UNIVERSIDADE ESTADUAL DE MONTES CLAROS

Centro de Ciências Exatas e Tecnológicas

Programa de Pós-Graduação Modelagem Computacional e Sistemas

Thiago Silva Prates

Proposta de um Algoritmo Baseado em Particle Swarm Optimization (PSO) para Classificação de Dados

Montes Claros - MG

Abril de 2017

Thiago Silva Prates

Proposta de um Algoritmo Baseado em Particle Swarm Optimization (PSO) para Classificação de Dados

Dissertação submetida à banca avaliadora designada pelo Colegiado do Programa de Pós-Graduação em Modelagem Computacional e Sistemas da Universidade Estadual de Montes Claros, como parte dos requisitos exigidos para a obtenção do título de Mestre em Modelagem Computacional e Sistemas.

Orientador: Prof. Dr. João Batista Mendes
Coorientador: Prof. Dr. Renato Dourado Maia

Montes Claros - MG

Abril de 2017

Agradecimentos

Ao meu orientador Prof. Dr. João Batista Mendes pela sabedoria e amizade durante as atividades e discussões sobre do andamento deste trabalho.

Ao meu coorientador Prof. Dr. Renato Dourado Maia pelos ensinamentos.

Aos professores do PPGMCS pelos ensinamentos ao ministrar as disciplinas, que foram muito importantes para a realização desta pesquisa.

Aos colegas pelo suporte e amizade.

Ao meu pai Manoel, minha mãe Aldenice e meu irmão Diego pela apoio e amor dedicado para vencer este momento tão importante em minha vida.

A Mariana pelo amor e compreensão dedicada, pois, sem dúvida, boa parte dos méritos conquistados devo a ela.

E, finalmente, a meu irmão Lucas que nos deixou cedo demais e sempre torceu pelo meu sucesso.

Obrigado a todos!

Resumo

A mineração de dados tem se tornado um importante fator estratégico dentro do ambiente empresarial, uma vez que permite que informações não triviais sejam identificadas de forma a alavancar os negócios ou mesmo promover inovações de produtos e serviços. Neste contexto, a classificação de dados é reconhecida como uma das mais importantes, e recorrentes, tópicos encontrados na mineração de dados. Dessa maneira, o seguinte trabalho tem por objetivo propor um algoritmo multiobjetivo baseado em PSO para classificação de dados por meio de extração de regras. Nesta abordagem, cada partícula (solução candidata) representa uma regra de classificação, que é convertida em um predicado lógico na construção de uma operação de seleção na linguagem SQL para avaliação de desempenho do algoritmo. Durante a realização de experimentos computacionais, foi observado que a abordagem proposta se mostrou competitiva, com resultados promissores quando comparados à outros métodos de classificação clássicos, reconhecidos na literatura, destacando-se especialmente em bases de dados desabalanceadas.

Palavras-chave: Otimização por Enxame de Partículas, Inteligência de Enxames, Abordagem Multiobjetivo, Extração de Regras e Classificação de Dados.

Abstract

Data mining has become an important strategic factor within the enterprise environment as it allows non-trivial information to be identified in order to leverage business or even promote product and service innovations. In this context, data classification is one of the most important and recurring topics encountered in data mining and in this way the following work aims to propose a multiobjective algorithm based on PSO for data classification through rule extraction. In this approach, each particle (candidate solution) represents a classification rule that is converted into a logical predicate for subsequent use in a selection operation in the SQL language for performance evaluation of the algorithm. In experiments carried out, the proposed approach proved to be competitive, with promising results when compared to other classical classification methods recognized in the literature and to present satisfactory results especially in unbalanced datasets.

Palavras-chave: Particle Swarm Optimization, Swarm Intelligence, Multi-objective Approach, Rule Mining and Data Classification.

Sumário

Lista de abreviaturas e siglas

ACO	*Ant Colony Optimization* ou Otimização por Colônia de Formigas
ANN	*Artificial Neural Network* ou Redes Neurais Artificial
BPI	*Best Pareto Improvement*
DGA	*Digital Gas Analysis*
DPSO	*Discrete Particle Swarm Optimization*
FN	Falso Negativo
FP	Falso Positivo
FPI	*First Pareto Improvement*
GPL	*GNU General Public License*
KDD	*Knowledge Discovery in Database*
NP	*Non-Deterministic Polynomial time*
NPI	*Neutral Pareto Improvement*
PG	Programação Genética
PLS	*Pareto Local Search* ou busca local Pareto
PSO	*Particle Swarm Optimization* ou Otimização por Enxame de Nuvens
RBF	*Radial Basis Function*
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
SMO	*Sequential Minimal Optimization*
SVM	*Support Vector Machine*
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
WEKA	*Waikato Environment for Knowledge Analysis*

Lista de símbolos

Ω	Espaço de Decisão
$\mathcal{P}$	Conjunto Pareto-Ótimo
$\mathcal{PF}$	Fronteira Pareto-Ótima
$\mathbb{R}$	Conjunto dos Números Reais
S	Conjunto das Soluções Factíveis

Lista de ilustrações

Lista de tabelas

1 Introdução

A classificação de dados tem-se firmado cada vez mais como um dos tópicos mais relevantes dentro da área de mineração de dados. Atualmente, diversos métodos são utilizados na tarefa de classificação de dados tais como Árvores de Decisão, as Redes Neurais Artificiais, os Algoritmos Evolucionários, dentre outros.

A Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) é um destes métodos que vem recebendo reconhecimento dentro da área científica devido a sua eficiência e custo-benefício quando comparados à outros métodos computacionais encotrados na literatura. Dessa forma, nada mais natural que tentar transportar sua eficácia para a área de mineração de dados.

Com o intuito de tentar otimizar o desempenho do PSO proposto foram introduzidos um método de geração do enxame inicial e mecanismos de busca local Pareto para refinamento das soluções encontradas.

Foram realizados experimentos em 07 (sete) bases de dados variadas, sendo comparados os resultados obtidos do PSO proposto com outros algoritmos clássicos, implementados pela ferramenta de mineração de dados WEKA (*Waikato Environment for Knowledge Analysis*)[1]. Ao final, análises estatísticas foram realizadas para comparação de desempenho com cada método utilizado em relação as bases de dados selecionadas.

1.1 Publicações

Prates, T. S.; Mendes, J. B.; D'Angelo M. F. S. V.; Lacerda, A. S.; Maia, R. D.; Rodrigues, R. V. Proposta de PSO multiobjetivo para classificação de dados. In: *XIX Encontro Nacional de Modelagem Computacional e VII Encontro de Ciência e Tecnologia de Materiais*, 2016, João Pessoa-PB.

1.2 Objetivos do Trabalho

O principal objetivo deste trabalho é propor um algoritmo multiobjetivo baseado em PSO para classificação de dados por meio de extração de regras de classificação.

Os objetivos específicos são listados à seguir:

[1] Disponível em: http://www.cs.waikato.ac.nz/ml/weka/. Acesso em Março de 2016.

- Analisar o problema de classificação de dados na ótica da Computação Evolucionária;
- Estudar Algoritmos Evolucionários Multiobjetivo destinados a problemas de classificação de dados;
- Identificar os principais trabalhos na literatura que tratam do problema de classificação de dados utilizando o PSO;
- Desenvolver um algoritmo multiobjetivo para classificação de dados baseado em PSO;
- Estudar técnicas de busca local do tipo *Pareto Local Search* (PLS) para refinamento das soluções;
- Analisar os mecanismo de PLS para aplicação no problema de classificação de dados;
- Realizar comparações estatísticas entre o PSO proposto e outros métodos e técnicas clássicadas encontradas na literatura.

1.3 Metodologia

Este trabalho tem o objetivo de proporcionar uma investigação acerca da proposta de um algoritmo multiobjetivo baseado em PSO para classificação de dados. Dessa maneira, consistiu basicamente na realização de pesquisas bibliográficas sobre o tema e experimentações em laboratório.

Para tanto, foram utilizados o *software* de mineração de dados *WEKA*, na versão 3.6.6, que serviu de ferramenta de apoio para testes e análises posteriores. O WEKA é uma coleção dos mais reconhecidos algoritmos e ferramentas computacionais encontrados na literatura destinados à mineração de dados. Foi desenvolvido pela Universidade de Waikato na Nova Zelândia e escrito na linguagem de programação Java sob os termos GPL[2] (WITTEN; EIBEN, 2005).

O algoritmo proposto e o WEKA foram utilizados em conjunto com o banco de dados *PostgreSQL*, uma vez que as bases de dados para testes foram portadas para este Sistema de Gerenciamento de Banco de Dados (SGBD) com o objetivo de facilitar a implementação.

[2] GNU General Public License

Foram utilizadas 07 (sete) bases de dados para os experimentos. São as bases *Hepatitis* e *Wine* do repositório *UCI Machine Learning*[3] em conjunto com as bases *Diabetes*, *Ionosphere* e *Unbalaced* presentes, como bases de testes do WEKA. Também são utilizadas bases reais de análise de concentração de gases (DGA, *Digital Gas Analysis*) em transformadores elétricos com o objetivo de classificar seu respectivo estado: *Normal*, *Falha Elétrica* e *Falha Térmica*, em duas versões: uma (i) balanceada e outra (ii) desbalanceada.

Com o intuito de subsidiar os resultados apresentados, foram também utilizadas técnicas estatísticas por meio da linguagem R com a finalidade de expressar as devidas análises e conclusões.

1.4 Organização do Trabalho

Este trabalho é organizado em 5 capítulos. No capítulo 1 é apresentado uma introdução acerca do trabalho, onde são discutindos o tema e os objetivos, geral e específicos, bem como a metodologia utilizada durante seu desenvolvimento.

Posteriormente, no capítulo 2 é realizada uma revisão bibliográfica referente a terminologia e principais conceitos relacionados ao PSO, bem como conceitos relativos ao PSO destinado à ambientes combinatoriais. Conceitos importantes referentes à otimização multiobjetivo, busca local Pareto e as áreas de mineração e classificação de dados utilizadas durante a implementação são também tratados. Ao final do capítulo são elencados trabalhos encontrados que discutem o tema estudado.

No capítulo 3 é apresentado o algoritmo proposto (mDPSO), uma variante multiobjetivo baseada em PSO para classificação de dados por meio de extração de regras, sendo pontuados funcionalidades e estratégias utilizadas durante sua implementação.

O capítulo 4 apresenta os experimentos realizados, relacionando os resultados obtidos pelo algoritmo proposto em comparação com outros 03 (três) algoritmos, implementados pela ferramenta WEKA, para bases de dados previamente selecionadas.

Finalmente, no capítulo 5 são apresentadas as considerações finais, assim como propostas para trabalhos futuros.

[3] Disponível em: http://archive.ics.uci.edu/ml/. Acesso em Março de 2016.

2 Fundamentação Teórica

Neste capítulo é apresentada a fundamentação teórica utilizada no desenvolvimento do presente trabalho. Numa primeira etapa, são apresentados os principais conceitos relacionados a metaheurística *Particle Swarm Optimization* (PSO) necessários à compreensão da abordagem proposta.

Posteriormente, conceitos e definições da área de otimização multiobjetivo serão apresentados, uma vez que são importantes durante a utilização da dominância Pareto pelos componentes *gbest* e *pbest* do PSO proposto. Conceitos referentes à busca local Pareto também são relacionados, pois são utilizados ao final de cada iteração com intuito de refinar as soluções encontras.

Ao final do capítulo são relacionados os principais conceitos e definições referentes às áreas de mineração e classificação de dados, sendo elencados trabalhos relevantes referentes ao tema abordado deste trabalho.

2.1 Particle Swarm Optimization

O PSO é uma metaheurística relacionada à área dos Algoritmos Evolucionários e inspirada na dinâmica da vida social de pássaros e cardumes de peixes durante a busca por alimentos (KENNEDY; EBERHART, 1995).

De acordo Cunha, Takahashi e Antunes (2014), a busca por alimentos efetuada de maneira coordenada por sociedades de animais envolve a realização de buscas em espaços de elevada dimensão, sobre funções que não apenas são de elevada complexidade, mas que também exigem uma execução contínua de mecanismos de adaptação. As soluções para tais problemas tratam-se de mecanismos bastante bem sucedidos, utilizadas pelos atuais seres biológicos existentes no planeta, pois permitiram a essas espécies estar hoje presentes após bilhões de anos. Devido a isso, acredita-se que ao emular tais processos, possa-se transpostar o sucesso destes processos na construção de mecanismos computacionais destinados a resolução de problemas de alta complexidade.

Relacionada com os Algoritmos Evolucionários encontra-se a sub-área chamada de *Inteligência de Enxame* que inclui qualquer tentativa de construir algoritmos e técnicas inspiradas no comportamento coletivo de colônias de insetos e outras sociedades animais. Segundo Millonas (1993), cinco princípios regem esses algoritmos:

(i) **Proximidade**: Os agentes do enxame devem ser capazes de interagir, de modo a formarem redes sociais.

(ii) **Qualidade**: Os agentes devem ser capazes de avaliar os seus comportamentos, isto é, as suas interações com o ambiente e com os demais indivíduos;

(iii) **Diversidade (de comportamentos)**: Esse princípio permite ao sistema reagir a situações desconhecidas ou inesperadas;

(iv) **Estabilidade**: Os agentes não devem alterar os seus comportamentos em resposta a qualquer variação ambiental;

(v) **Adaptabilidade**: Deve existir a capacidade de adaptação às variações ambientais e populacionais.

O termo "enxame" é utilizado de maneira genérica para se referir a qualquer coleção estruturada de agentes capazes de interagir (ZUBEN; ATTUX, 2007). Alguns exemplos encontrados na literatura baseados em tais conceitos são: a *Otimização por Colônia de Formigas* (ACO, do inglês *Ant Colony Optimization*) (DORIGO; MANIEZZO; COLORNI, 1996), *OptBees* (Algoritmo Inspirado em Colônias de Abelhas para Otimização em Espaços Contínuos) (MAIA; CASTRO; CAMINHAS, 2012), Algoritmos Imunológicos (CASTRO; TIMMIS, 2002), entre outros.

Neste contexto, o PSO vem sendo reconhecido por sua eficência quando comparado a outros métodos e técnicas matemáticas encontrados na literatura. Algumas carecterísticas tornam o PSO particulamente interessante, como a utilização de um conceito simples, ser de fácil implementação, tipo de memória inerente, poucos de parâmetros de controle e eficiência computacional quando comparados a outros métodos encontrados na literatura (VALLE et al., 2008). Entre as áreas de aplicação do PSO estão o *design* de sistemas, otimização multiobjetivo, classificação de dados, reconhecimento de padrões, modelagem de sistemas biológicos, programação (planejamento), processamento de sinais, jogos, aplicações robóticas, tomada de decisão, simulação e identificação, para citar alguns (KENNEDY; SHI, 2001).

2.1.1 Descrição do PSO

O PSO utiliza um enxame de partículas para varrer o espaço de busca na tentativa de encontrar novas soluções para resolução de um problema. Cada partícula do enxame possui dois componentes denominados **pbest** (*personal best*) e **gbest** (*global best*) onde ficam registradas informações acerca das melhores posições (no espaço de busca) encontradas por cada partícula (ou enxame). Enquanto *pbest* identifica a melhor posição

obtida pela própria partícula, *gbest* determina a melhor posição encontrada pelo enxame. Além disso, cada partícula está associada a uma velocidade que determina a intensidade (tamanho) do deslocamento da partícula pelo espaço de busca.

O PSO, descrito no Algoritmo 1, funciona da seguinte forma:

1) Gera-se um conjunto inicial de partículas (ou soluções candidatas);

2) Avalia-se a qualidade de cada partícula;

3) Atualiza, caso necessário, os valores de *pbest* e *gbest* das partículas;

4) Atualiza a velocidade de cada partícula, de acordo com a Eq. 2.1;

5) Atualiza as posições das partículas de acordo com a Eq. 2.2;

6) Volte ao passo 2 caso o critério de parada não seja atingido;

Algoritmo 1 Algoritmo PSO

```
1: Inicializa uma população de partículas com velocidades e posições aleatórias
2: enquanto critério de parada não atingido faça
3:     para cada partícula i faça
4:         Avaliar o fitness f(X_i) da partícula;
5:         se f(X_i) < f(pbest_i) então
6:             pbest_i ← X_i;
7:         fim se
8:         se f(X_i) < f(gbest_i) então
9:             gbest_i ← X_i;
10:        fim se
11:        Atualizar a velocidade da partícula i a partir da Eq. 2.1;
12:        Atualizar a posição da partícula i a partir da Eq. 2.2;
13:    fim para
14: fim enquanto
```

Para varredura do espaço de busca, a velocidade e a posição de cada partícula são atualizadas em cada iteração do algoritmo. Primeiramente atualiza-se a velocidade da partícula, segundo a Eq. 2.1, e, posteriormente, a posição da partícula é calculada segundo a Eq. 2.2. Esta operação resulta numa nova partícula que identifica uma nova posição no espaço de busca. Conceitualmente, o *gbest* e *pbest* guiam o processo de convergência do PSO na busca por novas soluções dentro do espaço de busca.

$$V_i^{t+1} = \omega.V_i^t + c_1.rand_1.(pbest_i^t - X_i^t) + c_2.rand_2.(gbest_i^t - X_i^t) \tag{2.1}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2.2}$$

Os termos c_1 e c_2 representam as *taxas de aprendizagem* do PSO. Ambos representam, respectivamente, os componentes dos *fatores cognitivo* e *social* da partícula e determinam o quanto a partícula irá utilizar de sua memória e do conhecimento de todo enxame na geração de novas soluções. $rand_1$ e $rand_2$ são valores aleatórios no intervalo $[0, 1]$ que atribuem aleatoriedade ao processo de deslocamento da partícula com objetivo principal de evitar que a partícula fique "presa" em um ótimo local.

A Figura 2.1.1 ilustra geometricamente o processo de atualização da posição/velocidade de uma partícula.

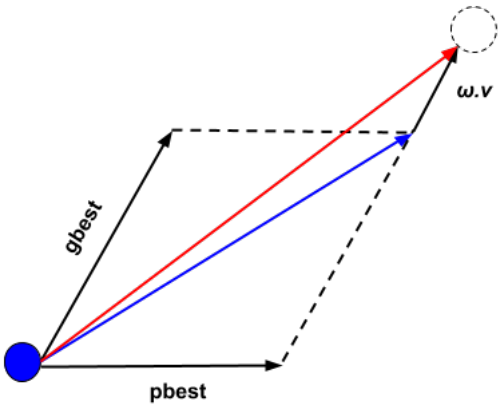

Figura 1 – Atualização da posição da partícula no PSO

Outro termo comumente utilizado é a constante ω (*ponderação de inércia*) que tem por objetivo servir como um mecanismo para controlar as capacidades de diversificação e intensificação das partículas do enxame. Basicamente, a ponderação de inércia atribui o quanto da direção de vôo anterior irá influenciar na nova velocidade da partícula numa próxima iteração do PSO (RINI; SHAMSUDDIN; YUHANIZ, 2011).

2.1.2 Representação da Partícula

A representação de uma partícula i e de dimensão D no PSO é, originalmente, codificada como um vetor númerico ($X^t = [x_1^t, x_2^t, x_3^t, ..., x_D^t]$) que representa a posição da partícula em determinado momento t no espaço de busca. Associado a esse vetor posição, tem-se o vetor velocidade $V^t = [v_1^t, v_2^t, v_3^t, ..., v_D^t]$ da partícula no instante t que é frequentemente ajustado em cada iteração do algoritmo.

2.1.3 PSO Combinatorial

Devido a sua essência originalmente contínua, o PSO é normalmente relacionado à problemas contínuos (KENNEDY; EBERHART, 1997). Segundo Rosendo e Pozo (2010), ainda existem poucas aplicações de PSO destinadas à problemas discretos, dado, em grande parte, à sua natureza inicial contínua.

Contudo, em razão do bom desempenho apresentado pelo algoritmo PSO na resolução de problemas de otimização de natureza contínua, intensificaram as pesquisas relacionadas à aplicação deste algoritmo na resolução de problemas de natureza combinatorial (WANG; ZHANG, 2011).

A primeira abordagem embasada no PSO para domínios discretos, chamada de DPSO (*Discrete Particle Swarm Optimization*), foi proposta por Kennedy e Eberhart (1997) e apresenta estrutura bastante similar à versão original do PSO. Diferentemente do PSO original, o DPSO nunca alcançou o mesmo sucesso do PSO (HOFFMANN et al., 2011).

Nesta abordagem a velocidade da partícula continua sendo representada de forma contínua. Por outro lado, a codificação da "posição" adota uma representação binária, cujo valor da i-ésima dimensão da partícula na iteração $t+1$, é determinada através da Eq. 2.3. A Eq. 2.4 representa uma função sigmoidal.

$$X_i^{t+1} = \begin{cases} 1\ ,\ se\ rand > S(V_i^{t+1}); \\ \\ 0\ ,\ sen\tilde{a}o. \end{cases} \tag{2.3}$$

$$S(v) = \frac{1}{1+e^{-v}} \tag{2.4}$$

Segundo Wang e Tang (2011), as aplicações envolvendo algoritmo PSO para problemas de natureza combinatorial dividem-se nas seguintes categorias:

- **Baseado em representação binária**: inspirada pelo trabalho de Kennedy e Eberhart (1997);
- **Baseada em permutação**: geralmente as partículas são representadas por uma sequência ordenada de símbolos. Neste modelo, a operação de troca de posições é implementada na busca por novas soluções no PSO.

Alguns trabalhos que envolvem a aplicação do algoritmo PSO em problemas de classificação de dados estão relacionados na seção 2.5. Normalmente, os autores adotaram

a representação binária na implementação do seu algoritmo, diferentemente da representação desenvolvida neste trabalho que segue a representação baseada em permutação.

2.1.3.1 PSO Baseado em Permutação

No contexto de PSOs baseados em permutação, Pan et al. (2008) propõem uma nova abordagem de PSO e, basicamente, o processo de atualização da posição da partícula envolve os seguintes conceitos, como apresentado na formulação da Eq. 2.5:

$$X_i^t = c_2 \otimes F_3(\overbrace{c_1 \otimes F_2(\underbrace{w \otimes F_1(X_i^{t-1})}_{\lambda},\ pbest_i^{t-1})}^{\delta},\ gbest^{t-1}) \tag{2.5}$$

Onde λ e δ são partículas temporárias utilizadas para simplificar o processo de atualização de cada partícula do enxame e w, c_1 e c_2 parâmetros definidos no intervalo $[0, 1]$.

- O **primeiro componente** $\lambda = w \otimes F_1(X_i^{t-1})$ representa a velocidade da partícula. F_1 representa um mecanismo de busca local que ocorre segundo uma probabilidade r_1. Deste modo, seja r_1 um número aleatório gerado no intervalo de $[0, 1]$, se r_1 for menor que w, então $\lambda = F_1(X_i^{t-1})$. Caso contrário, $\lambda = X_i^{t-1}$;
- O **segundo componente** $\delta = c_1 \otimes F_2(\lambda,\ pbest_i^{t-1})$ compreende o "termo cognitivo" do PSO original. F_2 representa um operador de recombinação (*crossover*) com probabilidade de ocorrer a partir de um valor aleatório $r_2 \in [0, 1]$. Assim, caso r_2 seja menor que o parâmetro c_1, então $\delta = c_1 \otimes F_2(\lambda,\ pbest_i^{t-1})$. Caso contrário, $\delta = \lambda$. É importante mencionar que λ e $pbest_i^{t-1}$ correspondem as soluções pais da operação de recombinação;
- O **terceiro componente** $X_i^t = c_2 \otimes F_3(\delta,\ gbest^{t-1})$ compreende o "termo social" do PSO. F_3 representa um operador de recombinação em que δ_i^t e $gbest^{t-1}$ representam as soluções pais. Assim, se um número aleatório r_3 for menor que o parâmetro c_2, então $X_i^t = c_2 \otimes F_3(\delta,\ gbest^{t-1})$. Caso contrário, $X_i^t = \delta$.

A Eq. 2.5 é implementada pelo Algoritmo 8, apresentado na seção 3.7.

2.1.4 Geração da População Inicial

Geralmente, as populações iniciais dos algoritmos evolucionários são construídas a partir da geração aleatória dos indivíduos (soluções candidatas) de maneira a constituir

uma amostra representativa do espaço de busca. Entretanto, conforme afirmam Haubelt, Gamenik e Teich (2005), despender um esforço inicial na implementação de heurísticas para construção das soluções iniciais de uma população pode auxiliar os algoritmos a convergirem mais rapidamente.

Diante deste fato, o PSO proposto neste trabalho também implementa uma estratégia para construção do enxame inicial com o objetivo de auxiliar na convergência do algoritmo. A seção 3.5 descreve a rotina de geração do enxame inicial implementada pelo algoritmo PSO proposto.

2.1.5 Técnicas de Nicho

A aplicação de técnicas de nicho pelos algoritmos populacionais consiste na divisão da população de soluções candidatas em diversas subpopulações (nichos). No contexto de problemas de classificação de dados, esta divisão da população em nichos ocorre de maneira que cada subpopulação (nicho) identifique as melhores regras de classificação de determinada classe da base de dados. Especificamente para este tipo de problema, o número de nichos é sempre igual ao número de classes.

Segundo Pereira et al. (2014), técnicas de nicho são uma funcionalidade importante, pois permitem ao algoritmo obter boas regras para todas as classes do problemas simultaneamente, em uma única execução. Além disso, o custo computacional total é reduzido em comparação com execuções múltiplas de uma versão do algoritmo que implemente uma população de soluções candidatas que descreve uma classe somente.

2.2 Otimização Multiobjetivo

De acordo Cunha, Takahashi e Antunes (2014), muitos problemas encontrados na natureza são modelados como sendo de múltiplos objetivos e neste sentido, diversos esforços têm sido empregados na construção de versões multiobjetivos de Algoritmos Evolucionários para resoluções de tais problemas.

Em problemas multiobjetivo não existe somente uma solução, mas sim um conjunto de soluções. Elas relacionam os "*trade-off*" entre os objetivos, e portanto, necessitam da utilização de uma noção de otimalidade para identificação destes conjuntos de soluções.

Por isso, para problemas multiobjetivo é comum a utilização do conceito de otimalidade proposto por Francis Ysidro Edgeworth (em 1881). Este conceito difundido, posteriormente, por Vilfredo Pareto (em 1896), é denominado de *Edgeworth-Pareto ótimo* ou, simplesmente, *Pareto ótimo* (COELLO-COELLO, 2006). Neste trabalho, é utilizado

pelo PSO proposto na identifição das soluções consideradas ótimas para o problema de classificação do dados.

Normalmente, os problemas multiobjetivo se caracterizam pela maximização (ou minimização) de um vetor de objetivos, sendo descrito por um modelo matemático conforme apresentado na formulação 2.6:

$$\textbf{Maximizar/Minimizar } \mathcal{F}(x) = [f_1(x), f_2(x), ..., f_m(x)] \tag{2.6}$$

$$\textbf{Sujeito a } x \in \Omega \tag{2.7}$$

Tal que Ω é o espaço de decisões. Sejam $u = (u_1, u_2, ..., u_m)$ e $v = (v_1, v_2, ..., v_m)$ tal que $u, v \in \mathbb{R}^m$. Diz-se que u domina v, ou $u \prec v$, se $\forall u_i \leq v_i$, para todo $i = 1, 2, ..., m$, e $\exists\ u_i < v_i$, respeitando, se existir, as restrições do problema. Quando u não domina e nem é dominado por v, diz-se que ambos u e v não possuem relação de dominância entre si ou são incomparáveis.

Baseado no conceito de relação de dominância apresentado é possível aplicar a condição de otimalidade em problemas de otimização multiobjetivo. Com isso, a região de soluções viáveis S, tal que uma solução $x \in S$, é identificada como *Pareto-ótima*, se e somente se, não existir outra solução $x' \in S$ tal que:

$$\mathcal{F}(x) \prec \mathcal{F}(x') \tag{2.8}$$

Logo, um *Conjunto Pareto-Ótimo* $\mathcal{P}$ é definido por:

$$\mathcal{P} = \{x \in S, \nexists\ x' \in S \mid \mathcal{F}(x) \prec \mathcal{F}(x')\} \tag{2.9}$$

Quando mapeadas no espaço dos objetivos, as *soluções Pareto-ótimas* são chamadas de *Fronteira Pareto*, $\mathcal{PF}$. Ou seja:

$$\mathcal{PF} = \{u = \mathcal{F}(x) \mid x \in \mathcal{P}\} \tag{2.10}$$

A tarefa de encontrar a fronteira Pareto exata de um problema de otimização multiobjetivo pode ser difícil. Dessa maneira, são aceitáveis aproximações de uma fronteira Pareto encontradas dentro de um intervalo limitado de processamento computacional (COELLO-COELLO, 2006). Tais soluções devem está o mais próximo da fronteira Pareto real com a maior diversidade possível de amostras.

A otimização multiobjetivo é uma importante área de pesquisa para cientistas e engenheiros. Técnicas tradicionais de otimização, tais como métodos baseados em gradientes, são difíceis, senão impossíveis, de serem utilizados em verdadeiros problemas de otimização multiobjetivo. Devido a isso, abordagens evolucionárias têm sido desenvolvidas a serem aplicadas em problemas de otimização multiobjetivo (HU; EBERHEART; SHI, 2003).

A classificação de dados por meio de extração de regras, tema deste trabalho, é formulada como um problema de otimização. Conforme afirma Liu et al. (2004), problemas de otimização multiobjetivo podem ser reduzidos à problemas de classificação de dados, cujo objetivo principal é encontrar regras de classificação com alta precisão, generalização e compreensão.

2.2.1 PSO Multiobjetivo

A relativa simplicidade do PSO, em conjunto com suas técnicas de compartilhamento de informação, tornaram-o um candidato natural para desenvolvimento de uma variante do PSO para aplicação em problemas de natureza multiobjetivo (MISHRA, 2016). Entretanto, apesar do PSO se mostrar particularmente adequado à abordagem multiobjetivo, devido ao seu poder de convergência para problemas mono-objetivo (COELLO-COELLO; PULIDO; LECHUGA, 2004), parece que ainda não há um consenso acerca da melhor forma de representação dos componentes *gbest* e *pbest* no contexto multiobjetivo (REDDY; KUMAR, 2007).

Cabe ressaltar que, na versão mono-objetivo do PSO, os componentes *pbest* e *gbest* representam uma solução somente. Porém, no contexto da otimização multiobjetivo, a representação de *pbest* e *gbest* é uma questão a ser analisada, visto que os tais componentes passam a se comportar como um repositório, contendo inúmeras soluções não-dominadas.

2.3 Busca Local Pareto

Os mecanismos de busca local são conhecidos pela sua eficiência em várias situações práticas, especialmente em problemas combinatórios complexos (BASSEUR; BURKE, 2007). A introdução destes mecanismos em algoritmos multiobjetivo é relativamente comum uma vez que eles possibilitam a esses algoritmos convergirem mais rapidamente e com mais precisão em direção à fronteira Pareto (MENDES, 2013).

Normalmente, estes mecanismos tentam aprimorar, iterativamente, uma solução s através de sucessivas substituições da solução corrente por uma solução melhor (solução

mutante) encontrada na vizinhança da solução s. A vizinhança de uma solução s, representada por $\mathcal{N}(s)$, compreende o conjunto de soluções (chamados vizinhos) que podem ser alcançados a partir de s através de modificações ou perturbações (operações de *move*) na solução corrente s (ZACHARIADIS; KIRANOUDIS, 2010).

Os algoritmos de *busca local Pareto* (PLS, *Pareto Local Search*) são técnicas de busca local nas vizinhanças de uma solução para resolução de problemas de otimização combinatorial multiobjetivo. As PLSs aplicam uma estratégia de exploração iterativa em um conjunto de soluções, objetivando obter consequentemente novas soluções não-dominadas, a partir de pesquisas nas vizinhanças de cada solução do conjunto (DRUGAN; THIERENS, 2012).

Os métodos de busca local Pareto propostos por Drugan e Thierens (2012) são descritos nas seções seguintes e têm em comum os seguintes parâmetros: uma solução inicial s, um conjunto de soluções não-dominadas A e uma função multiobjetivo $\mathcal{F}(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$. Ao final da execução de cada método, é retornado o conjunto de soluções não-dominadas A'.

2.3.1 Best Pareto Improvement - BPI

Descrito no Algoritmo 2, o *Best Pareto Improvement* (BPI) varre toda a vizinhança da solução s. O conjunto de soluções A', retornado pela rotina, contém as soluções encontradas nas vizinhanças $\mathcal{N}(s)$ que dominam ou são incomparáveis com as soluções do conjunto $\{s\} \cup A$.

O parâmetro *visited* identifica se a vizinhança de determinada solução já foi examinada anteriormente pelo método (*visited* = TRUE) ou não (*visited* = FALSE). O atributo *visited* de cada solução é inicializado com valor FALSE.

Algoritmo 2 Best Pareto Improvement

Entrada: s - Solução inicial; A - Conjunto Pareto de entrada.

1: $A' \leftarrow \{s\} \cup A$;
2: **para** $s' \in \mathcal{N}(s)$ **faça**
3: **se** $\forall s'' \in A', f(s') \prec f(s'') \vee f(s') \parallel f(s'')$ **então** // $\parallel$ - não há relação de dominância
4: $s'.visited \leftarrow$ FALSE;
5: $A' \leftarrow merge(A', \{s'\})$;
6: **fim se**
7: **fim para**
8: $A' \leftarrow A' \setminus (\{s\} \cup A)$;
9: **devolve** A';

A função *merge* (linha 4) agrupa os dois conjuntos Pareto em um novo conjunto

Pareto, removendo as soluções dominadas do conjunto resultante. No método BPI, as buscas na vizinhança de uma solução s envolvem todas as soluções do conjunto A' que não tenham sido visitadas ($visited$ = FALSE).

2.3.2 First Pareto Improvement - FPI

Descrito no Algoritmo 3, o processo de busca na vizinhança pelo método *First Pareto Improvement* (FPI) é finalizado quando a primeira solução localizada em $\mathcal{N}(s)$ que domina a solução corrente s é encontrada.

Algoritmo 3 First Pareto Improvement

Entrada: s - Solução inicial; A - Conjunto Pareto de entrada.

1: $A' \leftarrow \{s\} \cup A$;
2: **para** $s' \in \mathcal{N}(s)$ **faça**
3: **se** $\forall s'' \in A', f(s') \prec f(s'') \vee f(s') \parallel f(s'')$ **então**
4: $s'.visited \leftarrow$ FALSE;
5: $A' \leftarrow merge(A', \{s'\})$;
6: **se** $f(s') \prec f(s)$ **então**
7: $A' \leftarrow A' \setminus (\{s\} \cup A)$;
8: **devolve** A';
9: **fim se**
10: **fim se**
11: **fim para**
12: $A' \leftarrow A' \setminus (\{s\} \cup A)$;
13: **devolve** A';

2.3.3 Neutral Pareto Improvement - NPI

O método *Neutral Pareto Improvement* (NPI), apresentado pelo Algoritmo 4, interrompe a busca na vizinhança da solução s quando um vizinho que não é dominado nem pela solução corrente s e tampouco por nenhuma solução do conjunto A é identificado.

Algoritmo 4 Neutral Pareto Improvement

Entrada: s - Solução inicial; A - Conjunto Pareto de entrada.

1: $A' \leftarrow \{s\} \cup A$;
2: **para** $s' \in \mathcal{N}(s)$ **faça**
3: **se** $\forall s'' \in A', f(s') \prec f(s'') \vee f(s') \parallel f(s'')$ **então**
4: $s'.visited \leftarrow$ FALSE;
5: $A' \leftarrow merge(A', \{s'\})$;
6: $A' \leftarrow A' \setminus (\{s\} \cup A)$;
7: **devolve** A';
8: **fim se**
9: **fim para**
10: **devolve** $\emptyset$;

2.4 Mineração de Dados

Os avanços da tecnologia têm produzido uma abundância de dados. Logo, empregar métodos e técnicas capazes de analisar e extrair conhecimento de bases de dados tem se tornado um fator importante à tomada de decisões. De acordo com Fayyad et al. (1996b), por meio de KDD (*Knowledge Discovery in Database*)[1], é possível extrair informações implícitas, não triviais, previamente desconhecidas e potencialmente úteis em um banco de dados, que muitas vezes são inutilizadas ou pouco aproveitados dentro do ambiente de trabalho.

A *mineração de dados* (*data mining*) é uma fase dentro do processo de KDD e consiste basicamente na aplicação de algoritmos que, sob aceitável eficiência computacional, possam contribuir no reconhecimento de padrões, ou modelos, acerca dos dados estudados (FAYYAD et al., 1996a).

Nos dias de hoje, a mineração de dados tem despertado a atenção de diversos estudiosos, sendo citada pelos mesmos como um elemento fundamental em pesquisas relacionadas a área estratégica industrial, já que permite extrair informações que alavanquem seus negócios bem como a descoberta de novas oportunidades de inovação (CHEN; HAN; YU, 1996).

O termo mineração de dados foi cunhado em alusão ao processo de mineração, isto é, o processo de extração de minerais preciosos, que pode ser compreendido da seguinte maneira: explora-se uma base de dados (mina) usando algoritmos (ferramentas) para a obtenção de conhecimento (minerais preciosos) (CASTRO; FERRARI, 2016).

Diversas tentativas têm sido utilizadas na aplicação de algoritmos evolucioná-

[1] Tradução: Descoberta de conhecimento em banco de dados

rios para a realização das tarefas de mineração de dados. Neste contexto, o PSO se apresenta como uma ferramenta promissora na aplicação em problemas de mineração de dados devido a sua facilidade de implementação e poucos parâmetros a serem ajustados (WANG; SUN; ZHANG, 2007), sendo considerado um método robusto à resolução de problemas não-lineares, não-integráveis, multimodais e de alta dimensionalidade (KROHLING, 2004).

2.4.1 Classificação de Dados

Classificar um objeto significa atribuir um rótulo, chamado *classe*, de acordo com a categoria à qual ele pertence. Para que isso seja possível, um algoritmo ou método é utilizado na construção de um modelo de classificação - também chamado de *classificador*. Existe uma variedade de algoritmos destinados à tarefa de classificação de dados na literatura, como por exemplo as *Árvores de Decisão*, as *Rede Neurais Artificiais*, *Máquina de Vetores Suporte*, *Algoritmos Evolucionários*, etc, cujas aplicações incluem a identificação de *spams*, atribuição de crédito e detecção de fraudes, além de outras (CASTRO; FERRARI, 2016).

A *extração de regras* (do inglês *rule mining* ou *rule discovery*) é um método dentro da classificação de dados que busca encontrar um conjunto de regras *SE-ENTÃO* para classificar, de forma mais "natural", uma base de dados. Conforme explica Pereira (2012), a extração de regras é um tópico recorrente na mineração de dados, segundo Hassani e Lee (2013), caracterizado por ser um problema *NP-Completo*.

Cabe observar que a obtenção de regras com alto grau de precisão na etapa de treinamento não é garantia de que o mesmo sucesso seja obtido na fase de testes. Isso ocorre pelo fato de que regras muito adaptadas a um determinado conjunto de dados tendem a ser ineficientes na classificação de novos dados (*overfitting*)[2], sendo incapazes de promover a generalização (PEREIRA; JúNIOR; VASCONCELOS, 2010).

Deste modo, buscar um conjunto de regras que seja o mais simples (com menor número de termos) também é um objetivo normalmente explorado por alguns algoritmos que abordam o problema de classificação de dados (Pereira et al. (2014), Liu et al. (2004), Sousa, Silva e Neves (2004), etc).

Na Figura 2 é ilustrado o processo de classificação de dados através de conjunto de regras *SE-ENTÃO*:

(i) Coleta de dados;

[2] Tradução: Sobregeneralização

(ii) São extraídas as regras de classificação durante a fase de treinamento;

(iii) Constrói-se o modelo de classificação;

(iv) Novas amostras são classificadas na fase de teste.

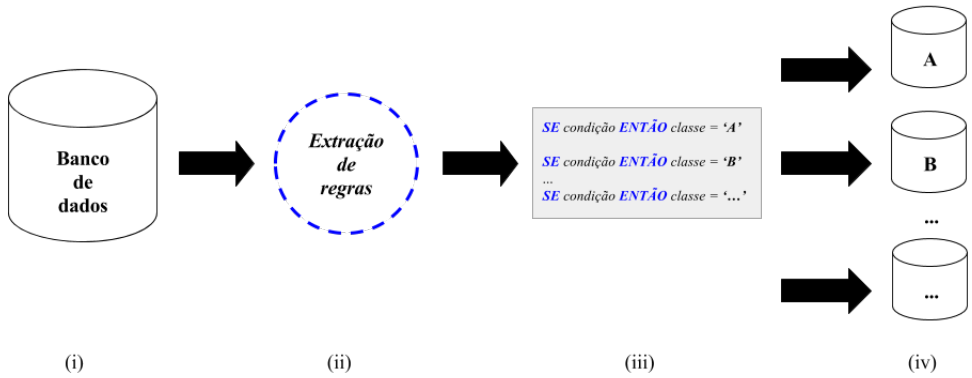

Figura 2 – Etapas que compõem a classificação de dados através de um conjunto de regras *SE-ENTÃO*.

Abordagens não-baseadas em extração de regras *SE-ENTÃO* apesar de representarem o conhecimento descoberto, funcionam como modelos de "caixa preta" e tendem a apresentar baixa compreensão, mesmo possuindo elevados níveis de precisão. São exemplos de métodos de classificação não-baseados em regras: *Máquinas de Suporte Vetorial* e *Redes Neurais Artificiais*, que mesmo apesar de apresentarem níveis de previsão considerados ótimos, são de difícil compreensão (HASSANI; LEE, 2013). Por outro lado, abordagens baseadas em regras *SE-ENTÃO* são mais intuitivas, já que tendem a beneficiar a representação simbólica do conhecimento, facilitando a sua compreensão por parte dos usuários (WANG; SUN; ZHANG, 2007).

Uma regra *SE-ENTÃO* é composta basicamente por dois componentes: os *antecedentes* (ou restrições) e os *consequentes* (a classe predita), conforme ilustrado no esquema da Figura 3.

$$\textbf{SE}\ \overbrace{termo_1\ \textbf{E}\ termo_2\ \textbf{E}\ \cdots\ \textbf{E}\ termo_N}^{antecedentes}\ \textbf{ENTÃO}\ \overbrace{classe}^{consequente}$$

Figura 3 – Estrutura das regras *SE-ENTÃO*.

Os *antecedentes* são um conjunto de termos, em que cada termo é descrito por uma tríade com a seguinte estrutura: *<atributo, operador, valor numérico ou outro atributo>*,

conectados por conectores lógicos **E** (*AND*), formando consequentemente uma série de restrições (condições) que devem ser atendidas. O *consequente* é posicionado após a cláusula **ENTÃO** da regra e define a classe que é predita pela regra. Em resumo, se os dados da base satisfazem os antecedentes, então estes são identificados com a classe associada à regra.

Para Freitas (2003) há duas abordagens para representação das regras pelos algoritmos evolucionários propostos para problemas de classificação de dados:

- **Abordagem Pittsburgh**: O indivíduo representa um conjunto de regras. Esta representação possui maior dificuldade de formulação e implementação pelos algoritmos;
- **Abordagem Michigan**: Cada indivíduo representa uma única regra. Esta representação é mais fácil de implementar e tende a reduzir o tempo de cálculo da aptidão (*fitness*) do indivíduo.

Normalmente, os algoritmos para problemas de classificação de dados raramente utilizam a abordagem Pittsburgh. De acordo Cervantes, Galvain e Isasi (2005), a abordagem Michigan apresenta certas vantagens em relação à abordagem Pittsburgh, como: (i) capacidade de fornecer boas soluções com um número menor de avaliações da função-objetivo e (ii) maior flexibilidade na representação das regras de classificação.

A classificação de dados por meio de extração de regras estudada neste trabalho é modelada como um problema de otimização biobjetivo em que se busca maximizar a efetividade das regras extraídas ao mesmo tempo em que se tenta minimizar a complexidade das regras (tamanho), uma vez que regras menores são mais simples de serem entendidas e tendem a evitar o *overfitting* (PEREIRA, 2012).

2.4.2 Avaliação da Classificação de Dados

Entende-se como "qualidade" de uma regra a capacidade em classificar corretamente o maior número de padrões (ou registros) de um banco de dados. Diversas formas de se medir a qualidade de uma regra de classificação, sendo que as mais comuns se baseiam em cálculos realizados a partir dos coeficientes da matriz de confusão (Ver Tabela 1) (PEREIRA, 2012).

Os coeficientes presentes na matriz permitem o cálculo de importantes métricas para averiguar o desempenho dos classificadores. São exemplos: a *acurácia* (Eq. 2.11),

Classif. / **Real**	**Positivo**	**Negativo**
Positivo	*Verdadeiro Positivo* (VP)	*Falso Negativo* (FN)
Negativo	*Falso Positivo* (FP)	*Verdadeiro Negativo* (VN)

Tabela 1 – Matriz de confusão

sensibilidade (Eq. 2.12) e *especificidade* (Eq. 2.13), dentre outras. Para o problema investigado neste trabalho:

$$Acurácia = \frac{VP + VN}{VP + FN + FP + VN} \tag{2.11}$$

$$Sensibilidade = \frac{VP}{VP + FN} \tag{2.12}$$

$$Especificidade = \frac{VN}{VN + FP} \tag{2.13}$$

Onde:

- **Verdadeiros positivos** (VP): total de registros (ou tuplas) recuperados do banco de dados utilizando a regra codificada pela partícula nas quais os respectivos registros coincidem com a classe predita pela partícula;
- **Falsos positivos** (FP): total de registros recuperados pela regra e que não pertecem a classe predita pela partícula;
- **Verdadeiros negativos** (VN): total de registros que não são cobertas pela regra e que não pertença à classe da partícula;
- **Falsos negativos** (FN): total de registros que não foram obtidas pela regra, mas que pertencem à classe predita pela partícula.

2.5 Trabalhos Relacionados

A tarefa de classificação de dados pode ser realizada por meio de diversas abordagens. A seguir são apresentados trabalhos relevantes encontrados no contexto de classifi-

cação de dados por meio de PSO que serviram de referência para desenvolvimento desse trabalho.

Sousa, Silva e Neves (2004) apresentam um estudo acerca de uma variante do PSO destinada à classificação de dados por meio da extração de regras *SE-ENTÃO* numa abordagem mono-objetivo, sendo a aptidão das partículas avaliadas a partir do cálculo do produto da sensibilidade pela especificidade. É interessante notar que as partículas, posteriormente à avaliação de aptidão, passam por uma fase de encolhimento (*prunning*), com o objetivo eliminar os atributos que não contribuem para eficácia da regra. O trabalho investiga as seguintes variantes do PSO: *Discrete PSO* (DPSO), *Constricted PSO* (CPSO) e *Linear Decreasing Weight PSO* (LDWPSO).

Liu et al. (2004) apresentam um classificador baseado no PSO para classificação de dados chamado de *Rule Discovery with Particle Swarm Optimization* (REPSO). Neste classificador, cada partícula representa uma única regra e, segundo os autores, tem a vantagem de poder ser aplicada tanto em dados categóricas como em dados contínuos. Durante os testes, foram utilizadas duas bases de dados do *UCI repository of Machine Learning*: a base de dados *Zoo*, no qual todos os atributos são categóricos e o conjunto de dados *Wine*, no qual todos os atributos, exceto o atributo de classificação, são contínuos. As regras descobertas pelo algoritmo foram avaliadas pelo somatório da acurácia obtida e pelo número de termos utilizado pela regra, sendo atribuídos "pesos" a cada um, na avaliação da aptidão (*fitness*) da solução.

Cervantes, Galvain e Isasi (2005) avaliam o desempenho de um PSO binário para classificação de dados utilizando as abordagens Pittsburgh e Michigan na codificação das partículas. O PSO proposto no trabalho apresenta as seguintes adaptações: (i) a adição de uma força competitiva que repele uma partícula de seu melhor vizinho e (ii) a utilização de vizinhanças dinâmicas baseadas num critério de proximidade. Os resultados obtidos mostraram que a abordagem Michigan foi superior à abordagem Pittsburgh na maioria das situações analisadas pelo artigo.

Zahiri e Seyedin (2007) propõem um classificador PSO inteligente chamado de *Intelligent Particle Swarm Classifier* (IPS-*classifier*). Este classificador tenta encontrar hiperplanos de decisão para determinação das diferentes classes dentro do espaço de busca. Um controlador *fuzzy* foi também projetado para melhorar o desempenho e eficiência do classificador proposto, adaptando os parâmetros de ponderação de inércia e os fatores cognitivo e social do PSO.

Wang e Zhang (2011) propõem um algoritmo mono-objetivo para classificação de dados e mineração de regras utilizando o DPSO. As regras são codificadas através de uma cadeia de *bits* de comprimento fixo, em que o número de termos utilizados pela regra

é ajustado dinamicamente por meio de um *bit* adicional para cada termo com objetivo de determinar a sua existência, ou não, em determinada regra. Nesta proposta, cada partícula é representada por um grupo de regras (abordagem Pittsburgh) e os resultados obtidos mostraram que o PSO proposto alcançou maior acurácia e uma lista de regras menor do que outros algoritmos analisados pelo artigo.

Chen e Ludwig (2012) apresentam uma implementação DPSO, chamada de *Discrete Particle Swarm Optimization With Local Search* (DPSO-LS), que utiliza a abordagem Pittsburgh para codificação da partícula. Cada partícula do enxame é representada por uma matriz em que cada linha descreve uma regra de classificação *SE-ENTÃO* durante a tarefa de classificação de dados. Além disso, avaliou-se neste trabalho a importância da utilização de técnicas de busca local para refinamento dos resultados encontrados.

Hassani e Lee (2013) apresentam um DPSO para mineração de regras de classificação que utiliza uma separação dos dados de treinamento em blocos em que cada um é atribuído a uma *thread*. Ao final, as regras obtidas em cada *thread* são integradas em uma base de regras para construção de um modelo de classificação.

Mishra (2016) apresenta um classificador para mineração de regras de associação com abordagem multiobjetivo para maximização do *suporte* (*cobertura*) e *confiança* da regra de associação ao mesmo tempo em que tenta minimizar a complexidade da mesma.

A maioria dos trabalhos relacionados acima propõem estruturas de dados baseadas em uma representação binária das partículas para extração de regras de classificação ou associação, similares ao DPSO proposto por Kennedy e Eberhart (1997).

Em problemas de classificação de dados que levam em consideração a complexidade das regras são utilizados mecanismos para ignorar certos termos da regras *SE-ENTÃO* codificadas nas partículas. Geralmente, estas codificações envolvem a utilização de um vetor com todos os atributos da base de dados e um *bit* que habilita ou não a utilização de determinado atributo na regra. Neste contexto, o seguinte trabalho se torna relevante ao utilizar uma nova abordagem baseada no trabalho de Pan et al. (2008) (Ver seção 2.1.3.1) para codificação e atualização da posição da partícula.

Pan et al. (2008) propõem um processo de atualização da posição da partícula alternativo. Neste estudo, o PSO proposto é aplicado em ambientes combinatoriais na resolução de problemas de alocação de recursos e escalomanento (*flow-shop*). De acordo os autores, tal abordagem pode ser facilmente aplicada em problemas de otimização combinatorial com resultados que demonstram ser competitiva, podendo inclusive obter resultados superiores a outros métodos encontradas na literatura especializada.

Finalmente, cabe também mencionar o trabalho de Pereira (2012) que serviu de

embasamento na utilização da metodologia aplicada durante à análises dos resultados. No seu trabalho é proposto uma variante multiobjetiva baseada em Programação Genética (PG) para extração de regras de classificação, sendo realizadas comparações de desempenho com três algoritmos implementados pelo WEKA.

2.6 Conclusão

Neste capítulo, foram apresentados os principais conceitos e definições relacionados ao PSO. Fez-se uma descrição de suas características principais, ressaltando suas funcionalidades e aplicações. Em seguida, foram relacionadas variantes do PSO destinadas ao ambiente combinatorial e as estratégias de geração do enxame inicial e técnicas de nicho.

Conceitos relacionados a otimização multiobjetivo, busca local Pareto e mineração de dados (classificação de dados), bem como a apresentação de estudos relevantes que serviram de referência para este trabalhos também foram apresentados. Nota-se que ainda existem poucos os que utilizam o PSO para a classificação de dados por meio da extração de regras. Não foi encontrado nenhum trabalho de classificação de dados baseado em PSO utilizando a formulação proposta por Pan et al. (2008), o que torna o presente trabalho inovador neste sentido.

3 mDPSO - Algoritmo Biobjetivo baseado em PSO para Classificação de Dados

Neste capítulo será detalhado o algoritmo biobjetivo, baseado na metaheurística PSO e denominado *multiobjective Discrete Particle Swarm Optimization* (mDPSO) proposto neste trabalho para classificação de dados por meio de extração de regras.

Ao contrário de outras variantes do algoritmo PSO destinadas à classificação e mineração de dados, o mDPSO utiliza uma abordagem diferente das que são comumente encontradas na literatura. Ele também implementa um conjunto de funcionalidades para explorar o espaço de busca e acelerar sua convergência.

3.1 Formulação do Problema

Neste trabalho, o problema de classificação de dados foi modelado como um problema biobjetivo, descrito pela Eq. 3.1, no qual os objetivos são: (i) maximizar a efetividade das regras extraídas; e (ii) minimizar a complexidade (número de termos) das regras.

$$\textbf{Maximizar } \mathcal{F}(x) = [f_1(I, X), f_2(I, X)] \tag{3.1}$$

- x é uma sentença da cláusula WHERE da linguagem SQL;
- Os parâmetros I e X representam respectivamente a regra codificada pela partícula e a classe X (nicho) a que a partícula pertence;
- $f_1(I, X)$, descrita por Eq. 3.2, corresponde à função que determina a efetividade (qualidade) da regra. Esta efetividade é resultante do produto da sensibilidade pela especificidade;
- $f_2(I, X)$, descrita na Eq. 3.3, representa a função que calcula a complexidade da regra. Neste trabalho ela é calculada como sendo o número inverso de termos da regra.

$$f_1(I, X) = sensibilidade \times especificidade \tag{3.2}$$

$$f_2(I, X) = \frac{1}{número\ de\ termos} \tag{3.3}$$

De acordo com Pereira (2012), o produto da sensibilidade pela especificidade apresenta propriedades interessantes, pois tende a evitar o *overfitting*, já que busca reduzir, ou até mesmo eliminar, a tendência de privilegiar as classes predominantes presentes em bancos de dados desbalanceados. Dessa maneira, se a regra apresentar um desempenho muito baixo em relação a um dos valores da sensibilidade ou da especificidade, o valor do *fitness* da partícula também será baixo, independente do outro indicador. A função *fitness* é penalizada por regras incapazes de identificar os padrões corretos, pois uso da acurácia pode resultar em um valor alto mesmo que o número de *verdadeiro positivos* (VP) seja alto e o valor de *verdadeiro negativo* (VN) próximo a zero.

3.2 Descrição do mDPSO

O objetivo do mDPSO é extrair um conjunto de regras durante as fases de treinamento para avaliar seu desempenho em novas amostras, desconhecidas durante a tarefa de treinamento, na fase de teste.

A abordagem Michigan foi utilizada para representação das partícula do mDPSO. Dessa maneira, cada partícula codifica apenas uma regra do problema, o que significa dizer que cada partícula provê uma solução possível para determinada classe e não uma solução geral para todo o problema.

O mDPSO, descrito no Algoritmo 5, implementa um conjunto de particularidades que o distinguem das demais implementações (variantes) da metaheuristica PSO encontradas na literatura. São funcionalidades adaptadas, implementadas pelo mDPSO:

- **Repositórios *gbest* e *pbest*** (Ver seção 3.4);
- **Geração do Enxame Inicial** (Ver seção 3.5);
- **Implementação de um Operador de Turbulência** (Ver seção 3.6);
- **Atualização da Posição da Partícula** (Ver seção 3.7);
- **Operação de Mutação de Partícula** (Ver seção 3.7.1);
- **Rotina de Recombinação das Partículas** (Ver seção 3.7.2).

A função *merge*, tal como descrita na seção 2.3, agrupa dois conjuntos Pareto em um novo conjunto Pareto, removendo as soluções dominadas do conjunto resultante.

Algoritmo 5 mDPSO

```
1: Geração do enxame inicial de partículas (Ver seção 3.5);
2: Avaliar as partículas do enxame (Ver Algoritmo 6);
3: enquanto critério de parada não atingido faça
4:     Aplicar o operador de turbulência (Ver seção 3.6);
5:     Atualizar a velocidade e posição de todas partículas do enxame (Ver seção 3.7);
6:     Avaliar as partículas do enxame;
7:     para cada partícula p ∈ C_k.gbest faça
8:         Aplicar busca local Pareto em p (Ver seção 3.8);
9:         Avaliar p;
10:    fim para
11: fim enquanto
```

Algoritmo 6 Avaliação de aptidão (*fitness*) de cada partícula no mDPSO

Entrada: $p.pbest$ - Soluções não-dominadas encontradas pela partícula p; $C_k.gbest$ - Soluções não-dominadas do nicho C_k.

```
1: para cada partícula p do enxame faça
2:     Avaliar p;
3:     C_k.gbest ← merge(C_k.gbest, {p});
4:     p.pbest ← merge(p.pbest, {p});
5: fim para
```

Nas próximas seções será feito o detalhamento de cada funcionalidade implementada pelo mDPSO.

3.3 Representação da Partícula

A representação adotada para cada partícula do mDPSO é ilustrada na Figura 4 e compreende dois componentes:

- Um vetor de tamanho variável de elementos (termos) que codificam a regra de classificação.
- Um rótulo identificador da classe (nicho).

Na avaliação da aptidão (*fitness*) da partícula no PSO, esse vetor de termos é convertido em um predicado lógico da claúsula WHERE de uma sentença SQL executada em um Sistema de Gerenciamento de Banco de Dados (SGBD) que retorna os registros (tuplas) que atendem à cláusula WHERE.

De posse dos registro retornados é feita a avaliação do desempenho da regra da partícula. A Figura 4 ilustra o processo de conversão da regra da partícula em uma setença SQL.

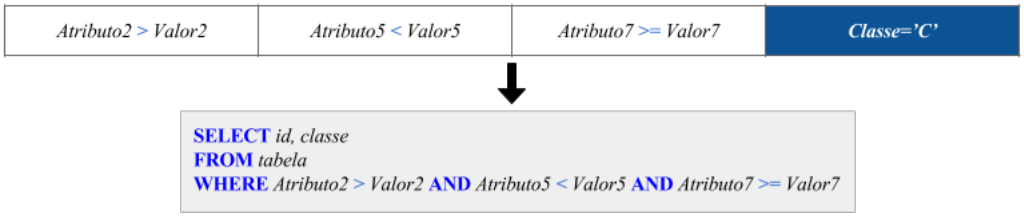

Figura 4 – Exemplo de codificação de uma partícula pelo mDPSO e a sentença SQL gerada para avaliação pelo SGBD.

3.4 Repositórios *gbest* e *pbest*

Um ponto importante no mDPSO refere-se aos componentes *gbest* e *pbest*. Em versões multiobjetivo do PSO, ambos *gbest* e *pbest* passam a atuar como repositórios de soluções não-dominadas, ao contrário da versão mono-objetivo, em que ambas se referem somente à melhor posição obtida pela partícula (*pbest*) e a melhor posição obtida pelo enxame (*gbest*).

O mDPSO implementa o *gbest* através de um repositório de soluções não-dominadas para cada nicho (classe) do problema ($C_k.gbest$). Além disso, o mDPSO também representa o *pbest* de cada partícula como um repositório de soluções não-dominadas.

Geralmente, o PSO implementa o conceito de partícula líder, que é utilizada na atualização das posições das partículas do enxame, isto é, uma solução que contribui no direcionamento do enxame dentro do espaço de busca. Nas versões mono-objetivo a partícula líder é o próprio *gbest*, sendo por isso necessário a proposição de um mecanismo de identificação da partícula líder em abordagens multiobjetivo, visto que, neste contexto, tanto o *gbest* quanto *pbest* são representadas por um repositório (conjunto) de soluções não-dominadas.

O processo de identificação da partícula líder (*gbest* ou *pbest*), necessária à recombinação no mDPSO durante a operação de atualização da posição de cada partícula, ocorre da seguinte maneira: (i) Determina-se a distância euclidiana no plano dos objetivos da partícula corrente em relação aos repositórios *gbest* ou *pbest*; (ii) A partícula mais próxima da partícula corrente em cada repositório é selecionada para atualização da sua posição durante a recombinação.

A escolha da partícula não-dominada mais próxima da partícula corrente na rotina de atualização da posição é exemplificada pela Figura 5. A partícula escolhida (partícula líder) é realçada na cor vermelha e a partícula corrente na cor azul.

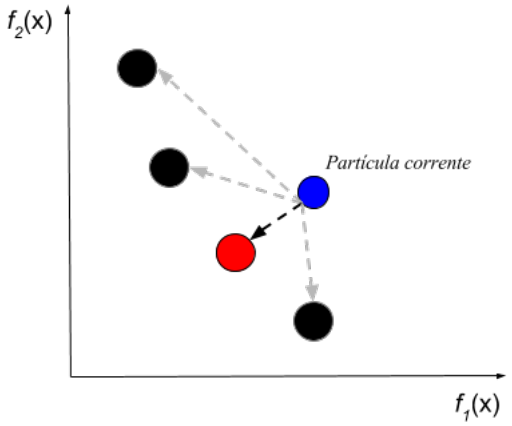

Figura 5 – Escolha da partícula líder

Tal abordagem é implementada pelo algoritmo de recrutamento de abelhas *OptBees* (MAIA, 2012). O *OptBees* é um algoritmo aplicado na resolução de problemas de otimização em espaços contínuos, sendo inspirado no comportamento das abelhas durante o forrageamento e em mecanismos envolvidos no processo de alocação de tarefas da vida em sociedade do enxame. É importante também frisar que tal abordagem pode ser aplicada tanto em problemas de maximização quanto de minimização.

Cabe também ressaltar que ao final de cada iteração do mDPSO, com o objetivo de refinar ou aprimorar as soluções encontradas até aquela iteração, é proposta uma busca local Pareto (Ver seção 3.8) no repositório *gbest* de cada nicho.

3.5 Geração do Enxame Inicial

O mDPSO implementa uma rotina para geração inicial do seu enxame de partículas. A rotina proposta, descrita pelo Algoritmo 7, funciona em duas etapas: (i) Primeiramente, aplica-se técnicas de nicho (Ver seção 2.1.5) para dividir o enxame segundo o número de classes do problema; (ii) Gera-se a regra de cada partícula do enxame.

A fase de construção (linha 7 do Algoritmo 7) da regra de cada partícula se resume a definir os η termos que entrarão na composição da regra associada à partícula. η é um número aleatório inteiro no intervalo $[1, \lceil \log(rand_{[1,N]}) \rceil]$, onde N o total de atributos da base de dados. Esta metodologia tem por objetivo simplicar as regras introduzidas na codificação da partícula.

Algoritmo 7 Geração do Enxame Inicial no mDPSO

Entrada: N - Número de partículas do enxame;.

1: $Enxame \leftarrow \emptyset$;
2: **para** Cada classe C_k do problema **faça**
3: $C_k.gbest \leftarrow \emptyset$;
4: **fim para**
5: **para** cada partícula do enxame **faça**
6: $\eta \leftarrow \left\lceil \log\left(rand_{[1,N]}\right) \right\rceil + 1$;
7: $partícula \leftarrow$ Adicionar regra de η termos;
8: $partícula \leftarrow$ Rotular partícula (C_k);
9: $partícula.pbest \leftarrow \emptyset$;
10: $Enxame \leftarrow \{partícula\} \cup Enxame$;
11: **fim para**

A seleção dos atributos que compõem cada termo da regra é feita de forma aleatoria a partir dos atributos presentes na base de dados. O operador relacional associado à cada termo da regra é escolhido do conjunto de operadores presentes na Tabela 2. Neste trabalho, foram atribuídas probabilidades de seleção distintas para alguns operadores. Estes valores (taxas de probabilidades) foram obtidos a partir de experimentos computacionais, e mostram relevância diferenciada entre os operadores.

Neste processo, observou-se que os termos das regras eram, em sua maioria, formulados com a seguinte estrutura: *<atributo, operador, valor numérico>*. No entanto, termos com estrutura do tipo *<atributo, operador, outro atributo>*, apesar de menos frequentes, também poderiam contribuir na construção de regras eficazes e mais flexíveis. Deste modo, por meio de experimentos computacionais, definiu-se que termos com a estrutura *<atributo, operador, valor numérico>* (*<atributo, operador, outro atributo>*) teriam probabilidade de 90% (10%) de serem adicionados à regra das partículas.

É importante mencionar que em situações específicas, boas regras de classificação com termos com estrutura do tipo *<atributo, operador, outro atributo>*, também facilitariam a adição de outros operadores lógicos ou mesmo funções de comparações topológicas (Ex. *contains*, *covers*, *crosses*, *disjoint*, *equals*, *touches*, *within* e *distance*, etc.) utilizadas em bancos de dados híbridos (geográficos), que promovem geralmente a comparação entre atributos.

A Tabela 2 também é utilizada pela rotina de mutação de partículas (Ver seção 3.7.1.2) implementada pelo mDPSO.

Operadores	=	!=	<	>	>=	<=
Probabilidades	6%	6%	22%	22%	22%	22%

Tabela 2 – Probabilidade de seleção dos operadores relacionais usada pelas rotinas de geração de partículas e de mutação do mDPSO.

3.6 Operador de Turbulência

Muitos autores citam a importância do operador de turbulência em razão de promover a diversidade do enxame no PSO (PARSOPOULOS; VRAHATIS, 2009), visto que a diversidade do enxame é importante para melhorar a capacidade exploratória do algoritmo PSO (REYES-SIERRA; COELLO-COELLO, 2006). Como afirmam Santana, Pontes e Bastos-Filho (2009), o PSO mono-objetivo apresenta rápida convergência quando comparado a outros métodos, contudo, tal vantagem pode ser também prejudicial no contexto de otimização multiobjetivo, já que o algoritmo tende a convergir prematuramente à um ótimo-local.

Para tentar atenuar tal efeito, o mDPSO utiliza um esquema adaptada do trabalho de Reyes-Sierra e Coello-Coello (2005). Nesta proposta, cada nicho é subdividido em três subpopulações C_1, C_2 e C_3 de mesmo tamanho, de modo que cada subpopulação C_i $(i = 1, 2, 3)$ pode, ou não, ser perturbada, via operador de mutação a fim de promover diversidade no enxame. Para isso utiliza a seguinte estratégia: (i) A subpopulação C_1 não sofre nenhuma operação de mutação; (ii) a mutação uniforme é aplicado nas partículas presentes em C_2; e (iii) a mutação não-uniforme é aplicada nas partículas da subpopulação C_3. Como alternativa à mutação não-uniforme é utilizado a mutação gaussiana (HIGASHI; IBA, 2003).

Na seção 3.7.1 são apresentadas as mutações implementadas pelo mDPSO.

3.7 Atualização da Posição da Partícula

A rotina de atualização da posição da partícula desenvolvida pelo mDPSO é descrita pelo Algoritmo 8. O mecanismo é uma adaptação da formulação proposta pelo trabalho de Pan et al. (2008) descrito na seção 2.1.3.1, sendo utilizados operadores de mutação e recombinação específicos ao problema de classificação de dados.

Algumas observações acerca do Algoritmo 8:

- A busca local na regra da partícula (linha 2) é implementada como uma operação

Algoritmo 8 Atualização da posição da partícula

Entrada: p - Partícula; ω, c_1 e c_2 - Parâmetros no intervalo $[0, 1]$.

1: // $rand_1$ valor aleatório uniforme no intervalo $[0, 1]$
2: **se** $rand_1 < \omega$ **então**
3: $\lambda \leftarrow$ Mutação(p, uniforme); // Busca Local
4: **senão**
5: $\lambda \leftarrow p$;
6: **fim se**
7: // $rand_2$ valor aleatório uniforme no intervalo $[0, 1]$
8: **se** $rand_2 < c_1$ **então**
9: $\delta \leftarrow$ Recombinar λ e p' (p' é a partícula de $p.pbest$ mais próxima de p);
10: **senão**
11: $\delta \leftarrow \lambda$;
12: **fim se**
13: // $rand_3$ valor aleatório uniforme no intervalo $[0, 1]$
14: **se** $rand_3 < c_2$ **então**
15: $p \leftarrow$ Recombinar δ e p'' (p'' é a partícula de $C_k.gbest$ mais próxima de p);
16: **senão**
17: $p \leftarrow \delta$;
18: **fim se**
19: **devolve** p;

de mutação no mDPSO (Ver seção 3.7.1);

- A operação de recombinação (linhas 7 e 12), descrita na seção 3.7.2, ocorre somente entre partículas pertencentes a uma mesma classe (nicho).

Ao analisar o Algoritmo 8 observa-se um paralelo entre a Eq. 2.1, a equação de atualização da partícula implementada pelo PSO original, e a Eq. 3.4 que corresponde à formulação proposta por Pan et al. (2008):

$$V_i^{t+1} = \underbrace{\omega.V_i^t}_{Busca\ Local} + \underbrace{c_1.rand_1.(pbest_i^t - X_i^t)}_{Recombinação} + \underbrace{c_2.rand_2.(gbest_i^t - X_i^t)}_{Recombinação} \tag{3.4}$$

Existem discussões acerca do mecanismo de atualização da posição da partícula no algoritmo PSO: (i) Para Reyes-Sierra e Coello-Coello (2006), o ajuste da velocidade da partícula atua como um operador de "mutação direcional"; (ii) Para Coello-Coello, Pulido e Lechuga (2004), o ajuste da posição da partícula é análogo à recombinação (*crossover*) dos Algoritmos Genéticos. Diante disto, o mecanismo proposto por Pan et al. (2008) mostra-se adequado ao processo de atualização da posição da partícula do PSO em problemas combinatoriais, visto que ele reúne características de ambas abordagens.

3.7.1 Operação de Mutação de Partícula

De acordo Cunha, Takahashi e Antunes (2014), para cada problema podem existir várias representações computacionais às suas resoluções, sendo normalmente necessário efetuar uma análise experimental para encontrar perturbações (mutações) aceitáveis.

Dessa forma, no decorrer do desenvolvimento deste trabalho, foram propostos três mecanismos de mutação nas regras das partículas objetivando garantir (ou mesmo introduzir) diversidade no enxame de partículas no mDPSO. A rotina de mutação implementada pelo mDPSO é detalhada no Algoritmo 9.

Algoritmo 9 Mutação

Entrada: p - Partícula; *mutação* - Tipo de mutação (perturbação).

se $rand_1 < 0.5$ **então**
    Adicionar um novo termo à regra da partícula p (Ver seção 3.7.1.1);
senão
    $t \leftarrow$ Selecionar aleatoriamente um termo da regra da partícula p;
    se $rand_2 < 0.5$ e $t[3]$ é um *valor numérico* **então**
        se *mutação* é uniforme **então**
            // Mutação Uniforme
            $t[3] \leftarrow \begin{cases} t[3] + (max(atributo) - t[3]) \times U(0,1), & \text{se } rand_3 < 0.5 \\ t[3] - (t[3] - min(atributo)) \times U(0,1), & \text{caso contrário} \end{cases}$
        senão
            // Mutação Gaussiana
            $\sigma \leftarrow 0.1 \times (max(atributo) - min(atributo))$;
            $t[3] \leftarrow t[3] + N(0, \sigma)$;
        fim se
    senão
        Alterar o *operador* do termo t de acordo com Tabela 2 (Ver seção 3.7.1.2);
    fim se
fim se

O parâmetro $N(0, \sigma)$ refere-se a uma amostra de uma distribuição normal (gaussiana) de média igual a 0 e desvio padrão σ. $U(0, 1)$ corresponde a um valor aleatório no intervalo $[0, 1]$ de uma distribuição uniforme.

É importante destacar que o algoritmo não implementa nenhum mecanismo de mutação que altere o rótulo (nicho) das partículas. Os três tipos de mutação nas regras das partículas implementados pelo mDPSO são detalhados nas seções 3.7.1.1, 3.7.1.2 e 3.7.1.3.

3.7.1.1 Adicão de Novo Termo

Durante a recombinação da partícula, foi percebido que as partículas teriam, no máximo, regras com tamanho de $(\lceil log(N) \rceil + 1)$ termos, onde N é o número total de atributos (Ver Algoritmo 7).

Diante deste fato, o mDPSO implementa um mecanismo que aumenta o número de termos de uma regra, objetivando construir regras com maior complexidade (tamanho) e elevada efetividade. A mutação do tipo "adição de novo termo" é exemplificada na Figura 6 com adição do termo $(Atributo5 < Valor5)$ à regra.

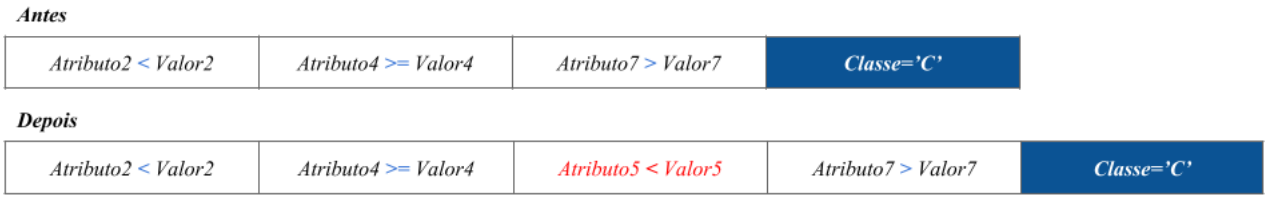

Figura 6 – Exemplo de mutação por adição de novo termo na regra codificada pela partícula.

Cabe ressaltar que o mDPSO não implementa nenhum mecanismo que permita a alteração do elemento *atributo* para tríades do tipo *<atributo, operador, valor numérico>* e do tipo *<atributo, operador, outro atributo>*. Assim, este é o único operador implementado pelo mDPSO que permite a introdução de novos atributos que não foram avaliados em nenhuma partícula do enxame. Por outro lado, os elementos da tríade: "operador" e "valor numérico", podem ser perturbados (mutados) pelos mecanismos de mutação implementados no mDPSO.

3.7.1.2 Mudança de Operador

Outro tipo de mutação implementado pelo mDPSO, ilustrado na Figura 7, permite a alteração de um operador lógico no termo de uma regra da partícula. Em resumo, este operador escolhe aleatoriamente um termo da regra e faz a sua substituição por outro operador segundo a distribuição de probabilidades apresentada na Tabela 2. O operador relacional modificado pelo operador "Mudança de Operador" está realçado na cor vermelha da Figura 7.

3.7.1.3 Adição ou Subtração do Valor Numerico

Esse tipo de mutação ocorre somente quando o termo da regra apresenta estrutura do tipo: *<atributo, operador, valor numérico>*. Em resumo, este operador tem o

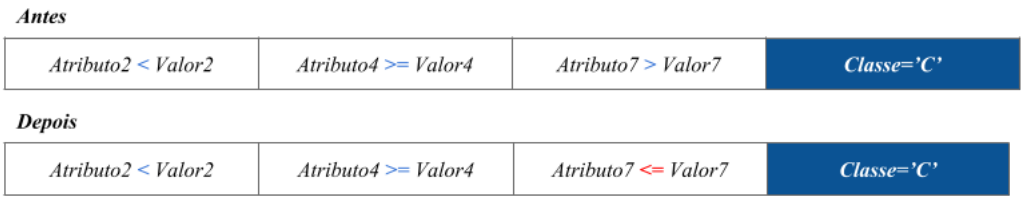

Figura 7 – Exemplo de mutação por mudança de operador.

seguinte funcionamento: (i) É selecionado aleatoriamente um termo da regra codificada na partícula; (ii) É adicionado ou subtraído um valor do terceiro elemento do termo (valor número).

O valor da perturbação, a ser adicionado ou subtraído do elemento "valor numérico" no termo da regra pode ser obtido de duas formas distintas, baseado no trabalho de Jancauskas (2014): (i) Valor aleatório gerado segundo uma distribuição uniforme (linha 12 do Algoritmo 9); (ii) Valor aleatório gerado segundo uma distribuição normal (gaussiana) (linha 9 do Algoritmo 9).

O funcionamento deste operador de mutação proposto é ilustrado na Figura 8: "NovoValor7" refere-se ao valor resultante da adição/substração de um valor numério "Valor7" em um termo da regra.

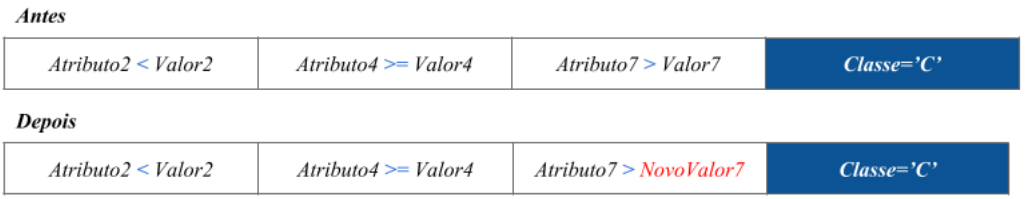

Figura 8 – Exemplo de mutação por mudança de um novo valor numérico.

3.7.2 Operação de Recombinação de Partículas

A recombinação entre as partículas tem o objetivo de combinar características de duas partículas pais com o intuito de gerar regras mais promissoras que as originais. O mecanismo de recombinação de partículas implementado pelo mDPSO é detalhado pelo Algoritmo 10. A variável *pl* é uma partícula genérica, utilizada tanto para designar a partícula líder tanto em $C_k.gbest$ quanto em $p.pbest$ (Ver Algoritmo 8), visto que para ambas as recombinações são realizadas de forma similar.

A representação proposta neste trabalho permite que partículas com tamanhos (número de termos) distintos de regras possam ser recombinadas. Diante disto, o opera-

dor de recombinação implementado pelo mDPSO retorna uma nova regra, resultante da recombinação da partícula corrente p e da partícula líder em $p.pbest$ ou $C_k.gbest$, com variação do número de termos entre 1 e o valor máximo de $[|p|, |p.pbest|, |C_k.gbest|]$, onde $|x|$ retorna o número de termos presente na partícula x.

Cabe mencionar que durante a operação de recombinação a partícula resultante pode sofrer uma redução no número de termos. Tal alteração no número de termos da regra durante a recombinação, deve-se à aleatoriedade do processo de seleção dos termos a serem adicionados à solução filha, bem como o mecanismo de eliminação de termos repetidos implementado pelas partículas no mDPSO.

Algoritmo 10 Recombinação entre partículas

Entrada: p - Partícula; pl - Partícula líder ($gbest$ ou $pbest$).

```
RF ← ∅; // Regra Filha
para k < |pl| faça
    se rand < 0.5 então
        RF ← RF ∪ {Termo aleatório da regra de pl};
    senão
        RF ← RF ∪ {Termo aleatório da regra de p};
    fim se
fim para
para k < |p| faça
    RF ← RF ∪ {Termo aleatório da regra de p};
fim para
devolve RF;
```

A Figura 9 ilustra o processo de recombinação entre duas regras pais e a regra filha resultante da recombinação. As setas posicionadas na parte superior de cada termo pai indicam quais termos foram selecionados aleatoriamente para geração da regra filha.

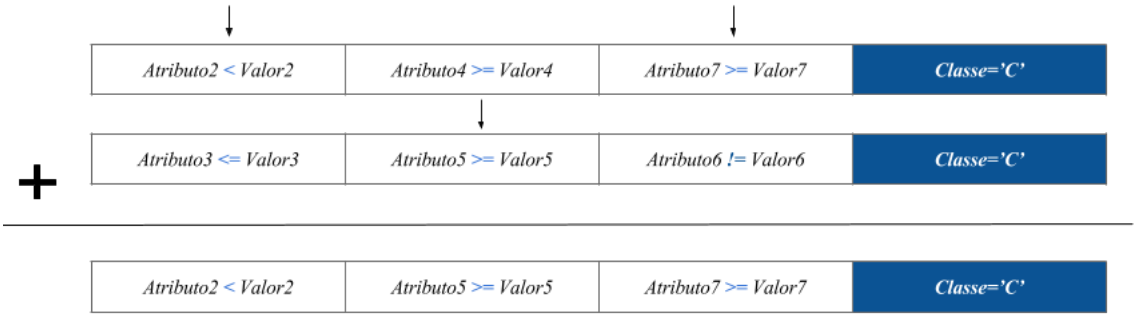

Figura 9 – Ilustração do operador de recombinação entre as partículas implementado pelo mDPSO.

3.8 Busca Local Pareto

Mecanismos de busca local são importantes por possibilitar refinamentos das soluções obtidas por um algoritmo de otimização combinatória. O Algoritmo 11 detalha o mecanismo de busca local Pareto, baseado no método *Neutral Improvement Pareto* (NPI) (Ver Algoritmo 4) executado ao final de cada iteração do mDPSO com objetivo de introduzir novas soluções não-dominadas no repositório $C_k.gbest$.

Durante a implementação da rotina de busca local Pareto proposta, observou-se que o NPI foi a abordagem que apresentou os melhores resultados quando comparado com o BFI e FPI, sendo por isso implementado pelo mDPSO.

Além disso, um mecanismo de geração de soluções vizinhas (linha 4 do Algoritmo) baseada na mutação gaussiana é também utilizado. Esta mutação se mostrou eficiente em várias situações estudadas por Higashi e Iba (2003) e para o problema analisado neste trabalho apresentou bons resultados ao ser combinada com NPI na rotina de busca local Pareto proposta.

Algoritmo 11 Busca Local Pareto

Entrada: p - Partícula; N - Número de atributos da base de dados; A - Conjunto de soluções não-dominadas de $C_k.gbest$;

1: $L \leftarrow \left\lceil \log_2(N) \right\rceil + 1$;
2: $i \leftarrow 0$;
3: **enquanto** $i < L$ **faça**
4: $s' \leftarrow$ Mutação(p, gaussiana);
5: // Se a regra associada a s' foi gerada anteriormente $s'.visited$
6: **se** $s'.visited$ é TRUE **então**
7: **continue**;
8: **fim se**
9: $s'.visited \leftarrow$ TRUE;
10: **se** $f(s') \parallel f(p) \vee f(s') \prec f(p)$ **então** // $\parallel$ - não há relação de dominância
11: $C_k.gbest \leftarrow merge(C_k.gbest, \{s'\})$;
12: $i \leftarrow L$;
13: **fim se**
14: $i \leftarrow i + 1$;
15: **fim enquanto**

3.9 Conclusão

Este capítulo descreve o algoritmo biobjetivo, denominado mDPSO, baseado no algoritmo PSO para classificação de dados por meio de extração de regras. Os objetivos (efetividade e complexidade) foram considerados a fim de obter boas regras de classifica-

ção e evitar o *overfitting*. Outra vantagem desta abordagem é a busca por regras menores, mais fáceis de serem entendidas e interpretadas.

Além do mais, o algoritmo proposto implementa várias funcionalidades, tais como, por exemplo, técnicas de nicho, rotina para geração da população inicial, mecanismos de recombinação e mutação específicos bem como um mecanismo de busca local Pareto.

4 Resultados Experimentais

O presente capítulo avalia o desempenho do **mDPSO** frente aos algoritmos implementados pelo WEKA: **J48** (Árvore de Decisão) (QUINLAN, 1986), Redes Neurais do tipo Função de Base Radial (**RBF**, *Radial Basis Function*) (BROOMHEAD; LOWE, 1988) e o **SMO** (*Sequential Minimal Optimization*) (KEERTHI et al., 2001), uma variante da Máquina de Vetores de Suporte (SVM, *Support Vector Machine*) em 07 (sete) bases de dados selecionadas para testes.

4.1 Introdução

O mDPSO foi implementado utilizando a linguagem *Java* 1.8 64-bits integrado com o banco de dados *PostgreSQL* 9.1 para armazenamento e manipulação das base de dados utilizadas. Os testes foram realizados em um PC Intel (R) Core (TM) i3-4330 CPU 3.50G Hz com 8.00GB de memória RAM e sistema operacional *Windows 10* 64-bits.

A metodologia de análise dos resultados é similar à utilizada por Pereira (2012). Os seguintes algoritmos J48, RBF e SMO foram implementados pelo *software* WEKA e avaliados considerando somente o produto da sensibilidade pela especificidade global. Este cálculo é realizado da seguinte forma: se determinado algoritmo para a base *Wine* obteve os seguintes resultados: 0.93 para classe A; 0.90 para B e 0.99 para classe C, o produto da sensibilidade pela especificidade global é calculado como a média dos desempenhos nas três classes: $(0.93 + 0.90 + 0.99)/3 = 0.94$.

O mDPSO, por sua vez, realiza o mesmo cálculo utilizando a melhor regra de cada classe (nicho), baseada no valor do primeiro objetivo $f_1(I, X)$ (Ver Eq. 3.2). Dessa forma, em uma base de dados composta por três classes, o mDPSO determina três classificadores distintos, um para cada classe. Como afirma Handl e Knowles (2007), espera-se que os resultados gerados pelos algoritmos multiobjetivo sejam tão bons (ou melhores) do que aqueles gerados por algoritmos mono-objetivo. No caso específico deste trabalho, é esperado que os resultados gerados pelo mDPSO sejam comparáveis com os resultados obtidos com os algoritmos estudados, implementados pelo WEKA.

Foram avaliados os desempenhos de 04 (quatro) algoritmos (mDPSO, J48, SMO e RBF) para 07 (sete) bases de dados, caracterizadas a seguir:

- **Diabetes**: Possui 8 atributos numéricos e 768 instâncias alocadas em duas classes:

tested positive (500) e *tested negative* (268);

- **DGA 1**: Possui 5 atributos numéricos e 149 instâncias divididas em três classes: *Normal* (84), *Falha Elétrica* (62) e *Falha Térmica* (78).
- **DGA 2**: Possui 5 atributos numéricos e 224 instâncias divididas em três classes: *Normal* (122), *Falha Elétrica* (10) e *Falha Térmica* (17).
- **Hepatitis**: Possui 19 atributos numéricos e 155 instâncias distribuídas em duas classes: *to die* (32) e *to live* (123);
- **Ionosphere**: Possui 30 atributos numéricos e 351 instâncias distribuídas nas classes: *good* (225) e *bad* (126);
- **Unbalanced**: Possui 32 atributos numéricos e 856 instâncias distribuídas em duas classes: *active* (12) e *inactive* (844).
- **Wine**: Possui 13 atributos numéricos e 178 instâncias divididas em três classes: A (59), B (71) e C (48);

Sendo as bases de dados *DG1*, *DG2*, *Hepatitis* e *Wine* foram as mesmas analisadas no trabalho de Pereira et al. (2014).

Durante a avaliação do desempenho dos algoritmos utilizou-se a *validação cruzada k=10-fold* (10 partições) que consiste em dividir a base de dados em k subconjuntos, sendo *k-1* partições para treinamento e 01 (uma) para teste. Este processo de treinamento e teste é repetido com todos os k subconjuntos e a média dos resultados obtidos é utilizada como indicador da qualidade do modelo (CASTRO; FERRARI, 2016).

A validação cruzada utilizada é estratificada e consiste em distribuir uniformemente as classes das amostras entre as partições. Por exemplo, dado um conjunto de dados dividido em duas classes e sabendo que a *classe A* possui 20% dos objetos e a *classe B* possui 80% das amostras. Ao se fazer a separação da base de treinamento e teste é garantido a manutenção da mesma proporção entre as k partições. Para os testes realizados, as mesmas partições são utilizadas em todos os algoritmos de maneira a tentar prover uma justa comparação entre todos os algoritmos analisados.

Com objetivo de atestar o desempenho dos algoritmos: mDPSO, J48, SMO e RBF, segundo o produto da sensibilidade pela especificidade global, foram aplicados os seguintes testes estatísticos: Teste *Kolgomorov-Smirnov* para verificação da normalidades das amostras; teste *One-Way Analysis of Variance* (ANOVA) e o teste *post-hoc Tukey* para construção de um *ranking* de desempenho dos algoritmos - O *ranking* pode indicar

performance superior, similar (empate) ou inferior entre os algoritmos. É importante frisar que os testes devem ser realizados na ordem apresentada.

Para realização dos testes *post-hoc Tukey*, adotou-se o valor de confiança igual a 95% (o mesmo adotado no trabalho de Pereira (2012)). Logo, se o p-valor retornado pelo teste for inferior a 0.05, sugere-se que o desempenho do par de algoritmos analisados é estatisticamente diferente. Do contrário, não há diferença estatística significativa entre ambos. Dessa maneira, se os desempenhos de ambos os algoritmos não apresentarem diferenças significativas entre eles, então ambos são alocados na mesma posição do *ranking*.

Segundo Carrano, Wanner e Takahashi (2011), a avaliação de desempenho de algoritmos não-determinísticos, tais como Algoritmos Evolutivos, não podem ser realizadas utilizando resultados de algoritmos determinísticos para cada instância do problema. A natureza estocástica desses métodos introduz uma variabilidade aleatória na resposta fornecida pelo algoritmo, uma vez que a solução obtida pelo mesmo pode variar consideravelmente a cada execução. Ainda que a mesma solução seja obtida, o tempo computacional necessário para alcançar tal solução é geralmente diferente em diferentes execuções do mesmo algoritmo.

4.2 Parâmetros dos Algoritmos

Para o mDPSO, os parâmetros ω, c_1 e c_2 foram definidas com os seguintes valores de 0.9, 0.8 e 0.8 respectivamente, sendo tais valores escolhidos por meio de experimentações. Foram utilizadas 90 partículas com critério de parada 30000 avaliações da função-objetivo em todas as bases de dados analisadas neste trabalho.

Os parâmetros utilizados pelos 03 (três) algoritmos implementados pela ferramenta WEKA, para cada base de dados, são listados na Tabela 3. A escolha destes parâmetros foi inspirada pelo trabalho de Pereira et al. (2014), sendo utilizadas as melhores configurações obtidas por meio de experimentações durante o desenvolvimento deste trabalho:

4.3 Análise Baseada na Métrica Sensibilidade $\times$ Especificidade Global

Nesta seção do trabalho, faz-se a descrição dos resultados obtidos com cada algoritmo utilizando a métrica sensibilidade $\times$ especificidade para cada base de dados. Nesta

Algoritmo	Parâmetros	Diabe.	DGA 1	DGA 2	Hepat.	Ionos.	Unbal.	Wine
J48	*Pruned*	Sim	Sim	Sim	Sim	Sim	Sim	Sim
	Confidence factor	0.25	0.025	0.25	0.025	0.25	0.25	0.25
SMO	*Complexity parameter C*	2.0	3.0	4.0	1.0	2.0	2.0	2.0
	Kernel function	PolyKernel	PolyKernel	PolyKernel	PolyKernel	PolyKernel	PolyKernel	PolyKernel
	Function exponent	2.0	3.0	1.0	2.0	2.0	2.0	2.0
RBF	*Clusters*	3	2	4	3	3	3	3
	Min. std. dev. clusters	0.1	0.01	0.01	0.1	0.1	0.1	0.1

Tabela 3 – Parâmetros de configuração do WEKA em cada base de dados.

fase do trabalho, somente serão avaliados os resultados gerados pelos algoritmos mDPSO e os algoritmos implementados no WEKA.

O produto da sensibilidade pela especificidade global relativo às 50 execuções (amostras) de cada algoritmo, na base de dados *Diabetes*, é apresentado na Figura 10. Os resultados apresentados pelo gráfico mostram que o mDPSO apresentou o melhor desempenho para a base *Diabetes*, com média superior aos demais. Observa-se também que os valores médios obtidos para a métrica sensibilidade × especificidade entre os algoritmos J48 e SMO ficaram bem próximos, com uma pequena vantagem para o SMO (Ver Tabela 13), chamando a atenção o desvio padrão obtido pelo J48, bem superior ao obtido pelo SMO. O RBF foi o algoritmo que apresentou o pior desempenho.

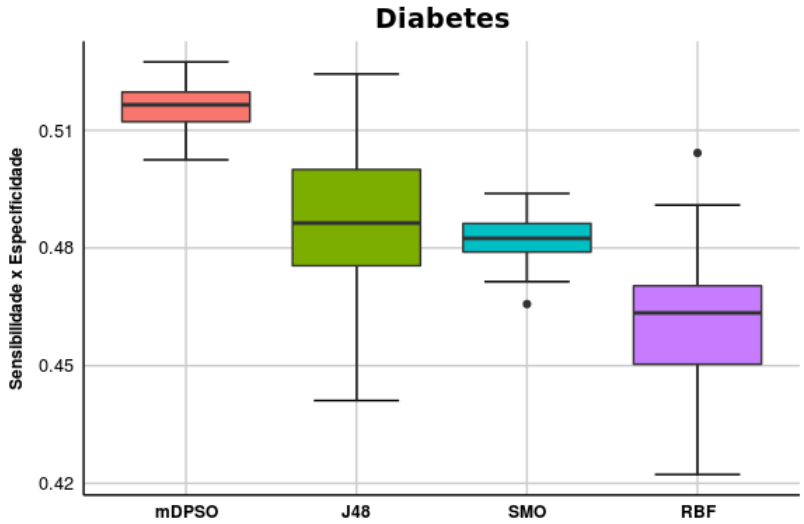

Figura 10 – Desempenho dos algoritmos na base de dados *Diabetes* utilizando a métrica sensibilidade × especificidade global.

Para todas as bases de dados analisadas, o p-valor obtido pelo teste ANOVA (com valor de confiança adotado de 95%) foi de aproximadamente 0 (zero). Deste modo, a Ta-

bela 4 apresenta a relação entre os pares de algoritmos de acordo com o p-valor obtido pelo teste *Tukey* para a base *Diabetes*. Os resultados mostram que não houveram diferenças significativas do desempenho pelo produto da sensibilidade pela especificidade global para as 50 execuções entre os algoritmos J48 e SMO, o que sugere que ambos os algoritmos J48 e SMO apresentaram desempenho similar para esta base de dados. O mDPSO não apresentou diferenças significativas em relação a todos algoritmos implementados pelo WEKA.

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.0000	0.0000
J48	0.0000	-	0.2804	0.0000
SMO	0.0000	0.2804	-	0.0000
RBF	0.0000	0.0000	0.0000	-

Tabela 4 – p-valor medido para o produto da sensibilidade pela especificidade global de cada algoritmo na base de dados *Diabetes*.

De forma semelhante, analisou-se o desempenho dos algoritmos nas demais bases de dados. As Figuras 11 e 12 ilustram o desempenho de cada algoritmo para as bases *DGA 1* e *DGA 2* respectivamente. Percebe-se que em ambas as bases de dados, o mDPSO gerou resultados superiores aos demais algoritmos. Também nota-se que os resultados gerados pelo SMO foram os que mais se aproximaram dos resultados do mDPSO na base *DGA 1*. Ainda nesta base, o pior desempenho foi o do algoritmo J48. E na base *DGA 2*, os resultados gerados pelo mDPSO foram bem superiores aos dos demais algoritmos. Por outro lado, o algoritmo SMO apresentou o pior desempenho nesta base que se caracteriza por ser bastante desbalanceada.

A Tabela 5 apresenta a relação entre os pares de algoritmos de acordo com o p-valor obtido do teste *Tukey* para a base *DGA 1*. Observa-se que apesar do desempenho superior do mDPSO, não houveram diferenças significativas entre os desempenhos dos algoritmos mDPSO e SMO. O mesmo comportamento foi observado entre os algoritmos J48 e RBF, ficando estes abaixo do desempenho obtido pelo algoritmos mDPSO e SMO.

Na Tabela 6 é apresentada a relação entre os pares de algoritmos de acordo com o p-valor obtido para a base *DGA 2*. Os resultados apresentados mostram que os valores gerados pelos algoritmos mDPSO são distintos dos demais algoritmos. Os resultados também sinalizam que não se pode afirmar que o algoritmo J48 possui desempenho superior ao desempenho do RBF. Diferentemente da base *DGA 1*, o SMO foi o que apresentou o

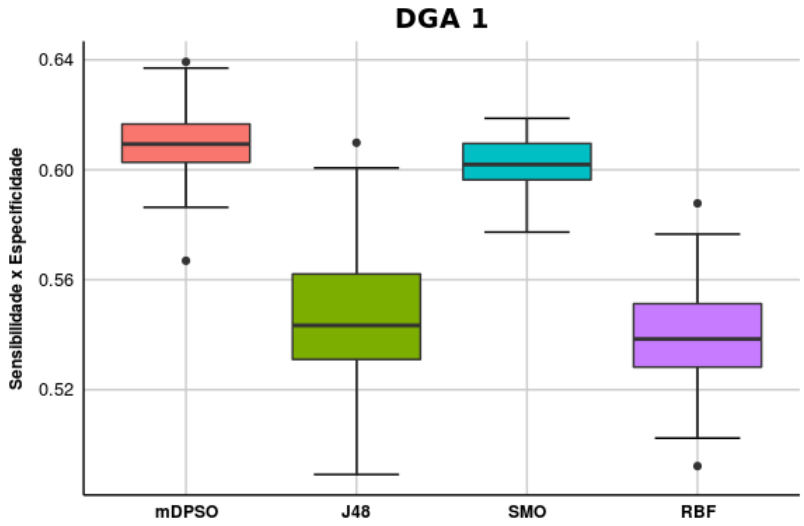

Figura 11 – Desempenho de cada algoritmo na base de dados *DGA 1* utilizando a métrica sensibilidade × especificidade global.

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.0684	0.0000
J48	0.0000	-	0.0000	0.0938
SMO	0.0684	0.0000	-	0.0000
RBF	0.0000	0.0938	0.0000	-

Tabela 5 – p-valor medido para o produto da sensibilidade pela especificidade global de cada algoritmo na base de dados *DGA 1*.

pior desempenho, isso possivelmente devido ao grau de desabalanceamento da base *DGA 2* em comparação a base *DG1*.

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.0000	0.0000
J48	0.0000	-	0.0000	0.8481
SMO	0.0000	0.0000	-	0.0000
RBF	0.0000	0.8481	0.0000	-

Tabela 6 – p-valor medido para o produto da sensibilidade × especificidade global de cada algoritmo na base de dados *DGA 2* global.

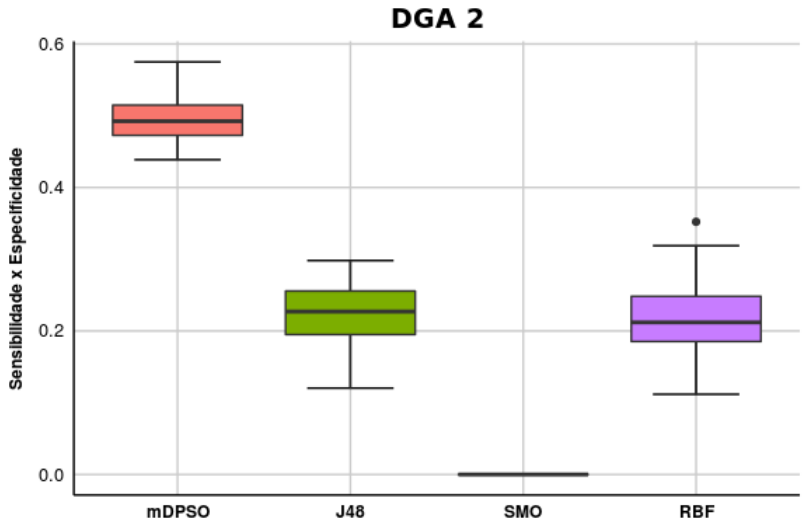

Figura 12 – Desempenho de cada algoritmo na base de dados *DGA 2* utilizando a métrica sensibilidade × especificidade global.

Para a base *Hepatitis*, o gráfico da Figura 13 mostra o bom desempenho dos algoritmos RBF, mDPSO e SMO, sendo possível perceber o desempenho considerado superior do algoritmo RBF. Durante a avaliação entre os pares de algoritmos segundo os p-valores obtidos para esta base e conforme apresentados na Tabela 7, revelam que não houveram diferenças estatísticas significativas entre o mDPSO e o SMO, constatando o desempenho superior do RBF. O algoritmo J48 foi o que apresentou o pior desempenho.

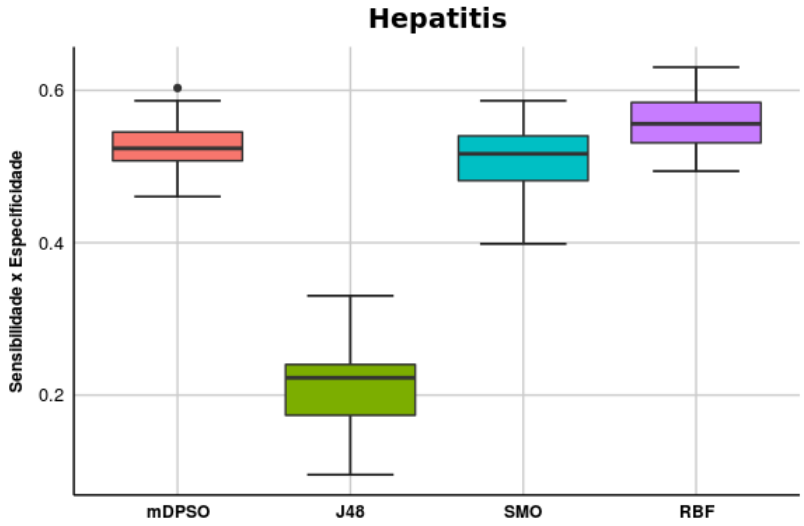

Figura 13 – Desempenho de cada algoritmo na base de dados *Hepatitis* utilizando a métrica sensibilidade × especificidade global.

Testes na base *Ionosphere* mostraram o desempenho ruim do mDPSO frente aos demais algoritmos implementados pelo WEKA. Além disso, como mostrado na Figura 14,

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.1767	0.0028
J48	0.0000	-	0.0000	0.0000
SMO	0.1767	0.0000	-	0.0000
RBF	0.0028	0.0000	0.0000	-

Tabela 7 – p-valor medido para o produto da sensibilidade pela especificidade global de cada algoritmo na base de dados *Hepatitis*.

percebe-se com certa facilidade que o algoritmo RBF apresentou desempenho superior em relação às outras implementações. Ademais, de acordo com a Tabela 8 os algoritmos apresentaram diferenças estatísticas significativas entre si, caracterizando desempenhos diferenciados entre os algoritmos para esta base de dados.

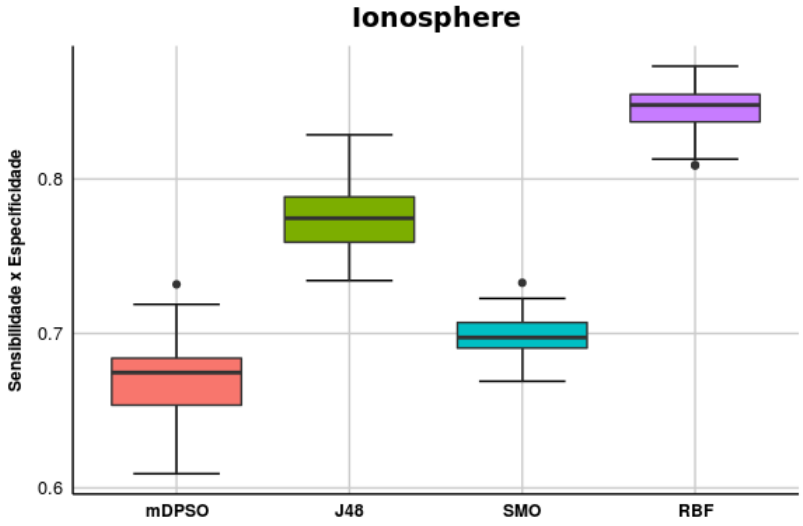

Figura 14 – Desempenho de cada algoritmo na base de dados *Ionosphere* utilizando a métrica sensibilidade × especificidade global.

Diante do baixo desempenho do algoritmo mDPSO na base *Ionosphere*, analisou-se o produto da sensibilidade pela especificidade de cada classe (classes *good* e *bad*) desta base para cada algoritmo. Os resultados obtidos, listados na Tabela 9, mostram que, apesar do mDPSO obter um bom desempenho para a classe *good*, os resultados obtidos para a classe *bad* influenciam negativamente o desempenho geral do algoritmo. É importante comentar que os resultados semelhantes gerados pelos algoritmos J48, SMO e RBF devem-se ao cálculo do produto da especificidade pela sensibilidade em bases binárias (com duas classes). Como o mDPSO utiliza técnicas de nicho e são escolhidas as melhores regras de

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.0000	0.0000
J48	0.0000	-	0.0000	0.0000
SMO	0.0000	0.0000	-	0.0000
RBF	0.0000	0.0000	0.0000	-

Tabela 8 – p-valor medido para o produto da sensibilidade pela especificidade global de cada algoritmo na base de dados *Ionosphere*.

cada nicho, consequentemente, cada classe (nicho) possui um classificador distinto com desempenhos diferentes.

	bad	**good**
mDPSO	0.5081 (0.0406)	0.8340 (0.0208)
J48	0.7752 (0.0229)	0.7752 (0.0229)
SMO	0.6988 (0.0132)	0.6988 (0.0132)
RBF	**0.8451 (0.0145)**	**0.8451 (0.0145)**

Tabela 9 – Desempenho de cada algoritmo usando o produto da sensibilidade pela especificidade em cada classe na base *Ionosphere* global.

Para o conjunto de bases analisadas neste trabalho, a base *Unbalanced* caracteriza-se como a mais desabalanceada. E novamente, tal como ocorre na base *DGA 2*, o mDPSO apresentou resultados superiores em relação aos demais algoritmos. Possivelmente, o desempenho ruim destes algoritmos deve-se ao fato do alto grau de desbalanceamento deste base de dados, uma vez que mostram que eles não foram capazes de identificar as duas classes da base, sendo penalizados quando avaliado o produto da sensibilidade pela especificidade.

Na Tabela 10 é apresentada a relação entre os pares de algoritmos de acordo com o p-valor obtido para a base *Unbalanced*. Os dados mostram que os resultados do mDPSO não possuem nenhuma similaridade com os dos outros três algoritmos.

Por fim, a Figura 16 mostra o gráfico de desempenho dos algoritmos para a base *Wine*, segundo os valores obtidos para a métrica sensibilidade $\times$ especificidade global. A figura ilustra o bom desempenho dos algoritmos SMO e RBF frente aos outros dois (mDPSO e J48).

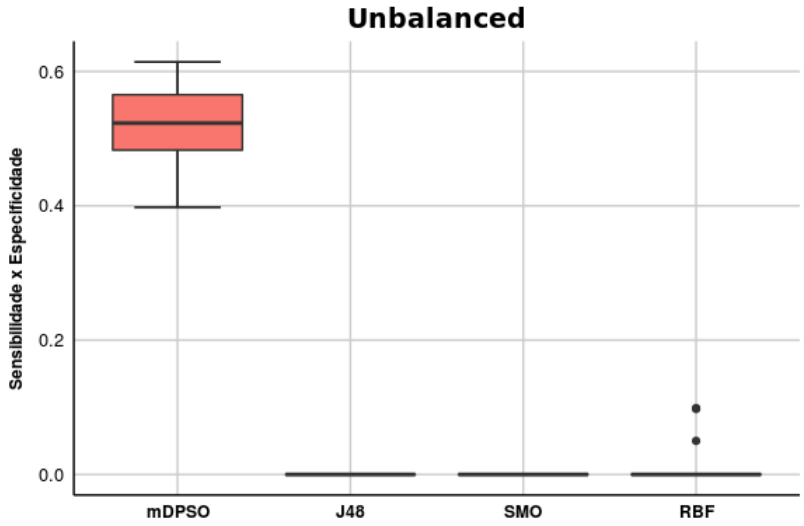

Figura 15 – Desempenho de cada algoritmo na base de dados *Unbalanced* utilizando a métrica sensibilidade × especificidade global.

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0000	0.0000	0.0000
J48	0.0000	-	1.0000	0.8599
SMO	0.0000	1.0000	-	0.8599
RBF	0.0000	0.8599	0.8599	-

Tabela 10 – p-valor medido para o produto da sensibilidade pela especificidade global de cada algoritmo na base de dados *Unbalanced*.

Além disso, ao avaliar os pares de algoritmos segundo o p-valor computado para a base *Wine*, conforme descrito na Tabela 11, verifica-se que não houveram diferenças estatísticas significativa entre os algoritmos SMO e o RBF. Novamente, o J48 apresentou o pior desempenho. Para esta base, o mDPSO obteve desempenho considerado intermediário, com diferenças estatísticas de desempenho entre os demais algoritmos implementados pelo WEKA.

Mais uma vez, motivado pelo desempenho ruim do mDPSO na base *Wine*, investigou-se o desempenho dos algoritmos para cada classe desta base. Os resultados obtidos são exibidos na Tabela 12. Um aspecto que chama a atenção em relação a base *Wine* é que o mDPSO apresenta um desempenho ruim por classe, especialmente para a classe B, na qual apresentou o pior desempenho entre os algoritmos analisados, influenciando negativamente no desempenho do mDPSO.

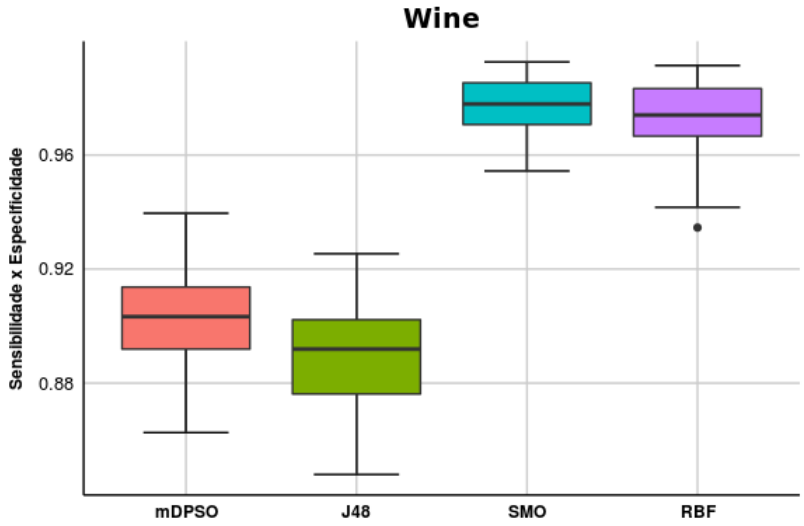

Figura 16 – Desempenho de cada algoritmo na base de dados *Wine* utilizando a métrica sensibilidade × especificidade global.

	mDPSO	**J48**	**SMO**	**RBF**
mDPSO	-	0.0617	0.0000	0.0000
J48	0.0617	-	0.0000	0.0000
SMO	0.0000	0.0000	-	0.4286
RBF	0.0000	0.0000	0.4286	-

Tabela 11 – p-valor medido para o produto da sensibilidade × especificidade global de cada algoritmo na base de dados *Wine*.

	Classe A	**Classe B**	**Classe C**
mDPSO	0.9487 (0.0190)	0.8178 (0.0357)	0.9426 (0.0229)
J48	0.9255 (0.0223)	0.8662 (0.0228)	0.8825 (0.0254)
SMO	**0.9937 (0.0091)**	0.9580 (0.0155)	0.9787 (0.0109)
RBF	0.9643 (0.0142)	**0.9664 (0.0164)**	**0.9889 (0.0131)**

Tabela 12 – Desempenho dos algoritmos segundo a métrica sensibilidade × especificidade para cada classe da base *Wine*.

Finalmente, na Tabela 13 é apresentada a média (desvio padrão) das 50 execuções do produto da sensibilidade pela especificidade global dos 05 (cinco) algoritmos para as 07 (sete) bases de dados analisadas neste trabalho. Observa-se que apesar do desempenho ruim nas bases *Ionosphere* e *Wine*, o mDPSO conseguiu resultados considerados bons

nas demais bases de dados. Particularmente, para as bases de dados *Diabetes*, *DGA 1* e *Unbalanced*, em que apresentou os melhores indicadores de desempenho. Além do mais, apesar do bom desempenho do algoritmo PG na base *DGA 2*, o desvio padrão calculado para este algoritmo é superior (aproximadamente 435%) ao desvio computado para o mDPSO, o que mostra o bom desempenho deste algoritmo para esta base. Para facilitar a compreensão dos dados da tabela, negritou-se os melhores resultados obtidos para cada base de dados.

	Diabe.	**DGA 1**	**DGA 2**	**Hepat.**	**Ionos.**	**Unbal.**	**Wine**
mDPSO	**0.5159 (0.0054)**	**0.6095 (0.0121)**	0.4935 (0.0304)	0.5275 (0.0336)	0.6710 (0.0230)	**0.5171 (0.0591)**	0.9030 (0.0151)
J48	0.4873 (0.0198)	0.5471 (0.0237)	0.2240 (0.0402)	0.2106 (0.0501)	0.7752 (0.0229)	0.0000 (0.0000)	0.8914 (0.0180)
SMO	0.4825 (0.0058)	0.6010 (0.0104)	0.0000 (0.0000)	0.5107 (0.0423)	0.6988 (0.0132)	0.0000 (0.0000)	**0.9768 (0.0092)**
RBF	0.4613 (0.0160)	0.5390 (0.0194)	0.2182 (0.0491)	**0.5564 (0.0356)**	**0.8451 (0.0145)**	0.0049 (0.0204)	0.9732 (0.0124)
PG*	-	0.5725 (0.1139)	**0.5409 (0.1325)**	0.5053 (0.2746)	-	-	0.9634 (0.0422)

Tabela 13 – Desempenho de cada algoritmo segundo o produto da sensibilidade pela especificidade global para cada base de dados.

A Tabela 14 apresenta o *ranking* de desempenho dos 04 (quatro) algoritmos para cada base de dados, de acordo estratégia similar à utilizada no trabalho de Pereira (2012). Ao analisar os resultados exibidos na Tabela 14, observa-se o desempenho considerado satisfatório do algoritmo mDPSO para classificação de dados nas várias bases de dados selecionadas. Em resumo, o mDPSO obteve o melhor desempenho em 04 (quatro) das 07 (sete) bases avaliadas (*Diabetes*, *DGA 1*, *DGA 2* e *Unbalanced*). Para a base de dados *DGA 1* o mDPSO obteve desempenho similar ao algoritmo SMO e superior aos algoritmos J48 e o RBF. Atenta-se também para o desempenho considerado ruim apresentado pelo algoritmo J48 que não conseguiu se sobressair em nenhuma das bases de dados analisadas, geralmente ocupando posições intermediárias.

	Diabe.	**DGA 1**	**DGA 2**	**Hepat.**	**Ionos.**	**Unbal.**	**Wine**
mDPSO	**A**	**A**	**A**	**B**	**D**	**A**	**B**
J48	**B**	**B**	**B**	**C**	**B**	**B**	**C**
SMO	**B**	**A**	**C**	**B**	**C**	**B**	**A**
RBF	**C**	**B**	**B**	**A**	**A**	**B**	**A**

Tabela 14 – *Ranking* de desempenho de cada algoritmo.

Cabe ressaltar que apesar de constar desempenho do algoritmo PG para algumas bases de dados não foi possível analisá-lo para inclusão no *ranking* de desempenho.

Assim, de maneira geral, foi possível observar que o mDPSO apresentou resultados considerados positivos, mostrando-se competitivo quando comparado à outros métodos clássicos encontrados na literatura especializada, sendo importante destacar também o bom desempenho obtido pelo mesmo em relação às bases de dados desbalanceadas quando comparado aos demais métodos analisados implementados pelo WEKA.

4.4 Conclusão

Neste capítulo foram apresentados os resultados obtidos em relação ao mDPSO e 03 (três) algoritmos clássicos (Árvore de Decisão, RBF e SVM) implementados pelo WEKA em 07 (sete) bases de dados previamente selecionadas.

Durante os experimentos, o mDPSO chamou a atenção pelo desempenho considerado satisfatório, uma vez que os resultados revelam que o algoritmo se mostrou promissor, principalmente ao se analisar o seu desempenho em bases de dados desbalanceadas. Em relação às demais bases de dados, mostrou-se competitivo com resultados comparáveis com a Árvore de Decisão, RBF e SVM implementadas pelo WEKA em várias bases de dados analisadas no decorrer dos experimentos analisados.

5 Considerações finais

A classificação de dados é um problema importante dentro da área de mineração de dados. Neste contexto, o presente trabalho teve por objetivo propor uma abordagem multiobjetivo baseada em PSO para classificação de dados por meio de extração de regras.

Denominado de mDPSO, o algoritmo foi comparada com três métodos clássicos, reconhecidos na literatura e implementados pela ferramenta de mineração de dados WEKA, em 07 (sete) bases de dados selecionadas para testes.

A partir dos resultados obtidos, observou-se que o mDPSO se mostrou competitivo, com desempenho considerado promissor, principalmente em bases de dados desbalanceadas, como apresentado durante a análise dos experimentos desenvolvidos neste trabalho.

Foram implementados uma rotina de geração da população inicial, um mecanismo de busca local Pareto, adaptado do NPI, para refinamento das soluções encontradas e operadores de recombinação e mutacção específicos ao problema de classificação de dados estudado. No entanto, percebe-se que alguns pontos do mDPSO ainda requerem maior estudo, visto que certas funcionalidades foram desenvolvidas por meio de testes e experimentações, como por exemplo, as taxas de probabilidades atribuidas a cada operador lógico utilizadas durante a geração da população inicial e durante a operação de mutação em um operador na regra da partícula.

Outro aspecto em relação ao mDPSO que é imporante mencionar, refere-se a capacidade do algoritmo atualmente em apenas manipular atributos númericos, sendo necessária a realização de tarefas de pré-processamento para a utilização de atributos textuais ou mesmo de atributos não-convencionais (geográficos). Para o futuro é esperado estender tais funcionalidades.

De maneira geral, é possível notar que o mDPSO obteve resultados considerados positivos, devido ao seu desempenho considerado satisfatório quando comparados aos algoritmos estudados neste trabalho e implementados pelo WEKA. É importante também destacar a capacidade do algoritmo proposto em lidar com bases de dados desbalanceadas, visto que boa parte dos problemas encontrados hoje possuem algum grau de desbalanceamento, logo o presente trabalho se torna relevante neste sentido, não somente do ponto de vista científico, como também do ponto de vista prático.

5.1 Trabalhos Futuros

Como trabalhos futuros, pode-se citar:

- Aprimorar o mecanismo de busca local Pareto do mDPSO visando refinar a qualidade das soluções encontradas pelo algoritmo. A motivação para este fato se deve ao baixo desempenho apresentado pelo mDPSO durante a classificação de dados em algumas bases de dados analisadas quando comparados aos algoritmos do WEKA selecionados neste trabalho;
- Em determinadas situações, algumas rotinas implementados do mDPSO foram desenvolvidos por meio de experimentações, dessa forma, estudos mais aprofundados poderiam auxiliar no aprimoramento destas funcionalidades;
- É esperado estender o mDPSO para classificação de dados híbridos (banco de dados geográficos), uma vez que são poucos os trabalhos referentes à classificação de dados não-convencionais encontrados na literatura. Acredita-se que o algoritmo poderia incluir tal funcionaliade sem grandes alterações em sua implementação atual;
- Outro aspecto importante seria estender o mDPSO para as três variantes propostas por Pereira et al. (2014) para comparação com três algoritmos (Árvore de Decisão, RBF e SVM) implementados pelo WEKA em seu trabalho. O mDPSO somente implementa o seguinte mecanismo (*One rule matching*) a fim de classificar o conjunto de testes. Deste modo, um comparativo geral entre os três mecanismos (*One rule matching*, *Counting of matching rules* e *Weighted count of matching rules*) poderia fornecer um melhor comparativo entre o mDPSO e as abordagens analisadas;
- Uma investigação de desempenho do mDPSO com outras bases de dados também seria relevante. Além do mais, outros algoritmos reconhecidos na literatura poderiam ser utilizados em análises e comparações de desempenho.

Referências Bibliográficas

BASSEUR, M.; BURKE, E. K. Indicator-based multi-objective local search. *Evolutionary Computation*, IEEE, p. 3100–3107, 2007.

BROOMHEAD, D. S.; LOWE, D. Multivariable functional interpolation and adaptive networks. *Complex Systems*, v. 2, p. 321–355, 1988.

CARRANO, E. G.; WANNER, E. F.; TAKAHASHI, R. H. C. A multicriteria statistical based comparison methodology for evaluating evolutionary algorithms. *Evolutionary Computation*, IEEE, v. 15, p. 848–870, 2011.

CASTRO, L. de; FERRARI, D. G. *Introdução à Mineração de Dados: Conceitos básicos, Algoritmos e Aplicações*. [S.l.]: Editora Saraiva, 2016.

CASTRO, L. de; TIMMIS, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. [S.l.]: Springer-Verlag, 2002.

CERVANTES, A.; GALVAIN, I.; ISASI, P. A comparison between the pittsburgh and michigan approaches for the binary pso algorithm. *IEEE Congress on Evolutionary Computation*, IEEE, v. 3, p. 290–297, 2005.

CHEN, M.; LUDWIG, S. A. Discrete particle swarm optimization with local search strategy for rule classification. *Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, p. 162–167, 2012.

CHEN, M.-S.; HAN, J.; YU, P. S. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, v. 8, n. 6, p. 866–883, 1996.

COELLO-COELLO, C. A. Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computation Inteligence Magazine*, v. 8, p. 28–36, 2006.

COELLO-COELLO, C. A.; PULIDO, G. T.; LECHUGA, M. S. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, v. 8, p. 256–279, 2004.

CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. *Manual de computação evolutiva e metaheurística*. [S.l.]: Imprensa da Universidade de Coimbra; Editora da Universidade Federal de Minas Gerais, 2014.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B Cybernetics*, v. 26, p. 29–41, 1996.

DRUGAN, M. M.; THIERENS, D. Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, v. 18, p. 727–766, 2012.

EBERHART, R.; SHI, Y. Particle swarm optimization: Developments, applications and resources. IEEE, v. 1, p. 81–86, 2001.

FAYYAD, U. et al. From data mining to knowledge discovery in databases. 1996.

FAYYAD, U. et al. Knowledge discovery and data mining: Towards a unifying framework. 1996. Acessado em 21 de Setembro de 2015. Disponível em: <http://www.cs.columbia.edu/~gravano/cs6111/Readings/fayyad.pdf>.

FREITAS, A. A. A survey of evolutionary algorithms for data mining and knowledge discovery. 2003.

HANDL, J.; KNOWLES, J. An evolutionary approach to multiobjective clustering. *Trans. Evol. Comp*, IEEE Press, v. 11, n. 1, p. 56–76, fev. 2007.

HASSANI, K.; LEE, W.-S. *An Incremental Parallel Particle Swarm Approach for Classification Rule Discovery from Dynamic Data.* [S.l.]: IEEE Computer Society, 2013. 430–435 p.

HAUBELT, C.; GAMENIK, J.; TEICH, J. *Initial Population Construction for Convergence Improvement of MOEAs.* [S.l.]: Springer-Verlag, 2005. 191–205 p.

HIGASHI, N.; IBA, H. Particle swarm optimization with gaussian mutation. *Swarm Intelligence Symposium*, IEEE, p. 72–79, 2003.

HOFFMANN, M. et al. *Discrete Particle Swarm Optimization for TSP: Theoretical Results and Experimental Evaluations.* [S.l.]: Springer-Verlag, 2011. 416–427 p. (ICAIS'11).

HU, X.; EBERHEART, R. C.; SHI, Y. Particle swarm with extended memory for multiobjective optimization. IEEE, p. 193–197, 2003.

JANCAUSKAS, V. Empirical study of particle swarm optimization mutation operators. *Baltic J. Modern Computing*, v. 2, p. 199–214, 2014.

KEERTHI, S. S. et al. Improvements to platt's smo algorithm for svm classifier design. *Neural Comput.*, MIT Press, v. 13, n. 3, p. 637–649, mar. 2001.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. IEEE, p. 1942–1948, 1995.

KENNEDY, J.; EBERHART, R. A discrete binary version of the particle swarm algorithm. IEEE, v. 5, p. 4104–4108, 1997.

KENNEDY, J.; SHI, Y. Particle swarm optimization: Developments, applications and resources. IEEE, v. 2, p. 81–86, 2001.

KROHLING, R. A. Gaussian swarm: a novel particle swarm optimization algorithm. *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, p. 372–376, 2004.

LIU, Y. et al. *Rule Discovery with Particle Swarm Optimization.* [S.l.]: Springer Berlin Heidelberg, 2004. v. 3309. 291–296 p.

MAIA, R. D. *Colônia de Abelhas como Modelo para Otimização Multimodal em Espaços Contínuos: Um Abordagem Baseada em Alocação de Tarefas*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Minas Gerais, Brasil, 2012. Editora UFMG, Minas Gerais, 2012.

MAIA, R. D.; CASTRO, L. N. de; CAMINHAS, W. M. Optbees - a bee-inspired algorithm for solving continuous optimization problems. *2013 BRICS Congress on 1st Computational BRICS Countries Intelligence Congress & on 11th Computational Brazilian Congress Intelligence on Computational Intelligence*, p. 142–151, 2012.

MENDES, J. B. *Uma Abordagem Multiobjetivo para o Problema de Despacho de Caminhões em Minas a Céu Aberto*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Minas Gerais, Brasil, 2013. Editora UFMG, Minas Gerais, 2013.

MILLONAS, M. M. Swarms, phase transitions, and collective intelligence. 1993. Acessado em 21 de Janeiro de 2017. Disponível em: <http://samoa.santafe.edu/media/workingpapers/93-06-039.pdf>.

MISHRA, A. K. Pso based swarm intelligence technique for multi-objective classification rule mining. *International Journal of Computer Applications*, v. 137, p. 18–22, 2016.

PAN, Q.-K. et al. A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. *Advanced Manufacturing Technology*, Springer-Verlag, v. 38, p. 337–347, 2008.

PARSOPOULOS, K. E.; VRAHATIS, M. N. *(Chapter II) Multi-Objective Particles Swarm Optimization Approaches*. [S.l.]: IGI Global, 2009. 20–41 p.

PEREIRA, M. de A. *Classificação de Dados Híbridos Através de Algoritmos Evolucionários*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Minas Gerais, Brasil, 2012. Editora UFMG, Minas Gerais, 2012.

PEREIRA, M. de A. et al. A niching genetic programming-based multi-objective algorithm for hybrid data classification. *Neurocomputing*, Elvesier, v. 133, p. 342–357, 2014.

PEREIRA, M. de A.; JúNIOR, C. A. D.; VASCONCELOS, J. A. de. A niched genetic programming algorithm for classification rules discovery in geographic databases. *SEAL 2010, Lecture Notes in Computer Science*, p. 260–269, 2010.

QUINLAN, J. R. Induction of decision trees. *Machine Learning 1*, p. 81–106, 1986.

REDDY, M. J.; KUMAR, D. N. Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation. *Wiley InterScience*, v. 21, p. 2897–2909, 2007.

REYES-SIERRA, M.; COELLO-COELLO, C. A. Improving pso-based multi-objective optimization using crowding, mutation and ∈-dominance. Springer-Verlag, p. 505–519, 2005.

REYES-SIERRA, M.; COELLO-COELLO, C. A. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, IEEE, p. 287–308, 2006.

RINI, D. P.; SHAMSUDDIN, S. M.; YUHANIZ, S. S. Particle swarm optimization: Technique, system and challenges. *International Journal of Computer Applications*, v. 14, n. 1, p. 19–27, 2011.

ROSENDO, M.; POZO, A. Applying a discrete particle swarm optimization algorithm to combinatorial problems. *IEEE Computer Society*, IEEE, p. 235–240, 2010.

SANTANA, R. A.; PONTES, M. R.; BASTOS-FILHO, C. J. A. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. IEEE, p. 237–242, 2009.

SOUSA, T.; SILVA, A.; NEVES, A. Particle swarm based data mining algorithms for classification tasks. *Parallel Comput.*, Elsevier Science Publishers B. V., p. 767–783, 2004.

VALLE, Y. del et al. Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, IEEE, p. 171–195, 2008.

WANG, H.; ZHANG, Y. Improvement of discrete particle swarm classification system. IEEE, p. 1027–1031, 2011.

WANG, X.; TANG, L. A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. *Applied Soft Computing*, v. 12, p. 652–662, 2011.

WANG, Z.; SUN, X.; ZHANG, D. A pso-based classification rule mining algorithm. Springer-Verlag, 2007.

WITTEN, I. H.; EIBEN, F. *Data mining: Practical Machine Learning Tools and Techniques*. [S.l.]: Morgan Kaufmann Publishers Inc., 2005.

ZACHARIADIS, E. E.; KIRANOUDIS, C. T. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, v. 37, p. 2089–2105, 2010.

ZAHIRI, S.-H.; SEYEDIN, S.-A. Swarm intelligence based classifiers. *Journal of the Franklin Institute*, v. 344, p. 362–376, 2007.

ZUBEN, F. J. V.; ATTUX, R. R. F. Inteligência de enxame. 2007. Acessado em 21 de Setembro de 2015. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia013_1s07/topico4_07.pdf>.