Universidade de São Paulo–USP
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica e de Computação
Programa de Pós-Graduação em Engenharia Elétrica

Eduardo Marmo Moreira

Sistema de aprendizado reconfigurável para classificação de dados utilizando processamento paralelo

São Carlos
2014

Eduardo Marmo Moreira

Sistema de aprendizado reconfigurável para classificação de dados utilizando processamento paralelo

Tese de doutorado apresentada ao Programa de Engenharia Elétrica da Escola de Engenharia de São Carlos como parte dos requisitos para a obtenção do título de Doutor em Ciências.

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Carlos Dias Maciel

São Carlos

2014

Trata-se da versão corrigida da tese. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

M838s

Moreira, Eduardo Marmo
Sistema de aprendizado reconfigurável para classificação de dados utilizando processamento paralelo / Eduardo Marmo Moreira ; orientador Carlos Dias Maciel. – São Carlos, 2014.

Tese (Doutorado) – Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Máquina de aprendizado. 2. Sistema de aprendizado. 3. Inteligência artificial. 4. Classificação de dados. 5. Particionamento de dados. 6. Banco de dados. I. Titulo.

FOLHA DE JULGAMENTO

Candidato: Bacharel **EDUARDO MARMO MOREIRA.**

Título da tese: "Sistema de aprendizado reconfigurável para classificação de dados utilizando processamento paralelo ".

Data da defesa: 07/05/2014

Comissão Julgadora: **Resultado:**

Prof. Associado **Carlos Dias Maciel (Orientador)**
(Escola de Engenharia de São Carlos/EESC) APROVADO

Prof. Associado **Alexandre Cláudio Botazzo Delbem** APROVADO
(Instituto de Ciências e Matemáticas e de Computação/ICMC)

Prof. Dr. **Carlos Alberto Murari Pinheiro** Aprovado
(Universidade Federal de Itajubá/UNIFEI)

Prof. Dr. **Enzo Seraphim** aprovado
(Universidade Federal de Itajubá/UNIFEI)

Prof. Dr. **José Antonio Perrella Balestieri** Aprovado
(Universidade Estadual Paulista "Júlio de Mesquita Filho"/UNESP-Guaratinguetá)

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

Este trabalho é dedicado aos
Espíritos amigos.

Agradecimentos

Registro sinceros agradecimentos a todas as pessoas que, direta ou indiretamente, contribuíram para a realização desta tese e, de forma particular, pela atenção inestimável, minha gratidão:

- a Deus;
- aos meus pais Antônio e Terezinha, pelo amor e apoio;
- ao meu irmão, professor Edmilson (Universidade Federal de Itajubá - UNIFEI), pois sem ele, não teria chegado a este momento;
- ao meu irmão, professor Leonardo (Universidade Federal de São João Del Rei - UFSJ), pela amizade e principalmente pelo apoio para não desistir;
- ao meu orientador, professor Maciel, que sempre entendeu meus momentos de conflitos e que contribuiu expressivamente para o meu crescimento profissional;
- à minha namorada Carol, pelo companheirismo e amor de sempre, entendendo principalmente meus momentos de estresse;
- à minha amiga Maria Aparecida, do Centro Espírita Teresa de Ávila em Três Corações, pela amizade eterna e pelas preces que tanto me ajuda;
- à minha cunhada Karina, que tanto me ajudou nos períodos em que hospedei em sua casa;
- à minha cunhada Juliana, pelas ajudas de sempre e pela nossa amizade;
- aos meus sobrinhos, Lucas, Matheus, Cecília e Henrique, pelos momentos tão especiais juntos;
- ao meu sogro Sr. Douglas, pelas conversas tão enriquecedoras;

- ❏ ao meu amigo João Paulo, por ser mais um irmão e por me apoiar neste período tão delicado que foi dirigir o campus do Instituto Federal de São Paulo (IFSP) de São João da Boa Vista e progredir no doutorado;
- ❏ à minha amiga e companheira Roselaine, pela amizade tão especial e pela torcida de sempre;
- ❏ ao meu amigo Professor Alexandre (IFSP), que tanto me ajudou nos estudos da língua Inglesa;
- ❏ ao meu amigo e colega Wagner, pelos momentos que passamos juntos neste programa de doutorado da USP;
- ❏ ao meu amigo Professor Marangoni (IFSP), pela força que sempre forneceu durante esses 5 anos;
- ❏ aos meus amigos e colegas especiais do IFSP campus São João da Boa Vista, que sempre torceram por mim e que todos os dias, fraternalmente, me direcionam energias de carinho e amizade.

Muito obrigado.

"Auxilia aos outros, tanto quanto puderes.
Cada pessoa que hoje te encontra talvez
seja amanhã a chave de que necessitas
para a solução de numerosos problemas.
A justiça funciona até que o amor
tome posse do coração e da vida."
Emmanuel

Resumo

Moreira, Eduardo Marmo **Sistema de aprendizado reconfigurável para classificação de dados utilizando processamento paralelo**. 133 p. Tese de doutorado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2014.

Esta tese apresenta a arquitetura de um sistema de aprendizado, com um escalonador de tarefas que possibilita a utilização de vários métodos de classificação e validação, permitindo a distribuição dessas tarefas entre os módulos do sistema. Esta arquitetura está estruturada de forma que classificações obtidas através de uma técnica sejam reutilizadas em paralelo pelo mesmo algoritmo ou por outras técnicas, produzindo novas classificações através do refinamento dos resultados alcançados e ampliando o uso em bases de dados com características diferentes. O sistema foi estruturado em quatro partes denominadas, respectivamente, **Módulo de Inicialização**, **Módulo de Validação**, **Módulo de Refinamento** e **Módulo Especial de Escalonamento**. Em cada módulo, podem ser usados vários algoritmos para atender aos seus objetivos. A estrutura deste sistema permite sua configuração, utilizando diversos métodos, inclusive com técnicas de inteligência artificial. Com isso, é possível a obtenção de resultados mais precisos por meio da escolha do melhor método para cada caso. Os resultados apresentados neste trabalho foram obtidos a partir de bases conhecidas na literatura, o que possibilita comparar as implementações dos métodos tradicionais que foram adicionadas ao sistema e, principalmente, verificar a qualidade dos refinamentos produzidos pela integração de técnicas diferentes. Os resultados demonstram que através de um sistema de aprendizado, minimiza-se a complexidade na análise de grandes bases de dados, permitindo verificar bases com estruturas diferentes e aumentar os métodos aplicados na análise de cada estrutura. Isto favorece a comparação entre os métodos e proporciona resultados mais confiáveis. Para uniformizar os dados provenientes de bases distintas, foi elaborada a modelagem de dados do sistema, o que favorece a escalabilidade do sistema de maneira uniforme.

Palavras-chave: Máquina de Aprendizado; Sistema de Aprendizado; Inteligência Artificial; Classificação de Dados; Particionamento de Dados; Banco de Dados.

Abstract

Moreira, Eduardo Marmo **Reconfigurable learning system for classification of data using parallel processing**. 133 p. Ph.D. Thesis – São Carlos School of Engineering, University of São Paulo, 2014.

This thesis presents the architecture of a System Learning with a task scheduler, which makes possible the utilization of several classification and validation methods, allowing the distribution of tasks between the module systems. This architecture is structured of such way that the classifications obtained through a specific technique can be reutilized in parallel by the same algorithm or by other techniques, producing new classifications through the refinement of the results achieved and expanding the use in databases with different characteristics. The system was structured in four parts denominated, respectively, **Initialization module**; **Validation module**; **Refinement module**; and **Especial scheduling module**. In each module, various algorithms can be employed to reach its objectives. The structure of this system allows its configuration, utilizing various methods, including artificial intelligence techniques. Thus, it is possible to obtain more precise results through the choice of the best method to each case. The results presented in this work were obtained from basis that are known in the literature, which allows to compare the implementations of the traditional methods that were added to the system and, especially, to verify the quality of the refinements produced by the integration of different techniques. The results demonstrated that through a learning system, the complexity of the analysis of great databases is minimized, allowing to verify basis with different structures and to increase the methods applied in the analysis of each structure. It favors the comparison between the methodologies and provides more reliable results. To standardize the data originated of distinct bases, the data modelling system was elaborated, which will favor the uniform scalability of the system.

Keywords: Machine Learning; System Learning; Artificial Intelligence; Clustering; Data

Partitioning; Database.

Lista de ilustrações

Figura 1 Modelo de processamento inteligente de Dietterich (1982) no qual a máquina de aprendizado aceita o ambiente de observação e adiciona as informações na base de dados, aprimorando a tarefa, visando à melhoria de desempenho 33

Figura 2 Figura modificada do artigo de Fayyad, Piatetsky-Shapiro e Smyth (1996), a qual representa uma visão geral das etapas que compõem um processo de extração de conhecimento através de um conjunto de dados 33

Figura 3 Mudanças nos *centróides* e dados na execução do *K-Means*, em concordância com Tan, Steinbach e Kumar (2013) (Os *centróides* estão marcados com o sinal '+') 37

Figura 4 Efeitos no resultado do *k-Means* devido a uma má inicialização conforme Tan, Steinbach e Kumar (2013) (Os *centróides* estão marcados com o sinal '+') 38

Figura 5 Distribuição de 10 pontos em um espaço e a definição inicial e aleatória de 3 *centróides* apresentados pela figura (**A**) e a atualização dos *centróides* utilizando o método *K-Means* apresentado pela figura (**B**) . 39

Figura 6 Atualização dos *centróides*, conforme dados distribuídos na figura 5, utilizando o método *K-Medoid*. A última iteração do processo identifica os dados marcados na figura como os dados mais centralizados de cada *cluster* 39

Figura 7 Distribuição de 20 pontos em um espaço. Identificação das Classes utilizando o método *K-Means* e o método *K-Medoid* respectivamente. Na área correspondente ao uso do método *K-Means*, as áreas marcadas representam os pontos identificados como centrais de cada *cluster*. Na área correspondente ao uso do método *K-Medoid*, os dados identificados representam os dados centrais (*centróides*) de cada *cluster*. O elemento marcado com um círculo na área correspondente a classificação do método *K-Medoid* apresenta uma classificação diferente quando comparado os dois métodos, isto é, utilizando o método *K-Means*, este elemento faz parte do *cluster* representado pela cor **marrom** e utilizando o método *K-Medoid*, este elemento faz parte do *cluster* representado pela cor **rosa** 40

Figura 8 Ilustração da estrutura de dados **Árvore Binária** definida por um elemento distinto, denominado raiz, com dois ponteiros para duas estruturas diferentes, denominadas sub-árvore esquerda e sub-árvore direita. Os nós de uma árvore binária possuem graus zero, um ou dois. Um nó de grau zero é denominado folha. Em uma árvore binária, por definição, cada nó poderá ter até duas folhas. Uma árvore "estritamente binária"é uma árvore na qual todo nó tem zero ou duas folhas 45

Figura 9 Modelo de PG - Inicialmente é criada, de forma aleatória, uma população. Essa população podem representar funções que sejam específicas ao domínio de um problema, operações matemáticas, funções lógicas ou operações de programação. O critério usado para avaliação é uma função chamada função de 'aptidão' (*fitness*), que avalia se cada solução contribuirá para a próxima geração de soluções. Esta medida é que dirige diretamente o processo evolutivo, fazendo que, quanto mais adaptado for um indivíduo maior a sua chance de permanecer ou ter suas características propagadas para as próximas gerações. 57

Figura 10 A figura mostra a estrutura do sistema e a forma pela qual os módulos estão relacionados. O conjunto de dados é primeiramente tratado pelo **Módulo de Inicialização** gerando o primeiro conjunto de dados particionados. Os conjuntos de dados particionados são avaliados pelo **Módulo de Validação**, que utiliza diferentes critérios e regras de especialistas para registrar a qualidade da partição. Paralelamente a este processo o **Módulo de Refinamento** gera mais partições a serem avaliados, a partir dos dados particionados definidos pelo **Módulo de Inicialização**, e pelo próprio **Módulo de Refinamento**. O **Módulo Especial de Escalonamento** é responsável pela administração das atividades 64

Figura 11 Método *Average Link*: Segundo Nassifa e Hruschkab (2011), a distância entre dois grupos é a distância média entre pares de objetos pertencentes a grupos diferentes 66
Figura 12 Método *Single Link*: Conforme Nassifa e Hruschkab (2011), a distância entre dois grupos é a distância mínima entre dois objetos pertencentes a grupos diferentes. Esse algoritmo induz grupos contíguos, baseados nas adjacências entre os objetos, podendo assumir formas variadas . . . 66
Figura 13 Fluxograma Geral de um Algoritmo Evolutivo adaptado de Zhang et al. (2011) 77
Figura 14 Ilustração dos dados A, B, C, D, E, F, G e K onde os elementos A, B, D, E fazem parte de uma partição, o elemento A é o *centróide* deste *cluster*, e na distância do elemento A com o elemento C existem 3 elementos candidatos a troca de partição baseando-se em um aumento do raio de cobertura em 58% 79
Figura 15 Escalonamento com Filas Multiníveis - O **Módulo Especial de Escalonamento** define prioridades na administração das atividades do sistema. O **Módulo de Validação** tem maior prioridade no sistema que o **Módulo de Refinamento** que por sua vez possui maior prioridade do que o **Módulo de Inicialização** 83
Figura 16 Modelo Entidade×Relacionamento representando a estrutura dos dados armazenados no sistema proposto 93
Figura 17 Modelo Entidade×Relacionamento representando a estrutura dos metadados do sistema proposto 94
Figura 18 Ilustração do estado das filas multiníveis após o **Módulo Especial de Escalonamento** ter feito a leitura do experimento definido na Tabela 9 99
Figura 19 Experimento analisando as 5 Bases de Dados (*Iris*, *Zoo*, *Glass*, *Wine* e *Balance*) com 1, 2 e 4 Máquinas no Cluster de Computadores 117
Figura 20 Experimento analisando as 5 Bases de Dados (*Iris*, *Zoo*, *Glass*, *Wine* e *Balance*) multiplicadas por 5 com 1, 2 e 4 Máquinas no Cluster de Computadores 117

Lista de tabelas

Tabela 1 Matriz de distância dos objetos A, B, C, D, E, F, G e K 78

Tabela 2 Animais por Grupo 90

Tabela 3 Tabela `EstruturaDaBase` com um registro referente a base de dados *Iris* 96

Tabela 4 Tabela `TipoDosAtributos` com um registro para cada atributo da base Iris 96

Tabela 5 Tabela `BaseDeDados` com os registros correspondentes as bases utilizadas nos experimentos 96

Tabela 6 Tabela 'Dados' correspondente as informações dos atributos da base *Iris* como os centímetros do comprimento e largura de sépala e do comprimento e largura de pétala 97

Tabela 7 Tabela `MétodoDeClassificação` correspondente a todos os métodos implementados em todos os módulos no sistema 98

Tabela 8 Tabela `Planejamento` indicando se a sequência será estática e pré-definida pelo usuário 98

Tabela 9 Tabela `Experimento` correspondente a sequência informada ao escalonador no processo de particionamento da base *Iris* e com os atributos conforme tabela resultante do mapeamento entre o Modelo Entidade×Relacionamento e o Modelo Relacional. Os dados representam o estado da tabela antes do processamento 98

Tabela 10 Tabela `Experimento` equivalente à Tabela 9, com os resultados do processamento realizado 100

Tabela 11 Particionamento da Base de Dados *Iris* obtido pelo `Método da Árvore Binária` no **Módulo de Inicialização** 100

Tabela 12 Melhor conjunto de dados particionado da base de dados *Iris* obtido pelo `Método Randômico` do **Módulo de Inicialização** entre **3.000.000** de execuções 101

Tabela 13 Melhor conjunto de dados particionado da base de dados *Iris* obtido pelo `Método Semi-randômico` do **Módulo de Inicialização** entre **3.000.000** de execuções 101

Tabela 14 Refinamento do conjunto gerado pelo `Método da Árvore Binária` a partir do `Método de Consulta por Abrangência` com o raio original dos respectivos conjuntos 102

Tabela 15 Refinamento do conjunto representado pela Tabela 14 102

Tabela 16 Refinamento do conjunto gerado pelo `Método da Árvore Binária` no **Módulo de Inicialização** a partir do `Método de Consulta por Abrangência` com o raio expandido em **0,2**, referente ao raio original dos respectivos conjuntos 103

Tabela 17 Atributos candidatos a realocação conforme refinamento na Tabela 16 . 103

Tabela 18 Refinamento do conjunto do `Método da Árvore Binária` a partir do `Método do Raio de Cobertura Mínimo` 104

Tabela 19 Refinamento, com o `Método do Raio de Cobertura Mínimo`, do particionamento obtido através do `Método Randômico` do conjunto de dados *Iris* 104

Tabela 20 Refinamento, com o `Método do Raio de Cobertura Mínimo`, do Particionamento gerado pelo `Método Semi-randômico` 105

Tabela 21 Análise dos dados da base *Iris* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento** 105

Tabela 22 Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Iris* 105

Tabela 23 Atributos e seus Tipo de Dados da Base de Dados *Zoo* 106

Tabela 24 Particionamento do Conjunto de Dados *Zoo* conforme Frank e Asuncion (2010) 106

Tabela 25 Particionamento do Conjunto de Dados *Zoo* conforme `Método da Árvore Binária` 106

Tabela 26 Refinamento pelo `Método do Raio de Cobertura Mínimo` a partir do particionamento do Conjunto de Dados *Zoo* conforme o `Método da Árvore Binária` 107

Tabela 27 Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método da Árvore Binária` 107

Tabela 28 Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Randômico` 108
Tabela 29 Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Randômico` 108
Tabela 30 Particionamento do Conjunto de Dados *Zoo* conforme `Método Semi-randômico` 108
Tabela 31 Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Semi-randômico` 109
Tabela 32 Análise dos dados da base *Zoo* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento** 109
Tabela 33 Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Zoo* 110
Tabela 34 Particionamento do Conjunto de Dados *Glass* conforme repositório Frank e Asuncion (2010) 110
Tabela 35 Particionamento do Conjunto de Dados *Glass* gerado pelo `Método Randômico` 111
Tabela 36 Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Randômico` 111
Tabela 37 Particionamento do Conjunto de Dados *Glass* gerado pelo `Método Semi-randômico` 112
Tabela 38 Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Semi-randômico` 112
Tabela 39 Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Randômico` 113
Tabela 40 Análise dos dados da base *Glass* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento** 113
Tabela 41 Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Glass* 113

Tabela 42 Particionamento do Conjunto de Dados *Wine* conforme Frank e Asuncion (2010) 114
Tabela 43 Particionamento do Conjunto de Dados *Wine* conforme `Método da Árvore Binária` 114
Tabela 44 Dados nas Partições da Base *Wine* utilizando o `Método da Árvore Binária` 115
Tabela 45 Valores do **Atributo de Qualidade** referente as análises realizadas no conjunto de dados *Balance* 115
Tabela 46 Análise dos dados da base *Balance* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento** 115
Tabela 47 Desempenho do Sistema de Aprendizado no *cluster* de computadores utilizando 5 Bases de Dados 116
Tabela 48 Desempenho do Sistema de Aprendizado no *cluster* de computadores utilizando as 5 Bases de Dados mencionadas multiplicadas por 5 117
Tabela 49 Simulação em uma única máquina de 7 conjunto de dados criados a partir dos registros da base *Covertype* 118

Lista de siglas

AE Algoritmo Evolutivo

AG Algoritmo Genético

AM Aprendizagem de Máquina

API *Application Programming Interface*

FCM Fuzzy C-Means

FM Fiduccia-Mattheyses

IA Inteligência Artificial

KDD *Knowledge Discovery in Database*

KL *Kernighan e Lin*

MPI *Message Passing Interface*

MRIP *Multiple Replication in Parallel*

PG Programação Genética

PVM *Parallel Virtual Machine*

SIG Sistemas de Informações Gerenciais

SO Sistema Operacional

SRIP *Single Replication in Parallel*

Sumário

1 Introdução **27**

1.1 Motivação 28

1.2 Objetivo 29

1.3 Organização do Trabalho 30

2 Revisão da Literatura **31**

2.1 Máquinas de Aprendizado 31

2.2 Reconhecimento de Padrões 34

2.2.1 Metodologias de Classificação de Dados 36

2.2.2 Métricas de Similaridade 51

2.2.3 Estabilidade de *Cluster* 56

2.3 Algoritmos Evolutivos 56

2.4 Programação Paralela 59

2.5 Considerações Finais 60

3 Materiais e Métodos **63**

3.1 Sistema Proposto 63

3.1.1 Módulo de Inicialização 63

3.1.2 Módulo de Validação 74

3.1.3 Módulo de Refinamento 75

3.1.4 Módulo Especial de Escalonamento 81

3.1.5 Gerenciamento dos Dados 85

3.2 Bases Utilizadas para Testes 89

3.3 Ambiente Computacional Utilizado 90

3.4 Considerações Finais 92

4 Experimentos e Resultados Obtidos **95**

4.1 Base de dados Iris 95

4.2 Outros Experimentos 105
4.3 Desempenho do Sistema 116

5 Considerações Finais **119**
5.1 Perspectivas Futuras 120

Referências **123**

Capítulo 1

Introdução

A Classificação de Dados tem sido estudada há vários anos em diversas áreas, envolvendo várias técnicas. É uma área promissora que tem experimentado um desenvolvimento rápido e tornou-se uma poderosa ferramenta para uma ampla gama de aplicações, visando descobrir padrões em dados, com base em teorias e metodologias diferentes (GIOTIS; PETKOV, 2012; ZHANG et al., 2011; XUTAO et al., 2010; MJOLSNESS et al., 2001). A Classificação tem como escopo realizar o particionamento de um conjunto de dados de modo que objetos semelhantes pertençam ao mesmo subgrupo.

Nesse processo de particionamento, existe uma alta relação entre os argumentos, métricas de similaridades e estruturas de dados. Diante das peculiaridades de cada base de dados, nem sempre é possível alcançar um resultado satisfatório, mesmo utilizando métricas, argumentos e estrutura de dados semelhantes. Por conseguinte, a manipulação ou configuração destes elementos torna-se essencial para a obtenção de resultados satisfatórios.

Diante das características dos dados a serem analisados e a complexidade na escolha dos métodos de classificação, existe um número elevado de metodologias de classificação. Todavia, não há um número extenso na literatura de trabalhos que comparam técnicas de particionamento aplicados em diferentes conjuntos de dados, por sua vez, existem trabalhos que comparam métodos de agrupamentos. Portanto, uma ferramenta que permita refinar os resultados obtidos nos métodos utilizados, bem como o uso de vários métodos em um mesmo conjunto de dados torna-se relevante para a obtenção de classificações mais precisas.

Neste contexto, este trabalho apresenta a arquitetura de um sistema de aprendizado, com um escalonador de tarefas implícito, que possibilita a utilização de vários métodos de classificação e validação, permitindo a distribuição dessas tarefas entre os módulos do sistema. Esta arquitetura permite que classificações obtidas através de uma técnica sejam reutilizadas em paralelo. A forma como esta arquitetura foi concebida permite que estes métodos sejam empregados simultaneamente, com a utilização de uma máquina paralela ou de uma rede de computadores, favorecendo uma reconfiguração do sistema em tempo

de execução e fornecendo resultados mais precisos e rápidos.

Nos testes de validação do sistema projetado foram utilizadas as bases de dados *Iris*; *Wine*; *Zoo*; *Balance*; *Glass* e *Covertype* do repositório de Aprendizado de Máquina UC Irvine (FRANK; ASUNCION, 2010). A base de dados da flor *Iris* (FISHER, 1936) é uma das mais utilizadas em testes de reconhecimento de padrões. A base possui 150 instâncias de três espécies da família das Iridáceas: *Setosa*, *Versicolor* e *Virginical*. Cada instância da base possui quatro atributos, com dados de largura e comprimento de pétalas e sépalas, e 50 registros de cada uma das espécies. As informações da base *Wine* representam os resultados da análise química de vinhos produzidos na mesma região da Itália, mas provenientes de três diferentes cultivos. A análise determinou as quantidades de 13 componentes químicos e/ou propriedades físico-químicas encontradas em cada um dos três tipos de vinhos. O conjunto de dados *Zoo* apresenta 101 instâncias com 17 características de sete grupos de animais. O conjunto de dados *Balance* foi obtido por meio de resultados experimentais de modelos psicológicos. O conjunto de dados *Glass* verifica tipos de vidro e o seu estudo é motivado para auxiliar na área de investigação criminológica. Finalmente, a base *Covertype* foi utilizada parcialmente por possuir um número relativamente alto de registros, 581012 registros, tornando-se um conjunto interessante para acompanhar e avaliar a eficiência computacional do sistema. O Sistema proposto realizou simulações em uma rede de computadores, de maneira sequenciais e paralelas, com o propósito de avaliar cada módulo implementado aplicado nas bases mencionadas.

1.1 Motivação

Atualmente, a maior parte dos problemas de tomada de decisão por especialistas, tais como administradores de empresa, pesquisadores acadêmicos, entre outros, está relacionada à conversão de dados em conhecimento concreto e efetivo. As bases mencionadas para a conversão são complexas e com grande volume de registros. Os recursos computacionais são robustos o suficiente para manipular grande quantidade de informações, favorecendo a interpretação de grandes bases de dados. Diante desta realidade, várias metodologias e sistemas estão sendo desenvolvidas para extrair conhecimento de diversos conjuntos de dados. Entretanto, diante da complexidade desta análise, as técnicas existentes que facilitam uma comparação de metologias de classificação ficam dependentes de uma análise dos respectivos especialistas de cada área, dificultando a extração de conhecimento diante dos dados tratados. Devido as características distintas de cada base de dados analisada, os métodos de classificação criados tornam-se específicos. Diante destas questões apresentadas, a importância de um método ou de um sistema reconfigurável capaz de aplicar diversas técnicas que colaboram entre si melhorando a classificação a cada iteração realizada através de um refinamento nos dados, apresenta-se como uma alternativa para aprimorar os resultados.

Conforme Gelbard, Goldman e Spiegler (2007), diferentes algoritmos de agrupamento produzem diferentes *clusters* e não há um método claro e padronizado para comparálos. Segundo Pons e Shulcloper (2011), não há nenhum método de agrupamento capaz de encontrar corretamente a estrutura ideal para todos os conjuntos de dados. Outra questão é que a interpretação para os diversos *clusters* formados, e sua implementação no ambiente original não está definida. O processo de agrupamento é imprevisível e os resultados às vezes incoerentes. Essa diversidade faz com que o processo de agrupamento seja complexo. Além disso, de acordo com Erlich, Gelbard e Spiegler (2003), Gelbard, Goldman e Spiegler (2007), não há mecanismos precisos para se medir e avaliar a qualidade de um algoritmo de agrupamento.

Em adição, um grande número de algoritmos de agrupamento estão disponíveis e descritos na literatura. Isso pode facilmente confundir um usuário tentando escolher aquele que melhor se adapta ao seu problema.

Por conseguinte, os desafios implícitos na análise e classificação de um grande quantidade de dados favorecem o uso de recursos paralelos no interesse de minimizar o tempo obtido na busca de conjuntos de partições relevantes.

1.2 Objetivo

O objetivo principal deste trabalho é desenvolver um sistema de aprendizado para a classificação de dados, como os dados biológicos, oferecendo uma arquitetura que permita reunir várias técnicas e métodos de classificação e que, através de um mecanismo de refinamento dos resultados obtidos, possibilite a interação entre os métodos de forma que o sistema seja capaz de gerar novas partições a partir de resultados obtidos por técnicas distintas. Além disso, o sistema deve ser capaz de identificar os melhores particionamentos obtidos nesses resultados. Por intermédio de recursos paralelos do sistema, tal proposta possibilita o aumento da quantidade de informações extraídas das bases de dados.

Consequentemente, o sistema em desenvolvimento apresenta os seguintes aspectos relevantes:

- ❑ Aumento do número de informações tratadas, através dos particionamentos criados, contribuindo nas análises em grandes bases de dados;
- ❑ Facilidade de uma comparação das características das bases a partir do impacto no uso de vários métodos na criação das primeiras partições, como no processo de refinamento mediante a interação entre as técnicas implementadas;
- ❑ Análise do método de classificação para cada base de dados;
- ❑ Possibilidade de estender o sistema através da inserção de novos métodos de classificação, o que aumenta a confiabilidade dos resultados;

- Emprego de n métodos como refinamento de particionamentos obtidos por outros métodos;
- Uso de recursos paralelos, aumentando os testes realizados no processo de refinamento dos resultados obtidos e, consequentemente a confiabilidade nos resultados encontrados.

1.3 Organização do Trabalho

Este documento está estruturado da seguinte forma: o capítulo 2 apresenta a revisão da literatura destacando os métodos relevantes para o propósito do trabalho. A seguir, o capítulo 3 corresponde aos materiais e métodos utilizados para a realização do trabalho, apresentando a estrutura do sistema de aprendizado proposto, as bases selecionadas para testes do sistema e informações a respeito do ambiente computacional que permite o uso do sistema. Finalmente, no capítulo 4, são apresentados os resultados obtidos e a análise das vantagens e desvantagens do sistema. As considerações finais e propostas para continuação deste trabalho são apresentados no capítulo 5.

Capítulo 2

Revisão da Literatura

Neste capítulo serão apresentados os conceitos introdutórios utilizados para a concepção do trabalho. A estrutura deste capítulo está dividida conforme segue: A seção 2.1 descreve os conceitos de Máquina de Aprendizado; a seguir, a seção 2.2 apresenta um importante processo da Máquina de Aprendizado, que é o Reconhecimento de Padrões, abordando algumas metodologias clássicas de Classificação de Dados, Métricas de Similaridade e uma menção sobre Estabilidade de *Cluster*, com o propósito de avaliar os dados classificados. A seção 2.3 descreve os conceitos básicos de Algoritmos Evolutivos com o propósito de otimizar a busca em soluções no processo de classificação de dados. A seção 2.4 apresenta alguns fundamentos teóricos da Programação Paralela e as vantagens e desvantagens no seu uso para a classificação de grandes quantidades de dados. A seção 2.5 apresenta as considerações finais deste capítulo.

2.1 Máquinas de Aprendizado

O aprendizado de máquina tem apresentado resultados significativos e superando as dificuldades geradas pela complexidade e pelo conhecimento incompleto principalmente sobre a natureza das estruturas estudadas na Biologia e em outras áreas do conhecimento humano.

Conforme Fayyad, Piatetsky-Shapiro e Smyth (1996), a nomenclatura utilizada na área que estuda padrões em dados tem recebido uma variedade de nomes, incluindo mineração de dados, extração de conhecimento, descoberta de informação, coleta de informações, arqueologia de dados e processamento de padrões de dados. O termo mineração de dados tem sido usado principalmente por estatísticos, analistas de dados e de comunidades de sistemas de informações gerenciais (SIG). A expressão *descoberta de conhecimento em bancos de dados* (*Knowledge Discovery in Database* - KDD) foi cunhada no primeiro *workshop* de KDD em 1989 (SHAPIRO, 1991) para enfatizar que o conhecimento é o produto final de uma descoberta de dados. Esta nomenclatura foi popularizada na área da Inteligência Artificial (IA) e em campos de aprendizagem de máquina (AM).

Segundo Alpaydin (2004), a aprendizagem de máquina é um campo da IA dedicada ao desenvolvimento de algoritmos e técnicas que permitam ao computador 'aprender', ou seja, aperfeiçoar seu desempenho em determinados processos. De acordo com Sinky et al. (2014), a aprendizagem de máquina pode ser dividida em duas categorias: aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado os dados são conhecidos e no aprendizado não supervisionado o sistema tem que descobrir sozinho relações, padrões, regularidades ou categorias nos dados que lhe vão sendo apresentados e codificando-as nas saídas. Os algoritmos que representam técnicas de aprendizagem ocasionam um alto custo computacional.

Realmente, para ser 'inteligente', um sistema que está em um ambiente de mudança deve ter a capacidade de aprender. Se o sistema é capaz de aprender e de se adaptar a cada mudança dos dados, o projetista do sistema não precisa prever e oferecer soluções para todas as situações possíveis. Neste caso, a automatização dos resultados seria aumentada, minimizando a necessidade de interferência do especialista dos dados. A aprendizagem de máquina é focada na característica indutiva da IA, extraindo regras e padrões de grandes conjuntos de dados (MICHALSKI, 1983).

O raciocínio indutivo consiste em uma reflexão a partir de um ou mais eventos específicos, buscando uma causalidade que possa ser generalizada de forma que venha a prever, com credibilidade minimamente satisfatória, a ocorrência de novos eventos semelhantes (CARROLL, 1993). O raciocínio indutivo, portanto, difere do raciocínio dedutivo, o qual parte de uma lei ou generalização previamente estabelecida para interpretação de um caso específico. Importante frisar que, quando não existem previamente as generalizações, a interpretação dos significados dos resultados não consiste em tarefa trivial, solicitando estudos profundos e extensos com grande número de amostras para gerar quaisquer inferências mínimas. Logo, tais sistemas requerem a elaboração, via raciocínio indutivo, de proposições que expliquem adequadamente a causalidade associada a tais fenômenos. Assim sendo, máquinas de aprendizado que desenvolvam tais classificações constituem alternativas de vanguarda para a análise de sistemas que apresentem múltiplos fatores causais, como é o caso de dados extraídos de sistemas biológicos (DZEROSKI et al., 1997).

Através de dados empíricos (experimentais) incompletos, é possível o desenvolvimento de interpretações no intuito de gerar conhecimento. De acordo com Fisher (1987), o objetivo da aprendizagem de máquina, neste contexto, é melhorar o desempenho de uma classificação de dados por meio de uma automatização na aquisição e no refinamento desses próprios dados. Este conceito é apresentado no modelo de processamento inteligente de Dietterich (1982), conforme Figura 1. O modelo foi utilizado para examinar os fatores que incidem sobre a concepção do elemento de aprendizagem. Estes incluem o nível da qualidade das informações fornecidas pelo ambiente, a forma e o conteúdo da base de conhecimento, bem como a complexidade e transparência do desempenho do elemento. Desses fatores, o mais importante é o nível das informações fornecidas pelo ambiente.

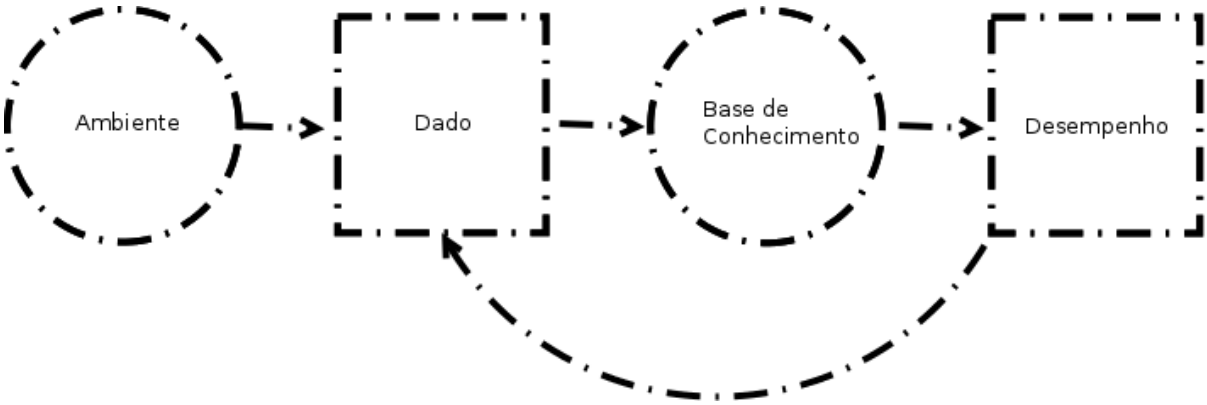

Figura 1 – Modelo de processamento inteligente de Dietterich (1982) no qual a máquina de aprendizado aceita o ambiente de observação e adiciona as informações na base de dados, aprimorando a tarefa, visando à melhoria de desempenho

Muitas técnicas utilizadas nas Máquinas de Aprendizado derivam dos trabalhos de psicólogos diante das suas teorias de aprendizagem animal e humana. De acordo com Nilsson (1998), uma máquina aprende sempre que muda sua estrutura, programa ou dados com base em suas entradas ou em resposta a informações externas, de tal forma que seu desempenho futuro previsto seja otimizado significativamente. Diante das suas propriedades, métodos para Máquinas de Aprendizado podem ser aplicados em diversas áreas (MITCHELL, 1999).

Na estatística, diferentes métodos elaborados para lidar com esses dados podem ser considerados instâncias de aprendizagem de máquina. Isso ocorre porque a decisão e as regras de cálculo dependem de amostras extraídas do ambiente do problema.

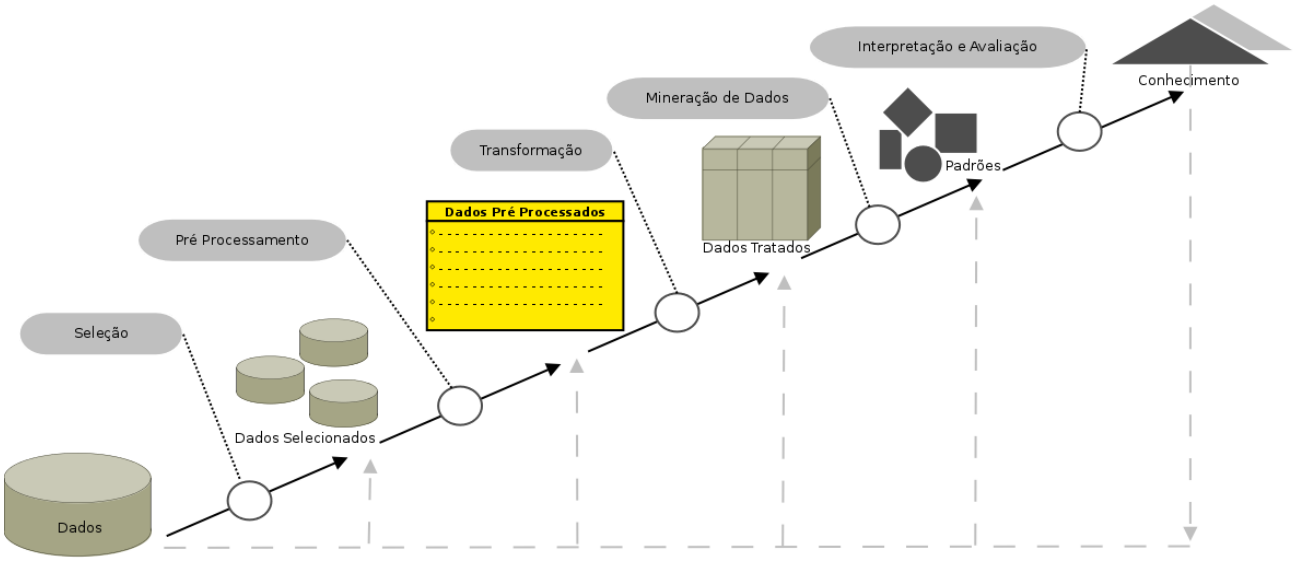

Figura 2 – Figura modificada do artigo de Fayyad, Piatetsky-Shapiro e Smyth (1996), a qual representa uma visão geral das etapas que compõem um processo de extração de conhecimento através de um conjunto de dados

Primeiramente, é possível visualizar na Figura 2, o processo de seleção dos dados, e finalmente, na última etapa, é realizada o processo de aprendizagem. Logo em seguida ao processo de seleção, ocorre a etapa de pré-processamento, na qual o processo realiza a filtragem dos dados utilizando operações básicas, que incluem a remoção de ruído, se for o caso, recolhendo a informação necessária para modelar. Subsequentemente, o processo de transformação corresponde principalmente a uma possível redução de dados sem prejudicar a extração do conhecimento. Após essa etapa, são aplicados métodos de mineração de dados como a classificação, no intuito de identificar padrões nos dados, e, finalmente, a interpretação dos padrões extraídos para determinação dos padrões que podem ser considerados novos conhecimentos.

2.2 Reconhecimento de Padrões

Uma importante tarefa de uma Máquina de Aprendizado é a classificação, que também é referenciada como reconhecimento de padrões. Este reconhecimento ocorre através de dados complexos, distinguindo classes distintas entre dados similares.

Durante a manipulação dos dados, os resultados obtidos por um método de classificação são frequentemente sensíveis a uma base de dados específica. Vários trabalhos estão focados em técnicas de classificação distintas, que são dependentes de parâmetros relacionados com os dados oriundos do respectivo estudo (LIU; WU; SHEN, 2011; ZAHRAIE; ROOZBAHANI, 2011; CELEBI; KINGRAVI; VELA, 2013).

É interessante notar que, de acordo com Kinnunen et al. (2011), grande número de metodologias de agrupamento tem sido proposto para essa tarefa em diversas áreas do conhecimento, mas, há poucos estudos extensivos, comparando os respectivos procedimentos com relação à maior eficiência computacional das metodologias. Além disso, considerando que o problema relacionado com o agrupamento de dados é NP-difícil (GAREY; JOHNSON, 1979; SHENG; LIU, 2006), o emprego de heurísticas é uma das abordagens mais utilizadas para a respectiva resolução. No entanto, existem dificuldades para a escolha da mais adequada para cada base de dados.

De acordo com Halkidi, Batistakis e Vazirgiannis (2001), Jain, Murty e Flynn (1999), os algoritmos podem ser amplamente classificados nos seguintes tipos:

- Métodos Hierárquicos: passa sucessivamente por qualquer fusão de *clusters* menores para os maiores, ou dividindo conjuntos maiores. Podem ser divididos em aglomerativos ou divisivos. Segundo os trabalhos de Ng e Han (2002), Mello, Silva e Souza (2007), nos métodos aglomerativos, inicialmente, uma classe é identificada para cada dado. A cada passo dos algoritmos, dois elementos mais similares são agrupados, criando-se uma nova classe e removendo-se as duas classes originais. Este processo iterativo finaliza quando é identificada uma única classe com todos os elementos da base, ou quando se atinge o número de partições desejado. No processo divisivo,

o mecanismo é o contrário. Os algoritmos iniciam com uma única classe contendo todos os dados da base e o processo iterativo da classificação separa os dados a partir de suas informações de similaridade. O resultado do algoritmo é uma árvore, chamada dendrograma, que mostra como os agrupamentos estão relacionados. Ao cortar o dendrograma a um nível desejado, um agrupamento de elementos de dados em grupos de disjunção é obtido. É possível apresentar como vantagens da classificação hierárquica a não necessidade de especificação do número de classes com antecedência. O processo gera classes menores que podem ser úteis para a análise, todavia, os dados podem ser 'incorretamente' agrupados em um estágio inicial. Este fato justifica a importância do acompanhamento para garantir sentido no processo de classificação.

- Métodos de Particionamento: realiza a decomposição do conjunto de dados em um conjunto de *clusters*. Mais especificamente, tentam determinar um número inteiro de partições que otimizam uma determinada função de critério. Esta função pode enfatizar a estrutura local ou global dos dados e sua otimização é um processo iterativo. Os métodos de particiomento trabalham com um número fixo de classes. Estes métodos foram desenvolvidos para agrupar elementos em k classes, sendo k a quantidade de classes definida previamente. Conforme Bussab, Miazaki e Andrade (1990), nem todos valores de k apresentam grupos satisfatórios, sendo assim, aplica-se o método várias vezes para diferentes valores de k, escolhendo os resultados que apresentem melhor interpretação das classes. De acordo com Anderberg (1973), a principal ideia da maioria dos métodos por particionamento é escolher uma partição inicial dos elementos e, em seguida, alterar os dados das classes para obter-se a melhor partição. Em relação aos métodos hierárquicos, o método por particionamento obtém melhor desempenho, tendo em vista a não necessidade de calcular e armazenar, durante o processamento, a matriz de similaridade.

  Considerando que os algoritmos hierárquicos encontram grupos sucessivos utilizando os *clusters* previamente estabelecidos, os algoritmos de particionamento determinam todos os *clusters* de uma só vez.

- Métodos baseados em Densidade: Os métodos baseados em densidade permitem identificar classes de formatos arbitrários. Nestes métodos, as classes são regiões densas de objetos no espaço de dados, sendo separadas por regiões de baixa densidade, que normalmente representam algum tipo de ruído. Um exemplo deste método é o algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*).

- Metodos baseados em Grid: Sua principal característica é quantificar o espaço em um número finito de células e, em seguida, realizar todas as operações no espaço

quantificado. Conforme Han, Kamber e Pei (2006), esta proposta realiza uma divisão dos dados em um número finito de células que formam uma estrutura de grade multidimensional, na qual todas as operações de agrupamento são realizadas. A principal vantagem é que as operações independem do número de objetos da base de dados, e sim do número de células da estrutura da grade, o que melhora o desempenho dos algoritmos baseados nesta heurística. Um exemplo deste método é o algoritmo CLIQUE (AGRAWAL et al., 1998).

Na próxima seção, serão apresentadas algumas metodologias utilizadas para classificação de dados.

2.2.1 Metodologias de Classificação de Dados

Considerando as dificuldades de criação de uma metodologia única para a classificação satisfatória de bases de dados distintas e a difícil interpretação das implicações relacionadas com os grandes volumes de dados em cada base, os esforços dos métodos de classificação podem, frequentemente, estar correlacionados com um aumento significativo do custo computacional (YAN et al., 2012). A escolha de uma ou várias técnicas de classificação está relacionada com a característica da base de dados a ser analisada e da estrutura computacional que será utilizada para análise.

2.2.1.1 K-Means

K-Means é um dos mais simples algoritmos de aprendizado não supervisionado para *clustering* (LIN et al., 2013) e, de acordo com Wu et al. (2008), é amplamente usado em mineração de dados. O procedimento segue uma forma simples para classificação de um determinado conjunto de dados através de um certo número de *clusters*. A principal proposta, no caso, é definir k pontos centrais do espaço dos dados. Cada ponto central está vinculado a uma classe, logo em seguida, o próximo passo é relacionar cada dado à classe referente ao ponto central mais próximo. Quando nenhum dado estiver pendente, os grupos estarão definidos. Conforme os dados vão sendo inseridos dentro do *cluster*, estas operações podem provocar alterações do *centróide*. Isso ocorre devido à mudança do centro do *cluster*, visto que o *centróide* pode ser caracterizado como um ponto médio da distância entre os dados do *cluster*.

De outra maneira, o algoritmo opera em um conjunto de N vetores no espaço Euclidiano denominado D, no qual $D = \{X_i \mid i = 1, ..., N\}$ no qual $X_i \in \mathbb{R}^d$ denota o i-ésimo vetor (ou *ponto de dados*). O algoritmo é inicializado relacionando k pontos em $\mathbb{R}^d$ como sendo os k representantes dos grupos iniciais ou *centróides*. As técnicas usadas para selecionar estes pontos iniciais podem incluir a realização de uma amostragem aleatória do conjunto de dados.

O algoritmo irá, então, iterar entre os dois passos abaixo até convergir.

1. **Atribuição de dados**. Cada ponto de dados (X_i) é associado ao seu *centróide* (y) mais próximo, utilizando a função de distância Euclidiana. Isto resulta na partição dos dados em seus respectivos *centróides*. O conjunto C de dados pertencente ao *centróide* y é representado por $C(y)$;

2. **Realocação dos *centróides***. cada *centróide* de um grupo é realocado para o ponto correspondente à média aritmética de todos os dados que lhe são associados. Se os pontos de dados incluem uma medida de probabilidade (peso), então ocorre a transferência para a média ponderada das partições de dados.

O algoritmo converge quando as atribuições param de sofrer alterações. O comportamento do algoritmo é visualmente exibido na Figura 3; os centróides são representados pelos símbolos "+" e as diferentes cores representam os conjuntos distintos de dados, conforme foram atribuídos durante a execução do *K-Means*.

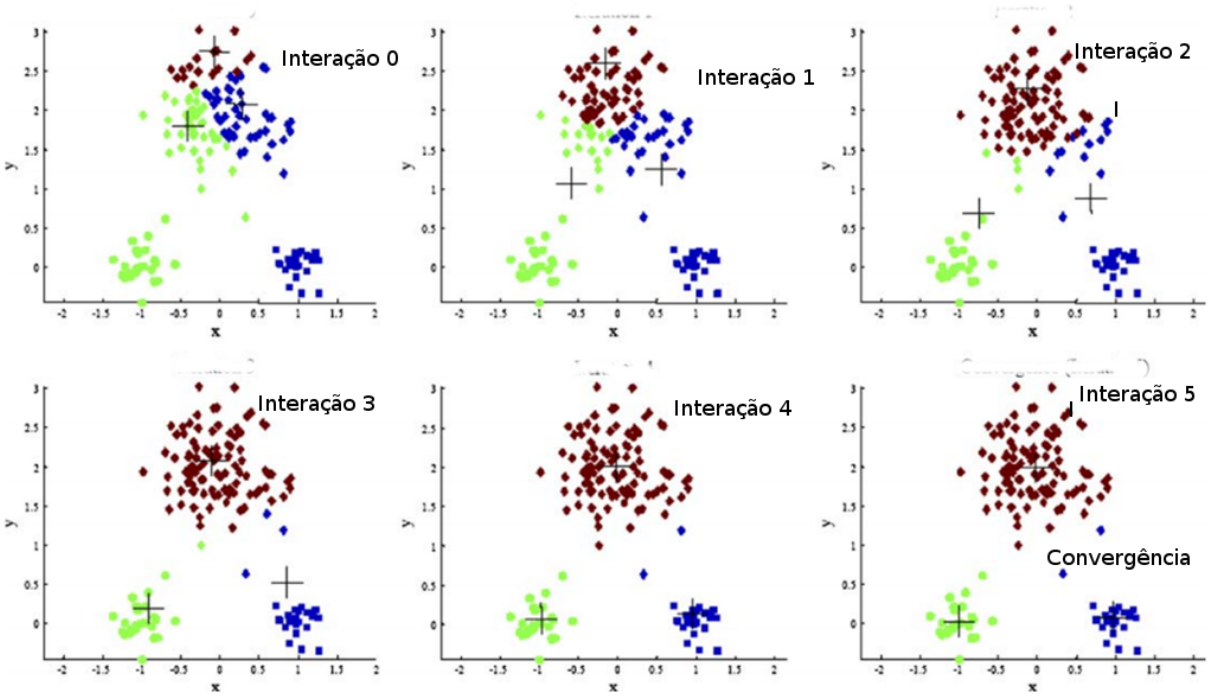

Figura 3 – Mudanças nos *centróides* e dados na execução do *K-Means*, em concordância com Tan, Steinbach e Kumar (2013) (Os *centróides* estão marcados com o sinal '+')

Nota-se que cada iteração necessita de $(N-1) \times k$ comparações, o que determina sua complexidade. O número de iterações necessárias para a convergência pode variar e depende do valor de N mas, como primeira alternativa, este algoritmo pode ser considerado linear no tamanho do conjunto de dados. O processo pode ser simplificado, conforme exibido no Algoritmo 1.

Algoritmo 1 Algoritmo K-Means

1: Escolha dos *centróides* iniciais de forma arbitrária $Y = y_1, ..., y_k \in \mathbb{R}^d$
2: **repeat**
3: Formar k *clusters* associando cada dado a seu *centróide* mais próximo
4: Recalcular cada *centróide* $y \in Y$ como sendo a média dos vetores de $C(y)$
5: **until** encontrar o critério de parada (*centróides* não sofrerem alterações)

As métricas serão abordadas na seção 2.2.2. A Figura 4 ilustra como um resultado insatisfatório pode ser obtido a partir do mesmo conjunto de dados da Figura 3 para três diferentes escolhas iniciais dos *centróides*. Este problema pode ser contornado através das iterações do algoritmo com diferentes *centróides* iniciais ou, também, realizando pesquisas locais limitadas sobre a solução de convergência.

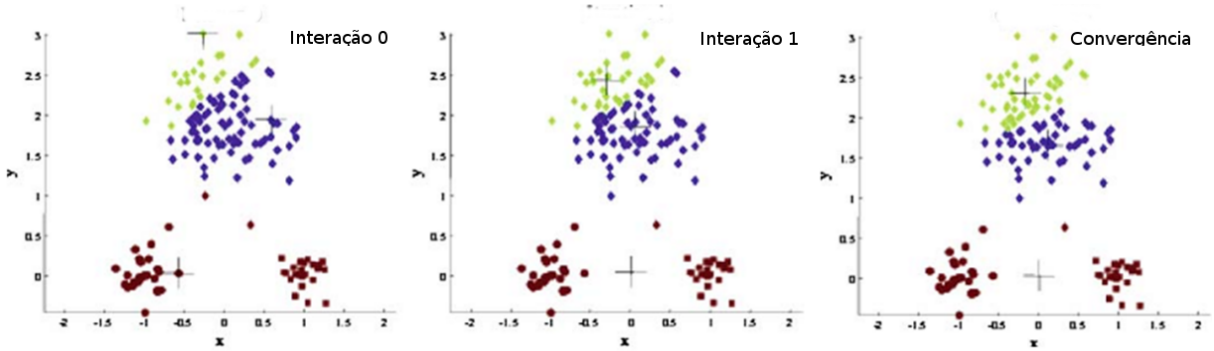

Figura 4 – Efeitos no resultado do *k-Means* devido a uma má inicialização conforme Tan, Steinbach e Kumar (2013) (Os *centróides* estão marcados com o sinal '+')

Um segundo exemplo foi elaborado utilizando uma *applet* implementada por Mirkes (2011)[1]. A Figura 5 (**A**) apresenta uma distribuição de 10 pontos em um espaço. O método inicialmente escolhe três pontos (o número três foi definido pelo usuário) como os *centróides* das classes. Considerando que há 3 classes nesta distribuição. Ao realizar as iterações do processo, o método atualiza os *centróides* que são apresentados na Figura 5 (**B**).

A vantagem do uso deste método é sua eficiência, todavia, ele é incapaz de lidar com dados ruidosos e de *outliers* (dados afastados do centro de seu *cluster*). Apresenta-se muito sensível com relação à escolha inicial de *clusters* e o método necessita especificar o número dos *clusters* com antecedência (ARAI; BARAKBAH, 2007; BRADLEY; FAYYAD, 1998).

Este algoritmo é um método iterativo para particionar um conjunto qualquer de dados em um número k de grupos especificados pelo usuário. Foi elaborado por vários pesquisadores nas mais diferentes disciplinas, com especial ênfase para Lloyd (1982), Forgy (apud JAIN; DUBES, 1988, p. 97), Friedman e Rubin (apud JAIN; DUBES, 1988, p. 97) e MacQueen (apud JAIN; DUBES, 1988, p. 97).

[1] http://www.math.le.ac.uk/people/ag153/homepage

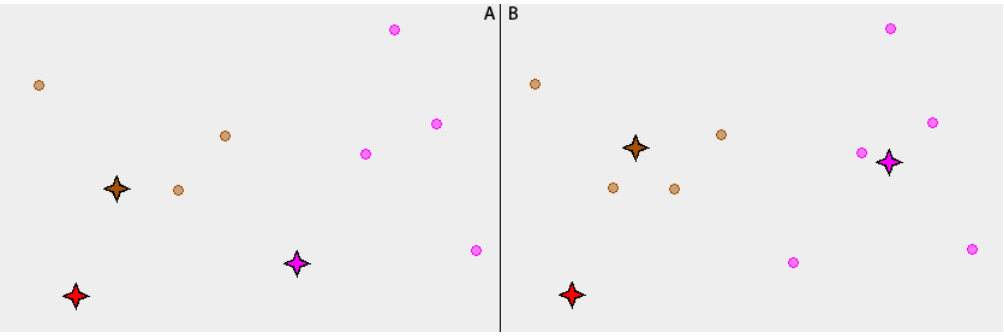

Figura 5 – Distribuição de 10 pontos em um espaço e a definição inicial e aleatória de 3 *centróides* apresentados pela figura (**A**) e a atualização dos *centróides* utilizando o método *K-Means* apresentado pela figura (**B**)

2.2.1.2 K-Medoid

Como foi visto na seção anterior, o método *K-Means* é sensível a *outliers*. Neste sentido, para minimizar a interferência dos *outliers*, em vez de tornar o ponto médio das distâncias entre os dados como ponto de referência do *cluster*, um *medóide* pode ser utilizado (VELMURUGAN; SANTHANAM, 2008). A estratégia é identificar k classes em n elementos e, arbitrariamente, encontrar um elemento representativo (*medóide*) para cada grupo. Cada elemento remanescente é agrupado com o *medóide* ao qual ele é mais similar. A estratégia, então, iterativamente, é atualizar as informações da classe, trocando um dos *medóides* por um dos não *medóides* caso haja melhoria da qualidade do agrupamento resultante. No trabalho de Velmurugan e Santhanam (2010), é realizada uma pesquisa a respeito dos algoritmos de classificação mais representativos, *K-Means* e *K-Medoid*, com base na sua abordagem básica. Alguns trabalhos aplicaram alterações na proposta, objetivando reduzir o cálculo da distância entre os dados e, consequentemente, minimizando o custo computacional (BARIONI et al., 2006). A Figura 6 ilustra o mesmo conjunto apresentado pela Figura 5.

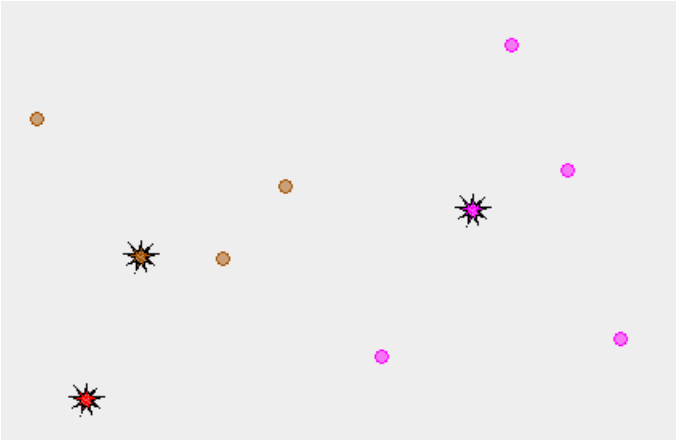

Figura 6 – Atualização dos *centróides*, conforme dados distribuídos na figura 5, utilizando o método *K-Medoid*. A última iteração do processo identifica os dados marcados na figura como os dados mais centralizados de cada *cluster*

Para uma melhor compreensão das diferenças entre os métodos *K-Means* e *K-Medoid* um segundo exemplo é utilizado. Foram distribuídos 20 pontos em um espaço e definiu-se 3 classes para serem analisadas com os dois métodos. A Figura 7 apresenta essa distribuição inicial e as convergências das classes, mediante a identificação dos seus *centróides*, conforme os métodos *K-Means* e *K-Medoid* respectivamente. O exemplo utilizado apresenta uma diferença entre as classes apresentadas. O elemento que foi classificado em classes distintas utilizando os dois métodos é destacado. A ideia básica da estratégia é apresentado no Algoritmo 2.

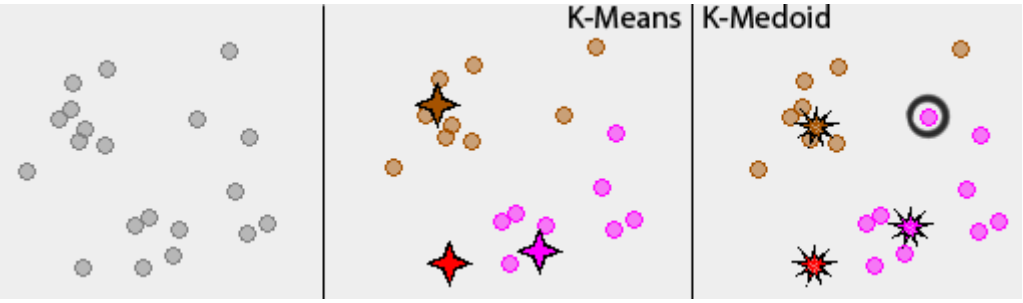

Figura 7 – Distribuição de 20 pontos em um espaço. Identificação das Classes utilizando o método *K-Means* e o método *K-Medoid* respectivamente. Na área correspondente ao uso do método *K-Means*, as áreas marcadas representam os pontos identificados como centrais de cada *cluster*. Na área correspondente ao uso do método *K-Medoid*, os dados identificados representam os dados centrais (*centróides*) de cada *cluster*. O elemento marcado com um círculo na área correspondente a classificação do método *K-Medoid* apresenta uma classificação diferente quando comparado os dois métodos, isto é, utilizando o método *K-Means*, este elemento faz parte do *cluster* representado pela cor **marrom** e utilizando o método *K-Medoid*, este elemento faz parte do *cluster* representado pela cor **rosa**

Algoritmo 2 Algoritmo K-Medoid

1: $kmedoid(k, n)$
2: 'Entrada apresentando o número de classes k e o número de dados n'
3: **repeat**
4: Random(n) - Escolher arbitrariamente um dado como *centróide* inicial da classe
5: **until** identificar *centróides* de todas as k classes
6: **repeat**
7: Registrar cada dado na classe referente ao *centróide* mais próximo.
8: **until** classificar inicialmente todos os n dados
9: **repeat**
10: Aleatoriamente selecionar um dado que não seja *centróide*, s e testar como *centróide* calculando o custo total, C
11: **if** C < Atual **then**
12: Atualizar o *centróide*
13: Verificar se o centróide mais próximo de cada dado continua o mesmo, caso contrário, atualizar
14: **end if**
15: **until** Não há atualizações a serem feitas nas classes
16: **return** Retornar as k partições

Uma das vantagens do método *K-Medoid* é a sua classificação, pela capacidade de

lidar com ruídos, todavia, necessita de especificação do número de *clusters* e em relação ao método *K-Means* é mais lento.

2.2.1.3 Algoritmo de Kernighan e Lin

Os elementos a serem classificados de um conjunto de dados podem ser representados por um grafo, que é uma estrutura que relaciona elementos de um conjunto. Um *grafo* G é um *par ordenado* $(V(G), E(G))$ que consiste em um conjunto $V(G)$ de *vértices* e um conjunto disjunto $E(G)$ de *arestas* de $V(G)$. Associado a este par ordenado, tem-se uma função denominada de *função de incidência*. Esta função associa cada aresta de G com um par de vértices de G. Se este par de vértices for não-ordenado, isto é, se o sentido da aresta não for importante, tem-se um *grafo não-direcionado*. Caso contrário, tem-se um *grafo direcionado* ou *dígrafo*. Os números de vértices e arestas de G são denotados por $v(G)$ e $e(G)$, respectivamente. Esses dois parâmetros básicos de G são chamados de *ordem* (v) e *tamanho* (e) de G.

Neste sentido, as arestas entre objetos do conjunto, representa o grau de pertinência entre estes objetos. A partir daí, técnicas de particionamento de grafos podem ser usadas como métodos de classificação.

O algoritmo de Kernighan e Lin (KL) (KERNIGHAN; LIN, 1970) aplica a técnica de melhoramento iterativo, sendo executado em dados classificados, refinando-os até que certa condição seja atingida. O processo troca pares de vértices de conjuntos disjuntos até se verificar a ocorrência de um mínimo local.

Denominando de G um grafo não-direcionado, com um número par de vértices $n = 2p$, e ponderado pela matriz de custos c, simétrica, deseja-se dividir o domínio de G em dois subconjuntos A e B, cada um com p vértices, de forma que o custo total de conexão entre os subconjuntos seja mínimo. O algoritmo de KL recebe dois possíveis conjuntos A e B iniciais como parâmetros de entrada e busca, por trocas sucessivas entre elementos de A e B, encontrar a melhor solução para o problema.

O método verifica os **custos internos** e **externos** de cada vértice. O **custo externo** de um vértice de um subdomínio é a soma dos custos de acesso deste vértice a todos os vértices dos demais subdomínios. Similarmente, o *custo interno* é a soma dos custos de seu acesso aos demais vértices pertencentes ao mesmo subdomínio. Assim, dado um vértice $a \in A$, define-se seus custos **externo** e **interno** como sendo, respectivamente,

$$E(a) = \sum_{n \in B} c(a, n) \tag{1}$$

$$I(a) = \sum_{q \in A} c(a, q), \tag{2}$$

Após este processo, pode-se calcular a diferença entre os custos externo e interno, obtendo a relação

$$D(a) = E(a) - I(a), \tag{3}$$

Ao trocar de subconjuntos dois vértices $a \in A$ e $b \in B$, obtém-se um fator de ganho g (que pode ser tanto positivo quanto negativo) expresso por

$$g(a, b) = D(a) + D(b) - 2c(a, b), \tag{4}$$

Estas são as principais variáveis analisadas pelo método de Kernighan-Lin. Em uma primeira fase, ele calcula D para todos os vértices de A e de B. Posteriormente, escolhe dois vértices a e b que maximizem o valor de $g(a, b)$ e, consequentemente, leva a uma maior proximidade do ponto de mínimo local. Após retirá-los de seus respectivos subconjuntos, o algoritmo recalcula D para todos os nós do que restou de A e B, ou seja, todos os elementos $r_1 \in \{A - (a)\}$ e $s_1 \in \{B - (b)\}$. Tal cálculo é evidenciado pelas expressões

$$D'(r) = D(r) + 2c(r, a) - 2c(r, b) \tag{5}$$

$$D'(s) = D(s) + 2c(s, b) - 2c(s, a), \tag{6}$$

O algoritmo é executado novamente para um novo par $(r_2,\ s_2)$ com $r_2 \in \{A - (r_1)\}$ e $s_2 \in \{B - (s_1)\}$. Após selecionar r_2 e s_2, recalcula-se D para os elementos restantes dos subdomínios, que agora são $\{A - (r_1, r_2)\}$ e $\{B - (s_1, s_2)\}$. Este procedimento ocorre iterativamente até que se analisem todos os pares $(r_1, s_1), ..., (r_p, s_p)$ e se calcule os seus respectivos ganhos.

Considerando dois subconjuntos $X = r_1, ..., r_k$ e $Y = s_1, ..., s_k$, pode-se analisar que a redução do custo g, ao trocar de subdomínios os conjuntos X e Y, é fornecida por

$$C = \sum_{i=1}^{k} g_i, \tag{7}$$

Para os casos em que $C > 0$, tem-se que a troca resulta em diminuição do custo total entre os subconjuntos. Desta maneira, esta troca é realizada e o resultado é passado como entrada para uma nova execução do algoritmo de Kernighan-Lin. Por outro lado, ao encontrar $C = 0$, uma solução para a entrada inicial A e B foi encontrada. Isto implica na possibilidade de se reexecutar o algoritmo para uma partição inicial distinta, de forma a tentar encontrar uma nova solução melhor do que a já encontrada (MENDES; JR., 2006).

Uma das diversas áreas de aplicação prática do algoritmo de Kerninghan-Lin, é a classificação de grandes bases de dados que necessitam de robusta estrutura de processamento. A divisão de carga entre múltiplas CPUs ou processadores é uma necessidade comum em muitos centros de pesquisa e, visando aumentar a eficiência no balanço de carga dentro de uma rede de computadores, Kernighan-Lin é uma escolha frequente (NETO, 1999).

2.2.1.4 Algoritmo de Fiduccia-Mattheyses

Similarmente ao Kernighan-Lin, este algoritmo usa uma técnica iterativa para refinar uma possível solução arbitrária. Realiza suas movimentações a partir de cálculos de ganho dos nós disponíveis, calculados de maneira similar à adotada no método KL, e também tem por objetivo minimizar o custo total de ligação entre as duas partições do grafo.

Apesar de seguir a mesma linha de raciocínio do KL, o método **Fiduccia-Mattheyses** (ou **FM**), proposto cerca de uma década depois do primeiro por dois pesquisadores da *General Electric* (FIDUCCIA; MATTHEYSES, 1982), apresenta algumas diferenças significativas deste outro algoritmo estudado anteriormente.

O algoritmo movimenta apenas um nó a cada passo, elegendo para esta mudança o nó disponível de maior ganho que mantenha as partições resultantes equilibradas. Este equilíbrio não implica necessariamente tamanho igual para as partições A e B, uma vez que o FM permite dividir o conjunto inicial em partições desbalanceadas ao inserir um fator r que representa a razão entre o tamanho total de ambos os subdomínios.

O método considera que não apenas as arestas, mas também os vértices, apresentam peso. Desta maneira, o "tamanho" total de cada partição é calculado através da soma dos pesos de todos os vértices nela presentes (o peso das arestas é desconsiderado nesta medida).

Com base nas premissas recém-apresentadas, diz-se que o um vértice qualquer selecionado para movimentação respeita a condição de balanço se e somente se

$$r \cdot V - S_{max} \leq |A| \leq r \cdot V + S_{max}, \tag{8}$$

em que $|A|$ é a soma dos pesos de todos os nós da partição A, V é a soma dos pesos de todos os nós do grafo ($V = |A| + |B|$) e S_{max} é o peso da maior célula presente no grafo.

Caso o nó selecionado não satisfaça a condição de balanço, o próximo nó disponível de maior ganho é selecionado e a condição é verificada novamente. Este processo itera até que todos os nós disponíveis no grafo tenham sido movimentados e, neste ponto, a sequência de movimentações que proporcione ganho parcial máximo é oficializado e mantido. Isto concretiza o fim de um passo do algoritmo; depois de tornar permanentes o melhor conjunto de alterações, todos os nós voltam a estado "*disponível*" e o processo recomeça (BRAUER, 2004).

O algoritmo tem como condição de parada o momento em que nenhum nó disponível satisfaça a condição de balanço ou a iteração em que todos os nós estejam disponíveis e apresentem ganho menor ou igual a zero (ou seja, quando todos os nós já se encontram nas suas partições otimizadas).

2.2.1.5 Algoritmos Espectrais

Kernigham e Lin desenvolveram um método combinatório eficaz baseado na troca de vértices (HOLZRICHTER; OLIVEIRA, 1999; KERNIGHAN; LIN, 1970). Extensões de vários

níveis do algoritmo de Kernighan-Lin têm se mostrado eficazes para grafos com grande número de vértices (HOLZRICHTER; OLIVEIRA, 1999; KARYPIS; KUMAR, 1998). Como métodos combinatórios, métodos espectrais têm se mostrado eficazes para grandes grafos (HOLZRICHTER; OLIVEIRA, 1999; BARNARD; SIMON, 1994). Técnicas de algoritmos espectrais também são utilizadas na classificação para aumentar a precisão e minimizar o custo computacional (HUAZHONG et al., 2010).

Conforme Abreu (2005) e Nascimento (2010) a Teoria Espectral em Grafos iniciou-se na Química Quântica, por meio de um modelo teórico de moléculas de hidrocarbonetos não saturadas. Tais moléculas possuem ligações químicas com diversos níveis de energia de elétrons. Alguns desses níveis de energia podem ser representados por autovalores de um grafo, o que caracteriza o estudo da Teoria Espectral. A fundamentação teórica da Teoria Espectral em Grafos iniciou-se com Hall (1970) e Cvetkovic, Doob e Sachs (1980).

Em teoria dos grafos, a matriz Laplaciana, também denominada matriz de Admitância ou matriz de Kirchhoff, é um representante matricial de um grafo, sendo ideal para realizar a teoria espectral de grafos.

Seja W a matriz de pesos de um grafo G, onde $W = [w_{ij}]_{n\times n}$, com $w_{ij} \in \Re$. O elemento w_{ij} de W indica o peso da aresta (i, j) de G. Seja $D = [d_{ij}]_{n\times n}$,com $d_{ij} \in \Re$, uma matriz diagonal, definida por $d_{jj} = \sum_{i=1}^{n} w_{ij}$, com $j = 1, ..., n$. A matriz Laplaciana não-normalizada de G, definida por L, é dada pela equação 9:

$$L = D - A, \tag{9}$$

O qual A é a matriz de adjacência, ou matriz de pesos, e D é a matriz diagonal contendo o somatório das linhas de A.

Como um exemplo de algoritmo espectral, o método Fiedler é um método espectral utilizado para particionar grafos com o intuito de formar *clusters* (FIEDLER, 1973). Este método é aplicado sobre a matriz Laplaciana de um grafo. A partir da matriz Laplaciana é necessário calcular o espectro, que corresponde aos autovalores da matriz Laplaciana ordenados de forma decrescente.

O método consiste, portanto, em calcular o autovetor associado ao segundo menor autovalor do espectro, este autovalor é denominado **conectividade algébrica** $a(G)$, que fornece informações sobre a conectividade de um grafo. Este autovetor informado é denominado Vetor de Fiedler (*Fiedler Vector*) e pode ser utilizado para particionar um grafo a fim de minimizar as conexões entre nós de partições distintas do grafo, isto é, ele contém informações que dividem um grafo em dois conjuntos A e B, tal que o corte de arestas entre esses conjuntos é o mínimo possível. O método utiliza os sinais dos valores do Vetor de Fiedler. As linhas que possuem os mesmos sinais devem ter os nós correspondentes na mesma partição e as linhas com sinal negativo são colocadas em outra. No entanto, é possível de se ter valor zero no vetor, e neste caso deve-se atribuir o nó arbitrariamente a uma das partições. Cada partição pode ser dividida novamente com o mesmo processo.

Conforme Benson-Putnins et al. (2011), o Método ganhou aceitação como uma técnica de agrupamento viável para grafos simples. Há, no entanto, algumas desvantagens para este método como o fato de exigir um custo computacional alto para grandes bases de dados.

2.2.1.6 Árvore Binária

Árvores são estruturas de dados que caracterizam uma relação de hierarquia ou de composição entre os dados que as compõem. Formalmente, uma árvore é um conjunto finito T de um ou mais nós, tais que:

1. existe um nó denominado raiz da árvore;
2. os demais nós formam $m \geq 0$ conjuntos disjuntos $S_1, S_2, \ldots, S_m$, nos quais cada um destes conjuntos é uma árvore. As árvores $S_i (1 \leq i \leq n)$ recebem a denominação de subárvores.

Uma **Árvore Binária** é aquela em que o grau de cada nó é no máximo igual a 2. O grau de um nó é o número de subárvores daquele nó. A Figura 8 ilustra uma **Árvore Binária**. O nó 1 representa a raiz da árvore e os nós 4, 5, 6 e 7 correspondem aos nós folhas.

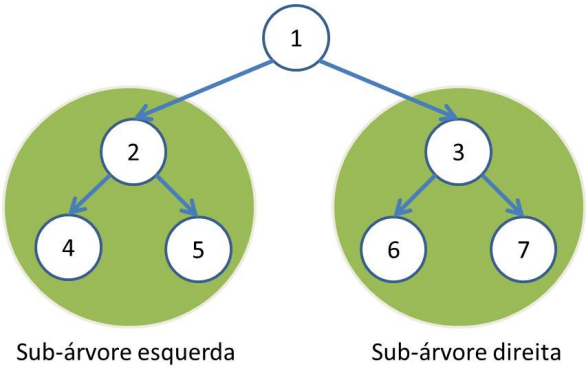

Figura 8 – Ilustração da estrutura de dados **Árvore Binária** definida por um elemento distinto, denominado raiz, com dois ponteiros para duas estruturas diferentes, denominadas sub-árvore esquerda e sub-árvore direita. Os nós de uma árvore binária possuem graus zero, um ou dois. Um nó de grau zero é denominado folha. Em uma árvore binária, por definição, cada nó poderá ter até duas folhas. Uma árvore "estritamente binária"é uma árvore na qual todo nó tem zero ou duas folhas

A construção de uma **Árvore Binária** é realizada no sentido *bottom/up*. Inicialmente, tem-se n conjuntos distintos ou partições, o qual n é a cardinalidade do conjunto de dados original. Os dois objetos mais próximos são identificados. Estes objetos serão unidos em uma mesma árvore, tornando-se as subárvores esquerda e direita dessa nova árvore; este processo é iterativo até montar toda a árvore a partir de todo o conjunto de dados. O critério de parada está vinculado até o número de subárvores atingir o número de classes definidas.

2.2.1.7 Árvores de Decisão

Métodos supervisionados pressupõem a existência de dados cuja classe é conhecida para a inicialização do processo de particionamento. Uma dessas técnicas de classificação é o método da árvore de decisão. Segundo Friedl e Brodley (1997), as árvores de decisão têm como vantagem o fato de serem simples e flexíveis, realizando testes de fácil compreensão, cuja semântica é praticamente intuitiva. Vale destacar outra dificuldade no processo de classificação, que é a existência de sobreposição de dados. Para isto, trabalhos com abordagem de *clustering* hierárquico foram desenvolvidos no intuito de implementar uma árvore de *cluster*. Segundo Xutao et al. (2010), estruturas como árvores foram implementadas como tentativa de identificar informações presentes nestes conjuntos.

De acordo com Aitkenhead (2008), para muitos problemas de classificação em que grandes conjuntos de dados complexos são usados, as árvores de decisão fornecem uma solução útil. A definição de uma árvore de decisão foi dada em Russell e Norvig (2002). As árvores de decisão representam funções booleanas. A combinação destas funções em uma estrutura hierárquica tem a forma de uma árvore de decisão. Conforme Aitkenhead (2008), em termos de capacidade, as árvores de decisão constituem um método rápido e eficaz de classificar entradas do conjunto de dados, podendo proporcionar boa capacidade de apoio à decisão (SALZBERG, 1995; SALZBERG et al., 1998; GREENSPAN, 1998; JENSEN; ARNSPANG, 1999; BERNSTEIN; PROVOST, 2001; MARKEY; TOURASSI; FLOYD, 2003; ZMAZEK et al., 2003).

Conforme Mitchell (1997), as árvores de decisão classificam instâncias partindo da raiz da árvore para algum nó folha que fornece a classe da instância. Cada nó da árvore especifica o teste de algum atributo da instância, e cada aresta ligada àquele nó corresponde a um dos possíveis valores deste atributo. Uma instância é classificada começando no nó raiz da árvore, e testando o atributo relacionado a este nó e seguindo a aresta que corresponde ao valor do atributo na instância em questão. Este processo é repetido para a sub-árvore abaixo até chegar a um nó folha.

A maioria dos algoritmos que foram desenvolvidos para aprendizado com árvores de decisão é uma variação de um algoritmo que emprega o método *top-down*. Uma árvore de decisão tem a função de particionar recursivamente um conjunto de dados, até que cada subconjunto obtido deste particionamento contenha informações de uma única classe. Para atingir esta meta, os algoritmos examinam e comparam a distribuição de classes durante a construção da árvore. O resultado obtido, após a construção de uma árvore de decisão, são dados organizados de maneira compacta, que são utilizados para classificar novos casos (HOLSHEIMER; SIEBES, 1994).

Os algoritmos ID3 e C4.5 são exemplos de árvores de decisão, sendo o algoritmo ID3 um dos primeiros algoritmos de árvore de decisão apresentados, que teve sua elaboração baseada em sistemas de inferência e em conceitos de sistemas de aprendizagem. Ele constrói árvores de decisão a partir de um dado conjunto de exemplos, sendo a árvore

resultante usada para classificar amostras futuras. Já o algoritmo C4.5 é um aprimoramento do algoritmo ID3, permitindo trabalhar com valores contínuos, realizar cortes na árvore e ajuste nas regras.

O algoritmo ID3, a partir de todos os dados do conjunto, escolhe o atributo que melhor divide a base, isto é, o dado que proporciona o melhor ganho de informação (para achar o melhor atributo é necessário encontrar a entropia para cada atributo possível naquele nó, realizando o agrupamento. Dado um conjunto S contendo dados positivos e negativos de algum conceito definido, a entropia de S relativa a esta classificação booleana é dada pela equação 10).

$$E(S) = -p_+ log_2 p_+ - p_- log_2 p_-, \tag{10}$$

o qual p_+ é a proporção de exemplos positivos em S e p_- é a proporção de exemplos negativos em S. Exemplificando, para uma situação onde um atributo identificado aceita c diferentes valores, a entropia de S relativa a esta classificação c-classes é definida na equação 11.

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i, \tag{11}$$

o qual p_i é a proporção de S pertencendo a classe i. O algoritmo ID3 é apresentado no Algoritmo de número 3.

Algoritmo 3 Algoritmo ID3

1: Escolher atributo que melhor divide a base.
2: Calcular Entropia para cada atributo possível.
3: **repeat**
4: Para o atributo escolhido, criar um nó filho para cada valor possível do atributo.
5: Calcular a entropia de todo o conjunto de dados.
6: Diminuir o atributo escolhido da entropia do sub-conjunto particionado.
7: Transportar os exemplos para cada filho, tendo em conta o valor do filho.
8: Repetir o procedimento para cada filho não "puro". *(Um filho é puro quando cada atributo X tem o mesmo valor para todos os exemplos)*
9: **until** Escolher todos os atributos do conjunto de dados - *O atributo com o maior ganho de informação será o selecionado para o próximo nó da árvore (próxima iteração do **loop**)*

O algoritmo ID3 usa o conceito de entropia para estabelecer o melhor atributo a ser utilizado para dividir os dados em sub-grupos. Após a construção de uma árvore de decisão é importante avaliá-la. Esta avaliação é realizada através da utilização de dados que não tenham sido empregados. Esta estratégia permite estimar como a árvore generaliza os dados e se adapta a novas situações, podendo, também, estimar a proporção de erros e acertos ocorridos na construção da árvore.

Na área de Física usa-se o termo entropia para descrever a quantidade de desordem associada a um sistema (SOUZA; NOVA; GONÇALVES, 1999). O Algoritmo ID3 usa este conceito para encontrar o melhor atributo de um dado para ser utilizado como nó de uma árvore de decisão. Logo, o objetivo do algoritmo ID3 é encontrar um atributo

que reduza significativamente a entropia de um conjunto de dados. Desta forma, estará essencialmente encontrando o melhor atributo para classificar a base.

A principal desvantagem do Algoritmo ID3 é que todo o processo só analisa atributos discretos, não sendo possível apresentar conjuntos de dados com atributos contínuos. Neste caso, os atributos contínuos devem ser discretizados. Este método também não trata valores desconhecidos, isto é, todos os exemplos do conjunto de treinamento devem ter valores conhecidos para todos os seus atributos. O algoritmo também não realiza cortes na árvore durante o processo, não permitindo minimizar a busca em árvores.

O Algoritmo C4.5, desenvolvido por John Ross Quinlan (QUINLAN, 1993), lida com atributos discretos e contínuos. O algoritmo apresenta dois estados durante o processo, folha que indica um ponto no final da classificação, sendo atribuída a uma classe e nó de decisão. Assim, baseando-se no atributo em análise, poderá conter uma ramificação seguida de uma folha ou uma sub-árvore para cada possível valor encontrado na base. O método ignora valores desconhecidos, portanto, não utiliza a amostra para cálculos. O método utiliza a medida de razão de ganho para selecionar o melhor atributo que divide os exemplos, sendo que a medida superior ao ganho de informação do ID3 gera árvores menos complexas. A fórmula da razão de ganho é representada na equação 12, dividindo o ganho pela entropia.

$$R = \frac{G}{E}, \tag{12}$$

Desta forma, o algoritmo C4.5, no primeiro passo, calcula o ganho de informação para todos os atributos. A partir dos atributos que obtiveram o ganho de informação acima da média, o método escolhe o atributo com a melhor razão de ganho para ser o nó raiz da árvore.

Algoritmo 4 Algoritmo C4.5

```
C4_5(R, C, S)
if S = Vazio then
    Retornar um Nó com valor negativo (Erro)
end if
if R = Vazio then
    Retornar uma folha com o valor mais frequente do atributo categórico
else
    Calcular o atributo D com maior ganho de informação, entre os atributos restantes de R
    Incluir Nó D
    Dividir S em subclasses, de acordo com os valores de D

    return Chamar novamente a Função para cada um desses valores (Recursividade)
end if
```

2.2.1.8 Fuzzy C-Means

Introduzido no trabalho de Dunn (1973), aperfeiçoado na obra de Bezdek (1981) e conforme discutido em Rocha et al. (2012), o algoritmo Fuzzy C-Means (FCM) é baseado na inserção de conceitos da Teoria de Conjuntos Fuzzy no algoritmo K-Means, sendo caracterizado como um Sistema Híbrido. Um Sistema Híbrido é definido pelo uso combinado de características referentes a duas ou mais técnicas da Inteligência Artificial, ou de áreas afins, em um único modelo, com o intuito de aproveitar o que de melhor cada uma das técnicas envolvidas pode oferecer à resolução de um problema.

O agrupamento fuzzy é uma generalização dos métodos por particionamento e, assim como nos métodos por particionamento, também é necessário indicar o número inicial de classes. Nos métodos tradicionais de particionamento, define-se em qual grupo ficará cada elemento. Já a classificação fuzzy permite visualizar o grau de pertinência de cada elemento em cada grupo, que geralmente se verifica em domínios de dados reais, onde um elemento pertence a diferentes grupos, com diferentes graus de associação.

A principal vantagem dos agrupamentos fuzzy em relação aos outros métodos é que ele fornece informações mais detalhadas sobre a estrutura dos dados, pois são apresentados os graus de associação de cada elemento a cada classe. A desvantagem desse método é que a quantidade de coeficientes de associação cresce rapidamente com o aumento do número de informações e de classes. Entretanto, trata-se de uma técnica válida, pois ela associa graus de incerteza aos elementos nas classes e, essa situação, em geral, se aproxima mais das características reais dos dados (KAUFMAN; ROUSSEEUW, 2009).

O FCM é um algoritmo de agrupamento fuzzy bastante utilizado. Trata-se de um algoritmo iterativo que inicia com c valores arbitrários, e com base nesses valores, associa cada elemento ao valor ao qual possui menor distância, formando c grupos. Em seguida, calcula-se o centro de cada grupo formado, e os elementos são reassociados ao centro mais próximo. Assim, os cálculos prosseguem, iterativamente, até que as diferenças entre os centros do passo atual e do anterior sejam mínimas.

Conforme Halkidi, Batistakis e Vazirgiannis (2001), FCM tenta localizar o ponto mais característico em cada *cluster*, o que pode ser considerado como o 'centro' do *cluster* e, em seguida, o grau de adesão para cada objeto nos *clusters*.

2.2.1.9 Enxame de Partículas

O método do enxame de partículas é um ramo da IA classificado por alguns autores como um ramo da computação evolutiva que otimiza um problema iterativamente ao tentar melhorar a solução candidata. O método do enxame de partículas foi proposto em Kennedy e Eberhart (1995), sendo um algoritmo de busca estocástica que é similar aos algoritmos de Computação Evolutiva. Entretanto, a forma como a informação é utilizada para a otimização não é inspirada em operadores genéticos, mas em dinâmica de enxames.

Basearam-se no modelo de comportamento social dos bandos de aves e tentaram simular os comportamentos imprevisíveis que esses conjuntos apresentavam, com a finalidade de descobrir um padrão para alguns de seus comportamentos. A técnica foi desenvolvida segundo algumas definições, nas quais a partícula consiste em um indivíduo que toma decisões a fim de produzir um evento que poderá otimizar a situação da população que, por sua vez, representa um conjunto de partículas com as mesmas funções objetivo.

O método de Enxame de Partículas é similar ao método da colônia de formigas. Segundo Lintzmayer, Mulati e Silva (2011), é uma meta-heurística de otimização combinatória, que se baseia no comportamento das formigas artificiais. Uma meta-heurística é um conjunto de conceitos algorítmicos que podem ser usados para definir métodos aplicáveis a um grande volume de diferentes problemas. Segundo Dorigo e Socha (2006), Lintzmayer, Mulati e Silva (2011) a execução deste algoritmo geralmente é composto por vários ciclos, onde atua a colônia. Cada formiga é geralmente um método construtivo e seu comportamento no algoritmo se manifesta quando há necessidade de definir o próximo passo do processo. Neste ponto, utiliza-se uma probabilidade com base em dois fatores: a trilha de ferormônio e a informação heurística (desejabilidade ou atratividade) relacionadas ao item. A informação heurística depende de cada problema.

Existem pesquisas, como o trabalho de Medeiros (2005), que mostram a capacidade do algoritmo de otimização por enxame de partículas como um método viável para busca de soluções em espaços complexos e de alta dimensão, como os problemas encontrados na área nuclear. Apresentando o método como um algoritmo simples cuja implementação é facilmente traduzida a partir de equações, torna-se natural a representação de modelos de funções reais em espaços reais de qualquer dimensão, mostrando-se eficaz na busca de soluções em espaços de alta dimensão sem necessidade de conhecimento prévio sobre a complexidade dos espaços de busca envolvidos. Em outros trabalhos, como Jeronymo, Borges e Coelho (2010), em que apresentam o uso deste método no planejamento de trajetória de um manipulador robótico, os resultados mostraram eficiência melhor do que as dos AGs.

Existem muitas variações do Enxame de Partículas, tais como métodos adaptativos, métodos específicos para movimentos do tipo abelha, formiga, entre outros. O processo se baseia em duas equações para cada partícula, o qual $x(k)$ é a posição de uma partícula e $v(k)$ sua velocidade (WAINTRAUB, 2009). A equação 13 apresenta o cálculo da posição da partícula.

$$x(k+1) = x(k) + v(k+1) \tag{13}$$

A equação 14 apresenta a equação da velocidade da partícula.

$$v(k+1) = v(k) + a * Random * (p - x(k)) + b * Random * (G - x(k)) \tag{14}$$

A partícula precisa verificar se o caminho é o mais apropriado para a sua classe. A variável p informa o valor mais adequado que foi atingido no processo em busca do ótimo. O processo registra uma estatística com os erros passados e grava o melhor valor, representado pela varíavel p. Se a partícula estiver traçando um caminho diferente das demais, será necessário ajustá-la para ficar mais próximo do ótimo da classe. O ótimo global é representado pela variável G. A evolução do conhecimento adquirido pela partícula é representado pela variável a. A diferença $(p - x)$ só representará algo se a for um valor maior.

O enxame é inicializado com os valores dos vetores de velocidade e posição gerados aleatoriamente. A primeira iteração do algoritmo inicia com a atribuição de valores aos parâmetros da equação de velocidade. Definem-se as informações e, principalmente, o critério de parada. Tendo já definido os valores para a posição das partículas e suas respectivas velocidades, aplica-se o cálculo do *fitness* a cada partícula. A Aptidão ou *fitness* é um dos principais conceitos na teoria da evolução. O valor adaptativo de um fenótipo é uma medida simples de sucesso reprodutivo, onde aqueles que sobrevivem e produzem maior número de prole capaz de se reproduzir, garantem a manutenção do gene da sua população, de outra forma, o valor extraído do *fitness* pode ser interpretado como a contribuição de um indivíduo perante todo o conjunto de genótipos existentes dentro da população em que está inserido. Referente ao enxame, a função *fitness* avalia o desempenho da partícula. Com as partículas do enxame avaliadas, extraem-se a melhor posição encontrada pela partícula e pelo enxame. Depois é realizado o processo de atualização com as velocidades e as posições de cada partícula do enxame. Caso o critério de parada tenha sido atingido, a solução do problema encontrada é apresentada. Caso contrário, aplica-se novamente a função *fitness* a este enxame. O processo de iteração prossegue até o critério de parada ser alcançado.

2.2.2 Métricas de Similaridade

O trabalho apresentado por Lin (1998) apresenta 3 conceitos intuitivos de similaridade:

1. A semelhança entre A e B está relacionada a sua característica comuns. Quanto mais informações características comum eles compartilham, mais eles são semelhantes.

2. A semelhança entre A e B está relacionado com as diferenças entre eles. Quanto maior as diferenças, menos similares eles são.

3. A similaridade máxima entre A e B é alcançado quando A e B são idênticos, não importa o quanto eles compartilham em comum.

Uma medida de similaridade pode então ser derivada a partir desses pressupostos. A fim de captar a intuição de que a similaridade de dois objetos estão relacionados à sua semelhança, é necessário uma medida. Lin (1998) apresenta 6 hipóteses:

1. O ponto em comum entre A e B é avaliado pela equação 15.

$$I(C(A,B)), \tag{15}$$

O qual $C(A,B)$ é uma proposição que afirma a similaridade entre A e B. $I(x)$ é a quantidade de informação contida em uma proposição x.

2. As diferenças entre A e B são medidas pela equação 16.

$$I(D(A,B)) - I(C(A,B)), \tag{16}$$

O qual $D(A,B)$ é a proposição que descreve o que A e B são.

3. A similaridade entre A e B, é uma função de suas semelhanças e diferenças.

$$S(A,B) = f(I(C(A,B)), I(D(A < B)), \tag{17}$$

O domínio de f é:

$$(x,y)|x \leq 0, y > 0, y \leq x, \tag{18}$$

4. A similaridade entre dois objetos idênticos é 1. Quando A e B são idênticos $I(C(A,B)) = I(D(A,B))$. Quando não há nenhuma semelhança entre A e B, a similaridade é 0.

5. A similaridade geral dos dois elementos A e B é uma média ponderada de suas semelhanças calculadas a partir de diferentes perspectivas.

6. Este item apresenta o teorema de similaridade que informa que a semelhança entre A e B é medida pela razão entre a quantidade de informação necessária para afirmar a comunhão de A e B e as informações necessárias para descrever completamente o que A e B são:

$$S(A,B) = \frac{logP(C(A,B))}{logP(D(A,B))}, \tag{19}$$

A seleção dos fatores causais ocorre de acordo com a semelhança entre eles e, subsequentemente, através de uma avaliação dos fatores preponderantes para gerar as propriedades observadas. Em outras palavras, os fatores mais determinantes para a definição de um perfil classificatório podem receber um peso maior na elaboração dos cálculos que constituirão a classificação propriamente considerada.

De fato, essa avaliação desempenha um papel importante na classificação dos dados. Classificar um dado semelhante a um outro dado baseia-se em uma métrica ou função de distância entre os padrões. Segundo Duda, Hart e Stork (2001), uma métrica deve ter quatro propriedades: para todos os vetores de a, b, e c, estas propriedades são as seguintes:

- não-negatividade: $D(a,b) \geqslant 0$,

- reflexividade: $D(a, b) = 0$ se e somente se $a = b$,
- simetria: $D(a, b) = D(b, a)$,
- triângulo desigualdade: $D(a, b) + D(b, c) \geqslant D(a, c)$,

Diante da importância das métricas de distância empregadas para a classificação de dados e da complexidade para definir a melhor métrica para determinado conjunto desses dados, vários trabalhos na literatura apresentam estudos comparando essas métricas (ZWICK; CARISTEIN; BUDESCU, 1987; SURAL; QIAN; PRAMANIK, 2002). Existem trabalhos que sugerem uma nova proposta de métrica de distância, como o caso de Giotis e Petkov (2012), que introduz uma métrica adaptativa, alcançando melhores resultados nos algoritmos de classificação com base na distância.

Ao analisar um conjunto de dados ou atributos como dimensões de um espaço multidimensional, a descrição de cada dado corresponde a um ponto nesse espaço. Por conseguinte, a distância entre os pares de objetos pode ser utilizada como grau de similaridade entre eles. Uma das métricas utilizadas para averiguar a distância entre os dados é a distância geométrica Euclidiana (Equação 20).

$$D_{i,k} = \sqrt{\sum_{j=1}^{n} (X_{ij} - X_{kj})^2} \tag{20}$$

Na literatura, encontram-se várias métricas para serem usadas em um processo de classificação (DUDA; HART; STORK, 2001), sendo que a distância de Minkowski é uma métrica no espaço Euclidiano que pode ser considerada como uma generalização da distância Euclidiana e da distância de Manhattan.

A distância de Minkowski de ordem p entre dois pontos é definida como:

$$(\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p} \tag{21}$$

Trabalhos como Vimal, Valluri e Karlapalem (2008), Maddah et al. (2008), Tsai et al. (2007) informam que a escolha da métrica de distância mais adequada não é um processo trivial. Existem diversas métricas, sendo que a Distância de Hamming (HAMMING, 1950), Distância de Levenshtein (LEVENSHTEIN, 1966), Distância de Jaro (JARO, 1995; JARO, 1989), Jaro Winkler (WINKLER, 1999), Monge Elkan (MONGE; ELKAN, 1997; MONGE; ELKAN et al., 1996), Damerau (DAMERAU, 1964), Soundex (PHUA; LEE; SMITH-MILES, 2006), Similaridade do Cosseno (BILENKO; MOONEY, 2003), são utilizadas para tratar *strings*. Para maiores esclarecimentos de métricas que analisam *strings*, a distância de Hamming e a distância de Levenshtein serão mais detalhadas a seguir:

- Distância de Hamming: A distância de Hamming corresponde ao menor número de substituições necessárias para transformar uma *string* em outra, ou o número

de erros que transformaram uma na outra. A função apresentada pelo algoritmo 5 ilustra o peso de Hamming, utilizado para aplicação da métrica. O cálculo da distância de Hamming tem complexidade $O(n)$.

Algoritmo 5 Cálculo da Distância de Hamming

1: $Hamming(x, y)$
2: dist = 0, val = x^y; // Contador do Número de Conjunto de *bits*
3: **repeat**
4: Incrementar dist;
5: val (Endereço de Memória) = val - 1;
6: **until** val
7: **return** dist

A função Hamming retorna um valor entre 0 e 1 que representa a proporção de caracteres iguais entre as duas *strings*, indicando uma medida de similaridade. Caso as *strings* não sejam do mesmo tamanho, faz-se a comparação levando em conta o tamanho da menor sequência de caracteres. O excedente da maior cadeia é descartado para a comparação. A função Hamming de similaridade é classificada como baseada em caractere.

- Distância de Levenshtein: Também conhecida como *edit distance*, a distância de Levenshtein entre duas *strings* é dada pelo custo do conjunto de operações necessárias para transformar uma *string* em outra, o qual uma operação pode ser uma inserção, deleção ou substituição de um caractere. A equação de cálculo do custo pode variar, dando origem a outras variantes. A distância de Levenshtein é um algoritmo de complexidade $O(m \times n)$ sendo m e n representadas pelos tamanhos das *strings* analisadas.

Outras métricas são apresentadas a seguir:

- Similaridade de Jaccard: O índice de Jaccard, também conhecido como o coeficiente de similaridade de Jaccard, é uma estatística utilizada para comparar a similaridade de conjuntos de amostras. O coeficiente Jaccard é definido como o tamanho da intersecção dividida pelo tamanho da união dos conjuntos de amostras:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{22}$$

A distância de Jaccard, que mede a desigualdade entre conjuntos, é complementar ao coeficiente de Jaccard e é obtido subtraindo o coeficiente de Jaccard de 1, ou, equivalentemente, dividindo-se a diferença entre os tamanhos da união e da interseção de dois conjuntos pelo tamanho da união:

$$D_j(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}, \tag{23}$$

- Distância e Similaridade de Taimoto: Este método é semelhante a similaridade de Jaccard, o trabalho de Rogers e Tanimoto (1960) apresenta maiores informações deste método.

- Coeficiente de Sobreposição: É uma medida de similaridade relacionada com o coeficiente de Jaccard, que mede a sobreposição entre os dois conjuntos, e é definido como o tamanho da intersecção dividido pelo menor valor dos dois conjuntos:

$$overlap(A,B) = \frac{|A \cap B|}{min(|A|,|B|)}, \tag{24}$$

  se X é um subconjunto de Y ou o inverso, então o Coeficiente de Sobreposição é igual a um.

- Hellinger/Bhattacharyya: A distância de *Bhattacharyya* mede a similaridade de duas distribuições de probabilidade discretas ou contínuas. Para as distribuições de probabilidade discreta p e q em relação há um domínio X, é definida a equação 25, sendo a equação 26 o coeficiente de Bhattacharyya.

$$D_B(p,q) = -ln(BC(p,q)), \tag{25}$$

$$BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)}, \tag{26}$$

  Para as distribuições de probabilidade contínua, o coeficiente de Bhattacharyya é definido na equação 27.

$$BC(p,q) = \int \sqrt{p(x)q(x)}dx \tag{27}$$

  A distância de *Bhattacharyya* é utilizada em pesquisas de extração de características, processamento de imagens, reconhecimento de fala entre outros (YOU; LEE; LI, 2009; REYES-ALDASORO; BHALERAO, 2006; GOUDAIL; RÉFRÉGIER; DELYON, 2004; CHOI; LEE, 2003; MAK; BARNARD, 1996).

- Média Harmônica: Métrica utilizada para tratar grandezas inversamente proporcionais, como exemplo velocidade e tempo. A equação 28 apresenta a fórmula para identificar a média.

$$Media = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3} + ... + \frac{1}{x_n}} \tag{28}$$

- Distância de Manhattan: A distância de Manhattan é definida pela distância entre dois pontos medidos ao longo dos eixos em ângulos retos, isto é generalizado para dimensões maiores. A distância de Manhattan é definida pela equação 29:

$$d_{xy} = |X_1 + Y_1| + |X_2 + Y_2| + ... + |X_p - Y_p| = \sum_{i=1}^{p} |X_i - Y_i|, \tag{29}$$

- Distância de Chebychev: A distância de Chebychev é apropriada no caso em que se deseja definir dois elementos como diferentes, se apenas umas das dimensões é diferente. Ela é definida pela equação 30:

$$d_{x2x3} = max(|X_1 + Y_1|, |X_2 + Y_2|, ..., |X_p - Y_p|), \tag{30}$$

Apesar da grande quantidade de métricas de similaridades, devido o grande volume de informações a serem tratadas e de suas peculiaridades, há ainda demanda por melhores métricas (CHA, 2007).

2.2.3 Estabilidade de *Cluster*

Mesmo melhorando os métodos aplicados, realmente, não há qualquer garantia de que um determinado método irá agrupar os dados corretamente, de acordo com o entendimento dos especialistas que contribuem com a tarefa subjetiva de interpretação dos dados. De fato, vários artigos apresentam esta dificuldade sobre o procedimento de classificação (LIU; WU; SHEN, 2011; CAO; LIANG; BAI, 2009; BLASHFIELD, 1976). O particionamento atingido pelo método escolhido pode, por conseguinte, ocasionar em uma extração de qualidade relativamente baixa. Assim sendo, estudos, tais como análises da estabilidade de *cluster*, como é o caso do trabalho apresentado por Milagre et al. (2009), possuem grande relevância na validação do particionamento. Conforme Maciel e Milagre (2008), a análise de estabilidade de *cluster* é uma maneira de validar o particionamento dos dados encontrados por meio de qualquer algoritmo de agrupamento.

2.3 Algoritmos Evolutivos

A teoria da evolução das espécies é utilizada como conceito para aplicação de algoritmos em dados complexos (DARWIN; BEER, 1998). Na natureza, não só animais aprendem a ter um melhor desempenho, mas as espécies evoluem de fato. As técnicas nas quais os modelos de evolução biológica têm sido propostos como métodos de aprendizagem podem melhorar o desempenho de programas de computador. Dentre os referidos modelos, os Algoritmos Genéticos (AGs) e os Algoritmos Evolutivos (AEs) destacam-se nas aplicações. De acordo com Poli, Langdon e McPhee (2008), Programação Genética (PG) é uma sistemática que, a partir de uma instrução de alto nível, permite que o método resolva

problemas de forma automática. Empregando ideias da evolução natural, a PG começa a refiná-los progressivamente, a partir da instrução inicial e de programas aleatórios. Isso ocorre através de processos de mutação e recombinação, até atingir uma avaliação aceitável.

A Figura 9 apresenta o fluxo de controle básico para programação genética, no qual a sobrevivência do mais forte (dados mais relevantes) é usada para encontrar soluções.

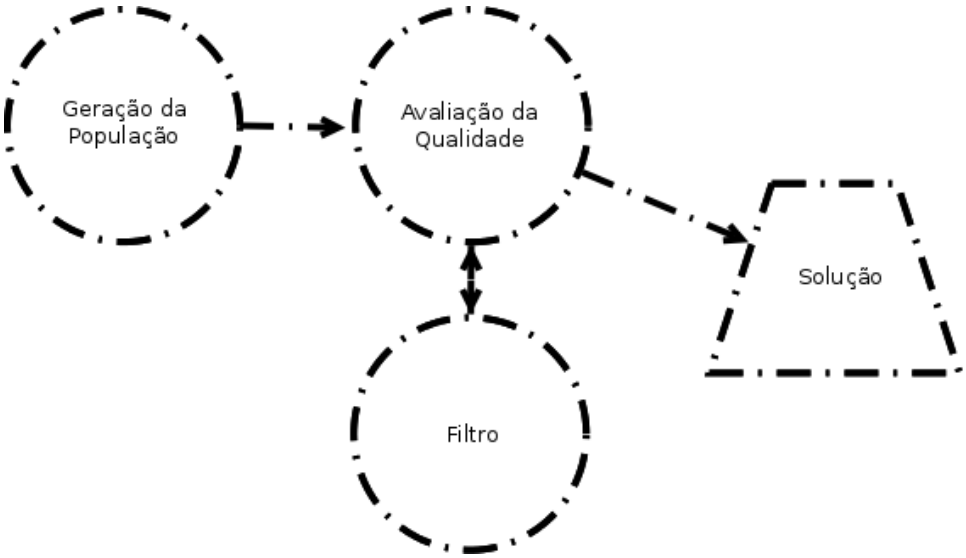

Figura 9 – Modelo de PG - Inicialmente é criada, de forma aleatória, uma população. Essa população podem representar funções que sejam específicas ao domínio de um problema, operações matemáticas, funções lógicas ou operações de programação. O critério usado para avaliação é uma função chamada função de 'aptidão' (*fitness*), que avalia se cada solução contribuirá para a próxima geração de soluções. Esta medida é que dirige diretamente o processo evolutivo, fazendo que, quanto mais adaptado for um indivíduo maior a sua chance de permanecer ou ter suas características propagadas para as próximas gerações.

Os AEs são baseados no uso de algoritmos de busca probabilísticos, inspirados na teoria da Evolução das Espécies de Darwin (DARWIN; BEER, 1998; SPEARS et al., 1993; EIBEN; SMITH, 2003; DAS; SUGANTHAN, 2011; CHATZIDIMITRIOU; MITKAS, 2013). Destaca-se que este paradigma tem auxiliado na aprendizagem de máquina (ZURADA et al., 2009; YEN, 2009; HASAN; JAFAR; RAMAKRISHNAN, 2011; ZHANG et al., 2011), havendo trabalhos, como o de Espejo, Ventura e Herrera (2010), que apresentam estudos utilizando aplicações de AGs para a classificação de dados, revelando a flexibilidade dos AGs neste tipo de aplicação. Além disso, também existem trabalhos cujo objetivo é avaliar as classificações obtidas, selecionando um subconjunto ótimo de dimensões a partir do original (PRASARTVIT et al., 2012; CHATZIDIMITRIOU; MITKAS, 2013).

Zhang et al. (2011) apresentam uma descrição do comportamento de um AE. Inicialmente há três operações fundamentais. O primeiro passo é a **Inicialização**, que a partir de uma configuração padrão ou de uma configuração aleatória gera a primeira população. A população, geralmente, é representada por uma classificação realizada diante dos conjuntos de dados analisados.

Conforme um exemplo de AE apresentado no Algoritmo 6, após a primeira população, o algoritmo entra no processo de gerações, realizando sua evolução em cada iteração através da função *Fitness*. Operadores genéticos são aplicados aos indivíduos de uma população para dar nascimento a uma nova população de indivíduos (a próxima geração). A função *Fitness* é empregada para medir a qualidade de um indivíduo. Quanto melhor o indivíduo, maior a probabilidade de ser selecionado para participar na criação de uma nova geração de indivíduos. Os principais operadores genéticos são *Crossover* (recombinação/cruzamento) e *Mutação*. *Crossover* corresponde ao operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações recebam essas características como herança biológica. O operador de *Mutação* é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida. Assim, meios são formados para a introdução de novos elementos na população. Desta forma, a *Mutação* assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção da busca.

Conforme Espejo, Ventura e Herrera (2010), o principal problema dos Algoritmos Genéticos é o seu custo computacional, principalmente devido a muitos acessos à função *Fitness*, o que tem que ser repetidamente avaliado através do processo evolutivo. Isto sinaliza a importância da aplicação desta técnica em uma rede de computadores.

Algoritmo 6 AE

1: $t = 0$
2: Criar População Inicial $P(0)$
3: Para cada Indivíduo i da População Atual $P(t)$ faça
4: Avaliar *fitness* do Indivíduo i
5: **repeat**
6: Incrementar t
7: Selecionar a População $P(0)$ a partir de $P(t-1)$
8: Aplicar operadores de cruzamento em $P(t)$
9: Aplicar operadores de mutação em $P(t)$
10: Avaliar $P(t)$
11: **until** Critério de Parada OK

Em Modelos Psicológicos, os psicólogos e estudiosos dessa área avaliaram o desempenho de humanos em tarefas de aprendizagem diferentes. Trabalhos relacionados levaram a uma série de árvores de decisão. Alguns trabalhos envolvem estímulos como recompensa, em consequência à aprendizagem (RIBEIRO; GABRIELLI, 2003).

Conforme Luger (2004), a aprendizagem permanece sendo uma área desafiadora para a IA. A maioria dos sistemas especialistas é prejudicada pela inflexibilidade das suas estra-

tégias para resolução de problemas e pela dificuldade em modificar grandes quantidades de código. A solução obvia para estes problemas é permitir que estes sistemas aprendam por conta própria.

Em Espejo, Ventura e Herrera (2010) foi realizado um estudo mostrando a relevância da utilização de Algoritmos Genéticos em processos de classificação. As características do PG tornaram esta técnica muito conveniente em relação à classificação, sendo a flexibilidade a principal característica dela, o que permite que a técnica se adapte às necessidades de cada problema, mostrando sua importância mediante as peculiaridades de cada base de dados.

2.4 Programação Paralela

Conforme mencionado na seção 1.1, a maior parte dos problemas de tomada de decisão por especialistas, tais como administradores de empresa, pesquisadores acadêmicos, entre outros, está relacionada à conversão de dados em conhecimento concreto e efetivo, principalmente a partir de bases de dados de sistemas complexos, como é o caso dos sistemas biológicos. Estas bases, em sua maioria, possuem grande número de atributos, ocasionando um alto custo computacional durante as respectivas análises. Por conseguinte, cada vez mais recursos computacionais são necessários para definir caminhos de melhor tratamento destes dados. Conforme Pavlo et al. (2009), os estudos utilizando redes de computadores têm crescido gradualmente no intuito de avaliar grandes bases de dados.

A demanda por desempenho computacional em aprendizado de máquina tem aumentado devido ao crescimento rápido da quantidade e das dimensões dos conjuntos de dados. Conforme estudos apresentados por Sinky et al. (2014), algoritmos em aprendizado de máquina como *clustering*, classificação e aprendizagem múltiplas, exigem a execução de tarefas paralelas para um processamento rápido.

As principais razões para utilizar a programação paralela são:

- ❑ Redução do tempo necessário para solucionar um problema;
- ❑ Resolução de problemas complexos de maior dimensão;
- ❑ Memória disponível;
- ❑ Processamento de atividades independentes.

Desta forma, a característica paralela de um sistema de classificação pode minimizar este custo computacional, oferecendo um maior número de dados analisados. Entretanto, o custo na comunicação em um algoritmo paralelo pode se apresentar como um problema no desempenho do sistema (TSENG; CHEN, 2005). Isso mostra que a programação paralela é uma tarefa complexa, pois há a necessidade de verificar questões ausentes em um programa

sequencial, como a questão de concorrência, comunicação e sincronização, balanceamento de carga e escalonamento.

Para realizar a comunicação entre os processos, é possível utilizar bibliotecas como *Parallel Virtual Machine* - PVM e *Message Passing Interface* - MPI. O trabalho de Geist, Kohl e Papadopoulos (1996) apresenta uma comparação entre estas duas bibliotecas.

Segundo Echevarria (s.d.), o MPI é o resultado do trabalho de aproximadamente 60 pessoas, pertencentes a 40 instituições, principalmente dos Estados Unidos e Europa. A maioria dos fabricantes de computadores paralelos participou, de alguma forma, da elaboração do MPI, juntamente com pesquisadores de universidades, laboratórios e autoridades governamentais.

O padrão MPI apresenta algumas características que favorecem a exploração do paralelismo, são eles:

- Eficiência: projetado para executar em máquinas diferentes, o padrão MPI define a mesma *Application Programming Interface* (API) para os usuários. Contudo, cada fabricante é livre para implementar as rotinas, utilizando características exclusivas de sua arquitetura;

- Facilidade: baseado em práticas comuns de paralelismo e de outras bibliotecas de passagem de mensagem, acrescenta também algumas extensões, permitindo maior flexibilidade;

- Portabilidade: compatível com sistemas de memória distribuída, memória compartilhada e outras arquiteturas;

- Transparência: permite que um programa seja executado em sistemas heterogêneos sem mudanças significativas;

- Escalabilidade: suporta comunicação coletiva com a criação de subgrupos de processos, favorecendo a troca de mensagens.

2.5 Considerações Finais

O número de informações em diversas áreas cresce a cada dia, proporcionando um aumento do volume de dados nas bases avaliadas. Conforme Batista (2003), o método clássico de análise consiste em um acompanhamento de especialistas atuando como uma interface entre usuários e bases. Esta ação torna-se muitas vezes cara, lenta e subjetiva, acrescentando que o próprio crescimento dos dados inviabiliza a ação do especialista, principalmente na extração de conhecimento das informações avaliadas. Um dos exemplos destas bases de dados é o repositório Frank e Asuncion (2010), utilizado neste trabalho, que atualmente mantém 246 conjuntos de dados como um serviço para a comunidade

de aprendizado de máquina. Entretanto, os repositórios disponíveis apresentam dados previamente analisados, diferenciando um pouco de outras bases que possam ser utilizadas para análise.

A área de Descoberta de Conhecimento em Bases de Dados (KDD), como em Máquina de Aprendizado, visa extrair conhecimento em base de dados, principalmente em bases com grandes inconsistências e com classes desbalanceadas, onde o particionamento dos dados não é uniforme. Apesar da existência de um elevado número de algoritmos que tratam desta extração de conhecimento, a partir desses conjuntos de dados, a falta de qualidade presente nos dados, isto é, a dificuldade de extrair conhecimento a partir do quantitativo de informações, afeta diretamente o desempenho dos métodos aplicados.

Segundo Fayyad, Piatesky-Shapiro e Smyth (1996), são várias as áreas que têm promovido um crescimento exponencial em suas bases de dados, gerando a necessidade de novas ferramentas e técnicas para análise automática e inteligente de bancos de dados. Fayyad, Piatesky-Shapiro e Smyth (1996) descrevem que a exploração de conhecimento em bases de dados, é um processo não trivial de identificação em dados padrões que sejam válidos, novos, potencialmente úteis e compreensíveis, visando melhorar o entendimento de um problema ou um procedimento de tomada de decisão. Existe uma necessidade urgente de uma nova geração de abordagens ou metodologias computacionais para contribuir com os especialistas na extração de informações úteis considerando o aumento do volume de dados.

Fayyad, Piatetsky-Shapiro e Smyth (1996) completam sua análise informando que bases de dados que contêm informações na ordem de $r = 10^9$ objetos são cada vez mais comuns nas áreas de astronomia, e de saúde, na qual o número de campos pode ser facilmente da ordem de $c = 10^2$ ou mesmo $c = 10^3$. Portanto, o trabalho de análise deve ser automatizado, pelo menos parcialmente. Máquinas que identificam gradualmente estas informações podem ser preparadas para explorar mais efetivamente os resultados obtidos nas respectivas análises.

Para materializar a implementação do sistema de aprendizado proposto neste trabalho, com o intuito de avaliar um número considerável de bases de dados, agregando um maior número de métricas e metodologias de classificação e permitir a obtenção de resultados satisfatórios em um tempo acessível, a estrutura do sistema foi planejada para uso em redes de computadores. O sistema utiliza técnicas de Inteligência Artificial e aproveita os recursos presentes em um sistema distribuído na análise dos dados. Alguns métodos de agrupamento e técnicas de refinamento foram implementados para demonstrar o funcionamento do sistema. Por fim, para implementação do sistema, que será descrito nos capítulos seguintes, foi utilizado a biblioteca MPI para tratar a comunicação do sistema.

Capítulo 3

Materiais e Métodos

Este capítulo apresenta a metodologia utilizada para implementação do trabalho, bem como os recursos disponibilizados no sistema proposto. Como metodologia, a seção 3.1 apresenta o sistema de aprendizado proposto, com a composição detalhada de todos os seus módulos, bem como a modelagem dos dados, apresentando a forma como as informações que trafegam no sistema estão relacionadas. Os recursos são descritos na seção 3.2 e 3.3 que apresentam brevemente os Bancos de Dados utilizados nos experimentos e o Ambiente Computacional o qual é instalado o sistema para uso. Finalmente, a seção 3.4 encerra o capítulo apresentando as considerações finais.

3.1 Sistema Proposto

O Sistema de aprendizado proposto neste trabalho foi projetado utilizando conceitos de algoritmos evolutivos e apresenta quatro módulos: **Módulo de Inicialização**, **Módulo de Validação**, **Módulo Especial de Escalonamento**, e o **Módulo de Refinamento**. Em cada módulo, podem ser utilizados diversos algoritmos para atender ao seu objetivo principal. A única exigência é que os dados estejam padronizados, favorecendo a integração dos módulos e possibilitando o processamento em paralelo.

A Figura 10 apresenta a estrutura deste sistema de aprendizado, permitindo observar como estes módulos interagem entre si. As atividades de cada módulo serão detalhadas nas subseções seguintes.

3.1.1 Módulo de Inicialização

O objetivo do **Módulo de Inicialização** é preparar os dados originais que serão classificados. Este módulo, além de criar uma partição inicial, estrutura esses dados em um padrão compreendido pelos módulos de Validação e Refinamento.

O primeiro passo do **Módulo de Inicialização** é a obtenção das matrizes de similaridade a partir de algum critério de comparação, como por exemplo, o cálculo da distância

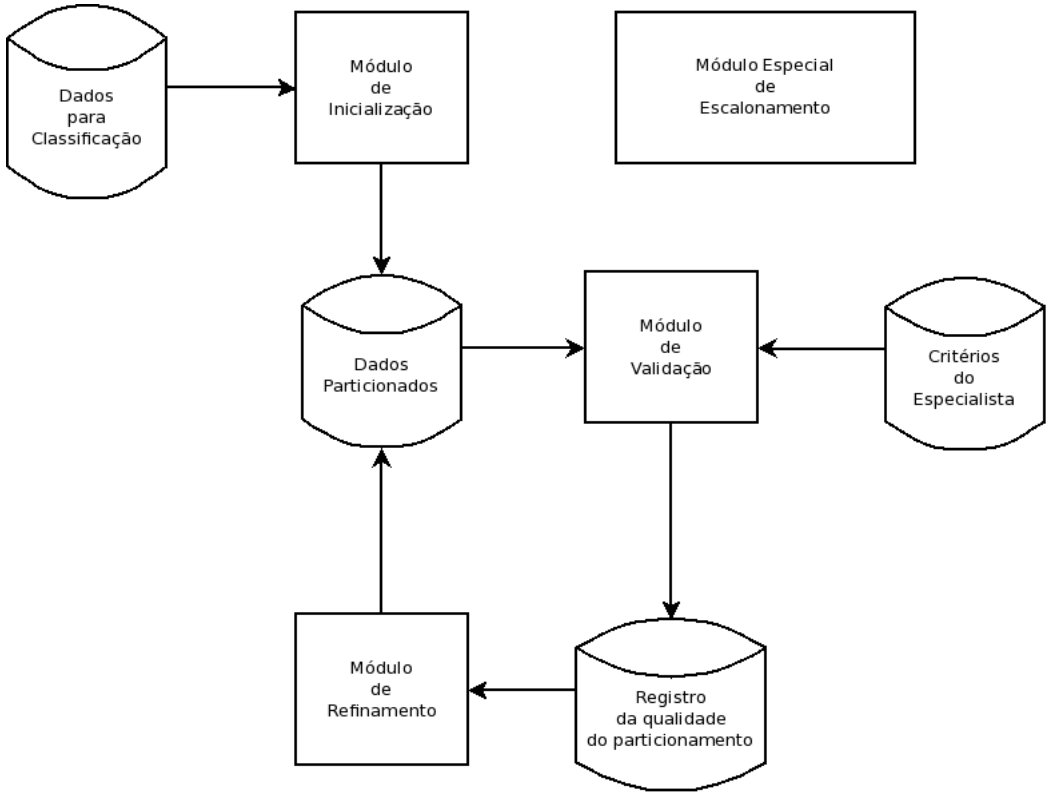

Figura 10 – A figura mostra a estrutura do sistema e a forma pela qual os módulos estão relacionados. O conjunto de dados é primeiramente tratado pelo **Módulo de Inicialização** gerando o primeiro conjunto de dados particionados. Os conjuntos de dados particionados são avaliados pelo **Módulo de Validação**, que utiliza diferentes critérios e regras de especialistas para registrar a qualidade da partição. Paralelamente a este processo o **Módulo de Refinamento** gera mais partições a serem avaliados, a partir dos dados particionados definidos pelo **Módulo de Inicialização**, e pelo próprio **Módulo de Refinamento**. O **Módulo Especial de Escalonamento** é responsável pela administração das atividades

em um espaço métrico. De fato, esse cálculo desempenha um papel importante na classificação dos dados. A partir das informações padronizadas da base, contendo os atributos de cada objeto, gera-se uma matriz que será usada para a construção da matriz de similaridade. O algoritmo 7 apresenta um código escrito na linguagem C++ que constrói a matriz de similaridade (`MatSim`) a partir da matriz de atributos (`MatAtrib`), conforme cálculo da distância Euclidiana, apresentado na seção 2.2.2. A matriz de similaridade é uma matriz quadrada da ordem do número de objetos a serem classificados (`NrObjetos`). A matriz de atributos contém, em cada linha, as informações pertinentes a um objeto, portanto, é uma matriz de ordem $NrObjetos \times NrAtributos$.

Algoritmo 7 Construção da Matriz de Similaridade

```
1  void MatrizSimilaridade(double **MatSim, int NrObjetos,
2                          double **MatAtrib, int NrAtributos){
3    double Result;
4    for (int i=0; i<NrObjetos; i++) {
5      for (int j=0; j<NrObjetos; j++) {
6        Result = 0.0;
7        for (int k=0; k<NrAtributos; k++)
8          Result += pow(MatAtrib[i][k]-MatAtrib[j][k],2);
9        MatSim[i][j] = pow(Result,0.5);
10     }
11   }
12 }
```

A partir das matrizes de similaridade, vários métodos podem ser utilizados para se obter uma partição inicial, conforme descrições apresentadas no capítulo anterior. Neste capítulo, três métodos serão detalhados para ilustrar como eles serão tratados no sistema. O mais importante é o `Método da Árvore Binária`, descrito na subseção 3.1.1.1, que, devido à análise que é realizada nos dados durante a sua execução, cria uma partição com grande potencial para ser o resultado final ou próximo dele. Os outros dois métodos são algoritmos baseados em mecanismos de escolha aleatória que, apesar de simples, podem alcançar resultados relevantes devido ao processamento paralelo e à integração entre os módulos de validação e de refinamento.

Ressalta-se que toda classificação produzida no **Módulo de Inicialização** ou no **Módulo de Refinamento** recebe um valor que representa a qualidade da partição criada. A este valor foi dado o nome de **Atributo de Qualidade**. A forma como este atributo é calculado pode variar de acordo com o tipo de dados ou de acordo com os métodos utilizados. A seção 3.1.2 discute o cálculo do **Atributo de Qualidade** utilizado nos experimentos realizados.

3.1.1.1 Método da Árvore Binária

O método descrito nesta seção utiliza-se da estrutura de dados `Árvore Binária` para construção das partições. Cada partição será representada por uma árvore que armazena os objetos semelhantes.

A partir da matriz de similaridade, são identificados os dois objetos mais próximos, ou seja, aqueles com maior similaridade entre si. Estes objetos serão unidos em uma mesma árvore, tornando-se as subárvores esquerda e direita dessa nova árvore. Desta forma, o número de partições diminui em uma unidade. A matriz de similaridade é refeita, excluindo as linhas e colunas referentes aos objetos escolhidos e inserindo-se uma nova

linha, cujo ponto de referência no espaço é a média entre as distâncias dos dois objetos escolhidos para as demais partições.

É importante destacar que a definição da nova coordenada de localização da partição pode ser realizada de diversas maneiras, como, por exemplo, a utilização da média da distância entre todos os pares de objetos, método conhecido como *Average Link* (HANDL; KNOWLES; DORIGO, 2003), conforme ilustra a Figura 11, ou utilizando a coordenada do objeto, o qual é pertencente à partição que estiver mais próxima em relação às demais partições, caracterizando o método conhecido como *Single Link* (HUBERT, 1974). A Figura 12 ilustra este método. Estas variações foram implementadas e, no sistema proposto, executam em paralelo, fornecendo, ao **Módulo de Validação**, partições distintas para análise.

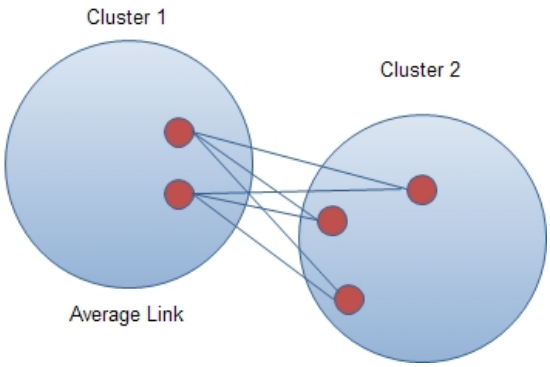

Figura 11 – Método *Average Link*: Segundo Nassifa e Hruschkab (2011), a distância entre dois grupos é a distância média entre pares de objetos pertencentes a grupos diferentes

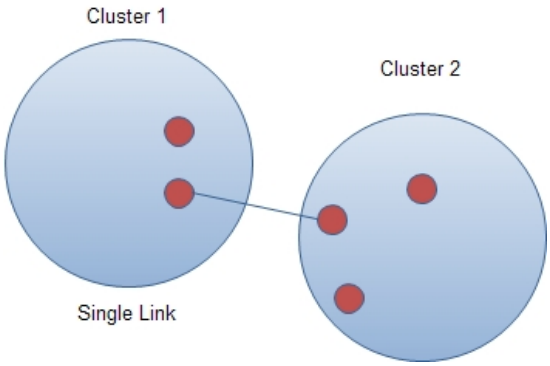

Figura 12 – Método *Single Link*: Conforme Nassifa e Hruschkab (2011), a distância entre dois grupos é a distância mínima entre dois objetos pertencentes a grupos diferentes. Esse algoritmo induz grupos contíguos, baseados nas adjacências entre os objetos, podendo assumir formas variadas

O procedimento de escolha das partições mais próximas é repetido até que se atinja o número de partições desejado. Para se ter uma visão mais clara, este método é descrito através do código em C++ apresentado nos Algoritmos 9 e 10, contendo as etapas necessárias para a geração de um particionamento.

O Algoritmo 8 apresenta a estrutura para cada nó da árvore no sistema, registrando a identidade de cada dado, o número de dados em cada classe, quem é o *medóide* da classe, isto é, os objetos de representação de um conjunto de dados cuja dissimilaridade média de todos os objetos no cluster é mínima e o raio de cobertura da classe. Os *medóides* são semelhantes em conceito aos *centróides*, mas *medóides* são sempre os membros do conjunto de dados. A informação referente ao *medóide* é mantida na estrutura da árvore para permitir consultas por abrangência e por vizinhos mais próximos, conforme discussão que será apresentada na seção 3.1.3. O atributo `ListIdObj` mantém a informação dos identificadores de todos os nós folhas que são filhos de uma determinada árvore. Esta informação também permite otimizar o processo de busca e caminhamento na árvore.

Algoritmo 8 Estrutura da Árvore Binária

```
1  struct NoArvDendo {
2    int *ListIdObj;
3    int Tam;
4    int KMedoid;
5    double Raio;
6    NoArvDendo *Esq;
7    NoArvDendo *Dir;
8  };
9
10 NoArvDendo **ArvPart;
```

Inicialmente, o algoritmo constrói os nós folhas (Algoritmo 9), sendo um nó para cada dado da base. Ao final da execução do algoritmo, o vetor resultante (**ArvPart**) possui k ponteiros, sendo cada ponteiro o identificador de uma subárvore contendo as informações de uma determinada partição. Após a inicialização, a árvore é gerada em uma sequência *bottom-up* através das junções dos objetos mais próximos.

Logo após o processo de redução da matriz, identificando os elementos mais próximos, o método cria um novo nó para a estrutura. Em seguida, o processo atualiza o *K-Medoid* e verifica o raio de cobertura da classe, conforme algoritmo 10. Encerrado a criação da árvore, o sistema gera o arquivo nos padrões estabelecidos para ser avaliado pelo **Módulo de Validação**.

Algoritmo 9 Gerar Árvore Binária - primeira parte

```
1  void GeraArvoreBinaria(NoArvDendo **ArvPart, int K,
2                         double **MatSim, int NrObjetos)
3  {
4    /* Etapa de inicialização dos nós representando os elementos */
5    for (int i=0; i<NrObjetos; i++) {
6      ArvPart[i] = new NoArvDendo;
7      ArvPart[i]->ListIdObj = new int[2];
8      ArvPart[i]->Tam = 1;
9      ArvPart[i]->KMedoid = i;
10     ArvPart[i]->Raio = 0.0;
11     ArvPart[i]->ListIdObj[0] = i;
12     ArvPart[i]->Esq = ArvPart[i]->Dir = NULL;
13   }
14
15   int linMenor, colMenor, contMedia, kMedoid, OrdemAtual=
       NrObjetos;
16   double Menor, Media;
17   NoArvDendo *Aux; double distMaior, raioMenor;
18
19   for (int fimLista=NrO; fimLista>K; fimLista--) {
20     /* Encontrar menor distância dos nós ligados na lista */
21     Menor = INFINITO;
22     for (int i=0; i<fimLista-1; i++) {
23       for (int j=i+1; j<fimLista; j++) {
24         Media = 0.0; contMedia = 0;
25         for (int m=0; m<ArvPart[i]->Tam; m++) {
26           for (int n=0; n<ArvPart[j]->Tam; n++) {
27             Media +=
28               M[ArvPart[i]->ListIdObj[m]][ArvPart[j]->ListIdObj[n
                 ]];
29             contMedia++;
30           }
31         }
32         Media = Media / contMedia;
33         if (Media < Menor){
34           linMenor = i; colMenor = j; Menor = Media;
35         }
36       }
37     }
38 ...
```

Algoritmo 10 Criação de um novo Nó - Árvore Binária

```
...
    /* Criação de um novo nó */
    Aux = new NoArvDendo;
    Aux->Esq = ArvPart[linMenor]; Aux->Dir = ArvPart[colMenor];
    Aux->ListIdObj = new int[ ArvPart[colMenor]->Tam +
                              ArvPart[linMenor]->Tam + 1 ];
    Aux->Tam = 0;
    for (int w=0; w < ArvPart[colMenor]->Tam; w++)
      Aux->ListIdObj[w] = ArvPart[colMenor]->ListIdObj[w];
    Aux->Tam = ArvPart[colMenor]->Tam;
    for (int w=0; w<ArvPart[linMenor]->Tam; w++) {
      int z=Aux->Tam-1;
      while (z > -1 &&
             Aux->ListIdObj[z] > ArvPart[linMenor]->ListIdObj[w])
                 {
        Aux->ListIdObj[z+1]=Aux->ListIdObj[z]; z--;
      }
      Aux->ListIdObj[z+1]=ArvPart[linMenor]->ListIdObj[w];
      Aux->Tam++;
    }
    /* Calcular valor do K-Medoid e raio de cobertura */
    raioMenor = INFINITO;
    for (int i=0; i<Aux->Tam-1; i++) {
      distMaior = 0.0;
      for (int j=0; j<Aux->Tam; j++)
        if (M[Aux->ListIdObj[i]][Aux->ListIdObj[j]] > distMaior)
          distMaior = M[Aux->ListIdObj[i]][Aux->ListIdObj[j]];
      if (distMaior < raioMenor) {
        raioMenor = distMaior; kMedoid = Aux->ListIdObj[i];
      }
    }
    Aux->KMedoid = kMedoid; Aux->Raio = raioMenor;
    /* Compactar vetor de partições */
    ArvPart[linMenor] = Aux;
    for (int w=colMenor; w<fimLista-1; w++)
      ArvPart[w] = ArvPart[w+1];
    ArvPart[fimLista-1] = NULL;
  }
  // Gerar Arquivo de Partições
}
```

3.1.1.2 Método Randômico

O `Método Randômico` é bastante simples. A partir de uma semente inicial são escolhidos, aleatoriamente, os objetos que irão compor cada partição. Obviamente, este método não parte de nenhum conhecimento prévio dos dados e, por conseguinte, produz partições cujo atributo de qualidade é ruim. A maior vantagem deste método, além de sua simplicidade, é a velocidade com que as partições são criadas, a partir de novas sementes de números aleatórios. Neste trabalho, para a geração dos números aleatórios, foi utilizada uma função de distribuição de probabilidade uniforme, entretanto, outras distribuições podem ser usadas desde que se realize um estudo do impacto dessas distribuições no atributo de qualidade das partições obtidas. A estrutura de dados apresentada na Algoritmo 11 é utilizada para as variáveis que compõem as informações dos dados analisados.

Algoritmo 11 Estrutura de dados e variáveis utilizadas no Método Randômico

```
1 struct NoListRand {
2   int *ListIdObj;
3   int Tam;
4 };
5 NoListRand *Parts;
6
7 int NroMelhorias;
8 double MenorAtribQuali, AtribQuali;
```

O código do Algoritmo 12 apresenta a função utilizada para gerar as partições. Esta técnica incialmente divide os objetos em um número igual de dados para cada classe. Dependendo do número total de objetos na base, a última partição terá uma quantidade inferior de elementos.

Após a criação dessas classes, o **Módulo de Validação** analisará a classificação gerada. Esta sequência de atividades será realizada inúmeras vezes até o `Critério de Parada` seja alcançado.

Algoritmo 12 Método Randômico

```
1  void MetodoRandomico(NoListRand *Parts, int K, int NrObjetos){
2    bool *Objetos = new bool[NrObjetos];
3    int Sort, NrOAux = NrObjetos;
4    for (int i=0; i<NrObjetos; i++) Objetos[i] = false;
5
6    int QuantObjsPorLista = NrObjetos / K; int FimLista=-1, Pos;
7
8    for (int i=0; i<K-1; i++) {
9      Parts[i].ListIdObj = new int[QuantObjsPorLista];
10     Parts[i].Tam = QuantObjsPorLista;
11     NrOAux -= QuantObjsPorLista;
12
13     for (int j=0; j<QuantObjsPorLista; j++) {
14       Sort = rand() % NrObjetos;
15       while (Objetos[Sort]) Sort = (Sort + 1) % NrObjetos;
16       Pos = FimLista;
17       while (Pos > -1) {
18         if (Parts[i].ListIdObj[Pos]<Sort) break;
19         Parts[i].ListIdObj[Pos+1] = Parts[i].ListIdObj[Pos];
20         Pos--;
21       }
22       Parts[i].ListIdObj[Pos+1] = Sort;
23       FimLista++;
24       Objetos[Sort] = true;
25     }
26     FimLista=-1;
27   }
28   Parts[K-1].ListIdObj = new int[NrOAux];
29   Parts[K-1].Tam = NrOAux;
30
31   for (int j=0; j<NrOAux; j++) {
32     Sort = rand() % NrObjetos;
33     while (Objetos[Sort]) Sort = (Sort + 1) % NrObjetos;
34     Pos = FimLista;
35     while (Pos > -1) {
36       if (Parts[K-1].ListIdObj[Pos]<Sort) break;
37       Parts[K-1].ListIdObj[Pos+1] = Parts[K-1].ListIdObj[Pos];
38       Pos--;
39     }
40     Parts[K-1].ListIdObj[Pos+1] = Sort;
41     FimLista++; Objetos[Sort] = true;
42   }
43 }
```

3.1.1.3 Método Semi-randômico

O `Método Semi-randômico` é uma variação do mecanismo anterior. Apesar de utilizar escolhas aleatórias, os objetos não são escolhidos dentro de todo o universo dos dados originais. Uma pré-classificação é realizada para que as escolhas dos objetos sejam feitas em subconjuntos com maior índice de semelhança entre seus componentes.

Os conjuntos iniciais são construídos a partir da matriz de similaridades. Inicialmente, são escolhidos k pares de objetos aleatoriamente. Os pares representam os primeiros objetos da cada partição. A distância entre os dois objetos do par é o valor inicial de referência para a incorporação dos demais objetos. O primeiro objeto escolhido para o par será considerado o *medóide* do conjunto, para as referências de distância. A partir destas distâncias, o algoritmo de montagem das partições escolhe, aleatoriamente, os objetos ainda não selecionados e realiza uma comparação entre a distância destes objetos em relação aos *medóides* de cada partição, definindo a partição com a menor distância. Caso a distância entre o *medóide* da partição escolhida e o objeto seja maior do que a definida anteriormente, este novo valor passará a ser a referência do conjunto para as próximas comparações.

O número de elementos em cada partição pode ser informado pelos usuários do sistema ou, caso contrário, será definido de forma homogênea entre as partições.

Da mesma forma como utilizado no `Método Randômico`, a distribuição uniforme foi a função de probabilidade utilizada na geração dos números aleatórios, porém, outras distribuições podem ser definidas.

O código disposto nos Algoritmos 13 e 14 apresenta a função que particiona os dados utilizando este método. A primeira parte tem o objetivo de escolher os dois objetos que iniciarão as respectivas partições. A segunda parte insere os demais objetos, considerando sempre a menor distância entre o objeto e as partições em construção.

Algoritmo 13 Método Semi-randômico - primeira parte

```
1  void MetodoSemiRandomico(NoListRand *Parts, int K,
2                           int NrObjetos, double **MatSim)
3  {
4    bool *Objetos = new bool[NrObjetos];
5    int Sort, NrOAux = NrObjetos;
6    int *ObjRef = new int[K]; double *DistObj = new double[K];
7    int *Fim = new int[K];
8    int QuantObjsPorLista = NrObjetos / K;
9
10   for (int i=0; i<NrObjetos; i++) Objetos[i] = false;
11   for (int i=0; i<K-1; i++) {
12     Parts[i].ListIdObj = new int[QuantObjsPorLista];
13     Parts[i].Tam = QuantObjsPorLista;
14     NrOAux -= QuantObjsPorLista;
15   }
16   Parts[K-1].ListIdObj = new int[NrOAux];
17   Parts[K-1].Tam = NrOAux;
18
19   for (int i=0; i<K; i++) {
20     Sort = rand() % NrObjetos;
21     while (Objetos[Sort])
22       Sort = (Sort + 1) % NrObjetos;
23     ObjRef[i] = Sort;
24     Objetos[Sort] = true;
25   }
26   for (int i=0; i<K; i++) {
27     Sort = rand() % NrObjetos;
28     while (Objetos[Sort])
29       Sort = (Sort + 1) % NrObjetos;
30     DistObj[i] = MatSim[ObjRef[i]][Sort];
31     if (ObjRef[i] < Sort) {
32       Parts[i].ListIdObj[0] = ObjRef[i];
33       Parts[i].ListIdObj[1] = Sort;
34     } else {
35       Parts[i].ListIdObj[0] = Sort;
36       Parts[i].ListIdObj[1] = ObjRef[i];
37     }
38     Fim[i]=1;
39   }
40 ...
```

Algoritmo 14 Método Semi-randômico - segunda parte

```
...
  NrOAux = NrObjetos - (2 * K);
  int PartAtual = 0; int Pos;

  while (NrOAux > 0) {
    Sort = rand() % NrObjetos;
    while (Objetos[Sort]) Sort = (Sort + 1) % NrObjetos;

    Objetos[Sort] = true;
    PartAtual = 0;
    for (int i=1; i<K; i++)
      if (M[ObjRef[i]][Sort] < MatSim[ObjRef[PartAtual]][Sort])
        PartAtual = i;
    while (Fim[PartAtual] == Parts[PartAtual].Tam - 1)
      PartAtual = (PartAtual + 1) % K
    if (M[ObjRef[PartAtual]][Sort] > DistObj[PartAtual])
      DistObj[PartAtual] = MatSim[ObjRef[PartAtual]][Sort];
    Pos = Fim[PartAtual];
    while (Pos > -1) {
      if (Parts[PartAtual].ListIdObj[Pos]<Sort) break;
      Parts[PartAtual].ListIdObj[Pos+1] = Parts[PartAtual].
         ListIdObj[Pos];
      Pos--;
    }
    Parts[PartAtual].ListIdObj[Pos+1] = Sort;
    Fim[PartAtual]++; NrOAux--;
  }
}
```

3.1.2 Módulo de Validação

Uma vez que uma classificação tenha sido criada, pelo **Módulo de Inicialização**, ou melhorada, pelo **Módulo de Refinamento**, é necessário verificar a qualidade da partição, ou seja, avaliar se a classificação foi bem sucedida. Geralmente, esta tarefa necessita de um profissional especialista que irá utilizar os resultados do particionamento. Entretanto, a partir de informações destes profissionais, pode-se criar uma base de regras (`Critérios do Especialista`) que automatize este processo ou, pelo menos, elimine as partições que estejam muito aquém de um padrão mínimo de qualidade. Assim, o especialista avaliará somente aquelas partições que tenham atingido um adequado valor

no **Atributo de Qualidade**.

Para os testes realizados com os dados deste trabalho, o **Atributo de Qualidade** empregado foi a soma das médias das distâncias métricas entre todos os pares de objetos de uma mesma partição, código apresentado pelo Algoritmo 15, descrito na linguagem C++. Vale destacar que outros parâmetros estatísticos podem ser utilizados segundo as regras dispostas na base de '`Critérios do Especialista`'.

Algoritmo 15 Atributo de Qualidade

```
1  double AtributoQualidade (NoListRand *Parts, int NroParts,
2                            double **MatSim)
3  {
4    double SomaMedias = 0.0;
5    double Media;
6    int TotElem;
7
8    for (int k=0; k<NroParts; k++) {
9      Media = 0.0;  TotElem = 0;
10     for (int i=0; i<Parts[k].Tam-1; i++) {
11       for (int j=1; j<Parts[k].Tam; j++) {
12         Media += M[(Parts[k].ListIdObj[i])][(Parts[k].ListIdObj[j
             ])];
13         TotElem++;
14       }
15     }
16     SomaMedias += Media / TotElem;
17   }
18   return SomaMedias;
19 }
```

Após calcular o **Atributo de Qualidade** de um particionamento, o **Módulo de Validação** registra esta informação na base `Registro de Qualidade do Particionamento`. A critério do especialista, podem ser definidos valores mínimos de qualidade para que um particionamento seja tratado pelo **Módulo de Refinamento**. Entretanto, toda validação de um particionamento proveniente do **Módulo de Inicialização** deve, necessariamente, passar pelo **Módulo de Refinamento**.

3.1.3 Módulo de Refinamento

Refinar é melhorar a qualidade de um particionamento realizado previamente. Isto é feito trocando-se objetos entre as partições. Esta troca altera o valor do **Atributo de Qualidade**. Portanto, o **Módulo de Refinamento** tem por objetivo realizar a troca de

objetos entre as partições, com base em algum mecanismo previamente definido. Assim que a troca é realizada e uma nova classificação obtida, o **Módulo de Validação** avalia o impacto causado por essa troca. Quando o processo de troca é bem sucedido, ou seja, quando há melhora no valor do **Atributo de Qualidade**, a classificação obtida fica disponível como resultado alcançado e como ponto de entrada para um novo refinamento. Por sua vez, se a partir do procedimento de troca o valor do **Atributo de Qualidade** sinaliza uma piora da classificação original, essa classificação é descartada.

A integração entre os **Módulo de Validação** e o **Módulo de Refinamento**, está relacionada ao processo iterativo entre os módulos, de acordo com a Figura 10, permitindo possíveis aplicações de métodos evolutivos com o objetivo de aperfeiçoar o processo de refinamento dos dados particionados. A Figura 13 ilustra os passos gerais de um método evolutivo.

Os conceitos relacionados aos Algoritmos Evolutivos utilizados neste trabalho são: População de indivíduos, Geração de Novos Indivíduos e Função *Fitness*.

Estes conceitos foram utilizados da seguinte forma:

- ❑ População de indivíduos: a população é representada por um particionamento e o seu respectivo **Atributo de Qualidade**;

- ❑ Geração: operadores genéticos são aplicados aos indivíduos de uma população para dar nascimento a uma nova população de indivíduos (a próxima geração). Os principais operadores genéticos são *Crossover* (recombinação/cruzamento) e *Mutação*, conforme abordado no capítulo anterior. *Crossover* corresponde ao operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. O operador de *Mutação* é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, fornecendo assim, meios para a introdução de novos elementos na população. Desta forma, a *Mutação* assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois, com este mecanismo, altera-se levemente a direção da busca. A operação de *Crossover* é implementada através de técnicas específicas desenvolvidas no **Módulo de Refinamento**. A operação de *Mutação* não é utilizada no trabalho, uma vez que as alterações ocorrem sempre no mesmo indivíduo.

- ❑ Função *Fitness*: é utilizada para medir a qualidade de um indivíduo. Quanto melhor o indivíduo, maior a probabilidade de ser selecionado para participar na criação de uma nova geração. Este conceito foi tratado no **Módulo de Validação** do sistema, discutido na seção 3.1.2.

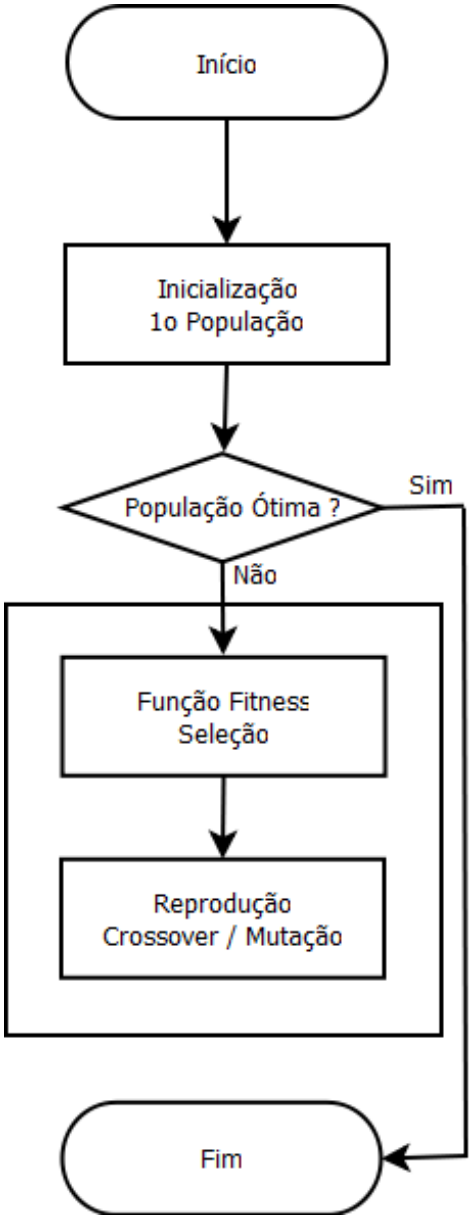

Figura 13 – Fluxograma Geral de um Algoritmo Evolutivo adaptado de Zhang et al. (2011)

Além dos métodos descritos no capítulo 2, esta seção apresenta três métodos que foram utilizados na implementação do **Módulo de Refinamento**: uma variação do algoritmo de Kernighan e Lin (1970) para particionamentos de grafos; o `Método de Consulta por Abrangência`; e o `Método do Raio de Cobertura Mínimo`, os quais são descritos a seguir. Os métodos utilizados para o refinamento partem sempre de uma classificação previamente obtida. Os mecanismos implementados utilizam um ponto de referência em cada partição. Para a identificação desses pontos, foram utilizados os métodos: *K-Means* e *K-Medoid*. Para utilizar o método *K-Means*, o sistema, através da Matriz de Similaridade, percorre todos os dados de cada classe, e a partir da localização de todos os pontos, gera a média de distância, identificando o ponto central do *cluster*. Para identificar o elemento como *medóide*, analisa-se, a partir da matriz de similaridade, qual é o objeto do

cluster que tem a menor distância entre todos os objetos do grupo. O objeto encontrado é definido como o *medóide* do *cluster*.

Uma vez definidos os pontos de referência, os mecanismos utilizados para o refinamento podem ser empregados. Para ilustrar como estes algoritmos foram tratados no sistema, as seções seguintes apresentam uma discussão sobre a implementação destes três métodos.

3.1.3.1 Método de Consulta por Abrangência

Através das partições definidas no **Módulo de Inicialização**, ou de resultados de particionamentos anteriores, este método calcula raios de cobertura a partir do *centróide* ou do *medóide* de cada partição, dependendo do método utilizado, isto é, o raio é a distância entre este ponto central e o objeto mais distante que pertença a partição. Neste sentido, pode-se identificar esferas envolvendo os objetos de cada partição, porém, é possível que objetos de partições diferentes sejam alcançadas pelos raios de cobertura de outras partições. Estes objetos estão em áreas de interseção entre duas ou mais partições. A partir da identificação das partições candidatas, ou seja, das partições cujos respectivos raios de cobertura alcançam o objeto, é escolhida a partição cuja distância entre o objeto e o respectivo *centróide* é a menor. O objeto é então deslocado da sua partição original para a partição escolhida.

Uma variação do método é a possibilidade de aumentar o raio de cobertura para identificar outros objetos no conjunto de interseção. Foram implementadas variações neste aumento. A taxa de aumento é uma relação direta com o valor do **Atributo de Qualidade**, ou seja, dependente dos dados do problema.

Para ilustrar o funcionamento deste método, a matriz apresentada na Tabela 1 informa a distância dos seguintes objetos A, B, C, D, E, F, G e K entre eles.

Tabela 1 – Matriz de distância dos objetos A, B, C, D, E, F, G e K

-	A	B	C	D	E	F	G	K
A	0	1,185	1,1866	0,626	0,55	1,516	1,492	2,194
B	1,185	0	0,971	1,271	1,686	2,701	2,122	1,923
C	1,1866	0,971	0	1,603	2,416	3,294	3,062	1,231
D	0,626	1,271	1,603	0	1,051	1,715	2,115	1,63
E	0,55	1,686	2,416	1,051	0	1,068	1,181	2,679
F	1,516	2,701	3,294	1,715	1,068	0	1,834	3,224
G	1,492	2,122	3,062	2,115	1,181	1,834	0	3,657
K	2,194	1,923	1,231	1,63	2,679	3,224	3,657	0

Considerando uma partição contendo apenas os objetos A, B, D e G, a Figura 14 ilustra a disposição dos objetos citados na Tabela 1, onde foi definido um determinado *cluster* com 4 elementos A, B, D e E. Utilizando-se o raio de cobertura do *K-Medoid* deste *cluster*, percebe-se que não há objetos candidatos a troca de partição a partir dos raios originais, uma vez que os objetos desta partição são apenas A, B, D e E, entre-

tanto, ampliando-se em 60% o raio de cobertura, os elementos F e G apresentam-se como candidatos à migração de classes.

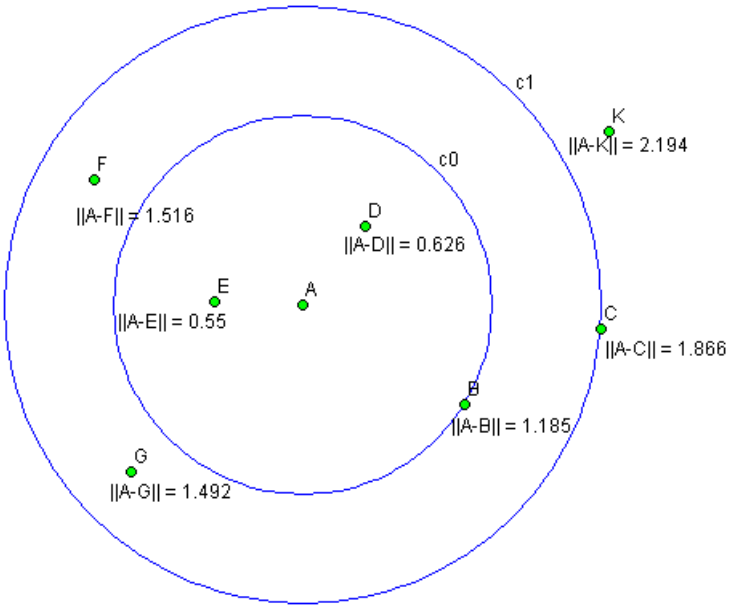

Figura 14 – Ilustração dos dados A, B, C, D, E, F, G e K onde os elementos A, B, D, E fazem parte de uma partição, o elemento A é o *centróide* deste *cluster*, e na distância do elemento A com o elemento C existem 3 elementos candidatos a troca de partição baseando-se em um aumento do raio de cobertura em 58%

O código em C++ do Algoritmo 16 apresenta uma função que realiza a consulta por abrangência nos dados já particionados. A partir do parâmetro `double` *raio*, o processo aumenta a área de circunferência da classe e verifica os elementos que estão ao seu redor. Esta consulta utiliza chamadas recursivas realizando a busca em uma estrutura de dados árvore, conforme descrição apresentada no Algoritmo 8. A partir de um dado (`objeto`) e um raio de referência, se a distância entre este objeto e o *centróide* da partição que está sendo analisada for menor que o raio de cobertura da respectiva partição com o acréscimo (`raio`) sugerido, o objeto poderá fazer parte desta partição por estar dentro do raio de cobertura da consulta realizada, sendo, portanto, um candidato à troca de partição. A sua informação será adicionada ao vetor de resultados (`Resultado`). Caso contrário, são realizadas buscas recursivas nas subárvores esquerda e direita por possíveis objetos que atendam aos critérios da consulta.

Algoritmo 16 Consulta por Abrangência

```
1  void ConsultaPorAbrangencia(NoArvDendo *A, double **MatSim,
2                              int objeto, double raio,
3                              int *Resultado, int& Tam) {
4    int i;
5    if (A != NULL) {
6      if (MatSim[A->KMedoid][objeto] <= A->Raio + raio) {
7        if (A->Tam == 1) {
8          i=Tam-1;
9          while (i > -1 && Resultado[i] > A->KMedoid) {
10           Resultado[i+1]=Resultado[i];
11           i--;
12         }
13         Resultado[i+1]=A->KMedoid;
14         Tam++;
15       } else {
16         ConsultaPorAbrangencia
17             (A->Esq, MatSim, objeto, raio, Resultado, Tam);
18
19         ConsultaPorAbrangencia
20             (A->Dir, MatSim, objeto, raio, Resultado, Tam);
21       }
22     }
23   }
24 }
```

3.1.3.2 Método do Raio de Cobertura Mínimo

O `Método do Raio de Cobertura Mínimo` realiza uma consulta em todos os dados, identificando o *medóide* mais próximo. Caso o *medóide* encontrado não seja o mesmo ao qual pertence o dado, o método transfere o dado para a partição deste *medóide*. Após o método verificar todos os dados, o sistema acessa o **Módulo de Validação** para obter um novo **Atributo de Qualidade**.

Uma característica interessante deste método é que, dependendo do conjunto de dados, há um processo de convergência das partições encontradas, visto que, conforme vão ocorrendo os refinamentos, os raios de cobertura tendem a diminuir, auxiliando o **Módulo Especial de Escalonamento** no critério de parada.

3.1.3.3 Método baseado em Kernighan e Lin

Como foi discutido na seção 2.2.1.3, o algoritmo de Kernighan e Lin (1970) funciona, basicamente, avaliando o ganho na redução do custo nos cortes entre partições durante a troca entre pares de vértices de partições diferentes.

O algoritmo desenvolvido neste trabalho seleciona uma partição aleatoriamente e escolhe um objeto que está mais distante do *medóide* desta partição. A partir deste ponto, o algoritmo, dentre as partições restantes, escolhe aquela cujo *medóide* está mais próximo do objeto selecionado. Além disso, o algoritmo seleciona o objeto desta segunda partição que está mais próximo do *medóide* da primeira. Após a escolha, o algoritmo calcula o ganho que será obtido no **Atributo de Qualidade** se essa troca for realizada. Este procedimento é realizado para todos os objetos da primeira partição em ordem decrescente de distância entre o objeto e o *medóide* da partição. Ao final, todas as trocas, cujos objetos selecionados irão produzir melhorias no **Atributo de Qualidade**, são realizadas.

Ao final do algoritmo, um novo particionamento é obtido e, portanto, está pronto para uma nova análise realizada pelo **Módulo de Validação**. A partição que serviu de referência (1ª partição) é marcada e uma nova partição é escolhida aleatoriamente repetindo-se a execução do método e, eventualmente, produzindo novos particionamentos até que todas as partições tenham sido base para o funcionamento do algoritmo. É importante destacar que só há geração de novas partições se houver alguma troca que produza uma melhoria no **Atributo de Qualidade**.

3.1.4 Módulo Especial de Escalonamento

Conforme apresentado anteriormente, a arquitetura deste sistema de aprendizagem possibilita a sua execução em um computador paralelo ou em uma arquitetura distribuída, como, por exemplo, um *cluster* de computadores.

O **Módulo Especial de Escalonamento** possui, nesse contexto, duas funções principais: escalonar as tarefas produzidas pelos demais módulos e identificar o momento de parada do sistema de aprendizagem. Estas funções serão discutidas nas subseções 3.1.4.1 e 3.1.4.2.

É importante esclarecer que para atender ao objetivo de diminuir o tempo de processamento dos métodos de classificação utilizados no sistema proposto, seria possível utilizar duas abordagens: a paralelização de cada método em si, abordagem conhecida como SRIP (*Single Replication in Parallel*) ou a abordagem adotada que realiza Múltiplas Replicações em Paralelo (MRIP - *Multiple Replication in Parallel*), ou seja, replicam-se várias execuções de um método sequencial. Apesar de ser possível executar implementações de métodos paralelos (SRIP) no sistema, a execução dos métodos tradicionais é mais simples e eficiente. Desta forma, a estrutura paralela da máquina funciona através do gerenciamento das replicações e execuções, em paralelo, de diversos métodos sequenciais

diferentes.

3.1.4.1 Escalonamento das Tarefas

Como foi explicado na seção 3.1.1, o **Módulo de Inicialização** possui vários métodos para obtenção de uma partição inicial. Além disso, alguns métodos podem obter particionamentos diferentes, dependendo das configurações iniciais, como é o caso dos valores das sementes de números aleatórios utilizados pelos métodos Randômico e Semi-randômico. Desta forma, várias execuções são necessárias para se encontrar um particionamento que ofereça um bom **Atributo de Qualidade**.

O **Módulo de Inicialização** mantém um processo em execução contínua (processo disparador). Basicamente, este processo é o responsável pela comunicação com o **Módulo Especial de Escalonamento**. Sua função é aguardar o sinal de disponibilidade (quando existe algum processador pronto para executar a tarefa) e criar a nova tarefa (a execução de um algoritmo para um método específico, que ainda não tenha sido utilizado, ou a execução de um método já utilizado com novos valores para seus atributos de configuração).

No **Módulo Especial de Escalonamento** há um `Processo Escalonador`. Este processo mantém um sistema de filas multiníveis de prioridades (SILBERCHATZ; GALVIN; GAGNE, 2001). As tarefas são agrupadas em três classes de prioridade: tarefas do **Módulo de Validação**; tarefas do **Módulo de Refinamento**; e tarefas do **Módulo de Inicialização**. A Figura 15 apresenta a estrutura das filas multiníveis. A fila referente aos processos do **Módulo de Validação** possui a mais alta prioridade, ou seja, nenhum processo das demais filas, que estiver aguardando, será escalonado se existir processo nesta fila. Isso é necessário, pois, uma vez que existem classificações já realizadas, a validação deve ser feita com maior prioridade em relação à obtenção de novos particionamentos. A fila referente ao **Módulo de Inicialização** tem a menor prioridade, ou seja, seus processos só poderão ser escolhidos se as demais filas estiverem vazias. É importante observar, no entanto, que no início do funcionamento da máquina de aprendizado, todas as filas estão vazias e, por consequência, as tarefas de geração de classificações iniciais serão executadas, pois não haverá classificações para serem validadas nem refinadas, estando as respectivas filas vazias.

O **Módulo de Validação** possui um único tipo de processo (`Avaliador da Qualidade`), que é responsável pela avaliação da qualidade de um particionamento obtido. Toda vez que um processo do **Módulo de Inicialização** ou do **Módulo de Refinamento** encerra sua execução, há um novo particionamento disponível para ter sua qualidade avaliada. Neste contexto, um novo processo `Avaliador da Qualidade` é criado para medir a classificação obtida. Este processo é, desta forma, inserido na respectiva fila de prioridade, aguardando pela disponibilidade de um processador para que o escalonador o selecione. Para melhorar a eficiência do sistema no ambiente paralelo, é possível, na configuração

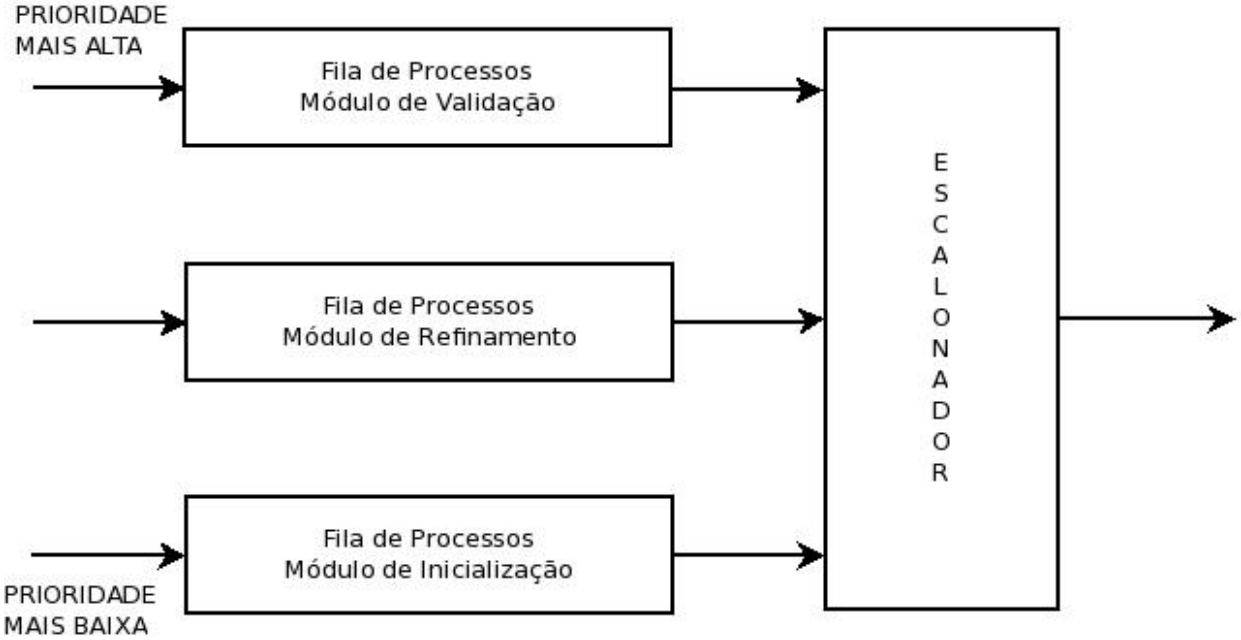

Figura 15 – Escalonamento com Filas Multiníveis - O **Módulo Especial de Escalonamento** define prioridades na administração das atividades do sistema. O **Módulo de Validação** tem maior prioridade no sistema que o **Módulo de Refinamento** que por sua vez possui maior prioridade do que o **Módulo de Inicialização**

do sistema, permitir que o processo `Avaliador da Qualidade` seja escalonado no mesmo computador que estava sendo executado o algoritmo de particionamento. Este procedimento minimiza o impacto da busca pelas informações dos dados gerados, diminuindo o tráfego na rede de comunicação de dados.

O código do Algoritmo 17, descrito na linguagem C++, apresenta a estrutura utilizada no administrador do **Módulo Especial de Escalonamento**. A função `EscalonaProc` procura pelo código pronto para escalonado, seguindo as prioridades das 3 filas.

Algoritmo 17 Estrutura Administrador - Módulo Especial de Escalonamento

```
1  #define NroFilas 3
2  #define TamStr 40
3  using namespace std;
4
5  struct Msg {
6    int funcao;
7    int prioridade;
8    char argumento[TamStr];
9  };
10 struct Processo {
11   int codFuncao;
12   char argumento[TamStr];
13   int pId;
14   int prioridade;
15 };
16 struct Host {
17   int codHost;
18   int codFuncao;
19 };
20 bool EscalonaProc(queue<Processo> FMN[NroFilas], Processo &Proc)
21 {
22   int nFila = 0;
23   while (nFila < NroFilas && FMN[nFila].size() == 0)
24     nFila++;
25
26   if (nFila == NroFilas)
27     return false;
28
29   Proc = FMN[nFila].front();
30   FMN[nFila].pop();
31   return true;
32 }
```

Em uma visão geral, o sistema funciona de modo semelhante a uma estrutura *pipeline*, compondo atividades que serão executadas em determinada sequência, ou seja, após a execução de tarefas anteriores e dependentes. Portanto, o sistema se aproveita do paralelismo quando são escalonadas várias tarefas de particionamento em bases diferentes, ou quando vários métodos são utilizados em uma mesma base, a partir das partições iniciais produzidas pelo **Módulo de Inicialização**.

3.1.4.2 Critério de Parada

Uma questão importante a ser considerada diz respeito aos critérios que o **Módulo Especial de Escalonamento** utiliza para interromper o escalonamento de novas tarefas. Quando o sistema verifica que o particionamento apresentado pelo **Módulo de Refinamento** já foi produzido por um método anterior (no **Módulo de Inicialização** ou em outro refinamento pelo mesmo método), não há necessidade de submetê-lo ao processamento pois resultaria em replicação de dados, a não ser que seja utilizado outro método de refinamento.

Outro critério de interrupção utilizado diz respeito a análise histórica dos valores obtidos para o **Atributo de Qualidade** utilizando o mesmo método de refinamento. Quando os valores circundam um valor de referência, sinalizando a proximidade de um mínimo local, o **Módulo Especial de Escalonamento** interrompe o escalonamento de novas tarefas. Observa-se que as abordagens para identificação do mínimo local podem variar conforme a base de dados utilizada.

Naturalmente, existem bases de dados cujas partições produzidas no **Módulo de Refinamento** não se repetem por causa da explosão combinatória provocada pela quantidade de informações. Além disso, as permutações obtidas a cada refinamento podem mudar significativamente o **Atributo de Qualidade** não sendo possível identificar a aproximação de um mínimo local. Nestas situações, é possível ainda definir como critério de parada uma quantidade máxima de refinamentos a partir de uma partição obtida através de um método do **Módulo de Inicialização**.

3.1.5 Gerenciamento dos Dados

Pela forma como os módulos do sistema proposto estão intrinsicamente relacionados, é possível identificar a necessidade de um bom gerenciamento dos dados que serão manipulados e produzidos durante a execução dos métodos de classificação e refinamento. Na Figura 10 estão representadas 4 bases de informação que, juntas, formam o repositório dos dados necessários para o funcionamento do sistema de aprendizado. De forma geral, as bases de dados utilizadas nos processos de classificação possuem quantidades significativas de dados armazenados. Além disso, o sistema proposto neste trabalho de doutorado permite que classificações obtidas através de uma técnica sejam reutilizadas em paralelo com a execução de outros algoritmos, atuando sobre os mesmos dados ou dados replicados, rapidamente gerando uma quantidade de informações cuja necessidade de gerenciamento é inerente.

Nesta seção, é apresentada a modelagem dos dados que compõe a estrutura de armazenamento do sistema. Para atingir este objetivo, o modelo conhecido como Entidade $\times$ Relacionamento foi escolhido, pois se trata do principal modelo de dados para aplicações de processamento de dados.

A Figura 16 apresenta o modelo conceitual que estrutura os dados manipulados pelos algoritmos inseridos na base. Esta descrição é importante, pois oferece uma estrutura padronizada. Neste sentido, as implementações de outros métodos que forem adicionadas ao Sistema deverão manipular os dados conforme a descrição representada pelo diagrama deste modelo de Entidade×Relacionamento, sendo necessário um processo de conversão dos dados originais para adaptá-los e inseri-los no banco de dados do sistema.

As entidades `BaseDeDados` e `Dados` representam os dados da base a ser classificada. `BaseDeDados` contém informações gerais que identificam a base de dados. No diagrama, estas informações estão representadas apenas pelo atributo `NomeDaBase`, mas outras informações relevantes fazem parte como, por exemplo, o endereço da Internet onde a base pode ser obtida ou, ainda, informações do significado de cada atributo. Os dados em si são armazenados na entidade `Dados`.

Quando um algoritmo de classificação, incorporado ao sistema, for executado sobre um conjunto de dados, as informações daquele processo de particionamento estarão armazenadas em seis entidades: `Planejamento`, `Experimento`, `MétodoDeClassificação`, `Métrica`, `CritérioDeParada` e `FunçãoDoAtributo`.

As entidades `Planejamento` e `Experimento` apresentam um relacionamento um-para-muitos, ou seja, um registro em `Planejamento` está associado a uma sequência de tarefas que serão executadas pelo sistema. As tarefas envolvem os algoritmos que serão utilizados no processo de inicialização, aqueles que serão executados no **Módulo de Validação** e os que serão utilizados no **Módulo de Refinamento**. O atributo `PreDefinido` da entidade `Planejamento` indica se a sequência será estática e pré-definida pelo usuário. É possível utilizar todos os algoritmos implementados no sistema, criando uma estrutura "inteligente" e dinâmica de retroalimentação do **Módulo Especial de Escalonamento**. Neste caso, os critérios de parada deverão estar bem definidos para que o sistema não fique em recursão sem fim. As funções que implementam os critérios de parada que serão utilizados nos experimentos de um determinado planejamento são definidas na entidade `CritérioDeParada`. O critério de parada para o planejamento é diferente do critério de parada utilizado por um método. No caso do planejamento, um critério de parada irá impedir que o escalonador programe novas execuções a partir de classificações cujos **atributos de qualidade** não alcançaram um bom índice previamente definido pelo usuário. Por sua vez, os critérios de parada de cada método serão definidos na entidade `ParâmetrosDeExecução`. Há métodos cujo critério de parada está implícito no algoritmo, como é o caso do `método da Árvore Binária`, enquanto outros dependem de testes de erro ou do número de iterações. É para estes que esta entidade foi definida, permitindo maior flexibilidade ao sistema. Além dos critérios de parada do método, esta entidade permite definir parâmetros que serão utilizados na execução de cada experimento.

A entidade `Métrica` armazena informações das funções que serão utilizadas pelos métodos de classificação para que o comportamento de cada método possa ser con-

figurado pelo usuário. Desta forma, as entidades `ParâmetrosDeExecução`, `Métrica` e `FunçãoDoAtributo` definem as funções que serão utilizadas para identificar se um algoritmo alcançou o ponto de convergência estabelecido ou se isso não será possível (caso do critério de parada do método), as métricas de similaridades que serão utilizados pelos métodos de classificação e o método utilizado para se calcular o **Atributo de Qualidade**, respectivamente.

Observa-se nas entidades `Métrica`, `CritérioDeParada` e `FunçãoDoAtributo` que, além da chave, só existe um atributo denominado, respectivamente, `FunçãoDeSimilaridade`, `ParadaDoMétodo` e `MétodoParaAtribQualidade`. Basicamente, estes atributos mantém as informações dos códigos que deverão ser utilizados. Em linguagens que oferecem a técnica de reflexão, como Java, estes atributos poderão armazenar o nome da classe que será vinculada dinamicamente durante a execução do algoritmo. No caso da linguagem C++, base para a construção do primeiro protótipo da ferramenta, estes atributos armazenam o endereço em disco do código que deverá ser executado. Outro ponto a ser destacado está no fato das entidades `CritérioDeParada` e `FunçãoDoAtributo` estarem relacionadas com a entidade `Planejamento` e não com a entidade `Experimento`. Isto ocorre devido à necessidade de se utilizar os mesmos critérios de comparação nos experimentos de um mesmo planejamento.

Observa-se que o modelo apresenta três entidades que são resultantes do procedimento de agregação de outras entidades do modelo. A primeira denomina-se `Particionado`, sendo resultante do relacionamento muitos-para-muitos entre as entidades `Dados` e `Experimento`. Através desta entidade, é possível identificar as partições geradas em cada experimento, com as informações dos respectivos dados de cada classificação realizada. A segunda entidade denomina-se `ConjuntoDeTeste`, resultante do relacionamento muitos-para-muitos entre as entidades `Dados` e `Supervisionado`. Esta entidade mantém as informações dos dados que serão utilizados para o pré-processamento e treinamento dos métodos supervisionados. A terceira, denominada `Configuração`, é um caso especial, pois não existirá fisicamente no banco de dados. Ela resolve a situação gerada pelo relacionamento implícito um-para-muitos entre as entidades `MétodosDeClassificação` e `Experimento`, pois, apesar de um método poder ser utilizado em vários experimentos, cada experimento só possui uma configuração para a sua execução.

A entidade `Experimento` possui dois atributos: `AtributoDeQualidade` e `NúmeroDe Partições(K)` para que sejam armazenados as informações do atributo de qualidade obtido em cada experimento e o número de partições que foram obtidas ou previamente definidas pelo usuário. Esta entidade possui um auto-relacionamento (`sucede`) para indicar se os dados, que estão sendo utilizados em um determinado experimento, já foram previamente classificados. Neste caso, haverá um campo na tabela resultante para manter esta ligação, permitindo, desta forma, rastrear a sequência de classificações que uma determinada base produziu. Além disso, este auto-relacionamento permite que o **Módulo**

Especial de Escalonamento escolha em qual fila de prioridades o processo deverá ser inserido. Caso o algoritmo parta de dados previamente classificados, então o escalonador irá escalonar a tarefa na fila de processos para o **Módulo de Refinamento**. Se for a primeira classificação, então o agendamento será feito na fila de processos para o **Módulo de Inicialização**.

Cada 'experimento' realizado indica um algoritmo de classificação que será escalonado para execução no sistema. Desta forma, a entidade `Experimento` está relacionada com a entidade `MétodoDeClassificação`, visto que cada método implementado poderá ser utilizado em vários experimentos. Esta entidade é especializada em duas outras, com base no critério que será utilizado por cada método. São elas: `NãoSupervisionado` e `Supervisionado`. Como explicado anteriormente, esta última apresenta um relacionamento muitos-para-muitos com a entidade `Dados`, produzindo a entidade intersectora com as informações dos dados que irão compor o conjunto de testes, representado pelo relacionamento `ConjuntoDeTeste`. A entidade `MétodoDeClassificação` mantém um atributo (`FilasDoEscalonador`). Este atributo irá permitir que o **Módulo Especial de Escalonamento** escolha a fila em que cada método será inserido. Existem métodos que serão utilizados apenas para se criar um particionamento inicial, outros serão utilizados para o refinamento e, há, também, métodos que podem ser utilizados em ambos os procedimentos.

Após a execução de um método de classificação, informações sobre o seu desempenho são armazenadas na tabela referente à entidade `InformaçõesEstatísticas`. Estas informações permitem que o **Módulo Especial de Escalonamento** possa otimizar o funcionamento do escalonador através da adaptação das políticas de escalonamento utilizadas, quando o mesmo algoritmo for reutilizado.

3.1.5.1 Os Metadados

Metadados são dados sobre dados, ou seja, são informações necessárias para se estruturar os arquivos de dados. Geralmente, estão vinculados a uma estrutura conhecida como dicionário de dados. Em bancos de dados tradicionais, o dicionário de dados é considerado um tipo especial de tabela que só pode ser acessado e atualizado pelo próprio sistema de banco de dados (não por um usuário comum). Um sistema de banco de dados consulta o dicionário de dados antes de ler ou modificar dados reais (SILBERSCHATZ et al., 2006).

A importância dos metadados surge devido à necessidade de gerenciar estruturas diferentes sem que, para isso, seja necessário implementar códigos distintos que executem os mesmos algoritmos. Os metadados permitem generalizar as estruturas, uma vez que informam como os dados de cada base estão estruturados.

O diagrama da Figura 16 apresenta o modelo Entidade×Relacionamento dos objetos que gerenciam os metadados do sistema proposto. Basicamente, são duas entidades:

`EstruturaDaBase` e `TipoDosAtributos` que definem a estrutura da entidade `Dados`, apresentada na Figura 17.

É importante considerar que a existência destes metadados permite flexibilizar o modelo apresentado pelo diagrama da Figura 16, uma vez que a estrutura das entidades `BaseDeDados` e `Dados` dependerá das informações disponíveis nas tabelas representadas pelas entidades do modelo representado pelo diagrama da Figura 17. Assim, cada base de dados será armazenada em uma estrutura adequada. Basicamente, a diferença reside na quantidade de atributos que os objetos de cada base oferecem. No entanto, não há necessidade de modificação nas estruturas das demais entidades, que não dependem da quantidade de atributos, mas das respectivas chaves de identificação.

3.2 Bases Utilizadas para Testes

O Sistema proposto foi idealizado para ser aplicado em n bases de dados. Foram utilizadas seis bases de dados disponíveis no repositório em Frank e Asuncion (2010). Este repositório, *UCI Machine Learning Repository*, é uma coleção de bancos de dados, teorias de domínio e geradores de dados que são utilizados pela comunidade de aprendizado de máquina para a análise empírica de algoritmos de aprendizado de máquina. Foi criado em 1987 por David Aha e colegas de pós-graduação na Universidade da Califórnia em Irvine. Desde então, tem sido amplamente utilizado por estudantes, educadores e pesquisadores de todo o mundo como a principal fonte de base de dados para aprendizagem de máquina. A versão atual do *site* foi projetado em 2007 por Arthur Assunção e David Newman.

Conforme apresentado no repositório de Frank e Asuncion (2010), dentre as bases mais utilizadas na área de AM, o sistema foi testado nas bases de dados *Iris*; *Zoo*; *Glass*; *Wine*; *Balance*; e *Covertype*.

O conjunto de dados Iris contém 3 classes com 50 instâncias cada, nos quais cada classe se refere a um tipo de planta. Os atributos presentes no conjunto de dados caracterizam 3 tipos de plantas, *Iris Setosa*, *Iris Versicolor* e *Iris Virgínica* e os atributos são caracterizados pelos comprimentos e pelas larguras de sépalas e pétalas. Existem duas classes que não são linearmente separáveis, tornando-se um exemplo para uso de métodos de classificação.

O conjunto de dados *Zoo* apresenta 101 instâncias com 17 características de 7 grupos de animais. Os grupos de animais foram separados conforme Tabela 2.

A base de dados *Glass* apresenta 214 instâncias com 10 características de 6 tipos de vidro. A importância do estudo de classificação de dados nesta base está relacionada com a possível contribuição na área de investigação criminológica.

As informações da base *Wine* foram obtidas a partir de uma análise química de vinhos cultivados numa mesma região da Itália, mas derivados de três cultivos diferentes. Esta

Tabela 2 – Animais por Grupo

Partição (N Dados)	Animais
0 (41)	porco da terra, antílope, urso, javali, búfalo, bezerro, *cavy*, chita, veado, golfinho, elefante, *fruitbat*, girafa, *girl*, cabra, gorila, hamster, lebre, leopardo, leão, lince, mink, toupeira, mangusto, gambá, antílope, ornitorrinco, doninha, pônei, toninha, puma, *pussycat*, guaxinim, rena, foca, leão, marinho, esquilo, vampiro, ratazana, wallaby, lobo
1 (20)	frango, corvo, pomba, pato, flamingo, gaivota, gavião, kiwi, cotovia, avestruz, periquito, pinguim, faisão, ema, *skimmer*, *skua*, pardal, cisne, abutre, *wren*
2 (5)	jararaca, cobra, verme lento, tartaruga, tuatara
3 (13)	*bass*, carpa, bagre, caboz, cação, *haddock*, arenque, lúcio, piranha, cavalo marinho, linguado, arraia, atum
4 (4)	dois tipo de rã, salamandra, sapo
5 (8)	pulga, mosquito, abelha, mosca, joaninha, traça, cupim, vespa
6 (10)	mariscos, caranguejo, lagostim, lagosta, polvo, escorpião, água viva, lesma do mar, vermes

avaliação determinou 13 características que foram encontradas em cada um dos três tipos de vinho, com 178 instâncias.

O conjunto de dados *Balance* apresenta quatro características com três classes e 624 instâncias. Este conjunto representa resultados experimentais de um modelo psicológico.

Finalmente, a base *Covertype* foi utilizada, parcialmente com 10000 registros, por possuir 581012 registros, com 55 atributos e 7 classes identificadas, tornando-se um bom exemplo para avaliar o desempenho do sistema. Esta base trata-se de informações relacionadas a quatro áreas selvagens encontradas no *Roosevelt National Forest of Northern* Colorado, informações estas importantes para apoiar gestores de recursos naturais em tomada de decisão para o desenvolvimento de estratégias de acompanhamento dos ecossistemas.

3.3 Ambiente Computacional Utilizado

A utilização de uma rede ou um *cluster* de computadores tornou-se uma poderosa ferramenta para solução de problemas complexos. Com o avanço e a popularidade da tecnologia, principalmente dos *hardwares* de computadores e os custos para a aquisição destes micros cada vez mais acessível, surgiu uma alternativa para utilização de processamento de alto desempenho utilizando vários PCs em paralelo.

Um *cluster* pode ser definido como um sistema no qual dois ou mais computadores trabalham de maneira conjunta para processar um número elevado de operações. Os

computadores interligados dividem as tarefas de processamento e trabalham como se fossem um único computador. Com isso, é possível realizar processamentos que até então somente computadores de alta performance seriam capazes de processar.

Cada computador de um *cluster* é denominado nó ou nodo. Todos devem estar interconectados, de maneira a formarem uma rede, de qualquer topologia. Essa rede precisa ser criada de uma forma que permita o acréscimo ou a retirada de um nó. O sistema operacional (SO) instalado deve ser de um mesmo tipo, isto é, linux, ou windows, dentre outros. Isso porque questões de compatibilidade poderiam impedir o funcionamento do *cluster*. Segundo Scherschel (2012), sobre a lista dos 500 supercomputadores mais rápidos do mundo, o sistema operacional Linux é utilizado por 462 dos 500 computadores. Isto é, mais de 92% dos computadores relacionados estão usando Linux. Além do SO de código aberto, 25 computadores no top 500 estão executando variantes de Unix, 2 estão executando o Windows e 11 supercomputadores estão usando uma combinação de SOs.

Independente do SO utilizado, é necessário software que auxilie na montagem do *cluster*. Esse software vai ser responsável pela distribuição do processamento e identificação de erros, oferecendo meios de providenciar reparos, mas sem interromper as atividades em processamento.

Mesmo que um aumento dos nós da rede intensifique os custos de implementação e manutenção, ainda assim os gastos geralmente são menores do que a aquisição e a manutenção de supercomputadores e, algumas vezes, o processamento é até mais eficiente.

O sistema proposto nesta tese permite o seu uso tanto em um único microcomputador quanto em uma rede de computadores. Os experimentos testados neste trabalho foram realizados em um *cluster* de computadores com a seguinte estrutura:

- ❑ Cinco computadores com as seguintes especificações:
  - Processador Intel(R) Core(TM)2 Quad CPU 2,66GHz;
  - Cache L2 8Mb;
  - 1066 Mhz Frontal bus;
  - 2 GBytes RAM.
- ❑ Rede de comunicação com a seguinte característica:
  - 100 Mbits/s Ethernet;
  - *Switch*: 3Com Baseline Switch 2948-SFP Plus 48-Port Gigabit (3CBL SG48).

As implementações dos programas foram desenvolvidas na linguagem C++, com a versão 1.5.4 da biblioteca OpenMPI. Utilizou-se o kernel 2.6.33.5-112.fc13 do sistema operacional Linux Fedora 13.

3.4 Considerações Finais

Este capítulo apresentou o projeto da Máquina ou Sistema de Aprendizado, cuja proposta é o tema desta tese. Apesar de terem sido apresentados alguns exemplos de códigos, descritos na linguagem C++, o objetivo principal foi discutir o projeto, através dos respectivos modelos e justificar como os diversos componentes se interconectam. A implementação do primeiro protótipo foi realizada utilizando-se a linguagem C++ e a biblioteca de troca de mensagens MPI. Por sua vez, com o detalhamento da modelagem deste projeto, há possibilidade de reescrever partes ou todo o código implementado, usando outras linguagens de programação. Isso permitiria oferecer novos recursos de interface e de acesso remoto, através da Internet, desde que o *hardware* que hospede este sistema permita este tipo de comunicação. É possível, da mesma forma, estender este modelo para hospedá-lo em ambiente de núvem (*cloud computing*). Isso favorecerá atender aos quesitos de escalabilidade e acessibilidade do sistema, quando estiver atuando em bases com grande volume de dados.

Como discutido, optou-se pelo modelo `Entidade×Relacionamento` para representar a modelagem dos dados do Sistema de Aprendizado proposto neste trabalho. Entretanto, para identificação de como as entidades irão se comportar fisicamente, outro modelo é mais adequado. Trata-se do `Modelo Relacional`. Por esta razão, existem procedimentos para se mapear os elementos do modelo `Entidade×Relacionamento` para o `Modelo Relacional`. Este mapeamento não será apresentado neste trabalho, mas o procedimento para realizá-lo pode ser encontrado em Silberschatz et al. (2006). Por sua vez, as tabelas de dados resultantes deste mapeamento serão apresentadas no próximo capítulo, durante a discussão dos experimentos realizados, permitindo a visualização das estruturas que comporão as bases de dados do sistema.

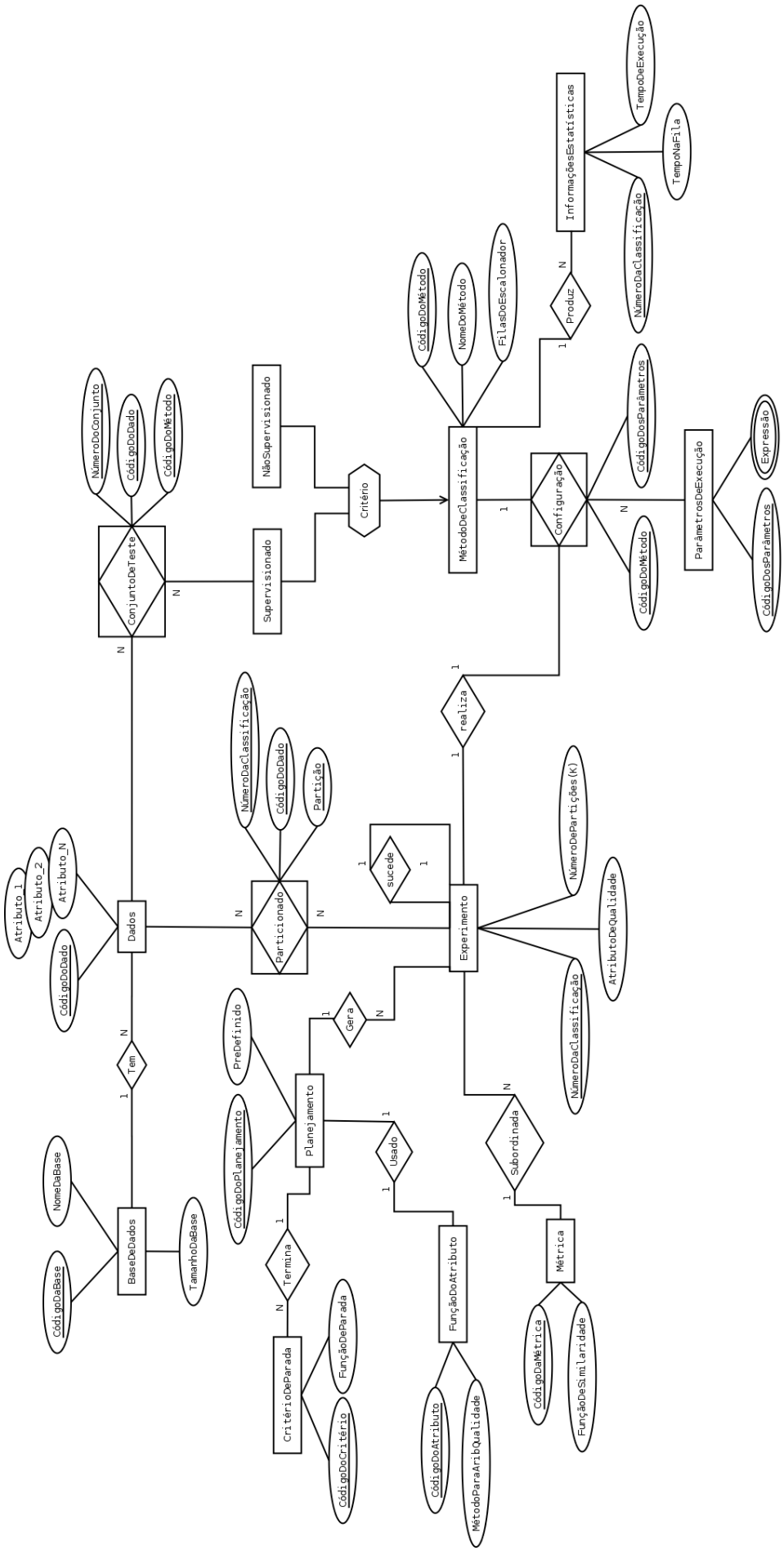

Figura 16 – Modelo Entidade×Relacionamento representando a estrutura dos dados armazenados no sistema proposto

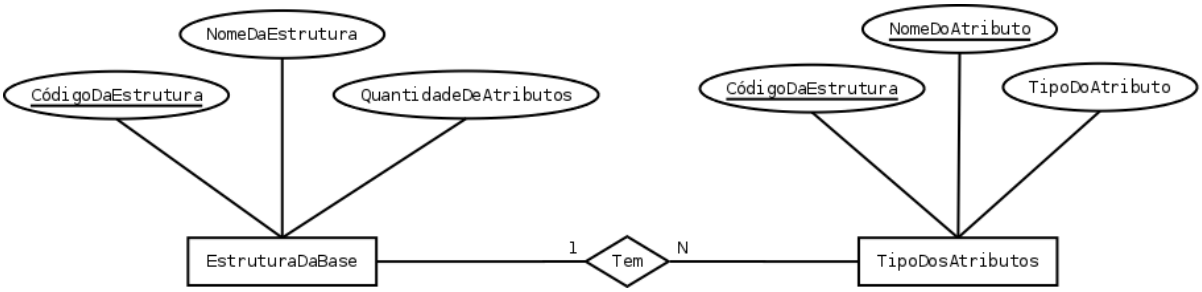

Figura 17 – Modelo Entidade×Relacionamento representando a estrutura dos metadados do sistema proposto

Capítulo 4

Experimentos e Resultados Obtidos

Os experimentos realizados pelo sistema de aprendizado proposto utilizaram as bases de Frank e Asuncion (2010). Conforme descrito no capítulo 1, os procedimentos foram aplicados nas bases de dados *Iris*; *Zoo*; *Glass*; *Wine*; *Balance* e *Covertype*. O objetivo principal não é verificar a eficácia dos métodos de classificação escolhidos para serem incorporados ao sistema, por se tratarem de algoritmos já consolidados na literatura, mas demonstrar o funcionamento do sistema de aprendizado proposto, destacando o mecanismo de escalonamento e a capacidade do sistema de refinar as classificações obtidas através das técnicas de refinamento. Além disso, o desempenho é um fator muito importante nesta análise, por se tratar de um sistema paralelo. A seção 4.1 apresentará de forma mais detalhada o comportamento do sistema e a maneira como a qual ele manipula a base de dados. A seção 4.2 ilustrará algumas informações em relação as outras bases testadas e finalmente a seção 4.3 fará uma discussão do desempenho do sistema durante os experimentos.

4.1 Base de dados Iris

O primeiro exemplo que será apresentado terá por objetivo ilustrar o fluxo de execução que é realizado no sistema. Serão apresentados os detalhes a respeito do comportamento dos módulos que compõem o sistema, além das estruturas dos dados nas respectivas bases de dados.

A base de dados escolhida para este propósito foi a base de dados Iris, por ser uma base pequena (possuir poucos dados, 4 atributos e 150 elementos) e ser bem conhecida na comunidade acadêmica.

Inicialmente, a base deve ser preparada, sendo necessário transferir os dados para os arquivos do sistema conforme descrição dos modelos `Entidade`×`Relacionamento` das Figuras 16 e 17. Por se tratar de uma base em que cada objeto possui atributos, os registros das tabelas referentes aos metadados devem manter esta informação, para que as tabelas `BaseDeDados` e `Dados` contenham exclusivamente esta quantidade de atributos.

É importante considerar que as tabelas apresentadas nesta seção descrevem a estrutura física dos arquivos, portanto, representam o resultado do mapeamento entre os modelos `Entidade×Relacionamento` e `Relacional`.

A Tabela 3 - `EstruturaDaBase` ilustra o registro que caracteriza a Base de Dados Iris e que representa a entidade `EstruturaDaBase` apresentada no modelo `Entidade× Relacionamento` da figura 17. Por sua vez, a Tabela 4 `TipoDosAtributos` armazena os dados dos campos referentes aos atributos correspondentes às 4 características da base de dados Iris.

Tabela 3 – Tabela `EstruturaDaBase` com um registro referente a base de dados *Iris*

EstruturaDaBase		
`CódigoDaEstrutura`	`NomeDaEstrutura`	`QuantidadeDeAtributos`
01	Iris	4

Tabela 4 – Tabela `TipoDosAtributos` com um registro para cada atributo da base Iris

TipoDosAtributos		
`CódigoDaEstrutura`	`NomeDoAtributo`	`TipoDoAtributo`
01	CompSépala	real
01	LargSépala	real
01	CompPétala	real
01	LargPétala	real

Uma vez configurados os metadados, as tabelas referentes aos dados em si podem ser preenchidas. A Tabela 5 - `BaseDeDados` ilustra como estarão os dados que foram inseridos, provenientes da base original. Esta Tabela corresponde a entidade de mesmo nome no modelo da Figura 16. São apresentados, como ilustração, apenas as 3 primeiras linhas dos 150 registros da base *Iris*.

Tabela 5 – Tabela `BaseDeDados` com os registros correspondentes as bases utilizadas nos experimentos

BaseDeDados		
`CódigoDaBase`	`NomeDaBase`	`TamanhoDaBase`
01	*Iris*	100kb
02	*Zoo*	154kb
03	*Glass*	212kb
04	*Wine*	99kb
05	*Balance*	512kb
06	*Forest CoverType*	1024kb
07	*Adult*	2048kb

A Tabela 6 - `Dados` corresponde às informações presentes em cada atributo da base. Essas informações serão utilizadas para a criação da matriz de atributos que, posteriormente, será a base para a geração da matriz de similaridades. Observa-se, nesta Tabela,

a presença do campo `CódigoDaBase`. Este campo é fruto do mapeamento do `Modelo Entidade×Relacionamento` para o `Modelo Relacional` e representa a chave estrangeira do relacionamento um-para-muitos entre `BaseDeDados` e `Dados`. O detalhamento deste mapeamento não será apresentado, mas as Tabelas seguintes serão apresentadas com todos os atributos resultantes do mapeamento.

Tabela 6 – Tabela 'Dados' correspondente as informações dos atributos da base *Iris* como os centímetros do comprimento e largura de sépala e do comprimento e largura de pétala

Dados					
`CódigoDoDado`	`CódigoDaBase`	`CompSépala`	`LargSépala`	`CompPétala`	`LargPétala`
1	1	5,1	3,5	1,4	0,2
2	1	4,9	3,0	1,4	0,2
...	1	...	...	...	...
150	1	5,9	3,0	5,1	1,8

A Tabela 7 - `MétodoDeClassificação` corresponde aos métodos implementados no sistema e que podem ser utilizados para o particionamento das bases. O campo `FilasDo Escalonador` recebe valores de 1 até 7. Cada fila é representada pelo coeficiente do polinômio de segundo grau de base 2. Assim, o valor $a_0 \times 2^0$ corresponde a fila do **Módulo de Inicialização**; o valor $a_1 \times 2^1$ corresponde a fila do **Módulo de Refinamento** e o valor $a_2 \times 2^2$ corresponde a fila do **Módulo de Validação**. Quando um método pode ser utilizado em mais de uma fila, os coeficientes correspondentes recebem o valor 1, caso contrário, o valor 0. Por exemplo, o método da `Árvore Binária` só será utilizado para gerar uma primeira partição. Por isso, o valor correspondente é 1. O mesmo acontece com os métodos `Randômico` e `Semi-randômico`. Por sua vez, o método de `Consulta por Abrangência` é utilizado para refinar uma classficação obtida anteriormente. Neste sentido, o valor deste campo é 2. Já o método `Kernighan-Lin` (KL) pode ser utilizado tanto para produzir a primeira partição quanto para refinar uma partição anterior. Neste caso, o Escalonador poderá inserir o método em ambas as filas. O valor do atributo correspondente será 3. Finalmente, no procedimento de mapeamento entre os modelos, as entidades `Supervisionado` e `NãoSupervisionado` definem o critério que será descrito no campo `Critério`.

No planejamento para os experimentos realizados com a base de dados *Iris*, os métodos foram previamente definidos. As tabelas 8 - `Planejamento` e 9 - `Experimento` descrevem o planejamento elaborado para este experimento. Desta forma, como os algoritmos foram escolhidos estaticamente, o registro correspondente na tabela `Planejamento` manteve o campo `PreDefinido` com o valor 'não'. O campo `CódigoDoAtributo` representa o método utilizado nos experimentos deste planejamento para se calcular o **Atributo de Qualidade**. No caso da base de dados *Iris*, este cálculo foi realizado conforme descrito na seção 3.1.2, identificado na tabela `FunçãoDoAtributo` com o valor de código igual a '01'. A tabela 9 - `Experimento` apresenta o estado antes do processamento de cada método.

Tabela 7 – Tabela `MétodoDeClassificação` correspondente a todos os métodos implementados em todos os módulos no sistema

MétodoDeClassificação			
`CódigoDo Método`	`NomeDo Método`	`FilasDo Escalonador`	`Critério`
01	Árvore Binária	01	Não Supervisionado
02	Randômico	01	Não Supervisionado
03	Semi-randômico	01	Não Supervisionado
04	Consulta por Abrangência	02	Não Supervisionado
05	Raio de Cobertura	02	Não Supervisionado
06	Variação KL	03	Não Supervisionado
07	Fiduccia	03	Não Supervisionado
08	Estabilidade	04	Não Supervisionado
09	Rede Neural 1	01	Supervisionado
10	Árvore de Decisão	01	Supervisionado

Por esta razão, a coluna `AtributoDeQualidade` não possui valor. Outro ponto a ser observado nesta Tabela é a existência de dois experimentos associados ao mesmo método de refinamento (`Consulta por Abrangência` - código '04') com refinamento sobre a mesma classificação obtida pelo `Método da Árvore Binária`. A diferença se encontra nos parâmetros utilizados, basicamente no valor do raio de cobertura. Esta diferença será comentada a seguir. Quando o método não necessita de parâmetros de execução, o valor '00' é atribuído ao campo `CódigoDosParâmetros`, conforme pode ser observado na primeira linha da tabela 9.

Tabela 8 – Tabela `Planejamento` indicando se a sequência será estática e pré-definida pelo usuário

Planejamento		
CódigoDoPlanejamento	CódigoDoAtributo	PreDefinido
01	01	não

Tabela 9 – Tabela `Experimento` correspondente a sequência informada ao escalonador no processo de particionamento da base *Iris* e com os atributos conforme tabela resultante do mapeamento entre o Modelo Entidade×Relacionamento e o Modelo Relacional. Os dados representam o estado da tabela antes do processamento

Experimento							
`NúmeroDa Classificação`	`CódigoDo Método`	`CódigoDos Parâmetros`	`CódigoDa Métrica`	`CódigoDo Planejamento`	`AtributoDe Qualidade`	`Classicação Anterior`	`NúmeroDe Partições`
01	01	00	01	01		00	3
02	02	01	01	01		00	3
03	03	01	01	01		00	3
04	04	01	01	01		01	3
05	04	02	01	01		01	3
06	05	01	01	01		00	3

A Figura 18(a) ilustra o estado da fila após a leitura dos registros associados ao planejamento '01'. Observa-se que a fila **Módulo Especial de Escalonamento** seleciona

apenas os registros cujo processamento está vinculado ao **Módulo de Inicialização**, através da identificação do valor '00' no campo `ClassificaçãoAnterior`. Como esta fila é de prioridade mais baixa, o escalonador não insere atividades nas demais filas, apesar de estaticamente isso já estar definido, para impedir que seja iniciado um método que necessita de uma particionamento prévio. Após o processamento do experimento '01', correspondente ao `Método da Árvore Binária`, o **Módulo Especial de Escalonamento** faz uma busca no banco de dados para identificar se existe algum experimento vinculado com o resultado do experimento '01'. A partir deste ponto, o **Módulo Especial de Escalonamento** insere na fila associada ao **Módulo de Refinamento** os experimentos de código '04' e '05', que passam a ter prioridade em relação aos experimentos vinculados à fila de Inicialização. A partir deste momento, a ordem de execução é não-determinística e dependerá da disponibilidade de recursos físicos (processadores) disponíveis.

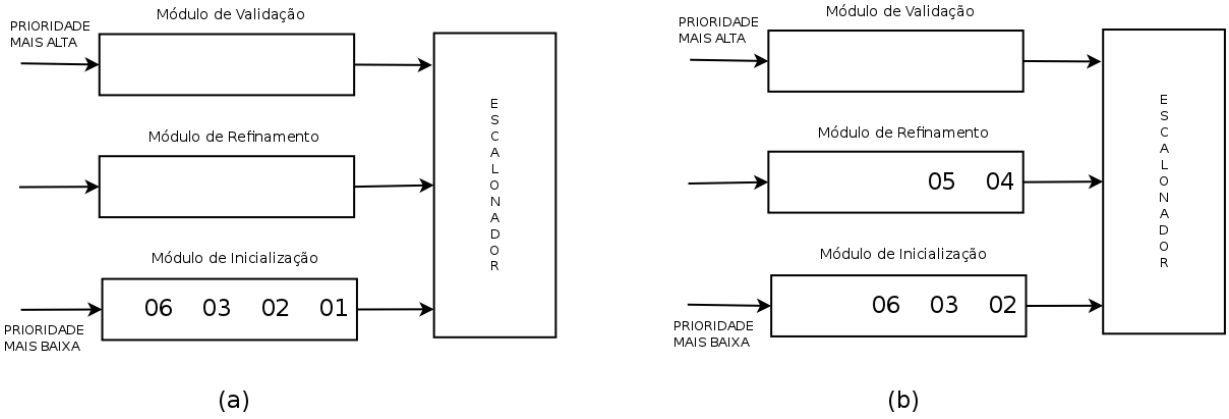

Figura 18 – Ilustração do estado das filas multiníveis após o **Módulo Especial de Escalonamento** ter feito a leitura do experimento definido na Tabela 9

A Tabela 10 apresenta o estado final destes registros após a execução dos respectivos métodos. Ao verificar o primeiro registro desta Tabela, observa-se que o resultado da execução do `Método da Árvore Binária` produziu um **Atributo de Qualidade** com o valor de **2,69466**. Considerando os dados utilizados nesta pesquisa, e a partir da geração da matriz de similaridades, descrito na seção 2.2.2, o sistema executou, para meios de comparação, o **Módulo de Validação** a partir do particionamento original de Fisher (1936), sendo seu **Atributo de Qualidade** com o valor de **2,82201**.

É importante considerar que o arquivo original de teste está bem organizado, um vez que os atributos das respectivas espécies estão agrupados, ou seja, as características cujos rótulos estão entre 0 e 49 referem-se a *Iris Setosa*; as características de 50 a 99 referem-se a *Iris Versicolor*; e as características dos rótulos 100 a 149 referem-se a *Iris Virgínica*.

O resultado apresentado pelo `Método da Árvore Binária`, o qual foi considerado significativamente melhor do que o original, demonstra a existência de sobreposições entre as classes, uma vez que o cálculo do **Atributo de Qualidade** utilizou como métrica

Tabela 10 – Tabela `Experimento` equivalente à Tabela 9, com os resultados do processamento realizado

Experimento							
`NúmeroDa Classificação`	`CódigoDo Método`	`CódigoDos Parâmetros`	`CódigoDa Métrica`	`CódigoDo Planejamento`	`AtributoDe Qualidade`	`Classicação Anterior`	`NúmeroDe Partições`
01	01	00	01	01	**2,69466**	00	3
02	02	01	01	01	**5,68444**	00	3
03	03	01	01	01	**2,71932**	00	3
04	04	01	01	01	**2,76877**	01	3
05	04	02	01	01	**2,72368**	01	3
06	05	01	01	01	**2,70427**	06	3

apenas a distância Euclidiana. Ainda assim, tal processo demonstrou ser um método bastante eficiente e rápido. A tabela 11 apresenta os rótulos dos atributos nas respectivas partições. Os rótulos são as informações da célula `CódigoDoDado` da tabela 6 `Dados`.

Tabela 11 – Particionamento da Base de Dados *Iris* obtido pelo `Método da Árvore Binária` no **Módulo de Inicialização**

Partição	Atributos (rótulos)
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1	50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 113 114 119 121 123 126 127 133 138 142 146 149
2	100 102 103 104 105 107 108 109 110 111 112 115 116 117 118 120 122 124 125 128 129 130 131 132 134 135 136 137 139 140 141 143 144 145 147 148

Ao escalonar o segundo registro da Tabela 9, o `Método Randômico` foi acionado para ser aplicado na base *Iris*. Como geração das partições iniciais, os resultados foram significativamente piores, o que já era esperado. Foram produzidos mais de **3.000.000** de particionamentos, partindo de diferentes sementes de números aleatórios, mas mantendo-se a mesma distribuição de probabilidade (uniforme). A especificação do número de iterações e de qual distribuição foi utilizada está definida na tabela `ParâmetrosDeExecução`, cujo código correspondente, neste caso, foi '01'. Nestas classificações, o melhor resultado encontrado foi o **Atributo de Qualidade** igual a **5,68444**. A tabela 12 apresenta o melhor particionamento encontrado.

Os resultados obtidos utilizando o `Método Semi-randômico` também ficaram próximos do conjunto definido por Fisher (1936). Os parâmetros utilizados neste experimento foram idênticos àqueles utilizados no `Método Randômico`, ou seja, 3.000.000 de iterações e a distribuição uniforme para a escolha das sementes aleatórias. O melhor particonamento que foi alcançado é apresentado na Tabela 13. Após a análise no **Módulo de Validação** este conjunto obteve o **Atributo de Qualidade** igual a **2,71932** e, como pode ser visto na Tabela 13, dos 150 elementos da base, apenas **13** elementos **(50, 52, 72, 76, 77,**

Tabela 12 – Melhor conjunto de dados particionado da base de dados *Iris* obtido pelo `Método Randômico` do **Módulo de Inicialização** entre **3.000.000** de execuções

Partição	Atributos (rótulos)
0	18 35 39 46 48 51 54 55 61 62 67 70 72 74 75 76 82 87 89 90 91 92 93 97 99 101 102 103 107 111 113 115 116 122 124 125 129 131 132 133 134 136 137 138 141 143 144 145 146 147
1	0 7 12 24 26 47 49 50 52 53 56 57 58 59 60 63 64 65 71 77 80 81 84 88 94 98 100 104 105 106 108 109 110 112 114 117 118 119 120 121 123 126 127 128 135 139 140 142 148 149
2	1 2 3 4 5 6 8 9 10 11 13 14 15 16 17 19 20 21 22 23 25 27 28 29 30 31 32 33 34 36 37 38 40 41 42 43 44 45 66 68 69 73 78 79 83 85 86 95 96 130

83, 86, 106, 113, 119, 121, 126 e 138) foram particionados em classes divergentes da classificação de Fisher (1936).

Tabela 13 – Melhor conjunto de dados particionado da base de dados *Iris* obtido pelo `Método Semi-randômico` do **Módulo de Inicialização** entre **3.000.000** de execuções

Partição	Atributos (rótulos)
0	50 52 72 76 77 83 86 86 100 101 102 103 104 105 107 108 109 110 111 112 114 115 116 120 122 123 124 125 127 128 129 130 131 132 133 134 135 136 137 139 140 141 142 143 144 145 146 147 148 149
1	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
2	49 51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 73 74 75 78 79 80 81 82 84 85 87 88 89 90 91 92 94 95 95 96 97 98 99 106 113 119 121 126 138

A partir do quarto registro da Tabela 9, e após a geração dos conjuntos iniciais, estes foram utilizados pelos algoritmos do **Módulo de Refinamento**. Como foi discutido na seção 3.1.3, a escolha dos objetos candidatos para trocar de partição pode ser feita por vários métodos distintos. Neste experimento, foram utilizados dois métodos: `Consulta por Abrangência` e `Raio de Cobertura Mínimo`.

Considerando o `Método de Consulta por Abrangência`, o **Módulo de Refinamento** gerou tarefas para o **Módulo Especial de Escalonamento**, partindo dos próprios raios de abrangência dos conjuntos originais, e aumentando, os raios em **15%** a cada passo na distância de abrangência Euclidiana (critérios definidos na Tabela `ParâmetrosDeExecução`). A condição de parada na geração dos conjuntos está condicionada a interrupção na diminuição do **Atributo de Qualidade** (critério definido na Tabela `CritérioDeParada`).

Na consulta utilizando os raios de cobertura originais, o **Atributo de Qualidade**

Tabela 14 – Refinamento do conjunto gerado pelo `Método da Árvore Binária` a partir do `Método de Consulta por Abrangência` com o raio original dos respectivos conjuntos

Partição	Atributos (rótulos)
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1	50 51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 110 113 114 119 121 123 126 127 138 142 149
2	52 77 100 102 103 104 105 107 108 109 111 112 115 116 117 118 120 122 124 125 128 129 130 131 132 133 134 135 136 137 139 140 141 143 144 145 146 147 148

atingiu **2,76877** no primeiro refinamento (partições correspondentes na Tabela 14) e **2,72368** no segundo refinamento, ou seja, na aplicação do `Método de Consulta por Abrangência` no resultado do 1° refinamento (partições correspondentes na Tabela 15). Em um teste alternativo, com o objetivo de verificar a interrupção automática na geração de refinamentos pelo **Módulo Especial de Escalonamento**, o resultado de um novo refinamento neste conjunto produziu o mesmo conjunto do primeiro refinamento, fazendo com que o Escalonador interrompesse a geração de novas tarefas. Vale destacar que o conjunto original havia alcançado **91,33%** de acerto e que, através deste refinamento, houve uma melhoria no índice de acerto para **93,33%**.

Tabela 15 – Refinamento do conjunto representado pela Tabela 14

Partição	Atributos (rótulos)
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1	50 51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 110 113 114 119 121 123 126 127 138 142 149
2	52 77 100 102 103 104 105 107 108 109 111 112 115 116 117 118 120 122 124 125 128 129 130 131 132 133 134 135 136 137 139 140 141 143 144 145 146 147 148

Nos refinamentos que aumentaram o raio, destaca-se o conjunto resultante do aumento em **0,2** no valor do raio original do conjunto referente à partição 2, um aumento correspondente a aproximadamente **16%** do raio da maior partição original. Neste resultado, o **Atributo de Qualidade** foi **2,74838** e o conjunto resultante pode ser visto na Tabela 16.

O processo de refinamento apresentado pela Tabela 16 informa os dados candidatos a serem reposicionados nos *clusters*. A Tabela 17 apresenta os dados correspondentes à

Tabela 16 – Refinamento do conjunto gerado pelo `Método da Árvore Binária` no **Módulo de Inicialização** a partir do `Método de Consulta por Abrangência` com o raio expandido em **0,2**, referente ao raio original dos respectivos conjuntos

Partição	Atributos (rótulos)
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1	51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 73 74 75 78 79 80 81 82 84 85 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 113 114 119 121 138 142
2	50 52 72 76 77 83 86 100 102 103 104 105 107 108 109 110 111 112 115 116 117 118 120 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 139 140 141 143 144 145 146 147 148 149

mudança.

Tabela 17 – Atributos candidatos a realocação conforme refinamento na Tabela 16

Partição Origem - Destino	Atributos (rótulos)
2 - 3	50 52 72 76 77 83 86 123 126 127 133 146 149

Em relação às partições originárias dos métodos de `Inicialização Randômico e Semi-randômico`, o **Módulo Especial de Escalonamento** realizou apenas quatro refinamentos sucessivos uma vez que as trocas estavam resultando em **Atributos de Qualidade** que permaneciam próximo de um mínimo local variando ora para mais ora para menos a cada refinamento.

O processo de refinamento também aplicou o `Método do Raio de Cobertura Mínimo` para os três conjuntos definidos respectivamente pelo `Método da Árvore Binária`, `Método Randômico` e `Método Semi-randômico`. Após aplicar o `Método do Raio de Cobertura Mínimo` no particionamento obtido pelo `Método da Árvore Binária`, o **Módulo de Validação** calculou o **Atributo de Qualidade** em **2,70427**. Os dados apresentados neste refinamento estão expostos na Tabela 18. O **Módulo Especial de Escalonamento** produziu apenas mais uma tarefa de refinamento com este método, pois não houve alteração das partições a partir deste resultado. Observa-se que este resultado se afastou ligeiramente daquele alcançado pelo `Método da Árvore Binária` no **Módulo de Inicialização**.

Após o **Módulo de Inicialização** utilizar o `Método Randômico` e obter um **Atributo de Qualidade** igual a **5,68444** em **3.000.000** de iterações, o **Módulo de Refinamento** aplicou o `Método do Raio de Cobertura Mínimo` no melhor particionamento definido pelo `Método Randômico`, o processo de refinamento obteve um **Atributo de Qualidade** com o valor de **3,65062** na primeira iteração do refinamento. Após mais três refinamentos, o `Método do Raio de Cobertura Mínimo` convergiu para o particionamento apresentado

Tabela 18 – Refinamento do conjunto do `Método da Árvore Binária` a partir do `Método do Raio de Cobertura Mínimo`

Partição	Atributos (rótulos)
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1	51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 113 114 119 121 123 126 127 133 138 142 146 149
2	50 52 77 100 102 103 104 105 107 108 109 110 111 112 115 116 117 118 120 122 124 125 128 129 130 131 132 134 135 136 137 139 140 141 143 144 145 147 148

na Tabela 19, cujo **Atributo de Qualidade** é **2,71379**.

Tabela 19 – Refinamento, com o `Método do Raio de Cobertura Mínimo`, do particionamento obtido através do `Método Randômico` do conjunto de dados *Iris*

Partição	Atributos (rótulos)
0	50 52 76 77 100 102 103 104 105 107 108 109 110 111 112 115 116 117 118 120 122 124 125 128 129 130 131 132 134 135 136 137 139 140 141 143 144 145 147 148
1	51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 101 106 113 114 119 121 123 126 127 133 138 142 146 149
2	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

A Tabela 20 apresenta os dados particionados após aplicação do `Método do Raio de Cobertura Mínimo` nas partições definidas pelo `Método Semi-randômico`. O método convergiu, após 2 refinamentos, para o **Atributo de Qualidade** de valor **2,85162**. Este valor é o mais próximo do valor obtido pela definição de Fisher (1936), a prova que esta representação contribui para uma análise do índice de acerto é que apenas 12 elementos não foram classificados conforme Fisher (1936), são eles: **50, 52, 76, 77, 83, 86, 106, 113, 119, 121, 126 e 138**.

Por fim, com o objetivo de comparação, elaborou-se a Tabela 21 que finaliza esta análise com a base de dados *Iris*, apresentando o índice de acerto conforme repositório em Frank e Asuncion (2010). A primeira coluna apresenta o método usado para a geração da primeira partição e o índice de acerto encontrado. A segunda coluna corresponde aos métodos de refinamento contendo, para cada um, duas informações. A primeira informação é a quantidade de iterações que foram necessárias até que se atingisse algum critério de parada, a segunda informação é o índice de acerto da melhor partição obtida.

Tabela 20 – Refinamento, com o `Método do Raio de Cobertura Mínimo`, do Particionamento gerado pelo `Método Semi-randômico`

Partição	Atributos (rótulos)
0	50 52 76 77 83 86 100 101 102 103 104 105 107 108 109 110 111 112 114 115 116 117 118 120 122 123 124 125 127 128 129 130 131 132 133 134 135 136 137 139 140 141 142 143 144 145 146 147 148 149
1	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 42 43 44 45 46 47 48
2	41 49 51 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 78 79 80 81 82 84 85 87 88 89 90 91 92 93 94 95 96 97 98 99 106 113 119 121 126 138

Tabela 21 – Análise dos dados da base *Iris* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento**

Conjunto de Dados *Iris*					
Módulo de Inicialização		Módulo de Refinamento			
Método	Índice de Acerto	*Raio de Cobertura Mínimo*		*Consulta por Abrangência*	
		Iterações	Índice de Acerto	Iterações	Índice de Acerto
Árvore Binaria	91,33%	1	88,66%	2	93,33%
Randômico	54,66%	3	88%	6	55,33%
Semi-Randômico	90,66%	1	91,33%	2	92%

As informações apresentadas pela Tabela 22 auxiliam na comparação das alterações realizadas nos valores do **Atributo de Qualidade** com a porcentagem de acerto do Índice conforme o repositório em Frank e Asuncion (2010).

Tabela 22 – Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Iris*

Módulo de Inicialização		*Raio de Cobertura Mínimo*	*Consulta por Abrangência*
Árvore Binária	**2,69466**	**2,70427**	**2,72368**
Randômico	**5,68444**	**2,71379**	**4,18003**
Semi-Randômico	**2,71932**	**2,70231**	**2,85162**

4.2 Outros Experimentos

Após apresentar mais detalhadamente o comportamento do sistema na seção 4.1 quando atua em uma base de dados, esta seção apresentará algumas informações obtidas nos testes realizados com outras bases de dados.

Os registros correspondente aos 17 atributos da base de dados *Zoo* utilizados para definir o particionamento e gerar a matriz de similaridades possuem tipos de dados lógicos e numéricos, conforme apresentado na Tabela 23.

Tabela 23 – Atributos e seus Tipo de Dados da Base de Dados *Zoo*

Tipo de Dado	Atributo
boolean	pêlos, penas, *eggs*, *milk*, aéreos, aquáticos, predador, dentes, espinha dorsal, respira, peçonhento, barbatanas, cauda, domésticos, *catsize*
numeric	pernas, tipo

Para facilitar a análise e a comparação dos particionamentos gerados pelo sistema em relação às informações apresentadas em Frank e Asuncion (2010), os dados foram organizados de acordo com a Tabela 24.

Tabela 24 – Particionamento do Conjunto de Dados *Zoo* conforme Frank e Asuncion (2010)

Partição	Atributos
0 (41)	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
1 (20)	41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
2 (5)	61 62 63 64 65
3 (13)	66 67 68 69 70 71 72 73 74 75 76 77 78
4 (4)	79 80 81 82
5 (8)	83 84 85 86 87 88 89 90
6 (10)	91 92 93 94 95 96 97 98 99 100

A validação (**Módulo de Validação**) aplicado nos dados, conforme Tabela 24, pontuou o **Atributo de Qualidade** com o valor de **12,7665**. A Tabela 25 apresenta o particionamento extraído do conjunto de dados *Zoo* utilizando o `Método da Árvore Binária`, no **Módulo de Inicialização**. O valor deste **Atributo de Qualidade** foi **9,70902**.

Tabela 25 – Particionamento do Conjunto de Dados *Zoo* conforme `Método da Árvore Binária`

Partição	Atributos
0 (38)	0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 30 31 32 33 35 36 37 38 39 40
1 (3)	9 29 34
2 (20)	41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
3 (16)	61 62 63 66 67 68 69 70 71 72 73 74 75 76 77 78
4 (6)	64 65 79 80 81 82
5 (14)	83 84 85 86 87 88 89 90 92 93 94 95 96 99
6 (4)	91 97 98 100

Ao aplicar o `Método do Raio de Cobertura Mínimo` no **Módulo de Refinamento** a partir do particionamento gerado pelo `Método da Árvore Binária`, o particionamento

convergiu em um índice de acerto de **88,12%**. O conjunto particionado é apresentado na Tabela 26.

Tabela 26 – Refinamento pelo `Método do Raio de Cobertura Mínimo` a partir do particionamento do Conjunto de Dados *Zoo* conforme o `Método da Árvore Binária`

Partição	Atributos
0 (37)	0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 30 31 32 33 36 37 38 39 40
1 (4)	9 29 34 35
2 (20)	41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
3 (16)	61 62 63 66 67 68 69 70 71 72 73 74 75 76 77 78
4 (6)	64 65 79 80 81 82
5 (14)	83 84 85 86 87 88 89 90 92 93 94 95 96 99
6 (4)	91 97 98 100

Ao aplicar o `Método de Consulta por Abrangência` (**Módulo de Refinamento**) no particionamento gerado pelo `Método da Árvore Binária`, o **Atributo de Qualidade** foi **10,2694**. O particionamento resultante (Tabela 27) apresenta o índice de acerto de **64,35%**. Neste processo, o **Módulo Especial de Escalonamento** escalonou 2 tarefas para o **Módulo de Refinamento** com base no `Método de Consulta por Abrangência`.

Tabela 27 – Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método da Árvore Binária`

Partição	Atributos
0 (25)	0 1 2 4 5 6 8 10 11 12 13 14 15 16 21 25 26 28 31 33 35 36 37 39 64
1 (3)	9 29 34
2 (16)	41 43 44 45 46 49 50 51 52 53 55 56 57 58 59 60
3 (16)	61 62 63 66 67 68 69 70 71 72 73 74 75 76 77 78
4 (19)	3 7 17 18 19 20 22 23 24 27 30 32 38 40 42 47 48 54 65
5 (18)	79 80 81 82 83 84 85 86 87 88 89 90 92 93 94 95 96 99
6 (4)	91 97 98 100

Na execução do `Método Randômico` foram produzidos mais de **3.000.000** de particionamentos partindo de diferentes sementes de números aleatórios, este número de iterações é similar aos testes realizados com a base de dados *Iris*. Nestas classificações, o menor **Atributo de Qualidade** encontrado foi **20,1517** representando um índice de acerto de **31,68%** conforme Frank e Asuncion (2010).

Como aconteceu na base *Iris*, a aplicação do `Método do Raio de Cobertura Mínimo` no melhor particionamento obtido pelo `Método Randômico`, melhorou o **Atributo de Qualidade** para **11,9582** (tabela 28), resultando em um índice de acerto de **66,33%**.

O particionamento extraído pelo `Método de Consulta por Abrangência`, a partir da classificação gerada pelo `Método Randômico`, não apresentou um índice de acerto. Isto

Tabela 28 – Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Randômico`

Partição (N de dados)	Atributos
0 (12)	1 4 5 8 10 12 14 25 26 28 33 64
1 (21)	0 2 3 6 7 16 17 18 19 20 21 22 23 24 27 30 31 32 38 40 65
2 (12)	79 80 81 82 83 84 85 86 87 88 89 90
3 (7)	9 13 29 34 35 52 54
4 (23)	11 15 36 37 39 41 42 43 44 45 46 47 48 49 50 51 53 55 56 57 58 59 60
5 (20)	61 62 63 66 67 68 69 70 71 72 73 74 75 76 77 78 91 97 98 100
6 (6)	92 93 94 95 96 99

ocorreu devido a mudança do número de classes que o método aplicou no conjunto. Este particionamento é ilustrado na Tabela 29.

Tabela 29 – Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Randômico`

Partição (N de dados)	Atributos
0 (25)	0 3 4 6 7 8 9 10 12 13 15 16 17 18 19 20 21 23 24 25 27 30 31 32 33
1 (9)	1 2 5 11 14 22 26 28 29
2 (25)	10 15 16 17 18 19 20 21 23 24 25 27 30 31 33 35 36 37 38 39 40 42 43 45 46
3 (15)	4 6 7 8 12 13 15 16 19 20 24 32 34 35 39
4 (4)	30 31 33 37
5 (39)	15 16 19 20 21 24 25 30 31 33 35 37 39 40 42 43 45 46 48 52 53 55 57 60 62 63 66 67 68 70 71 72 73 74 75 76 78 92 100

Para o conjunto de dados *Zoo* também foi aplicado o `Método Semi-randômico`, atingindo um **Atributo de Qualidade** de **12,892**, representando um índice de acerto de **38,61%**. Este particionamento é ilustrado pela Tabela 30.

Tabela 30 – Particionamento do Conjunto de Dados *Zoo* conforme `Método Semi-randômico`

Partição (N de dados)	Atributos
0 (14)	22 41 43 44 45 49 50 51 53 57 58 59 60 64
1 (14)	79 80 82 83 84 85 86 87 88 89 90 92 93 94
2 (14)	61 61 62 63 67 68 69 70 71 72 75 76 77 91
3 (14)	0 2 3 7 19 20 21 22 23 24 26 27 29 32
4 (14)	18 30 31 40 42 46 47 48 52 54 55 56 65 95
5 (14)	1 4 5 8 10 12 14 16 17 18 25 28 33 38
6 (17)	6 9 11 13 15 29 30 34 35 36 37 39 66 73 74 78 81

Após a obtenção do particionamento gerado pelo `Método Semi-randômico`, o sistema aplicou o `Método do Raio de Cobertura Mínimo`, obtendo o particionamento representado pela Tabela 31. O **Atributo de Qualidade** foi de **10,6774**, melhorando o índice de acerto para **60,39%**. O processo de chamadas do **Módulo Especial de Escalonamento** para os sucessivos refinamentos produziu, nesta base, 3 refinamentos.

Tabela 31 – Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Zoo* a partir do particionamento gerado pelo `Método Semi-randômico`

Partição (N de dados)	Atributos
0 (20)	41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
1 (14)	83 84 85 86 87 88 89 90 92 93 94 95 96 99
2 (20)	61 62 63 66 67 68 69 70 71 72 73 74 75 76 77 78 91 97 98 100
3 (16)	0 2 3 7 18 19 20 21 22 23 24 26 27 30 32 40
4 (6)	64 65 79 80 81 82
5 (15)	1 4 5 6 8 10 12 14 16 17 25 28 31 33 38
6 (10)	9 11 13 15 29 34 35 36 37 39

Ao aplicar o `Método de Consulta por Abrangência`, o melhor particionamento encontrado pelo sistema continuou sendo o do **Módulo de Inicialização**.

Finalmente, a Tabela 32 resume esta análise com a base de dados *Zoo*, apresentando o índice de acerto, conforme repositório em Frank e Asuncion (2010).

Tabela 32 – Análise dos dados da base *Zoo* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento**

Conjunto de Dados **Zoo**					
Módulo de Inicialização		Módulo de Refinamento			
Método	Índice de Acerto	*Raio de Cobertura Mínimo*		*Consulta por Abrangência*	
		Iterações	Índice de Acerto	Iterações	Índice de Acerto
Árvore Binaria	86,13%	1	88,12%	2	64,35%
Randômico	31,68%	1	66,33%	1	43,56%
Semi-Randômico	38,61%	3	60,39%	1	38,61%

As informações apresentadas pela Tabela 33 auxiliam na comparação das alterações realizadas nos valores do **Atributo de Qualidade** com a porcentagem de acerto do Índice, conforme o repositório em Frank e Asuncion (2010).

O sistema de aprendizado testou seus módulos no conjunto de dados *Glass*. Para efeito de análise, os dados foram organizados conforme Tabela 34, de acordo com o repositório Frank e Asuncion (2010).

O **Atributo de Qualidade** aplicado no particionamento referente ao repositório foi **13,7589**.

Para este conjunto de dados, o sistema aplicou, no **Módulo de Inicialização**, os métodos `Randômico` e `Semi-randômico`. O **Atributo de Qualidade** atingido pelo `Método`

Tabela 33 – Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Zoo*

Módulo de Inicialização		*Raio de Cobertura Mínimo*	*Consulta por Abrangência*
Árvore Binária	**9,70902**	**10,1647**	**10,2694**
Randômico	**20,1517**	**11,9582**	—
Semi-Randômico	**12,892**	**10,6774**	—

Tabela 34 – Particionamento do Conjunto de Dados *Glass* conforme repositório Frank e Asuncion (2010)

Partição	Atributos
0 (70)	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
1 (76)	70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
2 (17)	146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
3 (13)	163 164 165 166 167 168 169 170 171 172 173 174 175
4 (9)	176 177 178 179 180 181 182 183 184
5 (29)	185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213

`Randômico` foi **17,9587**. Este valor foi alcançado na iteração de número **2.904.584** em **3.000.000** de iterações solicitadas para o sistema. Neste conjunto de iterações, o sistema identificou **15** melhorias no particionamento. Os dados particionados da iteração com o menor **Atributo de Qualidade** são apresentados na Tabela 35.

O **Módulo de Refinamento** aplicou o `Método do Raio de Cobertura Mínimo` no particionamento gerado pelo `Método Randômico`. O valor do **Atributo de Qualidade** após esta aplicação diminuiu para **14,1058**. A Tabela 36 apresenta o particionamento alterado pelo `Método do Raio de Cobertura Mínimo`.

O valor do **Atributo de Qualidade** no particionamento gerado pelo `Método Semi-randômico` foi **11,8971**. A Tabela 37 apresenta como esses dados foram classificados.

O particionamento gerado pelo `Método Semi-randômico` foi utilizado pelo `Método do Raio de Cobertura Mínimo` do **Módulo de Refinamento**, todavia este processo aumentou o **Atributo de Qualidade** para **12,3607**. A Tabela 38 apresenta a classificação do conjunto de dados *Glass* pelo `Método Semi-randômico` após ser atualizado pelo `Método do Raio de Cobertura Mínimo`.

Tabela 35 – Particionamento do Conjunto de Dados *Glass* gerado pelo `Método Randômico`

Partição	Atributos
0 (35)	2 5 32 34 36 39 42 43 44 53 57 61 62 66 69 74 87 96 97 101 102 105 107 110 116 117 124 134 136 137 149 158 161 176 178
1 (35)	3 6 7 8 11 12 18 37 38 40 41 45 58 63 64 71 89 90 95 103 112 113 118 122 133 142 143 151 154 159 162 165 167 177 204
2 (35)	0 13 16 19 20 21 22 23 24 46 52 65 67 73 75 76 79 84 85 98 99 100 114 115 119 120 123 125 135 138 139 155 182 186 196
3 (35)	9 10 14 15 17 25 26 33 35 48 68 77 78 83 86 88 91 92 104 111 126 127 132 140 141 144 146 147 152 160 170 179 183 197 205
4 (35)	1 4 27 28 29 30 31 47 49 50 51 54 55 56 59 60 70 80 93 94 106 108 109 121 128 145 148 150 153 156 157 163 164 172 181
5 (39)	72 81 82 129 130 131 166 168 169 171 173 174 175 180 184 185 187 188 189 190 191 192 193 194 195 198 199 200 201 202 203 206 207 208 209 210 211 212 213

Tabela 36 – Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Randômico`

Partição	Atributos
0 (7)	105 106 107 110 111 112 131
1 (5)	165 166 167 170 173
2 (117)	0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 19 20 22 23 24 25 26 27 28 29 30 31 32 33 34 35 37 40 41 42 44 55 56 57 58 59 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 113 114 115 116 117 118 119 120 121 122 123 124 125 126 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 152 153 154 155 156 158 159 160 161
3 (39)	8 17 18 21 36 38 39 43 45 46 47 48 49 50 51 52 53 54 60 61 62 63 64 65 66 67 68 69 103 104 108 109 127 128 129 130 151 157 162
4 (14)	164 168 169 175 176 177 178 179 180 182 183 184 187 201
5 (32)	163 171 172 174 181 185 186 188 189 190 191 192 193 194 195 196 197 198 199 200 202 203 204 205 206 207 208 209 210 211 212 213

O `Método de Consulta por Abrangência` também foi aplicado nos particionamentos gerados pelos Métodos `Randômico` e `Semi-Randômico` do **Módulo de Inicialização**. O particionamento encontrado na análise do conjunto de dados gerado pelo `Método`

Tabela 37 – Particionamento do Conjunto de Dados *Glass* gerado pelo `Método Semi-randômico`

Partição	Atributos
0 (37)	163 168 169 179 180 181 182 183 184 185 186 187 188 189 190 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213
1 (34)	0 8 17 18 21 22 34 35 36 38 39 40 42 43 44 45 46 47 48 49 51 52 53 54 60 61 62 63 64 65 66 67 68 69
2 (34)	1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 19 20 23 24 25 26 27 28 29 30 31 33 37 41 56 57 58 59 70
3 (34)	32 55 71 90 95 96 97 102 113 114 115 116 120 124 125 126 132 136 144 145 146 147 148 149 150 151 152 155 156 157 159 160 161 162
4 (35)	72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 91 92 93 94 98 117 118 119 121 122 123 134 137 138 142 153 178
5 (39)	48 50 99 100 101 103 104 105 106 107 108 109 110 111 112 127 128 129 130 131 133 135 139 140 141 143 146 154 164 165 166 167 170 171 172 173 174 175 177

Tabela 38 – Refinamento utilizando o `Método do Raio de Cobertura Mínimo` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Semi-randômico`

Partição	Atributos
0 (35)	176 178 179 180 181 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213
1 (24)	0 17 18 21 36 38 39 43 45 46 47 48 49 50 51 52 60 61 62 63 64 65 68 69
2 (37)	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 19 20 23 24 25 26 27 29 31 32 33 35 42 56 57 58 59 70 78 84 117 119
3 (48)	16 22 28 30 34 37 40 41 44 53 54 55 66 67 71 90 95 96 97 98 99 100 101 102 103 104 113 114 115 116 120 124 125 126 127 128 129 130 132 133 135 136 141 142 144 145 151 157
4 (46)	72 73 74 75 76 77 79 80 81 82 83 85 86 87 88 89 91 92 93 94 118 121 122 123 134 137 138 139 140 143 146 147 148 149 150 152 153 154 155 156 158 159 160 161 162 163
5 (24)	105 106 107 108 109 110 111 112 131 164 165 166 167 168 169 170 171 172 173 174 175 177 182 183

`Randômico` é apresentado pela Tabela 39. Este conjunto foi encontrado na 6ª iteração do processo de refinamento atingindo um **Atributo de Qualidade** de **16,3886**, melhorando **13%** o resultado do particionamento inicial. Este processo foi interrompido com **11** iterações.

O `Método de Consulta por Abrangência`, aplicado no particionamento gerado pelo

Tabela 39 – Refinamento utilizando o `Método de Consulta por Abrangência` no Conjunto de Dados *Glass* a partir do particionamento gerado pelo `Método Randômico`

Partição	Atributos
0 (13)	103 104 105 106 107 108 109 110 111 112 131 170 173
1 (20)	129 130 165 166 167 168 169 175 176 177 178 179 181 182 183 187 190 195 196 208
2 (56)	0 8 13 16 18 19 20 22 23 24 36 38 39 43 45 46 47 49 50 51 52 53 54 60 61 62 63 64 65 66 67 69 73 75 76 79 84 85 98 99 100 114 115 119 120 123 125 127 128 135 138 139 151 155 157 162
3 (94)	1 2 3 4 5 6 7 9 10 11 12 14 15 17 21 25 26 27 28 29 30 31 32 33 34 35 37 40 41 42 44 48 55 56 57 58 59 68 70 71 72 74 77 78 80 81 82 83 86 87 88 89 90 91 92 93 94 95 96 97 101 102 113 116 117 118 121 122 124 126 132 133 134 136 137 140 141 142 143 144 145 146 147 148 149 150 152 153 154 156 158 159 160 161
4 (10)	164 174 184 192 197 200 201 202 204 205
5 (21)	163 171 172 180 185 186 188 189 191 193 194 198 199 203 206 207 209 210 211 212 213

`Método Semi-randômico`, não gerou melhorias no particionamento. A Tabela 40 resume esta análise apresentando o índice de acerto em cada teste.

Tabela 40 – Análise dos dados da base *Glass* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento**

Conjunto de Dados **Glass**					
Módulo de Inicialização		Módulo de Refinamento			
Método	Índice de Acerto	*Raio de Cobertura Mínimo*		*Consulta por Abrangência*	
		Iterações	Índice de Acerto	Iterações	Índice de Acerto
Árvore Binária	50,93%	2	51,40%	2	51,86%
Randômico	28,97%	3	53,73%	6	42%
Semi-Randômico	56,54%	3	60,74%	4	37,38%

As informações apresentadas pela Tabela 41 auxiliam na comparação das alterações realizadas nos valores do **Atributo de Qualidade** com a porcentagem de acerto do Índice, conforme o repositório em Frank e Asuncion (2010).

Tabela 41 – Alteração do **Atributo de Qualidade** conforme aplicação do `Método do Raio de Cobertura Mínimo` e do `Método de Consulta por Abrangência` do **Módulo de Refinamento** nos particionamentos gerados pelo **Módulo de Inicialização** na Base de Dados *Glass*

Módulo de Inicialização		*Raio de Cobertura Mínimo*	*Consulta por Abrangência*
Randômico	**17,9587**	**14,1058**	**16,3886**
Semi-Randômico	**11,8971**	**12,3607**	**13,0107**

Os experimentos realizados na base de dados *Wine*, considerando as informações do Repositório Frank e Asuncion (2010), atingiu o valor do atributo de qualidade de **558,118**.

O uso do `Método da Árvore Binária` obteve um **Atributo de Qualidade** de **403,448**. A Tabela 42 apresenta a forma como foi organizada os dados, de acordo com o repositório, para facilitar a comparação em relação ao sistema.

Tabela 42 – Particionamento do Conjunto de Dados *Wine* conforme Frank e Asuncion (2010)

Partição (N de dados)	Atributos
0 (59)	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
1 (71)	59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129
2 (48)	130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177

A Tabela 43 demonstra como o `Método da Árvore Binária` particionou os dados.

Tabela 43 – Particionamento do Conjunto de Dados *Wine* conforme `Método da Árvore Binária`

Partição (N de dados)	Atributos
0 (42)	0 1 2 6 7 8 9 11 12 13 15 16 17 22 23 26 27 29 30 32 33 34 37 38 41 42 45 46 47 48 49 50 51 52 53 54 55 56 57 58 73 95
1 (6)	3 5 10 14 18 31
2 (130)	4 19 20 21 24 25 28 35 36 39 40 43 44 59 60 61 62 63 64 65 66 67 68 69 70 71 72 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177

A Tabela 44 apresenta, de forma geral, a localização dos dados na classificação obtida.

O sistema organizou os dados da base de dados *Balance* conforme Frank e Asuncion (2010) e estes são apresentados da seguinte forma:

- 1ª Partição: Registro 0 a 48 (49 elementos)
- 2ª Partição: Registro 49 a 336 (288 elementos)
- 3ª Partição: Registro 337 a 624 (288 elementos)

Tabela 44 – Dados nas Partições da Base *Wine* utilizando o `Método da Árvore Binária`

Partição Original	Elementos	Identificados na Partição 1	Identificados na Partição 2	Identificados na Partição 3
1	59	40	6	13
2	71	2	0	69
3	48	0	0	48

O **Módulo de Validação** foi executado nesta distribuição para a obtenção do **Atributo de Qualidade** que apresentou o valor de **10,4785**.

Todos os métodos do **Módulo de Inicialização** foram aplicados nesta base. O `Método Randômico` identificou o melhor particionamento na iteração de número **1.105.293** em **3.000.000** de iterações.

A Tabela 45 apresenta os **Atributos de Qualidade** encontrados para esta base de dados.

Tabela 45 – Valores do **Atributo de Qualidade** referente as análises realizadas no conjunto de dados *Balance*

Módulo de Inicialização		*Raio de Cobertura Mínimo*	*Consulta por Abrangência*
Árvore Binária	**9,48555**	**9,42903**	**10,2209**
Randômico	**11,1819**	**9,44066**	**10,2188**
Semi-Randômico	**9,47514**	**9,40753**	**10,3554**

Por fim, a Tabela 46 informa a porcentagem de acerto do Índice conforme o repositório em Frank e Asuncion (2010). O uso desta base foi importante na análise devido ao seu tamanho em relação as outras bases testadas.

Tabela 46 – Análise dos dados da base *Balance* após particionamentos gerados pelo **Módulo de Inicialização** e tratamento realizado pelo **Módulo de Refinamento**

Conjunto de Dados **Balance**					
Módulo de Inicialização		Módulo de Refinamento			
Método	Índice de Acerto	*Raio de Cobertura Mínimo*		*Consulta por Abrangência*	
		Iterações	Índice de Acerto	Iterações	Índice de Acerto
Árvore Binária	38,46%	3	51,20%	7	41,76%
Randômico	40,70%	1	52%	4	41,44%
Semi-Randômico	39,74%	4	54,08%	6	49,12%

A base de dados *Covertype* foi utilizada para avaliar o desempenho do sistema conforme apresenta a seção 4.3.

4.3 Desempenho do Sistema

Todos os módulos implementados foram simulados sequencialmente e paralelamente para a análise de desempenho do sistema.

Para testar o sistema proposto e verificar os métodos implementados como módulos do sistema, um dos testes foi a avaliação das cinco bases de dados já mencionadas na seção 3.2 (*Iris*, *Zoo*, *Glass*, *Wine* e *Balance*) com um total de **1267** atributos. O tempo de processamento foi determinado considerando a geração das matrizes para cada base, utilizando o cálculo da distância Euclidiana, a aplicação do `Método da Árvore Binária`, do `Método Randômico` utilizando **100.000** interações em todos os cinco conjuntos de dados e do `Método Semi-randômico` utilizando também **100.000** iterações em todos os cinco conjuntos de dados no **Módulo de Inicialização**. Para aplicação do **Módulo de Refinamento** foi utilizado o `Método da Consulta por Abrangência`.

Para avaliar a característica paralela do sistema, foi calculado o *speedup*, o qual mede o fator de redução do tempo de execução em P processadores e, finalmente, a *eficiência*, que é determinada pela relação entre o *speedup* obtido e o número de processadores utilizados (FOSTER, 1995).

A eficiência é afetada por vários fatores, principalmente a sobrecarga na rede, o que torna improvável obter o *speedup* ideal. Existe uma restrição de memória que devido aos métodos implementados manipularem uma matriz cheia, as simulações foram testadas em torno de **10.000** registros, todavia para manipular bases maiores é necessário a implementação de outras técnicas de classificação. A Tabela 47 apresenta o comportamento do sistema por meio do experimento mencionado.

Tabela 47 – Desempenho do Sistema de Aprendizado no *cluster* de computadores utilizando 5 Bases de Dados

Nº de Máquinas	Tempo	**SpeedUp**	**Eficiência**
1	6m37,480s	1	100,00%
2	5m22,899s	1,23	61,55%
4	4m59,0s	1,33	33,23%

A segunda coluna da Tabela 47 ilustra os tempos médios para a execução completa de todas as experiências realizadas em cada teste. As tarefas correspondem à geração de partições iniciais em cada base, a aplicação do **Módulo de Refinamento** em cada partição obtida e a avaliaçäo dos **Atributos de Qualidade**. Neste experimento vale destacar que a execução apenas da base *Balance* em um único processador utiliza-se em média **5m35.280s**. Sendo evidente que, devido à característica do sistema paralelo e da estrutura do escalonador, é possível realizar vários refinamentos dos dados já particionados, enquanto outras tarefas efetuam a primeira partição em outros conjuntos de dados.

A Figura 19 ilustra esse experimento que foi aplicado em 1 Máquina, 2 Máquinas e 4 Máquinas.

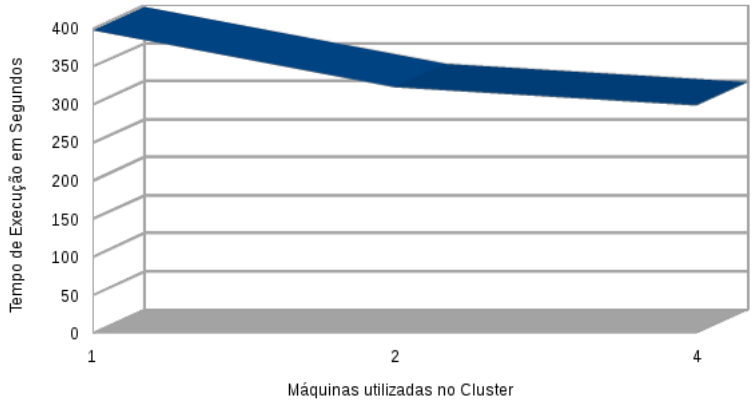

Figura 19 – Experimento analisando as 5 Bases de Dados (*Iris*, *Zoo*, *Glass*, *Wine* e *Balance*) com 1, 2 e 4 Máquinas no Cluster de Computadores

Para fortalecer a contribuição apresentada nesta tese, ilustrando melhor o comportamento do sistema implementado, o mesmo teste foi utilizado multiplicando as bases testadas, isto é, para o sistema, o **Módulo Especial de Escalonamento** escalonou 25 Bases de Dados para classificação. A Tabela 48 mostra o comportamento do sistema manipulando **6.335** atributos.

Tabela 48 – Desempenho do Sistema de Aprendizado no *cluster* de computadores utilizando as 5 Bases de Dados mencionadas multiplicadas por 5

Nº de Máquinas	Tempo	**SpeedUp**	**Eficiência**
1	32m13.849s	1	100,00%
2	16m14.887s	1,98	99,18%
4	9m7.396s	3,53	88,32%

A Figura 20 ilustra esse experimento que foi aplicado em 1 Máquina, 2 Máquinas e 4 Máquinas.

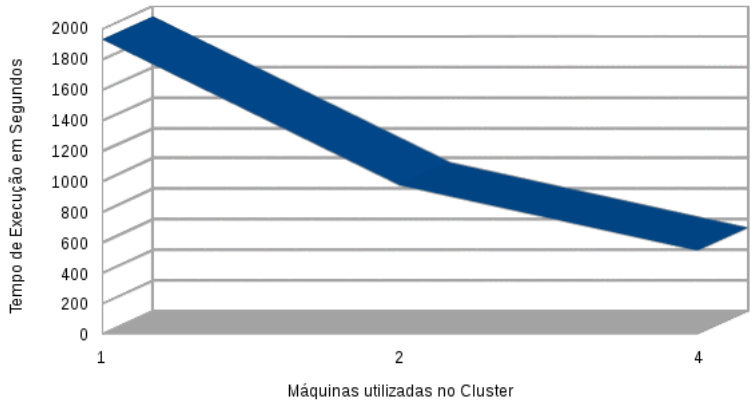

Figura 20 – Experimento analisando as 5 Bases de Dados (*Iris*, *Zoo*, *Glass*, *Wine* e *Balance*) multiplicadas por 5 com 1, 2 e 4 Máquinas no Cluster de Computadores

Para compreender melhor o comportamento do sistema e interpretar como o paralelismo contribui para a estrutura proposta, foi realizado um experimento, criando 7 conjunto de dados através de registros da base *Covertype*. O experimento foi realizado em um único computador com a seguinte característica:

- Processador Intel(R) Core(TM) I5 M450 CPU 2,40GHz;
- Cache L2 3Mb;
- 6 GBytes RAM.

Neste experimento, para os 7 conjuntos de dados, utilizou o `Método da Árvore Binária` no **Módulo de Inicialização** e o `Método da Consulta por Abrangência` e o `Método do Raio de Cobertura` como refinamentos do particionamento obtido dos 7 conjuntos de dados. A Tabela 49 apresenta as informações de tempo gasto na realização deste experimento. Observa-se que normalmente o **Módulo de Inicialização** exige mais processamento para realizar as partições devido as operações realizadas pela matriz de similaridades. Quando o **Módulo Especial de Escalonamento** transfere tarefas para o **Módulo de Refinamento**, as matrizes utilizadas no **Módulo de Refinamento** já estão calculadas. Todavia é intuitivo observar, analisando a Tabela 49, que o paralelismo permitirá o uso de mais métodos em um único experimento mediante ganho computacional.

Tabela 49 – Simulação em uma única máquina de 7 conjunto de dados criados a partir dos registros da base *Covertype*

Registros	**Módulo de Inicialização**	**Módulo de Refinamento**	
	Árvore Binária	**Consulta por Abrangência**	**Raio de Cobertura**
1000	0m10s	0m2s	0m2s
1500	0m35s	0m3s	0m2s
2000	1m22s	0m7s	0m4s
2500	2m37s	0m13s	0m6s
3000	4m35s	0m17s	0m7s
4000	10m38s	0m55s	0m15s
8000	93m17s	1m23s	0m58s

O capítulo 5 apresenta as considerações finais desta tese, verificando os resultados apresentados neste capítulo, bem como as perspectivas de atividades a serem realizadas em trabalhos futuros.

Capítulo 5

Considerações Finais

A principal característica do sistema de aprendizado de máquina apresentado nesta tese é a sua flexibilidade na composição de algoritmos que serão utilizados na classificação de dados. Além disso, através dos arquivos de configuração, é possível alterar facilmente os conjuntos de dados e, principalmente, aproveitar resultados de determinados métodos de classificação e aplicá-los em outros métodos utilizando o **Módulo de Refinamento** na busca de melhores classificações.

Os experimentos realizados no desenvolvimento deste trabalho foram escolhidos para demonstrar a facilidade em usar esta estrutura. Com poucas mudanças, novos métodos podem ser incluídos em cada módulo do sistema. Os testes realizados demonstram o potencial do sistema paralelo, uma vez que há naturalmente uma explosão combinatória de resultados, visto que são utilizados como pontos de partida para novas análises. Através dos critérios de parada estabelecidos e das possibilidades de combinar métodos de classificação distintos, este sistema oferece bons resultados tanto em termos de classificação como em desempenho, ou seja, no tempo de processamento necessário para se atingir classificações com bons **Atributos de Qualidade**.

Os algoritmos de classificação e as bases de dados aqui descritas foram utilizados para demonstrar o funcionamento do sistema e avaliar aspectos relevantes como o seu desempenho. A metodologia utilizada nos respectivos experimentos seguiu, naturalmente, o fluxo de execução que os testes realizados em uma determinada base de dados seguem no sistema, a partir da política de escalonamento imposta pelas filas de prioridades do **Módulo Especial de Escalonamento**.

Nos experimentos apresentados, este sistema mostrou-se eficiente, pois promove a busca por uma boa partição, mesmo quando o **Módulo de Inicialização** gera uma partição de baixa qualidade. O uso do **Módulo de Refinamento** pode corrigir uma má escolha de sementes de inicialização. Esta situação é verificada quando o sistema aplica o **Módulo de Refinamento** nos particionamento obtidos pelo **Método Randômico** no **Módulo de Inicialização**.

Considerando a principal característica do sistema, é possível inferir que ele permite

várias combinações de refinamento. No caso dos testes realizados, os particionamentos gerados inicialmente por cada base, *Iris*, *Zoo*, *Glass* e *Balance*, através dos métodos da **Árvore Binária**, **Randômico** e **Semi-randômico**, foram modificados aplicando os métodos do **Módulo de Refinamento**. Vale destacar que mesmo utilizando o método `Randômico` no **Módulo de Inicialização** o sistema mostrou potencial quando o **Módulo de Refinamento** utilizou o **Método do Raio de Cobertura** e atingiu **88%** de acerto, a partir de um conjunto de dados aleatórios da base de dados *Iris*.

Como discutido, o sistema proposto permite integrar vários métodos disponíveis na literatura e a criação de novas estruturas que explorem o melhor de cada técnica, porém sua maior qualidade é permitir que métodos distintos e em alguns casos divergentes possam ser utilizados de forma a alcançar as melhores características de cada um.

5.1 Perspectivas Futuras

O trabalho apresentado nesta tese permitiu desenvolver uma abordagem na qual a utilização de várias técnicas, de alguma forma relacionadas, possam contribuir nos processos de aprendizagem de máquina. O sistema proposto demonstrou ser eficiente e extensível no processo de avaliação do particionamento. Neste contexto, outros métodos para a definição do **Atributo de Qualidade** podem ser utilizados, permitindo uma análise mais precisa, com a possibilidade de inserir configurações que utilizem algoritmos evolutivos para critérios de parada, os quais podem aperfeiçoar o resultado final do sistema, bem como estudos de técnicas de estabilidade de *cluster* que podem ser introduzidas no sistema com o intuito de melhorar o **Módulo de Validação**, nas análises dos particionamentos obtidos.

Métodos que utilizam critérios de probabilidade para algoritmos de agrupamento hierárquico surgem como alternativas interessantes para implementação e teste no sistema. Conforme Christ, Talavera e Maciel (2007), os métodos Bayesianos obtém algumas vantagens sobre os tradicionais algoritmos aglomerativos, reduzindo as limitações de tempo devido à complexidade da memória, pois utilizam critérios de probabilidade, ao invés de métricas de distância. Exemplos para testes são o método da Força Bruta, algoritmo Naive Bayes ou Classificador Bayesiano Ótimo. O Teorema de Bayes calcula a probabilidade de diferentes hipóteses a medida que novas evidências são analisadas. Pela característica do sistema, estes métodos podem, facilmente, ser incorporados, aumentando as alternativas de configuração e interrelação entre os diversos algoritmos. Além disso, outros métodos podem ser incorporados, tanto no **Módulo de Inicialização** quanto no **Módulo de Refinamento**, verificando também a possibilidade de serem criados a partir da composição das vantagens de cada um dos métodos já disponíveis no sistema.

Critérios para padronizar o nível de abrangência do raio de cobertura no `Método de Consulta por Abrangência` do **Módulo de Refinamento** devem ser bem definidos,

visando melhorar o desempenho do sistema. Neste contexto, o estudo do impacto da abrangência do raio de cobertura nos resultados de cada base permitirá automatizar esta função, melhorando a eficiência e diminuindo a necessidade de interação do usuário durante a execução de um experimento, ou do seu planejamento.

Uma comparação mais criteriosa com outras ferramentas, a ponto de discutir melhorias no sistema proposto, bem como favorecer o apoio ao desenvolvimento de métodos presentes em sistemas de código aberto, deve continuamente ser realizada, o que permitirá a contínua atualização do sistema.

Um ponto ainda a ser tratado é a criação de uma interface de usuário mais amigável, para que os procedimentos de configuração e análise de resultados possam ser feitos com simplicidade e com recursos de manipulação de dados pelos usuários.

A partir da Modelagem dos Dados que compõe a estrutura de armazenamento do sistema, conforme apresentado na seção 3.1.5, será possível avaliar o impacto na utilização de um Sistema de Gerenciamento de Banco de Dados (SGBD) no Sistema Proposto.

Em relação as características do sistema paralelo, um trabalho que pode e deve ser iniciado em sequência é o estudo das adaptações necessárias para que este sistema possa ser instalado em um ambiente de *Cloud Computing*, que se refere, essencialmente, à ideia de utilizar, em qualquer lugar e independente de plataforma, as mais variadas aplicações, recursos de memória, armazenamento, capacidade de processamento de computadores e servidores compartilhados por meio da Internet, da mesma maneira que é utilizado quando instalado em um único computador ou em uma única rede de computadores. Este estudo permitirá identificar as vantagens e desvantagens na utilização desta tecnologia juntamente com a área de aprendizagem de máquina. Em especial, será importante analisar os requisitos para a implantação do sistema em núvens já disponíveis e avaliar questões de disponibilidade, escalabilidade, custo, eficiência, entre outros.

Apesar de ter sido colocado como desvantagem, principalmente em relação a dificuldade que usuários sem muitos conhecimentos em Computação Paralela, Redes de Computadores e Sistemas Distribuídos possuem, a estrutura do sistema proposto permite e abre campo para analisar o comportamento do sistema na execução de métodos classificação paralelizados na forma SRIP (*Single Replication In Parallel*). Para isso, pequenos ajustes devem ser realizados para permitir que o **Módulo Especial de Escalonamento** inicialize tarefas de um mesmo processo em paralelo.

Finalmente, será necessário disponibilizar a ferramenta aos usuários interessados para que testes de usabilidade possam ser realizados, possibilitando que o protótipo desenvolvido se torne uma ferramenta estável e robusta e permitindo que novos métodos de classificação possam ser desenvolvidos e testados com o uso da ferramenta.

Referências

ABREU, N. M. M. Teoria espectral dos grafos: um híbrido entre a álgebra linear e a matemática discreta e combinatória com origens na química quântica. **Trends in Applied and Computacional Mathematics**, 2005. São Carlos, v. 6, n. 1, p. 1–10, 2005.

AGRAWAL, R. et al. **Automatic subspace clustering of high dimensional data for data mining applications**. New York, NY, USA: ACM, jun. 1998. 94–105 p. Disponível em: <http://doi.acm.org/10.1145/276305.276314>.

AITKENHEAD, M. J. A co-evolving decision tree classification method. **Expert Systems with Applications**, 2008. 2008. V. 34, n. 1, p. 18-25.

ALPAYDIN, E. **Introduction to machine learning**. Massachusetts: MIT Press, 2004. Disponível em: <http://books.google.com.br> Acesso em: 01 nov. 2013.

ANDERBERG, M. R. **Cluster analysis for applications**. New York: [s.n.], 1973.

ARAI, K.; BARAKBAH, A. R. Hierarchical k-means: an algorithm for centroids initialization for k-means. **Reports of the Faculty of Science and Engineering**, 2007. v. 36, n. 1, p. 25–31, 2007. Disponível em: <http://dlwww.dl.saga-u.ac.jp/contents/.snapshot/hourly.3/mgzn/ZR00005460/ZR00005460.pdf> Acesso em: 25 jun. 2013.

BARIONI, M. C. N. et al. An efficient approach to scale up k-medoid based algorithms in large databases. **XXI Simpósio Brasileiro de Banco de Dados**, 2006. Florianópolis, p. 265–279, 2006. Disponível em: <http://webdocs.cs.ualberta.ca/ mn/SBBD2006/19-barioni-efficient-approach-sbbd.pdf> Acesso em: 15 mai. 2013.

BARNARD, S. T.; SIMON, H. D. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. **Concurrency: Practice and Experience**, 1994. Wiley Online Library, v. 6, n. 2, p. 101–117, 1994.

BATISTA, G. E. de A. P. A. **Pré- processamento de dados em aprendizado de máquina supervisionado**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-06102003-160219/pt-br.php> Acesso em: 02 fev. 2012.

BENSON-PUTNINS, D. et al. Spectral clustering and visualization: a novel clustering of fisher's iris data set. **SIAM Undergraduate Research Online**, 2011. Philadelphia, v. 4, p. 1–15, 2011. Disponível em: <http://www.siam.org/students/siuro/vol4/S01075.pdf> Acesso em: 01 jan. 2014.

BERNSTEIN, A.; PROVOST, F. An intelligent assistant for the knowledge discovery process. **Information Systems Working Papers Series, Vol**, 2001. New York, 2001. Disponível em: <http://hdl.handle.net/2451/14155> Acesso em: 26 fev. 2013.

BEZDEK, J. C. **Pattern recognition with fuzzy objective function algorithms**. [S.l.]: Kluwer Academic Publishers, 1981.

BILENKO, M.; MOONEY, R. J. Adaptive duplicate detection using learnable string similarity measures. In: ACM. **Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2003. p. 39–48.

BLASHFIELD, R. K. Mixture model tests of cluster analysis: accuracy of four agglomerative hierarchical methods. 1976. Psychological Bulletin, Washington, v. 83, n. 3, p. 377–388, 1976.

BRADLEY, P. S.; FAYYAD, U. M. Refining initial points for k-means clustering. In: **ICML INTERNATIONAL CONFERENCE ON MACHINE LEARNING**. San Francisco: Morgan Kaufmann Publishers, 1998. v. 98, p. 91–99.

BRAUER, J. **A linear-time heuristic for improving network partitions**. University of Wisconsin: [s.n.], 2004. Disponível em: <http://homepages.cae.wisc.edu/ ece556/fall2004/project/brauer.pdf.> Acesso em: 03 mar. 2012.

BUSSAB, W.; MIAZAKI, E. S.; ANDRADE, D. F. **Introdução a análise de agrupamentos**. São Paulo: Associação Brasileira de Estatística, 1990.

CAO, F.; LIANG, J.; BAI, L. A new initialization method for categorical data clustering. **Expert Systems with Applications**, 2009. v. 36, n. 7, p. 10223–10228, 2009.

CARROLL, J. B. **Human cognitive abilities: a survey of factor-analytic studies**. Cambridge: Cambridge University Press, 1993.

CELEBI, M. E.; KINGRAVI, H. A.; VELA, P. A. A comparative study of efficient initialization methods for the k-means clustering algorithm. **Expert Systems with Applications**, 2013. n. 40, p. 200–210, 2013.

CHA, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. **City**, 2007. Citeseer, v. 1, n. 2, p. 1, 2007.

CHATZIDIMITRIOU, K. C.; MITKAS, P. A. Adaptive reservoir computing through evolution and learning. **Neurocomputing**, 2013. Elsevier Science Publishers B. V., Philadelphia, v. 103, p. 198–209, mar 2013. ISSN 0925-2312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2012.09.022> Acesso em: 02 jan. 2011.

CHOI, E.; LEE, C. Feature extraction based on the bhattacharyya distance. **Pattern Recognition**, 2003. Elsevier, v. 36, n. 8, p. 1703–1709, 2003.

CHRIST, R. E. R.; TALAVERA, E. V.; MACIEL, C. D. Gaussian hierarchical bayesian clustering algorithm. **Intelligent Systems Design and Applications**, 2007. Rio de Janeiro, p. 133–137, 2007.

CVETKOVIC, D. M.; DOOB, M.; SACHS, H. **Spectra of graphs: theory and application**. [S.l.]: Academic press New York, 1980.

DAMERAU, F. J. A technique for computer detection and correction of spelling errors. **Communications of the ACM**, 1964. ACM, New York, v. 7, n. 3, p. 171–176, 1964.

DARWIN, C.; BEER, G. **The origin of species**. Oxford University Press, 1998. (Oxford World's Classics Series). ISBN 9780192834386. Disponível em: <http://books-.google.com.br/books?id=LDrPI52uFQsC>.

DAS, S.; SUGANTHAN, P. N. Differential evolution: a survey of the state-of-the-art. **IEEE Trans. Evolutionary Computation**, 2011. v. 15, n. 1, p. 4–31, 2011. Disponível em: <http://dblp.uni-trier.de/db/journals/tec/tec15.html>.

DIETTERICH, T. G. Learning and inductive inference. In: _____. Stanford: Stanford University Department of Computer Science, 1982. cap. 14, p. 323–605.

DORIGO, M.; SOCHA, K. **An introduction to ant colony optimization**. Bruxelles: Univeriste Libre de Bruxelles: [s.n.], 2006. 1–25 p. Disponível em :<http://123seminarsonly.com/Seminar-Reports/034/6948740-An-Introduction-to-Ant-Colony-Optimization.pdf.> Acesso em: 09 oct. 2013.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**. 2. ed. New York: [s.n.], 2001.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973. Journal of Cybernetics, v. 3, n. 3, p. 32–57, 1973.

DZEROSKI, S. et al. Using machine learning techniques in the construction of models. ii. data analysis with rule induction. **Ecological Modelling**, 1997. v. 95, n. 1, p. 95 – 111, 1997. ISSN 0304-3800. Disponível em: <http://www.sciencedirect.com/science-/article/pii/S0304380096000294>.

ECHEVARRIA, M. G. Message passing interface. s.d. s.d. Disponível em: <http://sites.google.com/site/quinho/mpi.pdf> Acesso em: 02 jul. 2013.

EIBEN, A. E.; SMITH, J. E. **Introduction to evolutionary computing**. Berlin: SpringerVerlag, 2003. ISBN 3540401849.

ERLICH, Z.; GELBARD, R.; SPIEGLER, I. Evaluating a positive attribute clustering model for data mining. **Journal of Computer Information**, 2003. v. 3, p. 100–108, 2003.

ESPEJO, P. G.; VENTURA, S.; HERRERA, F. A survey on the application of genetic programming to classification. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, 2010. v. 40, n. 2, p. 121–144, march 2010. ISSN 1094-6977.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, 1996. v. 17, n. 3, p. 37–54, 1996.

FAYYAD, U. M.; PIATESKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery: an overview. **Advances in Knowledge Discovery and Data Mining**, 1996. Cambrige, p. 1–36, 1996.

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. **Design Automation, 1982. 19th Conference on**, 1982. Las Vegas, p. 175–181, June 1982. ISSN 0146-7123.

FIEDLER, M. Algebraic connectivity of graphs. **Czechoslovak Mathematical Journal**, 1973. Institute of Mathematics, Academy of Sciences of the Czech Republic, v. 23, n. 2, p. 298–305, 1973.

FISHER, D. H. Knowledge acquisition via incremental conceptual clustering. 1987. Boston, v. 2, n. 2, p. 139–172, 1987. Disponível em:<http://download.springer.com/static/pdf/769/artAcesso em: 13 nov. 2010.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of Eugenics**, 1936. v. 7, p. 179–188, 1936.

FORGY, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. **Biometrics**, 1965. v. 21, p. 768–780, 1965.

FOSTER, I. Designing and building parallel programs: concepts and tools for parallel software engineering. mathematics and computer science division - argonne national laboratory. 1995. 1995. Disponível em: <http://www-unix.mcs.anl.gov/dbpp/text/node4.html> Acesso em: 30 dec. 2012.

FRANK, A.; ASUNCION, A. **UCI- machine learning repository**. 2010. Disponível em: <http://archive.ics.uci.edu/ml>.

FRIEDL, M. A.; BRODLEY, C. E. Decision tree classification of land cover from remotely sensed data. **Remote sensing of environment**, 1997. Elsevier, v. 61, n. 3, p. 399–409, 1997.

FRIEDMAN, H. P.; RUBIN, J. On some invariant criteria for grouping data. **American Statistical Association Journal**, 1967. v. 62, p. 1159–1178, 12 1967.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability. A guide to the theory of NP-completeness. A series of books in the mathematical sciences**. [S.l.]: WH Freeman and Company, San Francisco, Calif, 1979.

GEIST, G. A.; KOHL, J. A.; PAPADOPOULOS, P. M. Pvm and mpi: a comparison of features. **Calculateurs Paralleles**, 1996. v. 8, n. 2, p. 137–150, 1996.

GELBARD, R.; GOLDMAN, O.; SPIEGLER, I. Investigating diversity of clustering methods: an empirical comparison. **Data Knowl. Eng.**, 2007. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 63, n. 1, p. 155–166, oct 2007. ISSN 0169-023X. Disponível em: <http://dx.doi.org/10.1016/j.datak.2007.01-.002>.

GIOTIS, I.; PETKOV, N. Cluster-based adaptive metric classification. **Neurocomputing**, 2012. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 81, p. 33–40, apr 2012. ISSN 0925-2312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2011.10.018>.

GOUDAIL, F.; RÉFRÉGIER, P.; DELYON, G. Bhattacharyya distance as a contrast parameter for statistical processing of noisy optical images. **JOSA A**, 2004. Optical Society of America, v. 21, n. 7, p. 1231–1240, 2004.

GREENSPAN, M. A decision tree classifier for object recognition in range imagery. In: **Proceedings of vision interface 1998**. [S.l.: s.n.], 1998. p. 181–188.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. **Journal of Intelligent Information Systems**, 2001. Springer, v. 17, n. 2-3, p. 107–145, 2001.

HALL, K. M. An r-dimensional quadratic placement algorithm. **Management science**, 1970. INFORMS, v. 17, n. 3, p. 219–229, 1970.

HAMMING, R. W. Error detecting and error correcting codes. **BELL SYSTEM TECHNICAL JOURNAL**, 1950. AMER TELEPHONE TELEGRAPH CO, NEW YORK, v. 29, n. 2, p. 147–160, 1950. ISSN 0005-8580.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. [S.l.]: Morgan kaufmann, 2006.

HANDL, J.; KNOWLES, J.; DORIGO, M. Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. 2003. 2003.

HASAN, A.; JAFAR, M.; RAMAKRISHNAN, S. A survey: hybrid evolutionary algorithms for cluster analysis. **Artif. Intell. Rev.**, 2011. Kluwer Academic Publishers, Norwell, MA, USA, v. 36, n. 3, p. 179–204, oct 2011. ISSN 0269-2821. Disponível em: <http://dx.doi.org/10.1007/s10462-011-9210-5>.

HOLSHEIMER, M.; SIEBES, A. **Data mining: The search for knowledge in databases**. [S.l.]: CWI Amsterdam, 1994.

HOLZRICHTER, M.; OLIVEIRA, S. A graph based method for generating the fiedler vector of irregular problems. In: **Parallel and Distributed Processing**. [S.l.]: Springer, 1999. p. 978–985.

HUAZHONG, N. et al. Incremental spectral clustering by efficiently updating the eigen-system. **Pattern Recognition**, 2010. p. 113–127, 2010.

HUBERT, L. Approximate evaluation techniques for the single-link and complete-link hierarchical clustering procedures. **Journal of the American Statistical Association**, 1974. v. 69, n. 347, p. 698–704, 1974.

JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. Prentice Hall, 1988. (Prentice Hall advanced reference series). ISBN 9780130222787. Disponível em: <http://books.google.com.br/books?id=7eBQAAAAMAAJ>.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM computing surveys (CSUR)**, 1999. ACM, v. 31, n. 3, p. 264–323, 1999.

JARO, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. **Journal of the American Statistical Association**, 1989. Taylor & Francis, v. 84, n. 406, p. 414–420, 1989.

_____. Probabilistic linkage of large public health data files. **Statistics in medicine**, 1995. Wiley Online Library, v. 14, n. 5–7, p. 491–498, 1995.

JENSEN, K.; ARNSPANG, J. Binary decision tree classification of musical sounds. In: **Proceedings of the 1999 International Computer Music Conference**. [S.l.: s.n.], 1999.

JERONYMO, D. C.; BORGES, Y. C. C.; COELHO, L. dos S. Algoritmo genético e otimização por enxame de partículas aplicados ao planejamento de trajetória de um manipulador robótico. In: XVIII CONGRESSO BRASILEIRO DE AUTOMáTICA. Bonito-MS, 2010. p. 3308 – 3315.

KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. **SIAM Journal on scientific Computing**, 1998. SIAM, v. 20, n. 1, p. 359–392, 1998.

KAUFMAN, L.; ROUSSEEUW, P. J. **Finding groups in data: an introduction to cluster analysis**. [S.l.]: Wiley. com, 2009.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Neural Networks, 1995. Proceedings., IEEE International Conference on**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4.

KERNIGHAN, B.; LIN, S. An efficient heuristic procedure for partitioning graphs. **The Bell System Technical Journal**, 1970. v. 2, n. 49, 1970.

KINNUNEN, T. et al. Comparison of clustering methods: a case study of text-independent speaker modeling. **Pattern Recognition Letters**, 2011. n. 32, p. 1604–1617, 2011.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. In: **Soviet physics doklady**. [S.l.: s.n.], 1966. v. 10, p. 707.

LIN, D. An information-theoretic definition of similarity. In: **ICML**. [S.l.: s.n.], 1998. v. 98, p. 296–304.

LIN, Y. et al. Design and implementation of a biomass temporal-spatial variation simulate system. **Journal of Applied Sciences**, 2013. v. 13, n. 17, p. 3439 – 3445, 2013. ISSN 18125654.

LINTZMAYER, C. N.; MULATI, M. H.; SILVA, A. F. da. Rt-colorant: um algoritmo heurístico baseado em colônia de formigas artificiais com busca local para colorir grafos. In: **XLIII Simposio Brasileiro de Pesquisa Operacional 2011**. [S.l.: s.n.], 2011.

LIU, Y.; WU, X.; SHEN, Y. Automatic clustering using genetic algorithms. **Applied Mathematics and Computation**, 2011. n. 218, p. 1267–1279, 2011.

LLOYD, S. Least squares quantization in PCM. **IEEE Transactions on Information Theory**, 1982. v. 28, n. 2, p. 129–137, 1982.

LUGER, G. F. **Inteligência artificial: estruturas e estratégias para a solução de problemas complexos**. [S.l.]: Grupo A, 2004.

MACIEL, C. D.; MILAGRE, S. T. Performance analysis of restriction enzymes based in results of cluster stability. **Ciência e Engenharia, América do Norte**, 2008. 2008. Disponível em: <http://www.seer.ufu.br/index.php/cieng/article/view/654/628>. Acesso em: 08 de mar. 2011.

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: CAM, L. M. L.; NEYMAN, J. (Ed.). **Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability**. [S.l.]: University of California Press, 1967. v. 1, p. 281–297.

MADDAH, M. et al. A unified framework for clustering and quantitative analysis of white matter fiber tracts. **Medical Image Analysis**, 2008. p. 191–202, 2008.

MAK, B.; BARNARD, E. Phone clustering using the bhattacharyya distance. In: IEE INTERNATIONAL CONFERENCE ON INDUSTRIAL TECHNOLOGY. Philadelphia: INTERNATIONAL CONFERENCE SPOKEN LANGUAGE, 1996. v. 4, p. 2005–2008.

MARKEY, M. K.; TOURASSI, G. D.; FLOYD, C. E. Decision tree classification of proteins identified by mass spectrometry of blood serum samples from people with and without lung cancer. **Proteomics**, 2003. Wiley Online Library, v. 3, n. 9, p. 1678–1679, 2003.

MEDEIROS, J. A. C. C. **Enxame de partículas como ferramenta de otimização em problemas complexos de engenharia nuclear**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2005.

MELLO, C. E. R. de; SILVA, G. Z. da; SOUZA, J. M. de. Extensão do weka para métodos de agrupamento com restrição de contigüidade. In: **IX Brazilian Symposium on GeoInformatics**. Campos do Jordão, Brasil: [s.n.], 2007. p. 277–282.

MENDES, A. J.; JR., C. N. C. **Algoritmos aleatórios para particionamento de circuitos integrados**. [S.l.], 2006. Disponível em: <http://homepages.dcc.ufmg.br/ nivio/cursos/pa06/seminarios/seminario2/seminario2.pdf>.Acesso em: 29 jun. 2013.

MICHALSKI, R. S. A theory and methodology of inductive learning. **Artificial Intelligence**, 1983. v. 20, n. 2, p. 111 – 161, 1983. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/0004370283900164>.

MILAGRE, S. T. et al. Fuzzy cluster stability analysis with missing values using resampling. **International Journal of Bioinformatics Research and Applications**, 2009. v. 5, n. 2, p. 207–223, 2009.

MIRKES, E. M. **K-means and k-medoids applet**. 2011. University of Leicester. Disponível em: <http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids-/Kmeans_Kmedoids.html>.

MITCHELL, T. **Machine learning**. Boston: [s.n.], 1997.

MITCHELL, T. M. Machine learning and data mining. **Commun. ACM**, 1999. ACM, New York, NY, USA, v. 42, n. 11, p. 30–36, nov. 1999. ISSN 0001-0782. <http://doi.acm.org/10.1145/319382.319388>.

MJOLSNESS, E. et al. Machine learning for science: state of the art and future prospects. **Science**, 2001. v. 293, n. 5537, p. 2051–2055, 2001.

MONGE, A. E.; ELKAN, C. et al. The field matching problem: algorithms and applications. In: **KDD**. [S.l.: s.n.], 1996. p. 267–270.

MONGE, A. E.; ELKAN, C. P. Efficient domain-independent detection of approximately duplicate database records. **Department of Computer Science and Engineering**, 1997. p. 92093–0114, 1997.

NASCIMENTO, M. C. V. **Metaheurísticas para o problema de agrupamento de dados em grafo**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010. Disponível em: <http://mdereg1-fercarpetass.googlecode.com/svn/trunk/ComputaciAcesso em: 02 fev. 2012.

NASSIFA, L. F. da C.; HRUSCHKAB, E. R. Computação forense via agrupamento hierárquico de documentos. In: (ICOFCS), T. I. C. O. F. C. S. (Ed.). Brasília, 2011. v. 6, p. 166–172. Disponível em: < http://www.icofcs.org/2011/ICoFCS2011-PP19.pdf>. Acesso em: 05 jul. 2013.

NETO, D. M. **Efficient cluster compensation for Lin-Kernighan heuristics**. Tese (Doutorado) — University of Toronto, 1999.

NG, R. T.; HAN, J. Clarans: a method for clustering objects for spatial data mining. **Knowledge and Data Engineering, IEEE Transactions on**, 2002. IEEE, v. 14, n. 5, p. 1003–1016, 2002.

NILSSON, N. J. **Introduction to machine learning**. [S.l.: s.n.], 1998.

PAVLO, A. et al. A comparison of approaches to large-scale data analysis. In: **Proceedings of the 2009 ACM SIGMOD International Conference on Management of data**. New York, NY, USA: ACM, 2009. (SIGMOD '09), p. 165–178. ISBN 978-1-60558-551-2. Disponível em: <http://doi.acm.org/10.1145/1559845-.1559865>.

PHUA, C.; LEE, V.; SMITH-MILES, K. The personal name problem and a recommended data mining solution. **Encyclopedia of Data Warehousing and Mining**, 2006. Citeseer, 2006.

POLI, R.; LANGDON, W. B.; MCPHEE, N. F. **A field guide to genetic programming**. [S.l.: s.n.], 2008.

PONS, S. V.; SHULCLOPER, J. R. A survey of clustering ensemble algorithms. **International Journal of Pattern Recognition and Artificial Intelligence**, 2011. 2011.

PRASARTVIT, T. et al. Reducing bioinformatics data dimension with abc-knn. **Neurocomputing**, 2012. 2012.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [S.l.]: Morgan kaufmann, 1993.

REYES-ALDASORO, C. C.; BHALERAO, A. The bhattacharyya space for feature selection and its application to texture segmentation. **Pattern Recognition**, 2006. Elsevier, v. 39, n. 5, p. 812–826, 2006.

RIBEIRO, C. H. C.; GABRIELLI, L. H. Aprendizagem por reforço para times de robôs. 2003. 2003.

ROCHA, T. et al. **Tutorial sobre fuzzy-c-means e fuzzy learning vector quantization: abordagens híbridas para tarefas de agrupamento e classificação**. [S.l.], 2012.

ROGERS, D. J.; TANIMOTO, T. T. A computer program for classifying plants. **Science**, 1960. American Association for the Advancement of Science, v. 132, n. 3434, p. 1115–1118, 1960.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Prentice hall, 2002.

SALZBERG, S. Locating protein coding regions in human dna using a decision tree algorithm. **Journal of Computational Biology**, 1995. v. 2, n. 3, p. 473–485, 1995.

SALZBERG, S. et al. A decision tree system for finding genes in dna. **Journal of Computational Biology**, 1998. v. 5, n. 4, p. 667–680, 1998.

SCHERSCHEL, F. **Linux continues to dominate list of top 500 supercomputers**. June 2012. The H Open. Http://www.h-online.com/open/news/item/Linux-continues-to-dominate-list-of-top-500-supercomputers-1618155.html.

SHAPIRO, G. P. Knowledge discovery in real databases. In: REPORT ON THE IJCAI-89 WORKSHOP. [S.l.]: AI Magazine, 1991. v. 5, n. 11, p. 68–70.

SHENG, W.; LIU, X. A genetic k-medoids clustering algorithm. **Journal of Heuristics**, 2006. v. 12, n. 6, p. 447–466, 2006.

SILBERCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Sistemas operacionais: conceitos e aplicações**. Rio de Janeiro: Elsevier: Campus, 2001.

SILBERSCHATZ, A. et al. **Sistema de banco de dados**. Rio de Janeiro: Elsevier: Campus, 2006.

SINKY, H. et al. **Parallelism in machine learning: a survey on machine learning techniques related to computer architecture**. 2014. Disponível em: <http://web.engr.oregonstate.edu/ sinkyha/ECE570/ece570.project.report.pdf> Acesso em: 11 sep. 2013.

SOUZA, G. M.; NOVA, N. A. V.; GONÇALVES, A. N. Entropy, information, and water stress in eucaliptus camadulensis in vitro. **Revista Brasileira de Biologia**, 1999. SciELO Brasil, v. 59, n. 3, p. 471–476, 1999.

SPEARS, W. M. et al. **An overview of evolutionary computation**. 1993.

SURAL, S.; QIAN, G.; PRAMANIK, S. A comparative analysis of two distance measures in color image databases. In: **ICIP (1)**. [S.l.: s.n.], 2002. p. 401–404.

TAN, P. N.; STEINBACH, M.; KUMAR, V. **Introduction to data mining**. [S.l.]: Pearson College Division, 2013. ISBN 9780133128901.

TSAI, A. et al. Fiber tract clustering on manifolds with dual rooted-graphs. In: **Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on**. [S.l.: s.n.], 2007. p. 1–6. ISSN 1063-6919.

TSENG, F. S. C.; CHEN, P.-Y. Parallel association rule mining by data de-clustering to support grid computing. **PACIS 2005 Proceedings**, 2005. p. 89, 2005. Disponível em: <http://aisel.aisnet.org/pacis2005/89> Acesso em: 02 abr. 2013.

VELMURUGAN, T.; SANTHANAM, T. Performance analysis of k-means and k-medoids clustering algorithms for a randomly generated data set. In: **Proceedings of the International Conference on Systemics, Cybernetics and Informatics, Ian**. [S.l.: s.n.], 2008. v. 8, p. 578–583.

_____. Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. **Journal of Computer Science**, 2010. p. 363–368, 2010.

VIMAL, A.; VALLURI, S. R.; KARLAPALEM, K. An experiment with distance measures for clustering. In: **COMAD'08**. [S.l.: s.n.], 2008. p. 241–244.

WAINTRAUB, M. **Algoritmos paralelos de otimização por enxame de partículas em problemas nucleares**. Tese (Doutorado) — Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

WINKLER, W. E. The state of record linkage and current research problems. In: CITESEER. **Statistical Research Division, US Census Bureau**. [S.l.], 1999.

WU, X. et al. Top 10 algorithms in data mining. **Knowledge and Information Systems**, 2008. London, v. 14, n. 1, p. 1–37, jan. 2008. ISSN 0219-1377. Disponível em: <http://dx.doi.org/10.1007/s10115-007-0114-2>.

XUTAO, L. et al. On cluster tree for nested and multi-density data clustering. **Pattern Recognition**, 2010. p. 3130–3143, 2010.

YAN, W. et al. p-pic: parallel power iteration clustering for big data. **J. Parallel Distrib. Comput.**, 2012. 2012.

YEN, G. G. Learning and intelligence. **IEEE Comput. Intell. Mag.**, 2009. v. 5, n. 2, p. 2, 2009.

YOU, C. H.; LEE, K. A.; LI, H. An svm kernel with gmm-supervector based on the bhattacharyya distance for speaker recognition. **Signal Processing Letters, IEEE**, 2009. IEEE, v. 16, n. 1, p. 49–52, 2009.

ZAHRAIE, B.; ROOZBAHANI, A. Sst clustering for winter precipitation prediction in southeast of iran: comparison between modified k-means and genetic algorithm-based clustering methods. **Expert Systems with Applications**, 2011. n. 38, p. 5919–5929, 2011.

ZHANG, J. et al. Evolutionary computation meets machine learning: a survey. **IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE**, 2011. p. 68–75, 2011.

ZMAZEK, B. et al. Application of decision trees to the analysis of soil radon data for earthquake prediction. **Applied Radiation and Isotopes**, 2003. Elsevier, v. 58, n. 6, p. 697–706, 2003.

ZURADA, J. M. et al. Building virtual community in computational intelligence and machine learning [research frontier]. **IEEE Comp. Int. Mag.**, 2009. v. 4, n. 1, p. 43–54, 2009.

ZWICK, R.; CARISTEIN, E.; BUDESCU, D. V. Measures of similarity among fuzzy concepts: a comparative analysis. **International Journal of Approximate Reasoning**, 1987. v. 1, p. 221–242, 1987.