas relacionar elementos e dividi-los em grupos, de forma que os elementos de um mesmo grupo sejam semelhantes entre si e diferentes dos elementos dos demais grupos. Os níveis de semelhança e diferença são determinados por medidas de dissimilaridades, que estabelecem graus de relação entre os elementos, sendo possível assim, agrupar os padrões de forma adequada.

Segundo Camilo e Silva (2009), esta tarefa difere da classificação pois não necessita que os registros sejam previamente categorizados. O mesmo autor afirma que o agrupamento de dados não tem a pretensão de classificar, estimar ou predizer o valor de uma variável, mas identificar os grupos de dados similares. A tarefa de agrupamento é, em muitos casos, essencial para tornar a classificação possível, pois por meio dela surgem bases de dados com padrões já categorizados, necessárias no aprendizado supervisionado.

Para Neto (2012), "agrupamento pode ser formulado como um problema de otimização com múltiplos objetivos, sendo que o algoritmo a se escolher e seus parâmetros[1] dependem dos dados e do tipo de grupos procurados". Neste trabalho, assim como na ferramenta YADMT, é de interesse a tarefa de agrupamento de dados exclusiva, na qual cada dado só pode ser inserido em um único e não ambíguo grupo. Cada grupo gerado por essa tarefa é

[1] Valores como a função de distância, o limiar de densidade ou o número esperado de grupos.

caracterizado por possuir um centroide, o qual é definido como o centro geométrico entre todos os seus elementos.

2.1 Algoritmos de Agrupamento da YADMT

É fato que nenhum método de agrupamento é capaz de processar de maneira satisfatória todos os tipos de estruturas formadas pelos dados (KOTSIANTIS; PINTELAS, 2004). Portanto, é adequado a uma ferramenta de mineração de dados possuir métodos de agrupamento com abordagens de diferentes categorias para aumentar as chances de encontrar melhores resultados. Segundo Boscarioli (2008), as técnicas de agrupamento são organizadas nas seguintes categorias:

- Métodos baseados em Particionamento;
- Métodos Hierárquicos;
- Métodos baseados em Densidade;
- Métodos baseados em Grades;
- Métodos baseados em Modelos.

Na ferramenta YADMT são encontrados dois algoritmos que se encaixam na categoria de Métodos baseados em Particionamento (*K-means* e Colônia de Formigas), um grupo de algoritmos da categoria de Métodos Hierárquicos[2] e um algoritmo que se encaixa na categoria de Métodos baseados em Modelos (*Self-Organizing Maps*).

2.2 Critérios de Validação

Os critérios de validação são essenciais para comparar os resultados obtidos na tarefa de agrupamento. Estes são definidos por valores que devem ser maximizados ou minimizados com a intenção de obter agrupamentos relativamente satisfatórios segundo estas métricas. Pode-se conceituar o valor de um critério como um indicativo de qualidade, ou seja, considerando um critério que deve ser maximizado, o agrupamento que possui maior valor de critério pode ser considerado aquele com maior qualidade. Portanto, é possível verificar dentre estruturas de grupos qual é a melhor pontuada.

[2] Os Métodos Hierárquicos fazem agrupamentos iterativos de pares de grupos. Estes agrupamentos são feitos por meio de ligações. Os tipos de ligações implementados na ferramenta YADMT são: Ligação Simples, Ligação Completa, Ligação Média e Método Ward.

Durante cada iteração da execução, tais valores são utilizados para indicar ao algoritmo se o processamento deve prosseguir ou encerrar. Os critérios de validação também contribuem para encontrar melhores parâmetros: após diversas execuções dos algoritmos de agrupamento, pode-se observar qual é o conjunto de parâmetros que gerou a melhor estrutura de grupos segundo algum critério, com a intenção de realizar a seleção automática.

Pode-se avaliar a qualidade de agrupamentos e estruturas de grupos de diversas formas (CAMPELO, 2013): por meio de **critérios internos**, que medem a qualidade de uma partição obtida sem considerar quaisquer informações externas, por meio da avaliação do grau de compatibilidade entre a estrutura dos grupos formados e os padrões, utilizando apenas os dados; por **critérios externos**, baseados na taxa de acerto calculada com o número de padrões agrupados em sua classe correspondente (informação *a priori*), ou; por meio de **critérios relativos**, que avaliam qual dentre *n* estruturas é a mais pontuada, segundo algum aspecto. Os critérios relativos geralmente são critérios internos que, pela quantificação determinam a qualidade de uma estrutura de grupos, por meio de comparações entre os grupos que formam esta estrutura (CAMPELO, 2013). A seguir serão descritos os critérios de validação utilizados pelos algoritmos da YADMT, assim como o critério escolhido para implementação.

2.2.1 Variância Intra-Grupos

A Variância Intra-Grupos (*V*), ou *WGSS* (*Within-Group Sum of Squares*), é um critério interno, calculado por meio do somatório dos quadrados das diferenças de cada padrão para seu respectivo centroide, sendo que este somatório repete uma vez para cada grupo. Quanto menor o valor encontrado pelo cálculo da Variância, maior é a semelhança dos padrões com seus centroides e melhor é a qualidade do agrupamento segundo este critério, sendo assim um critério que deve ser minimizado. A equação da Variância Intra-Grupos é dada pela Equação 2.1:

$$V = \sum_{i=1}^{k} \sum_{x \in C_i} (x - m_i)^2 \tag{2.1}$$

onde *k* é o número de grupos, *x* é um elemento pertencente ao *cluster* C_i e m_i é o centroide do grupo *i*.

2.2.2 Medida F

A medida *F* (*F*) é um índice externo, que recupera informações sobre as classes

previamente estabelecidas de uma base de dados, e realiza uma análise de precisão entre as classes desejadas e os grupos gerados. Tais ações são mapeadas para itens denominados memória *r* e precisão *p*. Cada classe *i* contém n_i padrões desejados; cada grupo *j* contém n_j padrões que foram atribuídos pelo algoritmo; o número de padrões da classe *i* que estão no grupo *j* é definido por n_{ij} (HANDL; KNOWLES; DORIGO, 2003). A memória *r*, para cada classe *i*, é definida pela equação $r(i,j) = \frac{n_{ij}}{n_i}$ e, a precisão *p*, para cada grupo *j*, é definida pela equação $p(i,j) = \frac{n_{ij}}{n_j}$.

Estes itens são utilizados para o cálculo de um subitem necessário no cálculo da medida *F*, que é dado pela Equação 2.2:

$$F(i,j) = \frac{(b^2+1).p(i,j).r(i,j)}{b^2.p(i,j) + r(i,j)} \tag{2.2}$$

O valor de *b* deve ser igual a "1" para manter pesos iguais para *p* e *r*. Por fim, a medida *F* é obtida por meio da Equação 2.3:

$$F = \sum_i \frac{n_i}{n} max_j\{F(i,j)\} \tag{2.3}$$

onde *n* é o número de padrões no conjunto de dados. *F* é uma função limitada ao intervalo [0, 1], devendo ser maximizada (HANDL; KNOWLES; DORIGO, 2003).

2.2.3 Índice Aleatório

O Índice Aleatório (*R*) é outra medida externa. O cálculo desta medida é feito por meio das variáveis *a*, *b*, *c* e *d*, encontradas por meio das seguintes equações:

$$a = |\{i,j|c_U(i) = c_U(j) \text{ ^ } c_V(i) = c_V(\text{j})\}| \tag{2.4}$$

$$b = |\{i,j|c_U(i) = c_U(j) \text{ ^ } c_V(i) \neq c_V(\text{j})\}| \tag{2.5}$$

$$c = |\{i,j|c_U(i) \neq c_U(j) \text{ ^ } c_V(i) = c_V(\text{j})\}| \tag{2.6}$$

$$d = |\{i,j|c_U(i) \neq c_U(j) \text{ ^ } c_V(i) \neq c_V(\text{j})\}| \tag{2.7}$$

onde $c_U(i)$ e $c_U(j)$ representam as classificações corretas e $c_V(i)$ e $c_V(\text{j})$ representam os grupos encontrados pelo algoritmo de agrupamento (HANDL; KNOWLES; DORIGO, 2003). Finalmente, o índice é calculado pela Equação 2.8:

$$R = \frac{a + d}{a + b + c + d} \tag{2.8}$$

Assim como na medida *F*, o Índice Aleatório é uma função limitada ao intervalo [0, 1], que deve ser maximizada (HANDL; KNOWLES; DORIGO, 2003).

2.2.4 Índice Dunn

O Índice Dunn (*D*) é do tipo interno e determina para um conjunto de grupos qual é a razão mínima entre a distância dos centroides de dois grupos e o diâmetro de outro grupo (DUNN, 1973). A Equação 2.9 é definida a seguir:

$$D = min_{b,c \in C} \left[\frac{\delta(m_b, m_c)}{max_{a \in C} \left[diam(a) \right]} \right] \tag{2.9}$$

com *C* sendo o conjunto composto por todos os grupos, $diam(a)$ é o diâmetro de um grupo a e $\delta(m_b, m_c)$ é a distância dos grupos *b* e *c*, que pode ser encontrada pelo valor máximo da distância intra-grupo. *D* é uma função que deve ser maximizada (DUNN, 1973).

2.2.5 Calinski-Harabasz

Para este trabalho, interessou-se por um método de avaliação de agrupamentos que não exige conhecimento *a priori*, para poder processar também bases de dados sem a coluna definida como rótulo. Para avaliação dos grupos gerados foi adotado um critério do tipo interno e relativo, sendo chamado de critério Calinski-Harabasz (CALINSKI; HARABASZ, 1974). Ele avalia as diferenças dos padrões de um determinado grupo para seu centroide, determinando a similaridade intra-grupos, assim como avalia as diferenças dos centroides dos grupos para um centroide global, determinando a dissimilaridade inter-grupos.

Segundo Vendramin e Campelo (2008), a escolha é justificada por ser uma forma eficiente de avaliar grupos sem conhecimento *a priori*. Uma vantagem é que este critério independe do algoritmo aplicado em uma base de dados, pois o que será analisado são dados internos dos grupos gerados. Portanto, com esta medida é possível comparar algoritmos aplicados em uma mesma base de dados, determinando qual dentre eles obtém melhores resultados com relação aos grupos gerados. Este critério também pode ser considerado uma forma mais completa de avaliação, se comparado à Variância Intra-Grupos, que desconsidera as distâncias entre grupos.

Para obter o valor do critério Calinski-Harabasz (*CH*), é necessário calcular o *WGSS* (*Within-Group Sum of Squares*) e o *BGSS* (*Between-Group Sum of Squares*). Como definido, o *WGSS* é calculado pela Equação 2.1. O *BGSS* é o somatório do quadrado da diferença do centroide de cada grupo para o centroide global, multiplicado pelo número de elementos em cada grupo. O cálculo do *BGSS* é dado pela Equação 2.10:

$$BGSS = \sum_{i=1}^{k} n_i * (m_i - m)^2 \tag{2.10}$$

onde n_i é o número de padrões no grupo *i*, e *m* é o centroide global, calculado pela média (vetor de médias) entre os centroides de todos os grupos. Cada padrão possui um número indeterminado de atributos que varia entre as bases de dados. Portanto, deve-se considerar que cada padrão e cada centroide são na verdade um vetor, contendo valores respectivos aos atributos.

Por fim, o valor do critério Calinski-Harabasz é calculado pela Equação 2.11:

$$CH = \frac{BGSS * (n-k)}{WGSS * (k-1)} \tag{2.11}$$

com *k* sendo o número de grupos e *n* o número de padrões da base de dados. O valor *WGSS* trata-se de diferenças internas aos grupos, portanto, quanto menores forem os valores obtidos, maior será a similaridade interna nos grupos. O valor *BGSS* trata de diferenças entre grupos, com valores mais altos significando grupos com altas dissimilaridades entre eles (CALINSKI; HARABASZ, 1974). Observado isso, pode-se notar que a função do critério Calinski-Harabasz deve ser maximizada para alcançar maiores semelhanças intra-grupos e diferenças inter-grupos.

2.3 Considerações Finais

Separar padrões e agrupá-los é uma atividade cotidiana. A automatização deste processo deu origem à tarefa de agrupamento de dados, capaz de tratar uma quantidade robusta de dados e facilitar uma atividade que em muitos casos seria inviável se feita manualmente. Esta é de fato uma tarefa importante no meio dos negócios, sendo útil para garantir uma tomada de decisão mais rápida e segura.

Este capítulo tratou, de forma introdutória, conceitos de agrupamento de dados, assim como citou quais são os algoritmos encontrados na ferramenta YADMT e explicou os critérios de validação implementados, importantes na avaliação de agrupamentos e, portanto, na solução do problema *CASH*.

Capítulo 3

Soluções para o Problema *CASH*

Cada vez mais surgem usuários não especialistas de ferramentas de aprendizado de máquina que exigem soluções para uma utilização prática (THORNTON *et al*., 2012). Tais usuários possuem a possibilidade de utilizar diversas ferramentas livres, uma vez que estas exigem ao menos três interações: selecionar uma base de dados; selecionar um algoritmo de aprendizado de máquina e; configurá-lo. Pode ser muito desafiador realizar as últimas duas tarefas com um grau de liberdade tão elevado. Por isso, surgiram trabalhos para solucionar este problema, que estudam como realizar estas tarefas de forma a reduzir ao máximo a interação manual, aproximando os resultados aos melhores possíveis para uma determinada base de dados.

Como cada um dos algoritmos de aprendizado de máquina possui parâmetros distintos, não é possível desenvolver uma única solução para o problema *CASH*, as soluções e implementações devem ser específicas e tratar as particularidades de um certo algoritmo. Portanto, ao elaborar uma nova solução para uma ferramenta, é interessante estudar trabalhos correlatos para entender as diferentes abordagens possíveis.

O Auto-WEKA (THORNTON *et al*., 2012) tem foco em problemas de classificação: aprender uma função cuja imagem é finita. Um algoritmo de aprendizado *A* mapeia um conjunto de pontos de treinamento de dados, reunidos pela própria função e, relaciona um valor do domínio com um único do contradomínio, ou seja, consiste em treinar um modelo que, considerando um conjunto de dados de entrada, seja capaz de associar a esse conjunto uma classe dentre um número finito de classes bem definidas. Mas um algoritmo *A* pode ser alterado por seus próprios parâmetros, alterando a função e, assim, os resultados.

Este trabalho divide o problema *CASH* em duas categorias, a Seleção de Algoritmo e a Seleção de Parâmetros. A Seleção de Algoritmo considera um conjunto de algoritmos de

aprendizado *A* e um conjunto de dados, com o objetivo de indicar o algoritmo com maior desempenho para os dados selecionados.

A Seleção de Parâmetros considera um algoritmo *A*, um conjunto de instâncias *I* e um critério *c*, com o objetivo de encontrar a configuração paramétrica de *A* com o melhor valor de *c* em *I*. Sendo que *c* pode ser tempo computacional ou, no caso de solução ótima, o erro encontrado após um tempo pré-estabelecido, ou ainda, qualquer outro critério estabelecido.

Por mais que o problema seja dividido em duas categorias, a própria seleção do algoritmo poderia ser considerada como uma seleção de parâmetro, aumentando a dimensão do espaço de busca em mais um nível, mas a escolha desta arquitetura como resolução do problema acarreta em um grande aumento de custo computacional e tempo de processamento. Uma alternativa, que economiza este custo e tempo, é estudar uma forma heurística para indicar o possível melhor algoritmo.

Esta heurística é baseada em uma análise dos resultados obtidos pelos múltiplos processamentos dos algoritmos em diferentes bases de dados. O objetivo da análise é identificar, de maneira empírica, qual algoritmo obtém melhores resultados na maior parte dos casos, para que o usuário dê prioridade a sua utilização quando não souber qual algoritmo escolher. O relato completo sobre os resultados encontrados nesta análise pode ser visto no Capítulo 5.

Por mais que cada algoritmo de aprendizado de máquina exija uma implementação diferente para o processo de seleção de parâmetros, o fluxo de passos para realizar tal tarefa pode ser descrito de forma generalizada. Considerando um algoritmo *A*, um conjunto de dados *D*, um critério *c*, λ como um conjunto de parâmetros candidatos e Hλ uma lista com todos os conjuntos λ testados, o Algoritmo 3.1 descreve o processo.

Algoritmo 3.1 – Adaptação do Algoritmo *Sequential Model-Based Optimization* [*SMBO*] para resolução do problema *CASH*

Passo 1: Inicializar o gerador de configurações λ; Hλ ← ∅
Passo 2: **enquanto** o tempo de processamento não esgotar, **faça**
Passo 3: λ ← nova configuração candidata
Passo 4: Computar *c* = executa(*A*, λ, *D*)
Passo 5: Hλ ← Hλ ∪ {(λ, *c*)}
Passo 6: **fim-enquanto**

```
Passo 7:  retorna λ de ℍ com melhor c
```

Fonte: Adaptado de (THORNTON *et al*., 2012)

O Algoritmo 3.1 foi criado para resolução do problema *CASH* em tarefas de classificação e aqui adaptado de forma a ser compatível com a tarefa de agrupamento. Ele não descreve o gerador de configurações, que tem parte crucial no desempenho do processo de busca completo. Entretanto, servirá de base para as implementações de busca aleatória e algoritmos genéticos que serão descritas neste capítulo.

É necessário realizar diversos testes empíricos com combinações entre os valores dos parâmetros para descobrir quais são os melhores valores de configuração na geração de grupos para cada um dos algoritmos. A forma padrão de realizar a otimização de parâmetros é por meio de busca exaustiva (BERGSTRA; BENGIO, 2012), que é simplesmente uma busca por força bruta em um subconjunto do espaço paramétrico, sendo que este subconjunto pode ser especificado manualmente. Uma vez que a busca exaustiva é um método de força bruta e, por isso, potencialmente custoso, muitas alternativas foram propostas. Este trabalho foca em duas alternativas, Busca Aleatória e Algoritmos Genéticos, para ser possível realizar uma comparação entre as duas implementações e também aumentar as chances de obter melhores resultados na execução da tarefa de agrupamento. Ambas as abordagens foram escolhidas em virtude da existência de trabalhos[3] que utilizaram estes métodos para realizar seleção automática de parâmetros para agrupamento de dados.

A função que determina o conjunto de valores paramétricos ótimo tem uma dimensionalidade eficaz mais baixa do que a dimensionalidade total do próprio conjunto de parâmetros (CAFLISCH; OWEN, 1997). Isso significa que se um algoritmo possui *n* parâmetros, o número de parâmetros que farão diferença significativa no resultado é menor que *n*. Na Tabela 3.1 pode-se observar uma relação dos parâmetros encontrados nos algoritmos de agrupamento de dados da ferramenta YADMT:

Tabela 3.1 – Parâmetros encontrados nos Algoritmos de Agrupamento da Ferramenta YADMT

Algoritmo	**Nº De Grupos**	**Função de Distância**	**Máximo de Iterações**	**Parâmetros Únicos***
K-means	X	X	X	Sim
SOM		X	X	Sim
Formigas		X	X	Sim

***Determina se o Algoritmo em questão possui parâmetros não encontrados nos outros Algoritmos.**

É uma tarefa simples encontrar os melhores resultados para parâmetros que assumem

[3] (BERGSTRA; BENGIO, 2012) para busca aleatória e (CORRÊA, 2010) para algoritmos genéticos.

valores discretos, podendo ser resolvida com uma busca exaustiva, de baixo custo, se comparada ao alto custo de uma busca exaustiva com parâmetros de valores contínuos.

Os três primeiros parâmetros exibidos na Tabela 3.1 assumem valores discretos, não agregando dificuldade na resolução do problema proposto. Porém, por meio de testes empíricos, determinou-se que os algoritmos possuem itens que assumem valores contínuos que influenciam os resultados com bastante intensidade. Pode-se citar como exemplos os valores iniciais dos centroides no algoritmo *K-Means*, que podem ser determinados aleatoriamente, ou de acordo com uma porcentagem da base de dados e, o parâmetro *taxa de aprendizado* no algoritmo *SOM*.

É possível eliminar a escolha manual destes parâmetros contínuos por meio da pesquisa dos melhores valores, utilizando algum método de busca. Todos os algoritmos de agrupamento estudados contêm casos similares de determinação de parâmetros com valores contínuos.

3.1 Busca Aleatória

Esta é uma técnica capaz de testar um número alto de combinações entre os parâmetros pré-determinados, e encontrar os respectivos valores de configuração que geram grupos mais pontuados segundo algum critério pré-estabelecido. Este método trabalha atribuindo valores aleatórios aos parâmetros para se aproximar de melhores resultados, sendo denominado *random search* (busca aleatória). Este tipo de abordagem se mostra mais eficiente do que uma busca exaustiva ou busca em grade, pois a busca aleatória necessita de muito menos tempo e processamento para encontrar resultados em espaços de busca de dimensões mais altas (BERGSTRA; BENGIO, 2012).

A Figura 3.1 ilustra um comparativo entre as abordagens de busca exaustiva e busca aleatória, com nove tentativas de otimizar a função $f(x, y) = g(x) + h(y) \approx g(x)$ com baixa dimensionalidade efetiva. Acima de cada quadrado, a função $g(x)$ é mostrada em verde, e à esquerda de cada quadrado, a função $h(y)$ é mostrada em amarelo. Com a busca exaustiva, nove tentativas testam $g(x)$ em somente três lugares distintos. Com a busca aleatória, todas as tentativas testam $g(x)$ em nove lugares distintos, aumentando as chances de se obter melhores resultados. Esta falha da busca exaustiva é mais uma regra do que uma exceção, em otimização de espaços de alta dimensão (BERGSTRA; BENGIO, 2012).

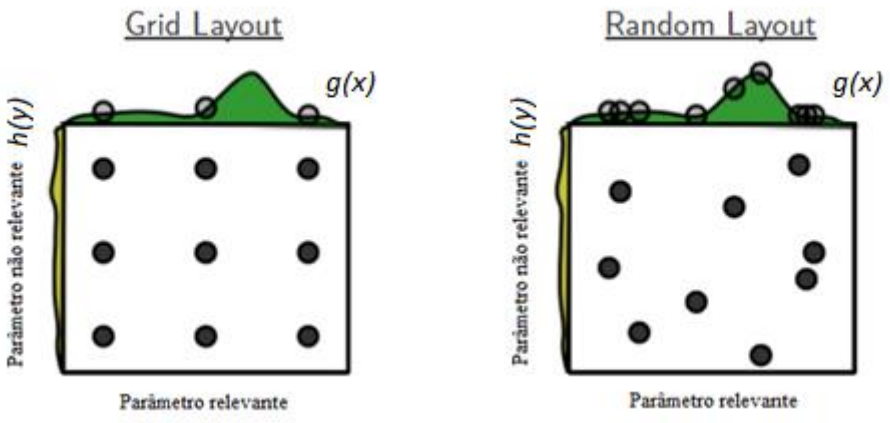

Figura 3.1: Comparativo entre *grid layout* e *random layout*

Fonte: Adaptada de (BERGSTRA; BENGIO, 2012)

A busca aleatória implementada realiza uma pesquisa em duas etapas:

- Primeiro são criados e testados diversos conjuntos de parâmetros, todos contendo valores gerados aleatoriamente segundo o método gerador de configurações. Em seguida, uma lista é preenchida com os conjuntos de parâmetros que obtiveram melhores resultados segundo o critério Calinski-Harabasz. A parada de adição dos conjuntos na lista acontece quando um tamanho máximo da lista é atingido ou quando não há mais conjuntos cujos valores do critério seja maior ou igual à metade do menor valor na lista (a lista só é criada após a computação de todos os conjuntos testados);

- No segundo nível, são realizadas buscas pontuais, mas ainda aleatórias, nas proximidades de cada conjunto de parâmetros da lista. Vence o conjunto de parâmetros que obter o maior valor para o critério Calinski-Harabasz.

3.2 Algoritmo Genético

Um Algoritmo Genético (AG) é um tipo de técnica de computação evolucionária que se baseia no princípio da seleção natural para abordar uma série de problemas, considerando especialmente os de otimização. Cada algoritmo genético é robusto, genérico e facilmente adaptável, sendo uma técnica amplamente estudada e utilizada nas mais diversas áreas (LUCAS, 2002). Esta técnica já foi utilizada para resolver problemas de configuração de parâmetros, conforme pode ser visto em (CORRÊA, 2010).

Estes algoritmos têm grande embasamento no darwinismo, que teoriza o processo de evolução das espécies por meio da sobrevivência dos seres mais fortes e aptos a se

reproduzirem. A característica que mais distingue o darwinismo das outras teorias evolucionárias é que a evolução é vista como uma função de alteração da população, não de um único indivíduo (RUSE, 1982).

As técnicas de busca e otimização mais recorrentes iniciam com somente um indivíduo, enquanto que as técnicas de computação evolucionária iniciam com um conjunto de indivíduos denominado população. Assim, é possível operar sobre uma população de indivíduos em paralelo. Desta forma, pode-se fazer a busca em diferentes áreas do espaço de busca, alocando um número apropriado de indivíduos para a busca em cada região (CARVALHO, 2009).

As etapas dos AGs foram propostas por Holland (1975) como sendo avaliação, seleção, cruzamento, mutação, atualização e finalização. A Figura 3.2 ilustra o ciclo de um algoritmo genético típico:

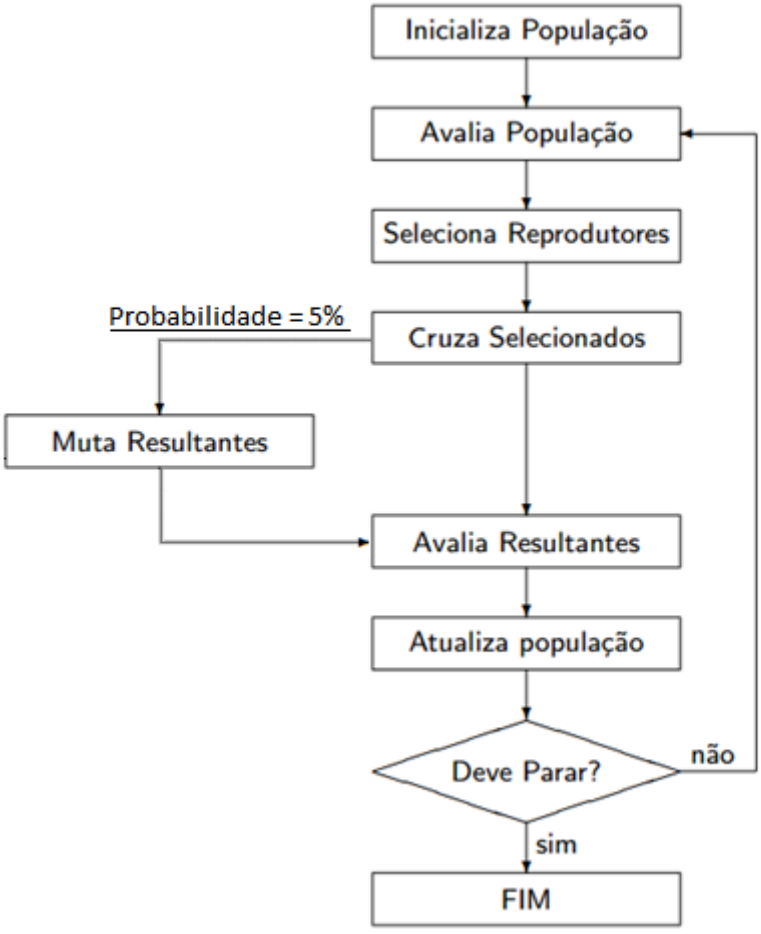

Figura 3.2: O ciclo tradicional de um Algoritmo Genético

Fonte: (LUCAS, 2002)

Inicialmente, valores são atribuídos aleatoriamente para cada um dos indivíduos que compõem a população. Enquanto o processo evolutivo tiver continuidade, esta população é avaliada, de forma que cada indivíduo recebe uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente (CARVALHO, 2009). Uma porcentagem dos mais aptos é mantida, enquanto que outros são descartados.

A maior parte dos indivíduos mantidos em cada processo de seleção possivelmente sofrerão modificações em suas características fundamentais. Estas alterações poderão ocorrer por meio de mutações e/ou cruzamentos (*crossover*), com o objetivo de gerar descendentes diferentes para a próxima geração, enquanto que uma menor parte permanecerá inalterada. Este processo de reprodução se repete até que um indivíduo satisfatório, segundo a condição de parada, seja encontrado (CARVALHO, 2009).

A complexidade do problema, assim como os recursos computacionais disponíveis, determinarão quantos e quais indivíduos serão alvos dos princípios de seleção e reprodução, em cada uma das iterações. A seleção escolhe cada um dos indivíduos que participarão da etapa de reprodução, para assim gerar um número determinado de descendentes na próxima geração, com uma probabilidade determinada pela seu índice de aptidão (CARVALHO, 2009).

A ideia fundamental por trás do funcionamento dos AGs é fazer com que um conjunto inicial de indivíduos, após muitas gerações e por meio de um critério de seleção, gere indivíduos que resolvam o problema inicial de maneira mais eficiente. Os métodos de seleção dão prioridade aos indivíduos mais aptos segundo os critérios, mas é importante manter a diversidade da população, para aumentar as chances de se encontrar a solução ótima. Portanto, é interessante que os métodos de seleção tenham um limite no grau de influência que sofrem por parte dos indivíduos (CARVALHO, 2009).

O Método da Roleta utiliza um sorteio para decidir quais indivíduos farão parte da próxima geração. Assim, as chances de se ter uma população diversificada são maiores do que quando indivíduos são escolhidos meramente por ordem de pontuação. A Figura 3.3 exibe um exemplo com um conjunto de indivíduos e suas respectivas aptidões:

	Indivíduo S_i	Aptidão $f(S_i)$	Aptidão Relativa
S_1	10110	2.23	0.14
S_2	11000	7.27	0.47
S_3	11110	1.05	0.07
S_4	01001	3.35	0.21
S_5	00110	1.69	0.11

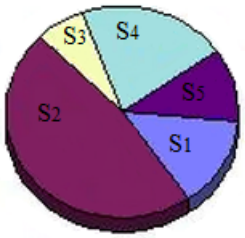

Figura 3.3: Exemplo de um conjunto de Indivíduos e suas respectivas aptidões

Fonte: (CARVALHO, 2009)

Cada indivíduo contém uma parte na roleta, proporcional à sua aptidão, assim, indivíduos com altas aptidões terão mais chances de participar da próxima geração. O número de vezes que a roleta é girada depende da quantidade de indivíduos participantes do sorteio (CARVALHO, 2009).

Para gerar populações com melhores aptidões ao longo do tempo, é necessário realizar duas operações, o cruzamento e a mutação (com 5% de chance de ocorrer a mutação para cada indivíduo não elitista), (CARVALHO, 2009). Ambas operações têm intenção de alterar a próxima geração, mas sem perder as características dos pais. Uma exceção é aberta aos melhores indivíduos, denominados elitistas, que farão parte da próxima geração sem sofrerem quaisquer alterações. Considerando o tempo atual *t*, *d* como o tempo determinado para finalizar o processo e *P* a população, o Algoritmo 3.2 exemplifica um algoritmo genético.

Algoritmo 3.2 – Exemplo de Algoritmo Genético

```
Passo 1:   t ← 0
Passo 2:   inicia população (P,t)
Passo 3:   avaliação (P,t)
Passo 4:   repita até t = d
Passo 5:     t ← t + 1
Passo 6:     seleção dos pais (P,t)
Passo 7:     recombinação (P,t)
Passo 8:     mutação(P,t) – 5% de chance por indivíduo
Passo 9:     avaliação(P,t)
Passo 10:    sobrevivem(P,t)
Passo 11:  fim-repita
```

Fonte: (CARVALHO, 2009)

3.3 Considerações Finais

O aumento no número de usuários não especialistas na área de mineração de dados gera uma nova demanda por parte das ferramentas: auxiliar o usuário na escolha e/ou configuração dos algoritmos, trazendo uma melhora nos resultados e tornando o processo mais atrativo. Estas vantagens também são úteis aos usuários mais experientes, já que muitos experimentos da área são empíricos e imprevisíveis.

Este capítulo tratou de duas possíveis soluções adaptadas do problema *CASH*, o método de busca aleatória, que se mostra melhor que a busca exaustiva, conforme descrito por Bergstra e Bengio (2012), assim como algoritmo genético. O Capítulo 5 mostra, por meio de testes, qual dos dois métodos apresentou melhores resultados.

Capítulo 4

Implementação na Ferramenta YADMT

Os métodos de busca aleatória e busca por algoritmo genético foram implementados na ferramenta YADMT – *Yet Another Data Mining Tool* (BENFATTI *et al*., 2010), que vem sendo desenvolvida na Universidade Estadual do Oeste do Paraná (UNIOESTE) pelo Grupo de Pesquisa em Inteligência Aplicada (GIA).

Os métodos foram integrados aos módulos de agrupamento de dados previamente implementados, de tal maneira que os demais módulos da ferramenta não são afetados. Cada método foi implementado respeitando os padrões de projeto já existentes na ferramenta, de forma que seu processamento não se tornou mais lento ou ineficiente.

Uma das vantagens desta implementação é não afetar de maneira negativa o trabalho já realizado, pois o usuário ainda tem a possibilidade de configurar um algoritmo de agrupamento de dados de forma totalmente manual.

A ferramenta YADMT trabalha com arquivos no formato *Attribute-Relation File Format* (ARFF), desenvolvido pelo departamento de Ciência da Computação da Universidade de Waikato, para uso na ferramenta WEKA (WEKA, 2015). Também há a opção de realizar a aquisição de dados por meio de SGBD – Sistema Gerenciador de Banco de Dados. Na YADMT esta funcionalidade é executada por meio do PostgreSQL, com o serviço de conectividade JDBC (*Java Data Base Connectivity*).

A tela inicial do sistema tem o objetivo de realizar a aquisição da base de dados, seja por SGBD ou em formato ARFF, então, é possível selecionar quais atributos serão utilizados pelos algoritmos e também realizar uma visualização dos dados em forma de tabela.

Neste trabalho o interesse se concentra em obter os dados por meio de arquivos de formato ARFF, pois estes arquivos podem ser facilmente criados a partir dos conjuntos de dados

obtidos no repositório da UCI (LICHMAN, 2013). A Figura 4.1 exibe a aquisição de dados na tela inicial.

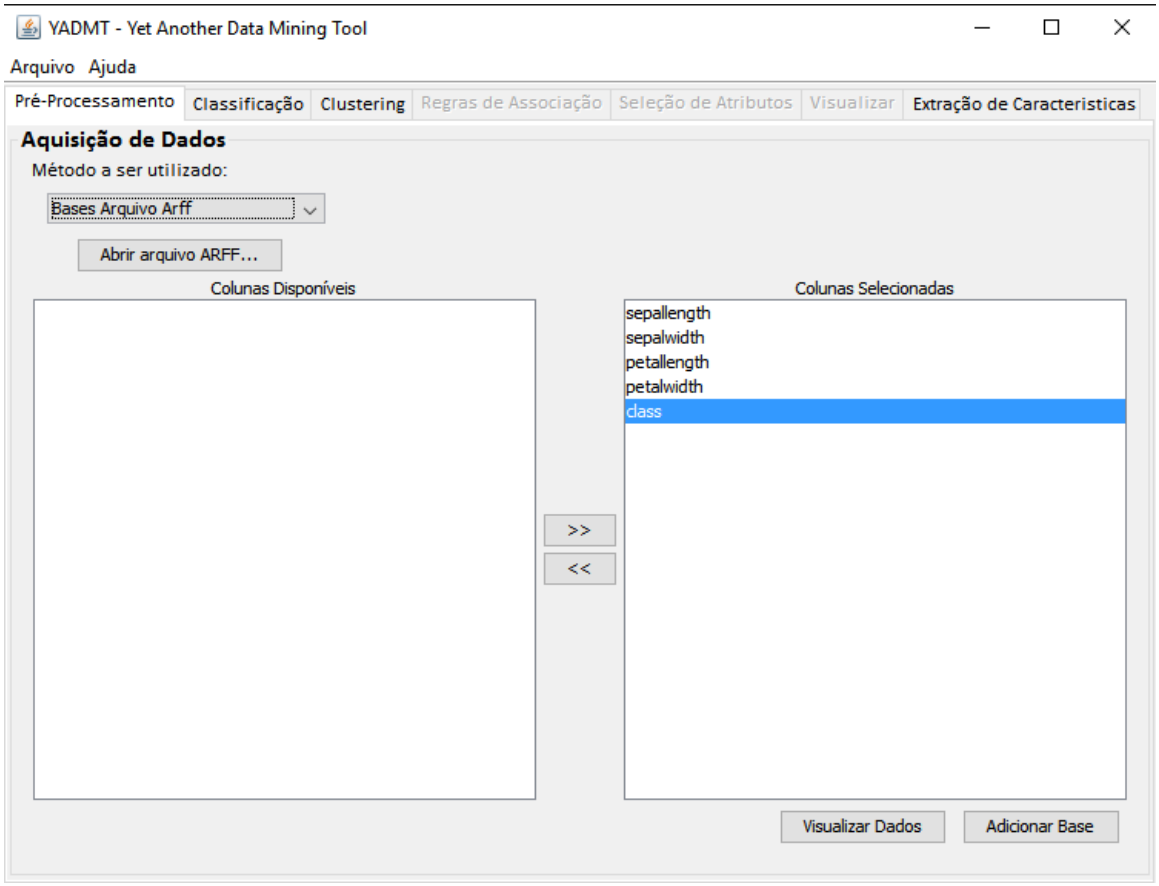

Figura 4.1 – Tela inicial da ferramenta YADMT

Nesta tela, deve-se especificar o local do arquivo ARFF a ser carregado. Após carregar uma base de dados, é possível executar os algoritmos de aprendizado de máquina compatíveis com os tipos de dados do repositório. Na aba *Clustering*, deve-se escolher um dos algoritmos de agrupamento.

Uma vez que o fluxo de ações por parte do usuário é semelhante para todos os algoritmos de agrupamento, para exibir o processo de execução, escolheu-se o algoritmo *K-Means* como exemplo, cuja Figura 4.2 exibe sua tela inicial.

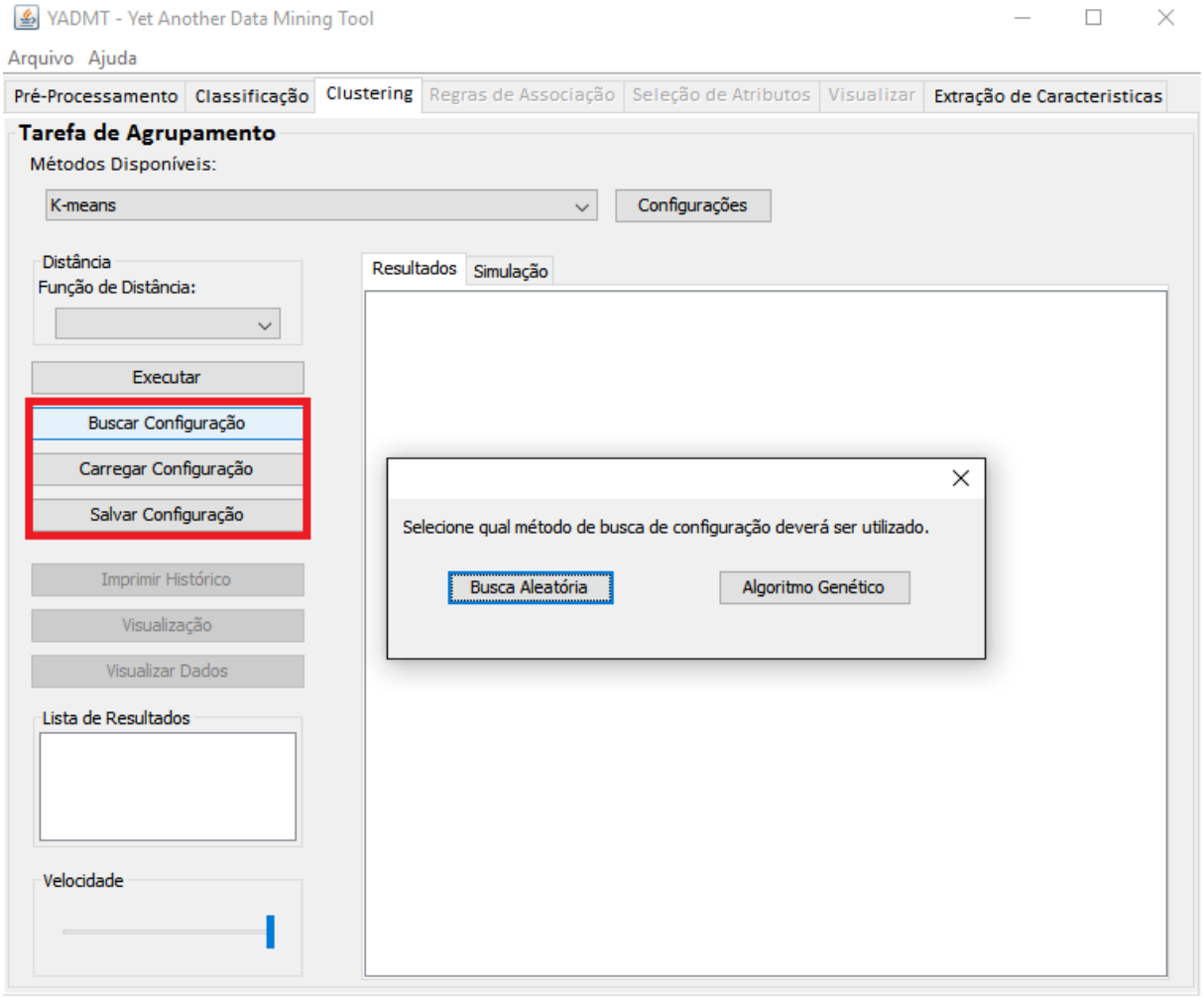

Figura 4.2 – Tela inicial do algoritmo *K-Means*

Nesta tela é possível configurar o algoritmo de forma manual, sendo necessário estabelecer os valores de parâmetros difíceis de se configurar de maneira intuitiva. Após este procedimento pode-se pressionar o botão *Executar* para que o algoritmo receba os parâmetros marcados e se inicie.

Na caixa destacada estão três botões que fazem parte do processo de seleção automática de parâmetros: *Buscar Configuração*; *Carregar Configuração*; e *Salvar Configuração*. Como demonstrado na imagem, ao pressionar o primeiro botão, uma mensagem é exibida, pedindo para selecionar um dos dois métodos de busca. Então o procedimento de busca por configuração é iniciado e ao final, os resultados são exibidos como na Figura 4.3.

```
===================== Informações =====================
                        YADMT.Clustering.AC
        Base: iris.arff
        Número de Instâncias: 150
        Atributos: 4
        Classes:
               Iris-setosa
               Iris-versicolor
               Iris-virginica

======== Módulo de Seleção Automática de Parâmetros ========

       Resultados encontrados:

       Número de Grupos: 3
       Parada Automática: true
       Seeds Aleatórios: true
       Porcentagem Seeds: 0.6586205869239419
       Número de Iterações: 150
       Distância Selecionada: Euclidiana
       Critério Calinski-Harabasz: 563,2869
```

Figura 4.3 – Resultado do processo de busca com o algoritmo *K-Means*

Os resultados incluem os parâmetros básicos, bem como parâmetros que não podem ser definidos de forma direta, como os centroides (centroides podem ser considerados parâmetros somente com o algoritmo *K-Means*). De forma manual, os centroides só podem ser definidos como valores aleatórios ou baseados em uma porcentagem da base de dados. A Figura 4.4 exibe estes valores.

```
==================Resultados=====================

Centroides:


Centroide: 0
Atributos: 5.88360655737705 2.740983606557377 4.388524590163935 1.4344262295081966

Centroide: 1
Atributos: 5.005999999999999 3.4180000000000006 1.464 0.2439999999999999

Centroide: 2
Atributos: 6.853846153846153 3.0769230769230766 5.715384615384615 2.053846153846153
```

Figura 4.4 – Centroides obtidos pelo algoritmo *K-Means*

Uma vantagem deste procedimento de busca de configuração é que, após a execução, é possível salvar estes valores e posteriormente carregá-los. No exemplo do algoritmo *K-Means*, quando uma configuração for carregada, os centroides iniciais serão definidos como aqueles resultantes do processo de busca.

Ao carregar uma configuração de parâmetros, não é necessário realizar uma busca por novos valores, por meio do botão *Executar* é possível encontrar o mesmo resultado obtido pela busca de configuração.

Finalmente, após o processo de busca ser concluído, uma aba é apresentada, na qual é possível observar um gráfico de desempenho, como representado na Figura 4.5.

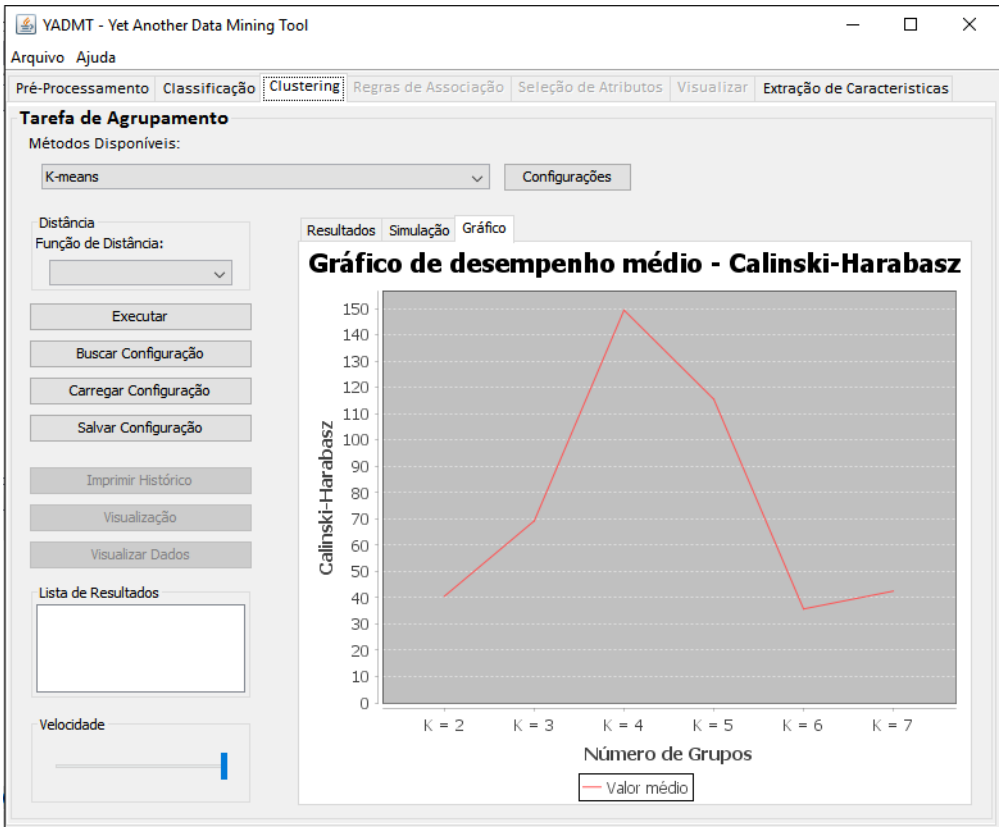

Figura 4.5 – Desempenho médio com o critério Calinski-Harabasz

Este gráfico exibe somente um parâmetro, mas também considera a variação dos outros parâmetros, pois efetua o cálculo da média de todos os valores do critério Calinski-Harabasz obtidos para cada um dos valores do parâmetro exibido. Desta forma, é possível ter uma ideia da localização de máximos locais e globais para determinado parâmetro.

4.1 Considerações Finais

A ferramenta YADMT oferece opções de execução, para que a experiência possa variar em questão de exigir mais interação, com a configuração realizada pelo usuário, ou menos

interação, com a busca pela configuração. A ferramenta ainda possibilita salvar e carregar a melhor configuração encontrada, para evitar passar novamente pelo mesmo processamento.

Em contrapartida aos detalhes que são exibidos sobre a configuração vencedora, os gráficos gerados pela busca de parâmetros disponibilizam uma forma de visualizar todos os resultados armazenados de maneira simultânea.

Capítulo 5

Resultados e Discussão

É interessante e essencial para o trabalho apresentar uma forma de comparação entre as técnicas de busca citadas no capítulo anterior. Para isso, foram realizadas múltiplas execuções dos algoritmos implementados em 12 bases de dados apropriadas à tarefa de agrupamento, adquiridas no repositório da UCI (LICHMAN, 2013). A Tabela 5.1 traz informações básicas de cada uma dessas bases de dados.

Tabela 5.1 – Bases de dados selecionadas para os testes empíricos

Base de dados	**Número de Atributos***	**Número de Instâncias**
3D Road Network (versão resumida)	5	4000
Cuff-Less Blood Pressure Estimation	3	12000
Diabetes	9	767
Geographical Original of Music	68	1059
Glass	10	213
Ionosphere	35	350
Íris	5	150
Stone Flakes	8	79
Student Performance	33	649
Synthetic Control Chart Time	60	600
Wine	14	177
Yeast	8	1484

***A última coluna em todas as bases de dados remete às classes.**

Ao escolher as bases de dados, o objetivo foi satisfazer as restrições de tipos de dados dos algoritmos, assim como escolher repositórios com quantidades de atributos e instâncias variadas. Alguns repositórios, tal como o *Cuff-Less Blood Pressure Estimation*, possuem um número pequeno de atributos. Em contrapartida, alguns repositórios, como *Synthetic Control Chart Time* e *Geographical Original of Music* possuem um número alto, com 60 e 68 atributos respectivamente. O mesmo ocorre com relação ao número de instâncias, com um mínimo de 79 instâncias, variando até o máximo de 12.000 instâncias.

Tal diversidade nas bases de dados é importante para garantir que os algoritmos sejam testados de maneira abrangente. De fato, um algoritmo pode ter baixo desempenho em certa base de dados, mas ser o melhor entre todos os testados com outro repositório.

Também é importante notar o tempo de processamento necessário para cada um dos algoritmos, tal fator pode ser decisivo na escolha dos usuários quando em necessidade de realizar experimentos com bases de dados mais complexas.

5.1 Resultados dos Experimentos

O ambiente de execução dos testes é formado por:

- Sistema Operacional *Windows 10 Home Single Language 64 bits*;
- Processador Intel® Core™ i5-5200U CPU @ 2.20GHz (4 CPUs);
- Memória RAM total: 8.192 MB;
- IDE NetBeans 8.1.

Cada processo de busca é composto por 5.000 iterações, sendo que cada iteração é de fato formada por uma execução do algoritmo de agrupamento de dados em questão. Há ainda um segundo critério de parada, que é acionado quando a variação no resultado é muito pequena em um número grande de iterações.

A Tabela 5.2 representa os dados obtidos com os testes realizados utilizando o método de busca aleatória.

Tabela 5.2 - Resultados para o critério Calinski-Harabasz via escolha padrão e busca aleatória

Base de dados	**Algoritmos de Agrupamento de Dados - YADMT**					
	K-Means		**Self-Organazing Maps**		**Colônia Formigas**	
	Escolha Padrão	Busca Aleatória	Escolha Padrão	Busca Aleatória	Escolha Padrão	Busca Aleatória

3D Road Network	331,07	**5.683,46**	174,79	**1.102,60**	42,35	**124,77**
Cuff-Less Blood Pressure Estimation	1.370,65	**45.334,75**	2.897,01	**23.961,93**	113,20	**525,59**
Diabetes	144,58	**267,41**	102,89	**175,18**	6,64	**28,72**
Geographical Original of Music	138,46	**392,05**	149,15	**391,47**	19,34	**56,13**
Glass	63,64	**184,00**	177,63	**352,92**	1,17	**2,54**
Ionosphere	31,12	**78,91**	29,46	**45,48**	3,42	**5,02**
Íris	49,05	**563,28**	136,07	**397,21**	12,43	**39,24**
Stone Flakes	23,50	**84,14**	25,82	**82,76**	1,26	**4,21**
Student Performance	55,87	**106,89**	52,11	**144,51**	7,58	**28,87**
Synthetic Control Chart Time	40,62	**218,30**	82,42	**94,03**	14,48	**16,01**
Wine	53,35	**83,29**	26,66	**43,62**	4,76	**9,79**
Yeast	168,47	**1.011,45**	149,44	**1.094,09**	13,98	**87,58**

Maiores valores para o critério Calinski-Harabasz (CALINSKI; HARABASZ, 1974) determinam grupos relativamente melhores (considerando a mesma base de dados). Portanto, a busca aleatória obteve melhores resultados, tendo superado a escolha padrão em todos os casos.

Também é observável a diferença de desempenho dos algoritmos segundo o critério *Calinski-Harabasz*, *K-Means* se mostrou melhor que os outros dois algoritmos em 09 casos. Em seguida, o algoritmo *SOM* obteve melhores resultados nos 03 casos restantes, enquanto que o algoritmo *Colônia de Formigas* não obteve o melhor resultado em nenhum dos casos.

A Tabela 5.3 representa os dados obtidos com os testes realizados utilizando o método de busca baseado em Algoritmo Genético.

Tabela 5.3 – Resultados para o critério Calinski-Harabasz via escolha padrão e Algoritmo Genético

Base de dados	**Algoritmos de Agrupamento de Dados - YADMT**					
	K-Means		**Self-Organizing Maps**		**Colônia Formigas**	
	Escolha Padrão	Algoritmo Genético	Escolha Padrão	Algoritmo Genético	Escolha Padrão	Algoritmo Genético
3D Road Network	331,07	**6.221,81**	174,79	**1.529,04**	42,35	**124,77**
Cuff-Less Blood Pressure Estimation	1.370,65	**39.691,40**	2.897,01	**20.773,24**	113,20	**525,59**
Diabetes	144,58	**283,22**	102,89	**175,18**	6,64	**29,67**

Geographical Original of Music	138,46	**392,05**	149,15	**391,47**	19,34	**57,46**
Glass	63,64	**184,00**	177,63	**326,16**	1,17	**3,85**
Ionosphere	31,12	**78,95**	29,46	**45,48**	3,42	**5,02**
Íris	49,05	**568,74**	136,07	**317,91**	12,43	**39,24**
Stone Flakes	23,50	**89,13**	25,82	**95,66**	1,26	**4,21**
Student Performance	55,87	**114,31**	52,11	**118,17**	7,58	**26,13**
Synthetic Control Chart Time	40,62	**218,30**	82,42	**87,40**	14,48	**18,18**
Wine	53,35	**83,29**	26,66	**41,76**	4,76	**11,07**
Yeast	168,47	**1.132,74**	149,44	**1.099,49**	13,98	**67,01**

O algoritmo genético também superou a escolha padrão em todos os casos. Desta vez o algoritmo *K-Means* apresentou o melhor resultado em 10 casos, com o *SOM* sendo melhor nos 02 restantes. Em ambas as tabelas, o *SOM* obteve melhores resultados que o K-Means com as bases de dados *Glass* e *Student Performance*.

Também é importante comparar o desempenho de ambos os métodos de busca de configuração. Considerando os três algoritmos, *K-Means*, *SOM* e *Colônia de Formigas*, é possível comparar 36 casos distintos. O algoritmo genético foi melhor em 15 casos, enquanto que a busca aleatória obteve melhores resultados em 09 casos. Nos 12 casos restantes ocorreram empates.

5.2 Considerações Finais

Os métodos de busca se mostraram viáveis por serem capazes de encontrar melhores resultados em todos os casos. Com estas implementações e disponibilidade de processamento extra, o usuário pode optar por qualquer um dos dois métodos e conseguir uma melhor divisibilidade dos agrupamentos segundo o critério Calinski-Harabasz (CALINSKI; HARABASZ, 1974).

Ambos métodos de busca apresentaram resultados satisfatórios, mas o algoritmo genético foi ligeiramente melhor que a busca aleatória. É notável a grande quantidade de empates entre os dois métodos. Há a possibilidade de que nestes casos os métodos tenham alcançado a configuração que gera o melhor agrupamento (agrupamento ótimo ou a configuração ótima) para determinado algoritmo.

Durante os testes, observou-se o tempo necessário de processamento com cada algoritmo/método. Não há diferença significativa entre os métodos de busca, mas o mesmo não acontece entre os algoritmos de agrupamento.

Em todos os casos, *K-Means* exigiu menos tempo de processamento do que os outros dois algoritmos. Nos casos mais extremos, com as bases de dados que possuem um grande número de instâncias e/ou atributos, o algoritmo *K-Means* executou em média 10 vezes mais rápido por busca que o algoritmo *SOM*. Já o *SOM*, executou em média 04 vezes mais rápido por busca que o algoritmo *Colônia de Formigas*.

Finalmente, devido aos resultados obtidos, pode-se concluir que o algoritmo de agrupamento *K-Means* é o mais indicado para escolha do usuário, tanto por apresentar melhores resultados, quanto por ser mais rápido. Sua desvantagem é não determinar (por meio do próprio algoritmo) automaticamente o número de grupos. Mas este problema é resolvido aqui pela busca automática.

Capítulo 6

Conclusões

Este trabalho discutiu e implementou duas formas de resolver o problema *CASH - Combined Algorithm Selection and Hyperparameter Optimization Problem* (THORNTON *et al*., 2012) na ferramenta YADMT. Este capítulo faz as últimas considerações sobre o trabalho, discutindo os resultados e citando possibilidades de trabalhos futuros.

6.1 Considerações Finais

O grande volume de dados armazenados projetou uma área conhecida como Descoberta de Conhecimento em Bases de Dados. Tal área provê ferramentas para realizar a análise de dados de forma empírica, sem a certeza de quais serão os resultados. Como os resultados são muitas vezes imprevisíveis, há a necessidade de se configurar corretamente os algoritmos, para ser possível obter máximo proveito em suas aplicações.

Este trabalho abordou as técnicas de busca aleatória e algoritmo genético para pesquisar valores de parâmetros em seus espaços de busca. O Capítulo 5 foi capaz de mostrar o bom desempenho dos métodos implementados. Ambos obtiveram melhores resultados que a escolha *default* em todos os casos abordados.

Segundo os resultados, o algoritmo *K-Means* é o mais indicado ao usuário, devido às taxas mais altas encontradas com o critério *Calinski-Harabasz*. Uma das vantagens do algoritmo *K-Means* é a pequena quantidade de parâmetros a serem configurados, em relação aos outros algoritmos. Entretanto, por mais que o *K-Means* tenha obtido os melhores resultados, deve-se considerar que, devido às diferenças na quantidade e nos tipos de parâmetros entre os algoritmos, não é possível realizar um comparativo entre algoritmos sob as exatas mesmas condições.

Com este trabalho espera-se que os usuários da ferramenta YADMT obtenham auxílio ao executar a tarefa de agrupamento de dados. Como maior contribuição, é possível realizar a configuração do algoritmo de maneira automática, em troca de tempo de processamento. Os resultados obtidos por meio do processo de busca automática são satisfatórios. Um usuário que optar pelos métodos de busca, conseguirá, na maior parte dos casos, encontrar melhores resultados que aqueles obtidos com a configuração *default*. Ainda tratando de resultados, o usuário é capaz de obter uma visão geral do espaço de busca dos parâmetros, observando o comportamento de uma base de dados e de um algoritmo para um determinado parâmetro.

A principal dificuldade deste trabalho foi lidar com problemas residuais das implementações anteriores. Infelizmente ainda existem problemas na ferramenta que não foram corrigidos. Pode-se citar como exemplos de problemas:

- A ferramenta sempre considera que a última coluna da base de dados define as classes. Desta forma, erros acontecerão se uma base de dados não possui tal coluna, ou se ela está localizada em outra posição;
- Em certos cálculos envolvendo grupos e centroides existe uma troca de valores, sendo assumidos valores de um grupo com o centroide de outro. Desta forma, é gerado um erro nos resultados. A fonte do problema ainda não foi identificada.

6.2 Trabalhos Futuros

Recomenda-se os seguintes aspectos para serem abordados em trabalhos futuros:

- Correção dos defeitos encontrados na ferramenta YADMT;
- Estudar formas de aumentar o desempenho do algoritmo Colônia de Formigas;
- Estudar quais motivos fazem com que a implementação do algoritmo *K-Means* seja mais rápida, assim como, se possível, formas de reduzir o tempo de processamento requerido pelas atuais implementações dos algoritmos *SOM* e Colônia de Formigas.

Referências

ALBERTO, B. L. A.; ALMEIDA, P. E. M. **Abordagens de Pré-processamento de Dados em Problemas de Classificação com Classes Desbalanceadas.** Dissertação (Mestrado em Modelagem Matemática e Computacional) – Centro Federal de Educação Tecnológica de Minas Gerais – Belo Horizonte – MG, 2012.

BATISTA, G. E.; MONARD, M. C. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado.** Dissertação (Doutorado em Ciências de Computação e Matemática Computacional). Instituto de Ciências Matemáticas e Computação – ICMC/USP – São Carlos - SP, 2003.

BENFATTI, E. W.; BONIFACIO, F. N.; GIRARDELLO, A. D.; BOSCARIOLI, C. **Descrição da Arquitetura e Projeto da Ferramenta YADMT - *Yet Another Data Mining Tool*.** Curso de Ciência da Computação, UNIOESTE, Campus de Cascavel, 2010.

BERGSTRA, J.; BENGIO, Y. **Random Search for Hyper-Parameter Optimization.** Journal of Machine Learning Research 13, 2012. Disponível em: http://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf. Acesso em 23 de Junho de 2015.

BOSCARIOLI, C. **Análise de Agrupamentos baseada na Topologia dos Dados e em Mapas Auto-organizáveis**. Dissertação (Doutorado em Engenharia Elétrica). Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Sistemas Eletrônicos - São Paulo - SP, 2008.

CAFLISCH, R. E.; MOROKOFF, W.; OWEN, A. **Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension**. J. Comp. Finance, 1(1):27–46, 1997.

CALINSKI, T.; HARABASZ, J. **A dendrite method for cluster analysis.** *Communications in Statistics*. Vol. 3, No. 1, 1974, pp. 1–27.

CAMILO, O. C.; SILVA, C. J. **Mineração de Dados: Conceitos, Tarefas, Métodos e**

Ferramentas. Instituto de Informática, Universidade Federal de Goiás, 2009.

CAMPELO, R. J. B. **Análise de Agrupamento de Dados: Validação de Agrupamento; Parte I.** Instituto de Ciências Matemáticas e Computação – ICMC/USP, 2013. Disponível em: http://wiki.icmc.usp.br/images/9/9b/Validacao_I_2013.pdf. Acesso em 23 de Junho de 2015.

CARVALHO, A. P. L. **Algoritmos Genéticos.** Instituto de Ciências Matemáticas e Computação – ICMC/USP, 2009. Disponível em: http://www.icmc.usp.br/~andre/research/genetic/. Acesso em 26 de Setembro de 2015.

CORRÊA, M. C. F.; PACHECO, M. A. C.; SOUZA, R. C. **Estimação de Hiperparâmetros para um Modelo de Previsão Hot-Winters com Múltiplos Ciclos por Algoritmos Genéticos.** Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica, PUC-Rio, 2010. Disponível em: http://rica.ele.puc-rio.br/media/Revista_rica_n9_a5.pdf. Acesso em 26 de Setembro de 2015.

DENEUBOURG, J. L.; GOSS, S.; FRANKS, N.; SENDOVA-FRANKS, A.; DETRAIN, C.; CHRÉTIEN, L. **The dynamics of collective sorting: Robot-like ants and ant-like robots.** In Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animals 1 (pp. 356 – 365). Cambridge, MA: MIT Press, 1991.

DUNN, J. C. **A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters.** Cybernetics and Systems, (pp. 32 – 57). Department of Theoretical and Applied Mechanics, Cornell University, 1973.

FAUSETT, L. **Fundamentals of Neural Networks - Architectures, Algorithms and Applications**. Prentice-Hall, Inc., Upper Saddle River, NJ, 1994.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **From data mining to knowledge discovery: an overview.** Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, Menlo Park, CA, 1996.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques.** San Francisco: Morgan Kaufmann Publishers Inc., 2000.

HANDL, J.; KNOWLES, J.; DORIGO, M. **On the Performance of ant-based Clustering.** Design and application of hybrid intelligent systems. In Design and application of hybrid

intelligent systems, p. 204-213, 2003.

HOLLAND, J. **Adaptation in natural and artificial systems.** Ann Arbor: Univ. of Michigan Press, 1975.

KOTSIANTIS, S. B.; PINTELAS, P. E. **Increasing the Classification accuracy of simple Bayesian classifier.** Em *Proceedings of AIMSA*, (pp. 198-207), 2004.

LICHMAN, M. **UCI Machine Learning Repository.** Irvine, CA: University of California, School of Information and Computer Science, 2013. Disponível em: http://archive.ics.uci.edu/ml. Acesso em 23 de Junho de 2015.

LUCAS, D. C. **Algoritmos Genéticos: Uma Introdução.** Apostila da disciplina de Ferramentas de Inteligência Artificial, 2002.

MATTEUCCI, M. **A Tutorial on Clustering Algorithms: K-means Clustering.** Notas de Aula, 2015. Disponível em: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html. Acesso em 23 de Junho de 2015.

NETO, J. T. **Clustering – Agrupamento de Dados**, 2012. Disponível em: http://legauss.blogspot.com.br/2012/06/clustering-agrupamento-de-dados.html. Acesso em 23 de Junho de 2015.

POZO, A. T. R. **Mineração de Dados: Conceitos, Aplicações e Experimentos com Weka.** Universidade Federal do Paraná – Departamento de Informática, 2002. Disponível em: http://www.inf.ufpr.br/aurora/disciplinas/ERBD/ERBD10.pptx. Acesso em 23 de Junho de 2015.

RUSE, M. **Darwinism Defended: A Guide to the Evolution Controversies**, 1982. Reading, Massachusetts: Addison-Wesley Publishing Company. 356 p. p. 76.

SCHMITT, J.; ANDRADE, D. F.; BARBETTA, P. A. **Pré-processamento para a Mineração de Dados: Uso da Análise de Componentes Principais com Escalonamento Ótimo.** Tese (Mestre em Ciência da Computação) – UFSC – Universidade Federal de Santa Catarina, Florianópolis – SC, 2005.

TEIXEIRA, M. F. ***Agrupamento e Visualização de Dados: Estudo e Implementações para a Ferramenta YADMT*.** Dissertação (Trabalho de Conclusão de Curso) – UNIOESTE –

Universidade Estadual do Oeste do Paraná, Cascavel – PR, 2013.

THORNTON, C.; HUTTER, F.; HOOS, H.H.; LEYTON-BROWN, K. **Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms.** Department of Computer Science, University of British Columbia, 2012.

VENDRAMIN, L.; CAMPELO, R. J. B. **Comparação de Critérios de Validade de Agrupamento.** In: Simpósio Internacional de Iniciação Científica da USP, 16, 2008.

WEKA. **Waikato Environment for Knowledge Analysis**, 2015. Disponível em: http://www.cs.waikato.ac.nz/ml/weka/. Acesso em 23 de Junho de 2015.

Anexo A – Parâmetros Utilizados pelos Métodos de Busca

Os dois métodos de busca estudados no trabalho utilizaram um conjunto de parâmetros para cada um dos algoritmos de agrupamento de dados. Os parâmetros utilizados com cada um dos algoritmos é representado na Tabela A.1:

Tabela A.1 – Relação entre algoritmo e parâmetros utilizados nos métodos de busca.

Algoritmo	**Parâmetro**	**Tipo**
K-Means	Número de *Clusters*	Discreto
	Máximo de Iterações	Discreto
	Seeds Aleatórias	Booleano
	Seeds	Contínuo, Intervalo [0-100]
	Função de Distância	Discreto
SOM	Número de Linhas e Colunas	Discreto
	Número de Neurônios	Discreto
	Tipo de Vizinhança	Discreto
	Raio Inicial	Discreto
	Função de Atualização	Discreto
	Aprendizagem Inicial	Contínuo

	Número de Iteraçôes	Discreto
	Método de Agrupamento	Discreto
Colônia de Formigas	Função Vizinhança	Discreto
	Vizinhança Máxima e Mínima	Discreto
	Controle de Vizinhança	Discreto
	Função Semelhança	Contínuo
	Semelhança Máxima e Mínima	Contínuo
	Controle de Semelhança	Contínuo
	Fase	Contínuo, Intervalo [0-100]
	Função de Distância	Discreto
	Método de Recuperação	Discreto

Os valores dos parâmetros são definidos de acordo com o método de busca. Alguns valores de parâmetros possuem um número definido de opções (por exemplo Função de Distância), mas outros, possuem um número infinito de possibilidades, sendo necessário definir limitações aos valores que podem ser atribuídos. Os intervalos de valores atribuídos foram definidos de maneira empírica para cada parâmetro.