Universidade Federal de Pernambuco
Centro de Informática

Mestrado em Ciência da Computação

Arquitetura Híbrida para Otimização Multi-Objetivo de SVMs

Péricles Barbosa Cunha de Miranda

Dissertação de Mestrado

Recife
22/02/2013

Universidade Federal de Pernambuco
Centro de Informática

Péricles Barbosa Cunha de Miranda

Arquitetura Híbrida para Otimização Multi-Objetivo de SVMs

Trabalho apresentado ao Programa de Mestrado em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de mestre em Ciência da Computação.

Orientador: *Prof. Dr. Ricardo Bastos Cavalcante Prudêncio*

Recife
22/02/2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Miranda, Péricles Barbosa Cunha de
Arquitetura híbrida para otimização multi-objetivo de SVMs. / Péricles Barbosa Cunha de Miranda. - Recife: O Autor, 2013.
xii, 60 folhas: fig., tab.

Orientador: Ricardo Bastos Cavalcante Prudêncio.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia e apêndice.

1. Inteligência artificial. 2. Aprendizagem de máquina. 3. Otimização. I. Prudêncio, Ricardo Bastos Cavalcante (orientador). II. Título.

006.3 CDD (23. ed.) MEI2013 – 059

Dissertação de Mestrado apresentada por **Péricles Barbosa Cunha de Miranda** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "Arquitetura Híbrida para Otimização Multi-Objetivo de SVMs" orientada pelo Prof. Ricardo Bastos Cavalcante Prudêncio e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Cleber Zanchettin
Centro de Informática / UFPE

_____

Prof. André Luíz Vasconcelos Coelho
Centro de Ciências Tecnológicas / UNIFOR

_____

Prof. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 22 de fevereiro de 2013

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico este trabalho à minha família, amigos e amor, que sempre me apoiam e confortam mesmo nos momentos mais difíceis da vida. Dedico este trabalho em especial ao meu pai Antônio Pádua de Miranda, pessoa fantástica e inesquecível. Amo você para sempre.

Agradecimentos

Agradeço a Deus pela saúde e força para me manter sempre dedicado e focado neste trabalho. Agradeço também a minha família, meus queridos amigos e minha amada noiva. Gostaria de fazer um agradecimento especial ao meu orientador Ricardo Prudêncio e aos professores André Carvalho e Carlos Soares, que me ajudaram no aperfeiçoamento deste trabalho. Com a orientação de vocês conseguimos desenvolver trabalhos de grande qualidade, sendo por sinal reconhecidos internacionalmente.

Resumo

Vem sendo dada grande atenção às Máquinas de Vetores de Suporte (SVMs) devido à sua fundamentação teórica e seu bom desempenho quando comparadas a outros algoritmos de aprendizado em diferentes aplicações. Porém, seu bom desempenho depende fortemente da escolha adequada de seus parâmetros de controle. Como a abordagem de tentativa e erro se torna impraticável devido às combinações entre os possíveis valores dos parâmetros, a seleção de parâmetros passou a ser tratada como um problema de otimização, de modo que o objetivo é encontrar a combinação de valores dos parâmetros mais adequada para um determinado problema. Embora a utilização de algoritmos de otimização e busca automatizem a seleção de parâmetros de SVM, ela pode se tornar inviável caso o número de parâmetros a serem selecionados aumente consideravelmente. Uma alternativa é o uso de Meta-Aprendizado (MA), que trata a tarefa de seleção de parâmetros como uma tarefa de aprendizado supervisionado. Cada exemplo de treinamento para o MA (*meta-exemplo*) armazena características de problemas passados e o desempenho obtido pelas configurações de parâmetros candidatas. Este conjunto de meta-exemplos forma a meta-base, sendo esta utilizada para auxiliar no módulo de sugestão ou meta-aprendiz. O meta-aprendiz tem a função de prever as configurações de parâmetros mais adequadas para um problema novo baseado em suas características. Deste modo, MA se torna uma alternativa menos custosa comparada aos algoritmos de otimização, pois faz uso de execuções passadas no processo de sugestão. Neste trabalho, as sugestões do meta-aprendiz são utilizadas como soluções iniciais da técnica de busca, sendo esta responsável pelo refinamento das soluções sugeridas.

Neste trabalho, foi criada uma arquitetura híbrida multi-objetivo, que combina MA com algoritmos de otimização, inspirados em enxames de partículas, com múltiplos objetivos aplicado ao problema de seleção de parâmetros de SVMs. Os algoritmos de otimização utilizados no experimento foram: MOPSO, MOPSO-CDR, MOPSO-CDRS, CSS-MOPSO, m-DNPSO e MOPSO-CDLS, e os objetivos levados em consideração foram: maximização da taxa de acerto na classificação e minimização do número de vetores de suporte. De acordo com os resultados alcançados, ficou comprovado o potencial do MA na sugestão de soluções para os algoritmos de otimização. O início da busca em regiões promissoras favoreceu a convergência e geração de soluções ainda melhores, quando comparada a aplicação de algoritmos de busca tradicionais. Os Pareto *fronts* gerados foram analisado em 4 perspectivas (*spacing, max. spread, hypervolume* e *coverage*), sendo os resultados da abordagem híbrida superiores aos das técnicas de otimização tradicionais.

Palavras-chave: Otimização Multi-Objetivo, Otimização por Enxame de Partículas, Meta-Aprendizado, Máquinas de Vetor de Suporte, Seleção de Parâmetros

Abstract

Support Vector Machines have achieved a considerable attention due to their theoretical foundations and good empirical performance when compared to other learning algorithms in different applications. However, the SVM performance strongly depends on the adequate calibration of its parameters including, for instance, the kernel function, the values of kernel parameters, the regularization parameter, among others. An exhaustive trial-and-error procedure for selecting good values for the parameters is obviously not practical.

The process of selecting SVM parameters is commonly treated by different authors as an optimization problem in which a search technique is used to find the adequate configurations of parameters to the problem at hand. Although it represents an automatic way to select SVM parameters, this approach can still be very expensive, since a large number of candidate configurations of parameters is often evaluated during the search process.

An alternative approach to SVM parameter selection is the use of Meta-Learning (ML), which treats the SVM parameter selection as a supervised learning task. Each training example for ML (i.e. each meta-example) stores the characteristics of a past problem and information about the performance obtained by a set of candidate configurations of parameters to the problem. By receiving a set of such meta-examples as input, a meta-learner is able to predict the most suitable configuration of parameters for a new problem based on its characteristics. ML is a less expensive solution compared to the search approach. In fact, once the knowledge is acquired by the meta-learner, configurations of parameters can be suggested for new problems without the need of empirically evaluating different candidate configurations.

In this work we created a hybrid multi-objective architecture which combines meta-learning with multi-objective particle swarm optimization algorithms for the SVM parameter selection problem. The optimization algorithms used in the experiment were: MOPSO, MOPSO-CDR, MOPSO-CDRS, CSS-MOPSO, m-DNPSO and MOPSO-CDLS, and the objectives considered were: the success rate and the number of support vectors.

The results of this work presented the potential of meta-learning in the process of suggest solutions to optimization algorithms. The fact that the search be started in promising regions favored convergence and generate even better solutions when compared to application of traditional search algorithms. The results were analysed in 4 perspectives (*spacing, max. spread, hypervolume* and *coverage*), and the results achieved by the hybrid approach overcame the traditional optimization techniques' ones.

Keywords: Multi-Objective Optimization, Particles Swarm Optimization, Meta-Learning, Support Vector Machines, Parameter Selection

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO 1

Introdução

1.1 Motivação

A inteligência artificial está ramificada em diversas subáreas, dentre as quais encontramos a aprendizagem de máquina, cujo foco é desenvolver algoritmos que ofereçam a capacidade de aprender um determinado padrão, através de exemplos ou observações. O aprendizado pode ser conceituado como qualquer processo no qual um sistema melhora seu desempenho, através da experiência [1]. Para tanto, conta com vários algoritmos e técnicas, dentre eles as máquinas de vetores de suporte (SVM).

A SVM é um algoritmo desenvolvido inicialmente pelo pesquisador Boser et al. em 1992 [2], a qual permite o reconhecimento de padrões tanto em nível de classificação quanto regressão de dados [3]. A SVM é fundamentada na teoria de aprendizado estatístico (TAE), que define a maximização da generalização através de termos matemáticos. A SVM também possui uma robustez para uma grande dimensionalidade de dados, o que trouxe resultados comparáveis e algumas vezes superiores aos obtidos por outras técnicas [4]. A SVM é considerada bem sucedida devido ao seu bom desempenho em diferentes problemas de aprendizado [8]. Esta vem sendo largamente aplicada em várias áreas, como: bioinformática [5], reconhecimento de face [6], predição em marketing [7], entre outras.

Na teoria, a SVM alcança uma generalização melhor comparada com outras técnicas, porém na prática isso nem sempre ocorre, pois o fator que maximiza a generalização é incerto. Esta generalização depende que os usuários definam os valores de alguns parâmetros empiricamente, tais como função de kernel, parâmetros da função de kernel, parâmetro de regularização, entre outros [9] levando usuários que não estão familiarizados com a SVM a obter, muitas vezes, resultados insatisfatórios [26]. Outro fator agravante, nesse processo, é que o desempenho da SVM pode variar bruscamente com a diferença sutil dos valores dos parâmetros [9]. Além de haver diversos tipos de SVM e de possíveis parâmetros, existem também diversos problemas e cenários diferentes, não havendo uma configuração de parâmetros ótima que possa ser aplicada a todos eles.

Diante da impossibilidade de se escolher valores dos parâmetros por tentativa e erro, diferentes autores modelaram o problema de seleção de parâmetros de SVM como um problema de otimização. Diante dos parâmetros escolhidos para serem otimizados, cada combinação de valores de parâmetros é mapeada como uma possível solução para um dado problema. Deste modo, técnicas de otimização são utilizadas para encontrar as configurações mais adequadas [18]. Embora esta proposta automatize o processo de seleção, esta pode se tornar bastante custosa devido ao número de possibilidades avaliadas durante o processo de busca [8].

Uma proposta alternativa é Meta-Aprendizado (MA), que automatiza o processo de seleção

de parâmetros da SVM tratando-o como uma tarefa de aprendizado supervisionado [8, 18]. Cada exemplo de treinamento (meta-exemplo) para o MA armazena as características (meta-características) de um problema no passado e informações sobre o desempenho obtido pelo conjunto de configurações de parâmetros candidatas para o problema. Em princípio, ao receber um conjunto de meta-exemplos como entrada, o meta-aprendiz, componente principal do MA, é capaz de definir as configurações de parâmetros mais adequadas para um problema novo, comparando as características deste com as dos problemas mais similares.

MA é uma solução menos custosa quando comparada à aplicação de técnicas de busca. Uma vez que a base de conhecimento do meta-aprendiz é gerada, configurações de parâmetros são sugeridas para novos problemas sem a necessidade da execução do algoritmo de aprendizado com as configurações candidatas. Porém, o sucesso do MA depende da qualidade dos meta-exemplos. Caso haja ruído nos dados e meta-características que descrevam mal o problema, a construção da base de conhecimento pode ser prejudicada, impactando negativamente no processo de sugestão.

Um trabalho recente desenvolvido por [60] combinou MA e técnicas de busca inspiradas em enxames de partículas para solucionar o problema de seleção de parâmetros de SVMs. Neste trabalho, MA foi adotado para sugerir um número de soluções (configurações de parâmetros) como população inicial da técnica de busca. Deste modo, a busca é iniciada em regiões promissoras, sendo as soluções refinadas ao longo do processo. Em [60] foram utilizadas técnicas com um único objetivo: minimizar a taxa de erro obtida pela SVM. Embora os resultados apresentados pela abordagem híbrida tenham superado os do MA puro e das técnicas de busca, seria mais adequado modelar o problema em questão usando-se múltiplos objetivos. De acordo com Narzisi [20], o problema de seleção de parâmetros de SVMs é inerentemente um problema de otimização multi-objetivo, cujo foco é encontrar a combinação de valores de parâmetros adequada levando em conta objetivos possivelmente conflitantes. Diversas pesquisas foram desenvolvidas utilizando-se os mais variados algoritmos de otimização e busca multi-objetivo aplicados à seleção de parâmetros de SVM [20][51][52][75][76]. Porém, a combinação destes algoritmos com MA ainda não foi estudada.

1.2 Trabalho Realizado

Nesta dissertação foi desenvolvida uma extensão do trabalho de Gomes e Prudêncio [60], fazendo uma combinação das técnicas multi-objetivo baseadas em enxames e meta-aprendizado para o problema de otimização de parâmetros da SVM. O objetivo principal desta dissertação é avaliar se a abordagem híbrida gera soluções de maior qualidade quando comparadas a técnicas de otimização multi-objetivo tradicionais.

As atividades necessárias para a realização deste trabalho são apresentadas abaixo:

1. Implementar algoritmos multi-objetivo existentes na literatura e adaptá-los ao problema de seleção de parâmetros de SVMs.

2. Selecionar meta-características para representação de problemas de classificação.

3. Criar a base de conhecimento de problemas de classificação.

4. Criar *framework* de MA para recomendação de configurações de valores de parâmetros.

5. Avaliar o desempenho das abordagens híbridas e compará-las com as técnicas puras.

Neste trabalho foi construída uma meta-base a partir de 40 problemas de classificação, sendo cada meta-exemplo caracterizado por 8 meta-características. Além disso, foram selecionados e implementados 6 algoritmos de otimização multi-objetivo inspirados em enxames. Cada algoritmo foi comparado com sua versão híbrida (fazendo uso de MA), e foi analisado o desempenho das duas abordagens. A população inicial, sugerida pelo MA, favoreceu a busca por soluções ótimas, sendo constatado, estatisticamente, o grande potencial da aplicação de MA no processo de otimização de parâmetros em SVMs.

1.3 Organização da Dissertação

Além deste capítulo, esta dissertação será apresentada em mais cinco capítulos que estão organizados da seguinte forma:

Capítulo 2: Neste capítulo é apresentado o problema de seleção de parâmetros de SVMs, suas motivações e desafios. Além disso, são apresentados a aplicação de técnicas de otimização e meta-aprendizado no contexto de seleção de parâmetros de SVMs, apresentando-se trabalhos recentes.

Capítulo 3: Neste capítulo são apresentados algoritmos de otimização inspirados em enxames de partículas. É feita uma breve revisão sobre o algoritmo canônico de único objetivo e apresenta-se o conceito de otimização multi-objetivo, seus conceitos e algoritmos existentes na literatura que foram utilizados nos experimentos.

Capítulo 4: Neste capítulo é apresentado o trabalho desenvolvido: uma arquitetura híbrida, fundamentada na combinação de algoritmos de otimização multi-objetivo com meta-aprendizado, para o problema de seleção de parâmetros de SVMs. São apresentados cada componente da arquitetura e suas interações, os desafios encontrados no processo de sugestão do meta-aprendizado para cenários multi-objetivo e a implementação da arquitetura.

Capítulo 5: É apresentado todo o arranjo experimental utilizado: os parâmetros dos algoritmos implementados, as métricas para avaliação de qualidade dos resultados e a metodologia de análise do experimento. Neste mesmo capítulo também são apresentados os resultados obtidos.

Capítulo 6: Neste capítulo são apresentadas as conclusões do trabalho desenvolvido, sugestões de trabalhos futuros, lista de publicações aprovadas e as contribuições realizadas.

CAPÍTULO 2

Seleção de Parâmetros de SVMs

A SVM é um algoritmo de aprendizado supervisionado que infere de um conjunto de exemplos rotulados (cuja classe é conhecida) uma função capaz de predizer os rótulos de novos exemplos desconhecidos. A derivação do algoritmo SVM pode ser compreendida a partir da classe mais simples das funções de decisão: funções lineares. Para ilustrar estes tipos de funções de decisão, na Figura 2.1, retirada do trabalho de Lima [35], são apresentadas três funções de decisão lineares que corretamente classificam um conjunto de treinamento de duas dimensões (os quadrados e os círculos). No entanto, cada função de decisão determina uma região completamente diferente para cada classe.

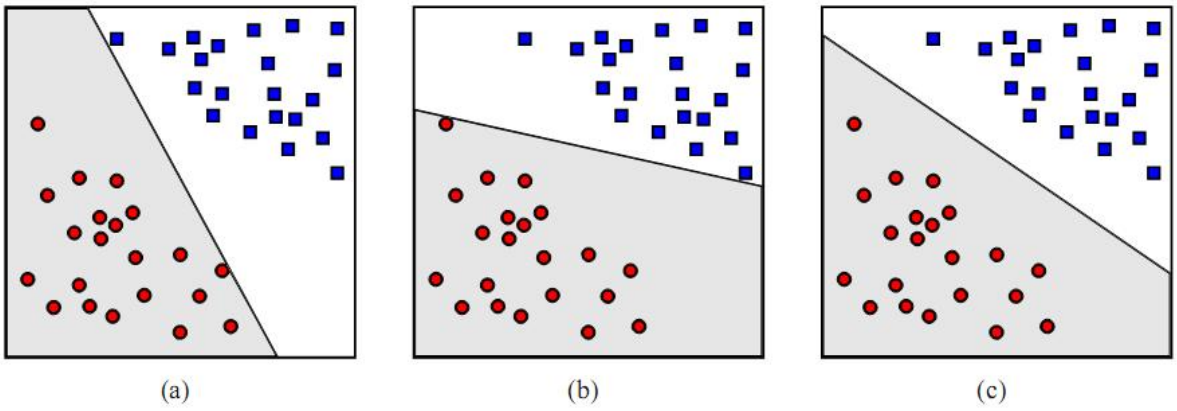

Figura 2.1 Classificação usando-se 3 funções de decisão distintas.

Funções de decisão lineares consistem de um limiar de decisão que é um hiperplano (uma linha em 2D, um plano em 3D, etc) separando duas regiões diferentes no espaço. Tal função de decisão ($g(\vec{x})$) é expressa por uma função matemática do vetor $\vec{x}$ e pode assumir os valores $+1$ ou -1 (a função *sign* é responsável por esta discretização), que representam o rótulo predito para o vetor $\vec{x}$. Assim, o classificador linear pode ser expresso como:

$$g(\vec{x}) = sign(f(\vec{x})), \tag{2.1}$$

$$f(\vec{x}) = <\vec{w},\vec{x}> +b \tag{2.2}$$

Desta maneira, a Equação 2.2 parametriza a função pelo vetor de pesos $\vec{w}$ e o escalar b. A notação $<\vec{w},\vec{x}>$ denota o produto interno de $\vec{w}$ e $\vec{x}$, definido por:

$$<\vec{w},\vec{x}> = \sum_{i=1}^{d} w_i x_i \tag{2.3}$$

em que d é a dimensionalidade e w_i é o i-ésimo componente de $\vec{w}$, sendo que $\vec{w}$ é da forma $(w_1, w_2, w_3, ..., w_d)$.

Uma vez que a formalização da função de decisão foi apresentada, pode-se definir o problema que o SVM linear se propõe a solucionar: *Dado um conjunto de treinamento de vetores (exemplos)* $x_1, x_2, x_3, ..., x_n$*, com suas correspondentes classes* $\vec{y_1}, \vec{y_2}, \vec{y_3}, ..., \vec{y_n}$ *que podem tomar valores* $+1$ *e* -1*, escolha os parâmetros* $\vec{w}$ *e* b *da função de decisão linear que melhor generaliza para exemplos desconhecidos.*

Intuitivamente, a Figura 2.1(c) é escolhida como a melhor função de decisão dentre as 3 funções apresentadas na Figura 2.1. Tal escolha pode ser justificada devido à grande margem de separação entre exemplos das duas classes.

A margem é obtida pela distância entre o hiperplano e os vetores que estão mais próximos a ele, sendo estes vetores denominados de vetores suporte. De acordo com Smola et al. [34], os vetores suporte são padrões críticos, que sozinhos determinam o hiperplano ótimo, sendo os outros padrões (não-críticos) irrelevantes, podendo ser removidos do conjunto de treinamento sem afetar os resultados. Na Figura 2.2 os vetores suporte são destacados por círculos externos nos padrões. Deste modo, o objetivo é encontrar o hiperplano que maximize a margem, distanciando as classes ao máximo.

O espaçamento da margem tem grande influência quando se trata do efeito de perturbações no momento de aquisição dos dados. Pequenas variações na medida de tais dados podem mudar os rótulos preditos para as funções de decisão das Figuras 2.1(a) e 2.1(b), enquanto a função de decisão da Figura 2.1(c) é a mais robusta nesse aspecto. Lovell e Walder [10] justificaram a escolha de funções de decisão de máxima margem com a seguinte afirmação: "É preferível limites de decisão que não somente separem corretamente duas classes, mas também estejam tão distantes quanto possível dos exemplos de treinamento".

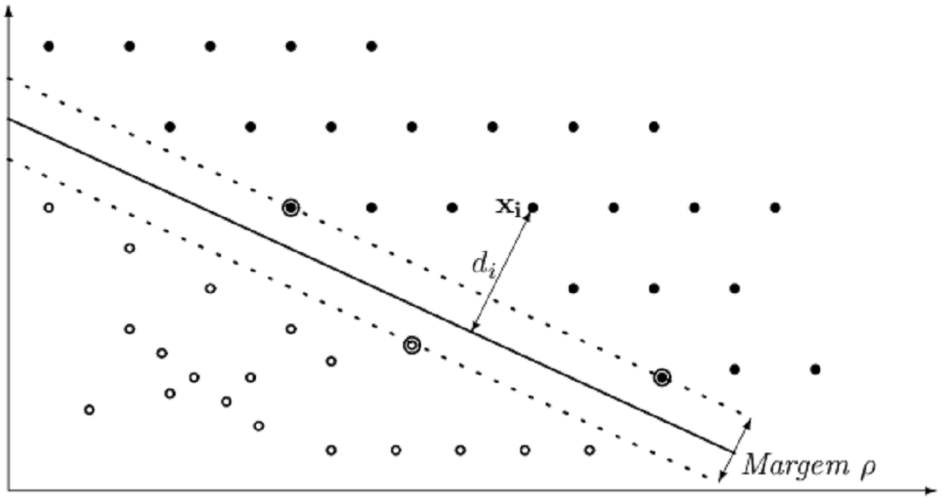

Figura 2.2 Identificação da margem p e dos vetores suporte sobre a linha pontilhada.

O exemplo apresentado na Figura 2.1 é linearmente separável, porém, problemas reais normalmente são não-linearmente separáveis, ou seja, não há como traçar um hiperplano separando as classes linearmente. A ideia básica do SVM é mapear não-linearmente os vetores

(exemplos de treinamento) do espaço de entrada em um espaço de características F de maior dimensão, em que os dados podem ser classificados pelo algoritmo linear apresentado anteriormente. A Figura 2.3 apresenta o mapeamento não-linear ϕ dos vetores de entrada para o espaço de características F. Tais vetores no novo espaço F, de maior dimensão, podem ser linearmente separáveis via hiperplano.

Uma vez que os vetores são mapeados para espaços de maiores dimensões, o produto interno destes vetores torna-se computacionalmente complexo. Para contornar tal complexidade computacional, existem funções de kernel que tornam tal cálculo eficiente. O hiperplano ótimo obtido em F corresponde a uma função de decisão não-linear cuja forma é determinada pelo kernel escolhido. Apesar do hiperplano ser linear no espaço de características, no espaço de entrada este corresponde a uma função de decisão não-linear, cuja forma é determinada pelo kernel utilizado (linear, polinomial, base radial, entre outros).

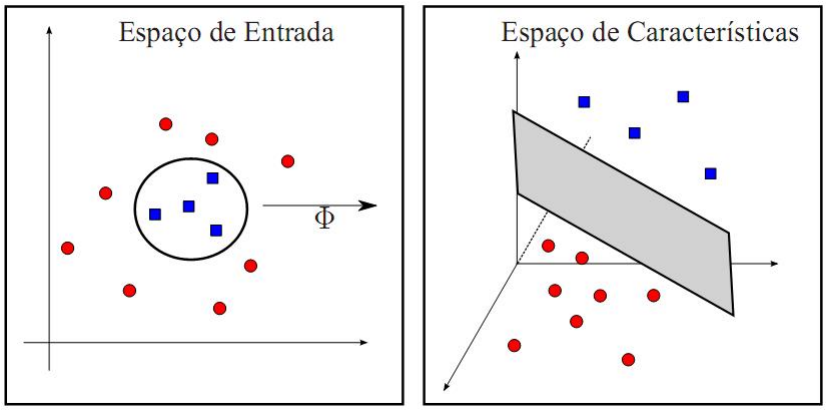

Figura 2.3 Mapeamento não-linear dos dados de treinamento em um espaço de características de maior dimensão via ϕ.

A SVM assemelha-se às redes neurais, entretanto o seu embasamento na TAE faz com que a capacidade de generalização esteja além da capacidade de métodos mais tradicionais [36]. De acordo com [36], a capacidade de generalização da SVM é satisfatória, mesmo tendo uma quantidade grande de atributos e poucos exemplos disponíveis para treinamento. O mesmo dificilmente aconteceria com outros métodos. Apesar de a SVM oferecer resultados satisfatórios, a mesma é bastante sensível à calibração dos parâmetros.

Atualmente a SVM pode ser trabalhada com valores padrões para os seus parâmetros ou os parâmetros podem ser definidos pela experiência do usuário, ou ainda por algumas abordagens: medidas de erro de generalização, técnicas de otimização, meta-aprendizado e técnicas híbridas.

2.1 Metodologias para avaliação de Erro de Generalização

O erro de generalização é baseado no erro empírico. Existem várias metodologias para avaliação do erro de generalização, tais como validação cruzada ou *cross validation* [12, 14], contador de vetores de suporte [15], *bootstrapping* [11], *radius-margin bound* [13], *span bound*

[16], entre outros. Essa abordagem possui a desvantagem de exigir uma demanda computacional significativa na fase de treinamento, pois o modelo SVM precisa ser induzido para cada configuração de valores de parâmetros. Além do mais, quanto maior a base de dados maior o custo computacional requisitado. A metodologia mais utilizada é a validação cruzada, cujo conceito central é o particionamento do conjunto de dados em subconjuntos mutuamente exclusivos, e posteriormente, utiliza alguns destes subconjuntos para a estimação dos parâmetros do modelo (dados de treinamento) e o restante dos subconjuntos (dados de validação ou de teste) são empregados na validação do modelo. Diversas formas de realizar o particionamento dos dados foram sugeridas, sendo que as três mais utilizadas são: o método *holdout*, o *k-fold* e o *leave-one-out*.

2.1.1 *Holdout*

Este método consiste em dividir o conjunto total de dados em dois subconjuntos mutuamente exclusivos, um para treinamento (estimação dos parâmetros) e outro para teste (validação). O conjunto de dados pode ser separado em quantidades iguais ou não. Uma proporção muito comum é considerar $2/3$ dos dados para treinamento e o $1/3$ restante para teste [17]. Após o particionamento, a estimação do modelo é realizada e, posteriormente, os dados de teste são aplicados e o erro de predição calculado.

Esta abordagem é indicada quando está disponível uma grande quantidade de dados. Caso o conjunto total de dados seja pequeno, o erro calculado na predição pode sofrer muita variação.

2.1.2 *K-fold*

O método de validação cruzada denominado *k-fold* consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos de mesmo tamanho e, a partir disto, um subconjunto é utilizado para teste e os $k-1$ restantes são utilizados para estimação dos parâmetros e calcula-se a acurácia do modelo. Este processo é realizado k vezes alternando de forma circular o subconjunto de teste.

No decorrer das k iterações o erro produzido é acumulado para obter a média absoluta do erro do conjunto de validação, obtendo assim uma medida mais confiável sobre a capacidade do modelo de representar o processo gerador dos dados.

2.1.3 *Leave-one-out*

O método *leave-one-out* é um caso específico do *k-fold*, com k igual ao número total de dados N. Nesta abordagem são realizados N cálculos de erro, um para cada dado. Apesar de apresentar uma investigação completa sobre a variação do modelo em relação aos dados utilizados, este método possui um alto custo computacional, sendo indicado para situações onde poucos dados estão disponíveis.

2.2 Técnicas de Otimização

Nesta seção, abordamos a aplicação de algoritmos de otimização e busca no problema de seleção de parâmetros de SVM. De acordo com Cawley [33], a tarefa de selecionar parâmetros em SVMs é frequentemente realizada combinando-se os valores dos parâmetros e armazenando-se as melhores combinações em desempenho com relação aos dados do problema em questão. De modo a automatizar este processo e evitar uma exploração manual e aleatória de parâmetros, adaptam-se algoritmos de busca para este contexto.

Estes algoritmos requisitam um espaço de busca composto de um conjunto de possíveis valores para cada parâmetro escolhido. O algoritmo converge para a melhor solução medida de acordo com uma função objetivo (fitness), que é normalmente uma medida de erro de generalização [46, 47]. Essa abordagem pode ser cara computacionalmente, pois também é necessária a execução do algoritmo SVM para cada configuração de parâmetros selecionados pelo método de otimização.

A arquitetura geral de uma solução através de algoritmos de busca é representada pela Figura 2.4 a seguir:

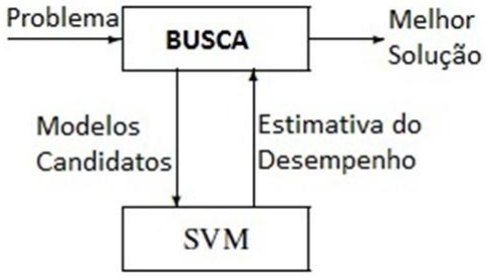

Figura 2.4 Arquitetura de algoritmos de otimização para SVM.

Nos últimos anos, diversas técnicas de otimização têm sido propostas para seleção de modelos de SVM. Estas técnicas diferem basicamente em dois aspectos: critério de seleção e método de busca utilizado. Alguns critérios de seleção são especificamente relacionados à formulação da SVM, tais como: *radius margin bound*, *span bound*, e contador de vetores de suporte. Outros são clássicos, tais como erro de validação cruzada e *holdout*. Por outro lado, os algoritmos de otimização mais comuns são: baseado em gradiente [30], otimização por enxame de partículas (PSO) [31, 74] ou técnicas evolucionárias, tais como algoritmos genéticos [22, 24] e *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [23]. Dentre os trabalhos citados, as seguintes medidas de erro de generalização foram escolhidas: em [22, 23, 24, 31, 74] utilizaram como função objetivo o erro de validação cruzada *10-fold*, [30] utilizou a validação cruzada *leave-one-out* e [73] adotou a medida *radius-margin bound* para cálculo do erro. Com relação aos parâmetros otimizados, todos os trabalhos citados foram aplicados na otimização dos parâmetros C, e o γ da função de kernel RBF da SVM.

Embora a aplicação dos algoritmos automatize o processo de seleção de valores de parâmetros, esta pode ser limitada devido à complexidade do processo de seleção de modelo [31]. Uma limitação de algoritmos baseados em gradiente é a exigência de uma função objetivo diferenciável. Outro problema é com relação à múltiplos mínimos locais, que depreciam o desem-

penho destes algoritmos. Para superar estes problemas, a aplicação de técnicas evolucionárias e inspiradas em enxames são bastante atrativas [31].

Embora os trabalhos apresentados tenham proporcionado um avanço no estudo de métodos automáticos de seleção de parâmetros de SVM, estes métodos não abordaram o problema de forma adequada [20]. As pesquisas citadas modelaram o problema de seleção de parâmetros levando-se em consideração apenas um objetivo. De acordo com Narzisi [20], o problema de seleção de parâmetros de SVMs é inerentemente um problema de otimização multi-objetivo. O projeto de sistemas de aprendizado supervisionado exige encontrar o *trade-off* adequado entre diversos objetivos. Tipicamente, deseja-se reduzir a complexidade do modelo e ao mesmo tempo obter um modelo com alto nível de precisão (ou baixa taxa de erro). Normalmente, ter um modelo com o melhor erro de generalização pode não ser a melhor escolha se o preço a ser pago é um modelo complexo em termos de tempo e espaço.

Geralmente, o problema multi-objetivo é resolvido agregando-se os objetivos em uma função escalar (ponderação linear dos objetivos), usando-se esta função como objetivo a ser otimizado em uma abordagem de único objetivo [20]. Porém, tem sido mostrado que esta proposta não é uma boa solução, pois exige que a função de agregação represente corretamente o problema, e esta tarefa não é simples, sendo a melhor solução aplicar, diretamente, algoritmos multi-objetivo [20].

Deste modo, diversas pesquisas foram desenvolvidas utilizando-se os mais variados algoritmos de otimização e busca multi-objetivo aplicados ao problema em questão [20, 51, 52, 75, 76]. Dentre os parâmetros a serem otimizados, destacam-se o C, e o γ do kernel RBF; já com relação às funções objetivo consideradas, destacam-se a maximização da taxa de acerto da SVM e a minimização do número de vetores de suporte, objetivando a geração de modelos de SVM de alto potencial de classificação e de baixa complexidade.

Embora as técnicas de otimização automatizem o processo de seleção de parâmetros, estas podem se tornar extremamente caras computacionalmente, pois é necessário a execução do algoritmo SVM para cada configuração de parâmetros selecionados pelo método de otimização. Outra questão a ser considerada é que os algoritmos de busca possuem uma natureza estocástica, ou seja, eles podem encontrar o valor de ótimo global ou ficar presos em mínimos/máximos locais. Por isso existe uma necessidade de executá-los várias vezes, para que seja possível avaliar com um certo grau de estabilidade a solução média da busca.

2.3 Meta-Aprendizado

Uma alternativa para os problemas apresentados na seção anterior seria o Meta-Aprendizado (MA), que trata a tarefa de seleção de parâmetros como uma tarefa de aprendizado supervisionado [27, 19].

Toda a fundamentação do meta-aprendizado está na associação entre as características do problema e o desempenho alcançado pelo algoritmo de aprendizagem [48]. De modo que através das características apresentadas pelo problema, seja possível se definir o melhor modelo de algoritmo para resolver o problema.

O primeiro estudo nesta área foi desenvolvido por Rendell e Cho [48], dando início às tentativas de caracterizar problemas de classificação e examinar o seu impacto no comportamento do

algoritmo, usando características relacionadas ao tamanho e concentração das classes. Aha [49] estendeu esta ideia, utilizando algoritmos de aprendizado baseados em regra para criar regras como: se o conjunto de dados apresenta as características $C_1, C_2, ..., C_n$, então use o algoritmo A_1 e não use o algoritmo A_2. Neste trabalho foram utilizados algoritmos de classificação para a resolução do problema de reconhecimento de caracteres. As características utilizadas incluíam o número de instâncias, número de classes, número de atributos relevantes e irrelevantes.

Outro trabalho bastante importante para o desenvolvimento da área de meta-aprendizado foi o projeto *Statlog*. O principal objetivo deste projeto era apresentar o potencial e as fraquezas existentes nos diversos algoritmos de classificação existentes. A análise experimental deste trabalho foi referenciada no livro de Brazdil e Henery [21], que relaciona o desempenho dos algoritmos às suas caraterísticas ou medidas de conjuntos de dados de classificação. O resultado final deste trabalho foi um conjunto de medidas para caraterizar conjuntos de dados agrupadas em medidas simples (número de instâncias, classes, etc), estatísticas (média e variância dos atributos) e de teoria da informação (entropia). Estas categorias de características de dados são utilizadas até hoje [68].

Baseado no sucesso do projeto *Statlog* e o progresso da pesquisa em MA, foi fundado o projeto *ESPRIT*, em 1998, entitulado *METAL: a meta-learning assistant for providing user support in machine learning and data mining*. O objetivo deste projeto foi desenvolver propostas para seleção de modelos de algoritmos aplicados tanto em classificação como em regressão e disponibilizar para o usuário. Este projeto, quando criado, incorporava 53 bases de classificação, 10 algoritmos de aprendizado e medidas de desempenho do algoritmo baseadas na validação cruzada e no custo computacional. Atualmente, os números são bem maiores.

A criação do projeto *METAL* proporcionou um grande avanço na pesquisa do meta-aprendizado, o que favoreceu sua aplicação não apenas na seleção de algoritmos, mas também em tarefas relacionadas como a seleção de parâmetros. A Figura 2.5 mostra a arquitetura do meta-aprendizado.

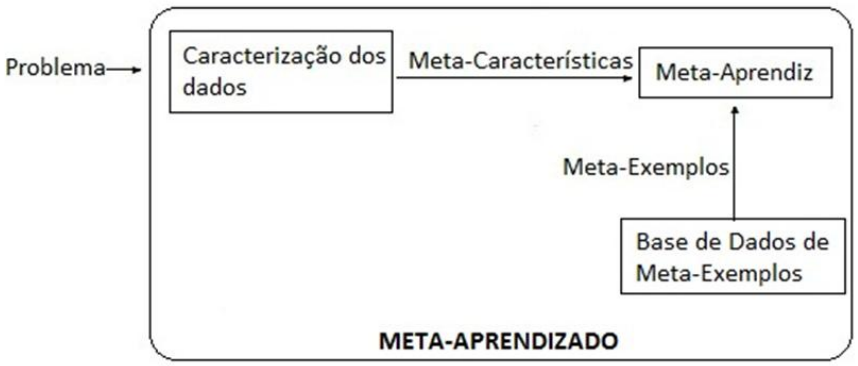

Figura 2.5 Arquitetura do Meta-Aprendizado.

Em MA, dada uma nova base de dados de problema, o sistema indica quais as melhores soluções (v.g. algoritmos, configurações de parâmetros) que geram os melhores resultados a partir das características do problema, proporcionando ao usuário economia de tempo, pois reduz a quantidade de algoritmos ou configurações de parâmetros que ele iria testar.

De modo a facilitar o entendimento do conceito de meta-aprendizado, utilizamos a arquitetura definida em [57] como referência. A arquitetura do meta-aprendizado pode ser dividida em dois modos de operação, aquisição de conhecimento e consulta, que serão detalhados a seguir. O modo de aquisição de conhecimento objetiva aprender sobre o próprio processo de aprendizagem. A Figura 2.6 ilustra este modo de operação.

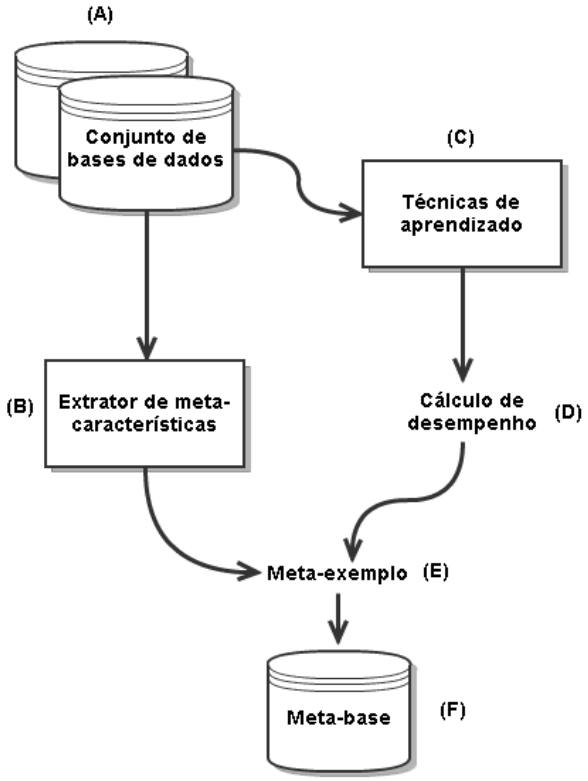

Figura 2.6 Meta-Aprendizado: Modo de Aquisição de Conhecimento.

Assume-se que a entrada do sistema consiste em um conjunto de bases de dados de exemplos (atributos e classes), como pode-se ver na Figura 2.6-A. Para cada base de dados são extraídas características (Figura 2.6-B) que transcendem um domínio particular da aplicação. Estas características ou meta-características, por sua vez, formam um conjunto de aspectos descritivos das bases (v.g., o número de atributos, número de *outliers*, número de atributos simbólicos, entre outros) que sejam capazes de capturar informação útil. Além disso, técnicas de aprendizado (Figura 2.6-C) são escolhidas (v.g. classificadores) para serem aplicadas nas bases de dados, gerando-se seu desempenho no dado problema (Figura 2.6-D). Informações derivadas da extração de meta-características e do cálculo de desempenho são combinadas, formando o meta-exemplo ou meta-conhecimento (Figura 2.6-E), e armazenadas na base de conhecimento ou meta-base (Figura 2.6-F). A meta-base é o principal resultado da fase de aquisição do conhecimento, refletindo a experiência acumulada ao longo de diferentes tarefas.

A eficácia do meta-aprendizado aumenta à medida que são acumulados meta-exemplos. A falta de experiência no início faz com que o MA faça escolhas sem uma real preferência.

No modo de consulta, dada uma nova base de dados como entrada (Figura 2.7-A), o objetivo é recomendar a estratégia de aprendizado (v.g. algoritmo, configurações do algoritmo) com base na comparação entre as características da base de entrada e dos meta-exemplos.

A meta-base gerada no modo anterior é utilizada como um repositório de execuções passadas (meta-exemplos). Com o surgimento de uma nova entrada, são extraídas suas características (Figura 2.7-B) comparando-as com as características dos meta-exemplos da meta-base (Figura 2.7-C) de modo a produzir recomendações (Figura 2.7-D). Deste modo, selecionam-se as soluções dos meta-exemplos mais similares com relação à nova entrada, que apresentam o melhor desempenho (Figura 2.7-E).

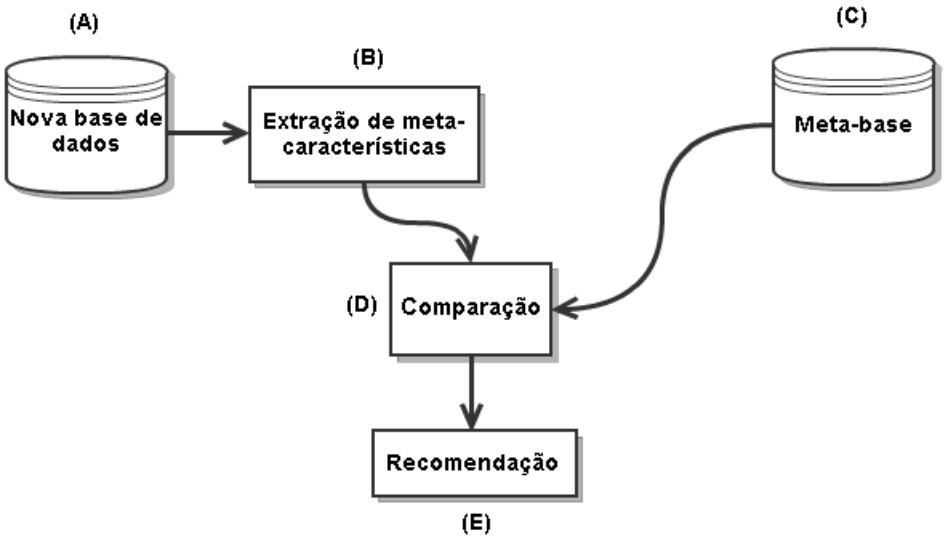

Figura 2.7 Meta-Aprendizado: Modo de Consulta.

Alguns trabalhos foram desenvolvidos utilizando-se meta-aprendizado para auxiliar o processo de automatização da escolha de parâmetros de SVMs [8, 54]. O principal ponto estudado pelos trabalhos citados é o desenvolvimento do meta-aprendiz, responsável pela exploração do conjunto de meta-dados (meta-conhecimento) para a obtenção de modelos e soluções para o problema de entrada. No trabalho de Smith-Miles e Shawkat [54], o meta-aprendiz é implementado como árvore de decisão para geração de regras que se adequem à melhor utilização dos tipos de kernel. Este trabalho realiza a comparação entre os diversos kernels da SVM, utilizando como medida de desempenho o erro da validação cruzada *10-fold*. O objetivo é identificar o kernel mais adequado para a SVM de acordo com o problema de entrada.

No trabalho de Soares et al. [8], o meta-aprendiz utilizado para seleção de parâmetros foi o ranking $k-NN$. O ranking *k-NN* é um método para recomendação de valores baseado nos *k*-vizinhos mais próximos e pode ser adaptado para a tarefa de ranking de parâmetros. Este trabalho realiza a comparação entre os parâmetros C, e o γ da função de kernel RBF, utilizando como medida de desempenho o erro de validação cruzada $10-fold$. O objetivo é identificar o conjunto valores de parâmetros mais adequado para a SVM de acordo com o problema de entrada.

Os trabalhos citados comprovaram que é possível prever o desempenho relativo da base de entrada observando-se as informações sobre os desempenhos passados de bases de dados similares. Através dessa previsão, é possível diminuir o custo computacional (e.g., o tempo de processamento) devido à diminuição de valores de parâmetros testados. Porém, para a possibilidade da previsão, é necessário adquirir todo o meta-conhecimento (conjunto de meta-dados) das bases de dados disponíveis, sendo esta a fase mais dispendiosa em relação aos recursos computacionais. Além disso, é necessário o pré-processamento dos dados, pois dados faltantes e ruidosos podem depreciar o conhecimento construído. Outra grande dificuldade de se trabalhar com meta-aprendizado é definir as meta-características que vão descrever o problema. A má escolha de meta-características pode influenciar todo o processo de sugestão realizado pelo meta-aprendiz, gerando soluções inadequadas para o problema de entrada. Trabalhos importantes foram desenvolvidos objetivando fazer o levantamento de meta-características adequadas para problemas de classificação [21] e regressão [27, 8].

2.4 Técnicas Híbridas

Diante dos problemas apresentados pelos algoritmos de otimização e meta-aprendizado, trabalhos recentes desenvolveram alternativas híbridas, envolvendo as duas áreas, para solucionar o problema de seleção de parâmetros de SVMs.

Um trabalho recente e inovador, desenvolvido por Gomes e Prudêncio [60], combinou MA com algoritmos de otimização e busca aplicado à seleção de parâmetros de SVMs. A proposta é utilizar MA para sugerir um número específico de soluções e usá-las como população inicial do algoritmo de otimização. Este trabalho utilizou o algoritmo de otimização por enxame de partículas ou PSO para realizar o refino das soluções. A inspiração para esta metodologia surgiu do fato de que problemas similares tendem a possuir espaços de busca similares. Dessa forma, o espaço de busca de um problema com características similares pode prover informações úteis sobre regiões ótimas de um novo problema [61]. Assim a proposta pelo Meta-Aprendizado permitirá o aumento da velocidade de convergência do algoritmo de busca para boas soluções.

A Figura 2.8 mostra a arquitetura de como a proposta está fundamentada. A criação da meta-base é o primeiro passo nesta abordagem. Inicialmente, seleciona-se uma série de problemas e extraem-se as meta-características de cada um deles. Além disso, para cada meta-exemplo gerado, realiza-se a avaliação de desempenho, aplicando-se a SVM, para todas as possíveis combinações de valores de parâmetros.

Com a meta-base criada, o meta-aprendiz se torna capaz de sugerir combinações de valores de parâmetros bem sucedidas de acordo com as características do problema que está sendo utilizado na entrada. Estas sugestões são passadas para o módulo de busca como população inicial. No processo de busca, as soluções são otimizadas levando-se em consideração a minimização da taxa de erro na regressão.

Em [60], as configurações foram compostas por dois parâmetros da SVM: γ do kernel RBF e o parâmetro de regularização C. A escolha do kernel RBF é devido à sua flexibilidade em diferentes problemas quando comparado a outros kernels [64][65]. O γ e o C têm uma importante influência no desempenho da aprendizagem, sendo o primeiro responsável pelo controle da linearidade e o segundo da complexidade da SVM [65].

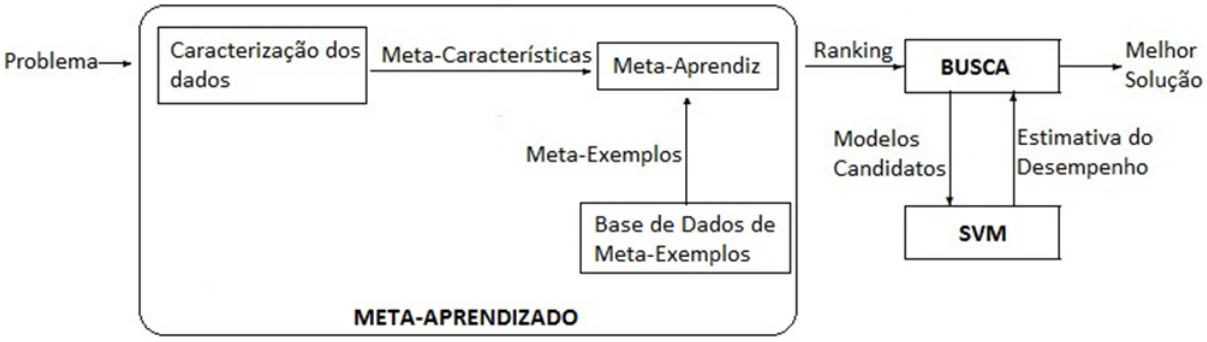

Figura 2.8 Arquitetura da Proposta de [60].

A implementação do algoritmo de otimização precisou ser configurada para a otimização das configurações escolhidas C, e γ do kernel RBF. De modo que cada partícula representa uma configuração (C, γ), que indica a sua posição no espaço de busca. Como foi dito anteriormente, a função objetivo utilizada para avaliação da qualidade das soluções é a própria SVM de kernel RBF, cuja saída é a taxa de erro na regressão com validação cruzada *10-fold*. Deste modo, o objetivo do algoritmo é encontrar a configuração (C, γ) que minimize a taxa de erro para um dado problema.

O trabalho desenvolvido por Reif e Shafait [55] seguiu a mesma ideologia, utilizando MA e raciocínio baseado em casos para prover soluções iniciais para algoritmos genéticos de modo a otimizar parâmetros para SVM e para o algoritmo *Random Forest* (*RF*) em problemas de classificação. Para a SVM foram otimizados os mesmos parâmetros de [60], C e γ. O algoritmo *RF* teve 5 parâmetros otimizados, mais detalhes em [55]. Em [55] foram utilizadas 15 meta-características (numéricas e categóricas). A técnica *k-NN*, foi utilizada para definir a similaridade entre os meta-exemplos e o erro de validação cruzada *10-fold* como função objetivo.

Os resultados apresentados nos dois trabalhos mostraram o grande potencial da combinação de MA com técnicas de otimização, gerando-se melhores soluções em comparação com a aplicação do algoritmo tradicional para o problema de seleção de parâmetros de SVM.

2.5 Considerações Finais

Neste capítulo, inicialmente, foram apresentados a SVM, seus conceitos, limitações, bem como o problema de seleção parâmetros, foco deste trabalho. Após a motivação do problema, foram apresentadas as abordagens de medidas de erro de generalização, técnicas de otimização, meta-aprendizado e técnicas híbridas, aplicadas para auxiliar no processo de seleção de parâmetros. Foram mencionados diversos trabalhos relacionados apresentando suas fundamentações e experimentos, constatando-se o grande potencial de técnicas híbridas, combinando algoritmos de otimização com meta-aprendizado.

O capítulo seguinte: "Algoritmos de Otimização Inspirados em Enxames de Partículas", apresenta com mais detalhes a história, o algoritmo canônico de único objetivo, a otimização multi-objetivo e variações de algoritmos multi-objetivo que foram utilizados neste trabalho.

CAPÍTULO 3

Algoritmos de Otimização Inspirados em Enxames de Partículas

O conceito de algoritmos de otimização baseados em enxames de partículas surgiu com estudos a respeito do comportamento de bandos de pássaros, quais regras os regiam, e como poderiam mudar o curso de forma tão repentina e sincronizada [32]. O PSO (do inglês *Particle Swarm Optimization*) foi proposto após a simulação de modelos sociais simplificados baseados em observações feitas nos bandos de aves à procura de alimento.

Há anos vem sendo estudado o comportamento social de alguns grupos de animais como a organização das abelhas, formigas e pássaros na busca de alimentos ou novos locais para estabelecer sua nova moradia. Este último grupo, em especial, despertou um grande interesse de alguns pesquisadores, na década de 80, dentre os quais se destaca dentre eles o biólogo Frank Heppner [38].

Após diversas observações sobre o comportamento de bando de pássaros em revoada, Heppner decidiu por modelar aquela inteligência coletiva para usá-la em métodos de busca para solução de problemas. Os estudos de Heppner consideravam que o comportamento de várias espécies de pássaros em bando, ao longo do seu voo, fazia o uso de alguma lógica e de alguma forma de comunicação. Após vários estudos e observações, Heppner descreveu o comportamento, qualificando-o como comportamento social [38].

James Kennedy e Russel Eberhart, inspirados no comportamento social dos pássaros estudado por Heppner, desenvolveram uma técnica de otimização que veio a ser conhecida como enxame de partículas. Essa denominação se deu, pois se notou que o modelo descrito por Heppner demonstrava características de um enxame inteligente, cujos membros que apresentavam tal comportamento foram generalizados para o termo partículas. Não só o nome do algoritmo, como os demais aspectos do modelo estudado por Heppner, ganharam uma nova conotação. A busca pelo alimento ou pelo ninho e a interação entre os pássaros ao longo do voo são modelados como um mecanismo de otimização. Fazendo uma analogia, o termo partícula foi adotado para simbolizar os pássaros e representar as possíveis soluções do problema a ser resolvido. A área sobrevoada pelos pássaros é equivalente ao espaço de busca e encontrar o local com comida, ou o ninho, corresponde a encontrar a solução ótima. Para que o bando de pássaros sempre se aproxime do objetivo, ao invés de se perder ou nunca alcançar o alvo focado, utiliza-se uma função que irá avaliar o desempenho das partículas (fitness). Para alcançar o alvo focado, sejam os alimentos ou os ninhos, os pássaros fazem uso de suas experiências e da experiência do próprio bando. Isso ressalta o senso do social, em que os indivíduos aprendem com o sucesso dos seus vizinhos e sempre priorizam o objetivo do grupo. Desta forma, não há centralização de controle e o grupo consegue atingir as soluções desejadas em conjunto.

Um enxame possui inteligência se certas características estiverem presentes [37]. Essas características são descritas por princípios básicos:

1. Princípio da proximidade: Afirma que os membros do enxame devem ter a noção do espaço e tempo durante seu deslocamento, de modo que mudanças possam ser realizadas sem interferir na movimentação de seus vizinhos.

2. Princípio da qualidade: Afirma que os membros do enxame devem ser capazes de se adaptar aos fatores de qualidade do ambiente, isto é, a depender do(s) objetivo(s) almejado(s), a população deve buscar um caminho em que resulta na(s) melhor(es) solução(ões), garantindo qualidade para o seu bando.

3. Princípio da resposta diversa: As mudanças no processo são sempre desejadas a fim de que o enxame possa evoluir em sua busca. Isso previne que os membros fiquem presos em um local em que não se encontra a melhor solução (mínimo local). Porém, é desejável que haja ações corretivas caso ocorram mudanças repentinas, pois do contrário podem ocorrer desvios no processo de busca que podem acarretar em um aumento no esforço computacional.

4. Princípio da estabilidade: Diz que os membros do enxame não devem mudar o seu comportamento a cada mudança do ambiente. O enxame deve ser capaz de avaliar se tal mudança é necessária a fim de aplicar um esforço conjunto de todo o bando.

5. Princípio da adaptabilidade: Afirma que a população deve mudar seu comportamento quando o custo computacional envolvido valer a pena, isto é, o enxame deve possuir a capacidade de avaliar as mudanças ocorridas e decidir se é válida a mudança no comportamento do bando.

A existência dessas características é extremamente importante para garantir o equilíbrio e a estabilidade do bando durante a evolução do processo, qualificando-o, assim, como inteligente.

Diversas variações de PSO vêm sendo propostas e aplicadas em diversos tipos de problemas, sejam elas de único ou múltiplos objetivos. A seguir serão detalhadas as duas abordagens: otimização com objetivo único e com múltiplos objetivos.

3.1 O Algoritmo PSO

Na maioria das implementações do PSO, as partículas movem-se no espaço de busca sendo guiadas por uma combinação entre a melhor posição encontrada pela partícula e a melhor posição encontrada pela vizinhança em que ela está inserida. Esta vizinhança é definida como um conjunto de partículas com as quais a partícula pode se comunicar, podendo este conjunto se estender para todo o enxame ou não. O que define esta comunicação entre as partículas é a topologia de comunicação entre elas. Existem duas topologias consideradas como base de todas as outras: Anel e Estrela (ver [32]).

Cada partícula está sob influência de três forças que podem ser representadas matematicamente como vetores:

Vetor inércia: representa o movimento atual da partícula, ou seja, a velocidade atual. Na Equação 3.1 está representado por $\vec{v}_i(t)$.

Vetor cognitivo: representa o componente cognitivo da partícula, uma relação entre a posição atual e a melhor posição encontrada por aquela partícula. Na Equação 3.1 é representado pelo termo $c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t))$.

Vetor social: representa a influência do enxame em uma determinada partícula. É uma relação entre a melhor posição encontrada pela vizinhança da partícula e a posição atual da partícula. Na Equação 3.1 é representado pelo termo $c_2 r_2(\vec{n}_i(t) - \vec{x}_i(t))$.

Um esquema gráfico pode ser observado na Figura 3.1, onde $\vec{x}_i(t)$ representa a posição atual da partícula e $\vec{x}_i(t+1)$ representa a posição da partícula após o processo de atualização da velocidade, que é determinante na atualização da posição.

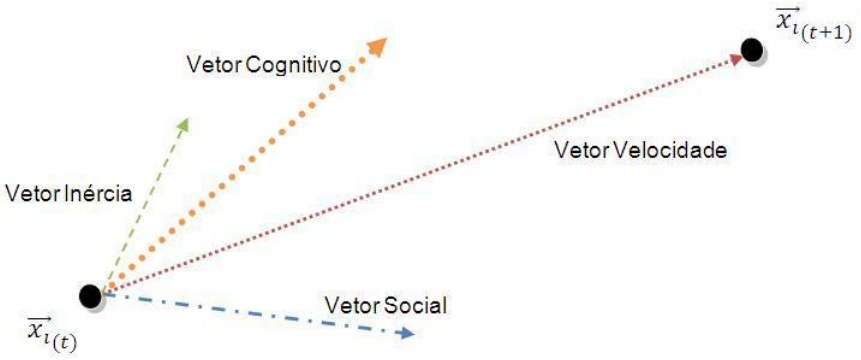

Figura 3.1 Vetores que influenciam o movimento das partículas.

O algoritmo básico de otimização por enxame de partículas pode ser descrito brevemente utilizando os seguintes passos: dada uma população inicial de partículas, atualiza-se o vetor posição a partir do vetor velocidade de cada partícula até que se atinja o critério de parada pré-definido [32]. A Figura 3.2 ilustra essa lógica:

Existe uma variedade de equações para atualização da velocidade. Neste trabalho foi utilizada a equação proposta por Shi e Eberhart [32] como mostra a Equação 3.1.

$$\vec{v}_i(t+1) = \omega \vec{v}_i(t) + c_1 r_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2 r_2(\vec{n}_i(t) - \vec{x}_i(t)), \tag{3.1}$$

$$\vec{x}_i(t+1) = \vec{x}_i + \vec{v}_i(t+1), \tag{3.2}$$

onde $i = 1, ..., N$, N é o número de partículas; ω é o fator de inércia, utilizado para controlar a convergência; c_1 e c_2 são os coeficientes de aceleração cognitivo e social; r_1 e r_2 são dois números aleatórios gerados pela função de densidade de probabilidade uniforme no intervalo [0,1]. A variável $\vec{p}_i(t)$ representa a melhor solução encontrada pela partícula, também chamada de *pbest*, e $\vec{n}_i(t)$ representa a melhor solução encontrada pela vizinhança, também chamada de

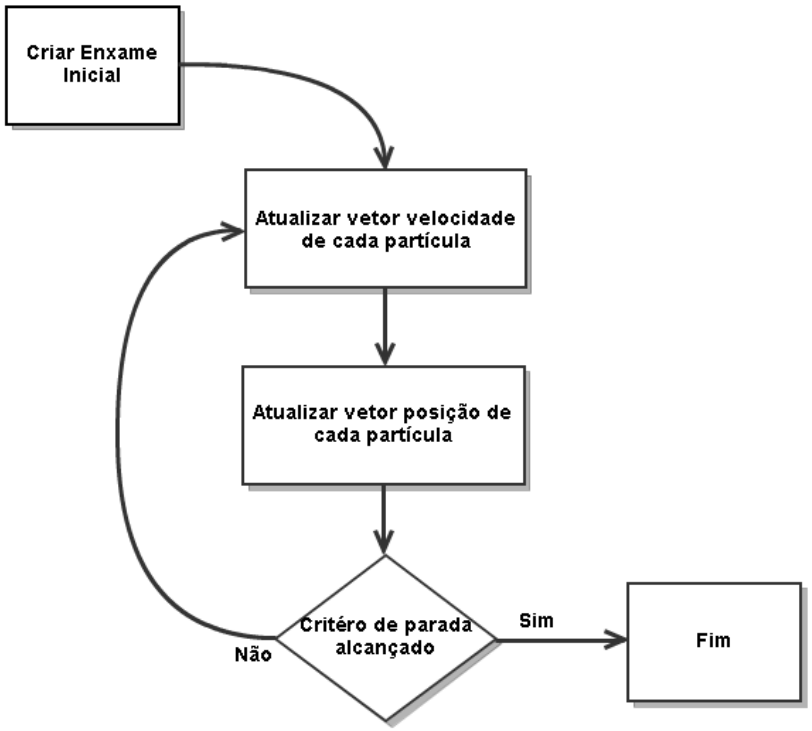

Figura 3.2 Fluxograma básico do funcionamento do PSO.

gbest. O pseudo-código abaixo mostra o funcionamento do PSO canônico de forma detalhada:

Algoritmo 3.1: Pseudo-código do PSO.

1: **Inicialização**:
Criar partículas com posições ($\vec{x}_i$) e velocidade ($\vec{v}_i$) aleatórias.
Atribuir a $\vec{p}_i(t)$ a posição atual das partículas e a $\vec{n}_i(t)$ a melhor posição encontrada pela vizinhança de cada partícula.
2: **enquanto** ***critério de parada*** *não satisfeito* **faça**
para cada *partícula p* **faça**
3: **Passo de Avaliação:**
Calcular *fitness* f($\vec{x}_i$).
4: **Passo de Atualização:**
Atualizar velocidade e posição utilizando as Equações 3.1 e 3.2.
Atualizar valores de *pbest* e *gbest*.
fim
fim

O algoritmo apresentado foi desenvolvido, inicialmente, para a resolução de problemas de único objetivo. Porém, a maioria dos problemas reais apresenta múltiplos objetivos potencialmente conflitantes, sendo necessária uma alternativa que leve em consideração a otimização e busca de configurações com relação a todos os objetivos paralelamente. Deste modo, foi desenvolvido, por Coello Coello et al. [29], um trabalho propondo modificações na estrutura básica do PSO permitindo que o enxame possa otimizar múltiplos objetivos. A seguir, a otimização multi-objetivo será detalhada.

3.2 Otimização com Múltiplos Objetivos

Em contraste com a otimização de um único objetivo, a otimização com multi-objetivos (MOO, do inglês Multi-Objective Optimization) possui vários objetivos a serem atingidos. MOO pode ser definido como o problema de encontrar um vetor de valores de variáveis de decisão que satisfazem restrições e otimizam um vetor de funções cujos elementos representam as funções objetivo. De modo a facilitar o entendimento da otimização multi-objetivo, a seguir são apresentadas algumas definições fundamentais, levando-se em consideração um cenário de minimização de objetivos:

Definição 1 (Mínimo global): No cenário de único objetivo, dada uma função f: $\Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq 0$, para $\vec{x} \in \Omega$ o valor $f^* \triangleq f(\vec{x^*}) > -\infty$ é chamado um mínimo global se e apenas se

$$\forall \vec{x} \in \Omega : f(\vec{x^*}) \leq f(\vec{x}). \tag{3.3}$$

Então, $\vec{x^*}$ é o mínimo global da solução, f é a função objetivo, e o conjunto Ω é o conjunto de possíveis valores ($\Omega \in \mathscr{S}$), onde $\mathscr{S}$ representa o espaço de busca.

Definição 2 (Problema geral de otimização multi-objetivo ou MOP): Encontrar o vetor $\vec{x^*} = [x_1^*, x_2^*, ..., x_n^*]$ que satisfaça as q restrições de desigualdade

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, ..., q, \tag{3.4}$$

as s restrições

$$h_i(\vec{x}) = 0 \quad i = 1, 2, ..., s, \tag{3.5}$$

e otimize o vetor de funções

$$\vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})], \tag{3.6}$$

onde $\vec{x} = (x_1, x_2, ..., x_n)$ é o vetor de variáveis de decisão.

Definição 3 (Dominância): Um vetor $\vec{u} = (u_i, ...u_k)$ domina $\vec{v} = (v_i, ..., v_k)$ (simbolizado por $\vec{u} \preceq \vec{v}$) se e somente se $\vec{u}$ é parcialmente menor que $\vec{v}$, v.g, $\forall_i \in 1, ..., k, u_i \leq v_i \wedge \exists i \in 1, ..., k : u_i < v_i$. A Figura 3.3 apresenta a relação de dominância entre pontos. Como pode-se ver, o ponto 1 domina todos os demais pontos, pois o mesmo é melhor ou igual nos objetivos f_1 e f_2.

Definição 4 (Pareto Ótimo): $\vec{x^*} \in \Omega$ é um Pareto ótimo se para cada $\vec{x} \in \Omega$ e $I = 1, 2, ..., k$ e

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x^*})) \tag{3.7}$$

existe ao menos um $i \in I$ que

$$f_i(\vec{x}) > f_i(\vec{x^*}). \tag{3.8}$$

Definição 5 (conjunto de Pareto ótimo): Dado um MOP $f(x)$, o conjunto do Pareto ótimo (P^*) é definido como:

$$P^* := \{x \in \Omega \mid \neg\exists x' \in \Omega \vec{f}(x') \preceq \vec{f}(x)\}. \tag{3.9}$$

Definição 6 (Pareto *front*): Dado um MOP $\vec{f}(x)$ e um conjunto do Pareto ótimo P^*, o Pareto *front* (PF^*) é definido como

$$PF^* := \{\vec{u} = \vec{f} = (f_1(x), ..., f_k(x)) \mid x \in P^*\}. \tag{3.10}$$

Em casos gerais é impossível encontrar uma expressão analítica da linha ou superfície que contenha estes pontos. O procedimento normal de se gerar o Pareto *front* é calcular os pontos alcançáveis e seus correspondentes. Quando existe um número suficiente destes pontos, é então possível determinar os pontos não dominados e produzir o Pareto *front*, ver Figura 3.4.

Cada problema de múltiplos objetivos apresenta um Pareto-Ótimo associado, que representa o conjunto de soluções ótimas que melhor o soluciona. Deste modo, um processo de otimização multi-objetivo objetiva encontrar o conjunto de soluções do Pareto *front* a partir do conjunto de todos os vetores de variáveis de decisão que satisfaçam 3.4 e 3.5; de modo que a distribuição destas variáveis de decisão se assemelhe com a do Pareto-Ótimo do problema [39].

Como foi visto na Seção 3.1, o PSO é uma técnica que visa solucionar problemas de único objetivo; de modo que para aplicá-lo em MOP o seu algoritmo original deve ser alterado. Deste modo, surgem algumas dúvidas técnicas referentes à adaptação que deve ser feita ao algoritmo do PSO [29]:

1. Como selecionar partículas para serem usadas como *pbest* e *gbest*?

2. Como armazenar as soluções não dominadas encontradas e atualizá-las durante o processo de busca?

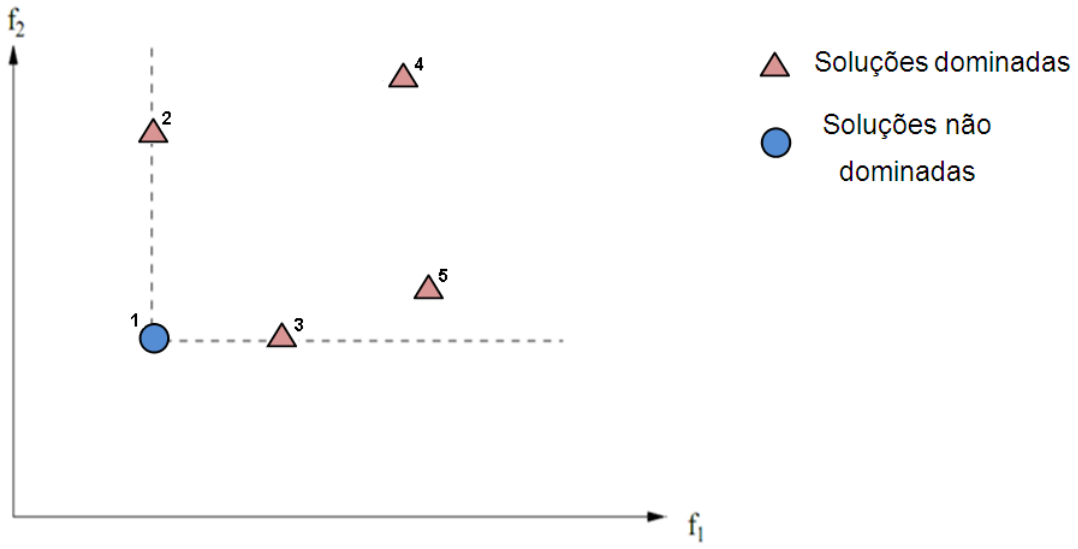

Figura 3.3 Relação de dominância em um espaço bi-objetivo.

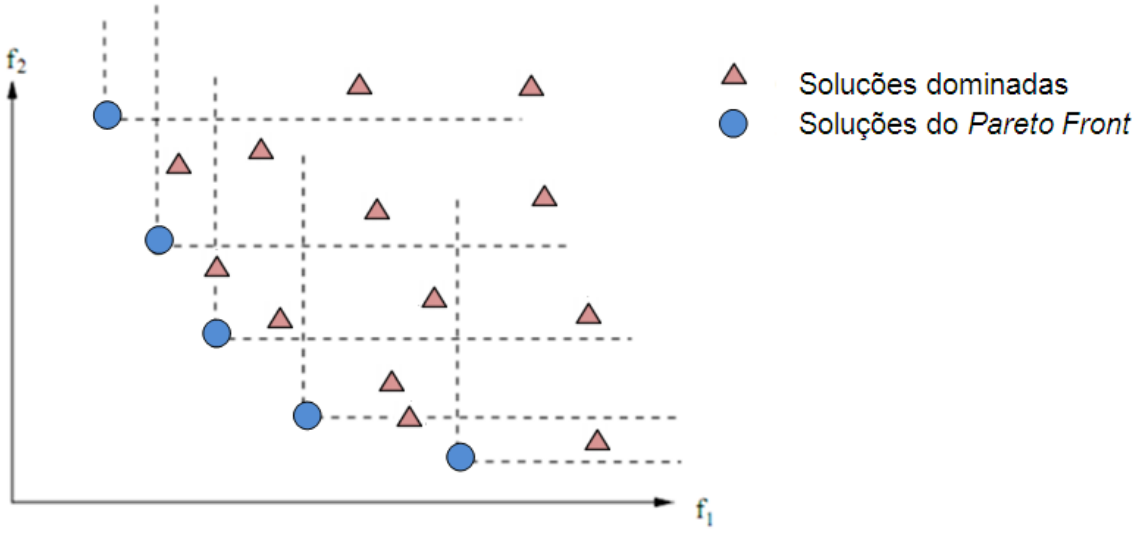

Figura 3.4 Conjunto de soluções de um Pareto Front.

3. Como manter diversidade no enxame de modo a evitar convergência para uma única solução?

As respostas para as perguntas listadas são encontradas através do pseudo-código do algoritmo genérico da aplicação de PSO em MOP, Algoritmo 3.2. Os procedimentos que foram adicionados ao algoritmo do PSO estão marcados em itálico.

O primeiro passo é a inicialização do enxame. Posteriormente, o repositório de soluções não dominadas é atualizado. Este procedimento se dá através da avaliação de dominância entre as soluções provenientes do espaço de busca e o conjunto de soluções presentes no repositório,

Algoritmo 3.2: Pseudo-código do algoritmo genérico do PSO em MOP.

Inicialização:
Criar partículas com posições ($\vec{x}_i$) e velocidade ($\vec{v}_i$) aleatórias
Atribuir a $\vec{p}_i(t)$ a posição atual das partículas.
Atribuir a $\vec{n}_i(t)$ a melhor posição encontrada pela vizinhança.
Atualizar repositório de soluções não dominadas (PF^*).

enquanto ***critério de parada*** *não satisfeito* **faça**
  para cada *partícula p* **faça**
    Passo de Avaliação:
    Selecionar líder social do PF^*.
    Calcular *fitness* $\vec{f}(\vec{x}_i)$.
    Passo de Atualização:
    Atualizar velocidade e posição utilizando as equações de velocidade e posição 3.1, 3.2.
    Atualizar valores de *pbest*.
  fim
  Atualizar PF^*.
fim

objetivando-se gerar um repositório de soluções não dominadas resultante. Esta atualização é fundamental, pois é através do repositório, (Pareto *front*), que seleciona-se um líder para cada partícula do enxame, sendo utilizado como *gbest* na Equação 3.1. Este passo repete a cada iteração. Outro detalhe importante acontece na atualização do *pbest* das partículas. No caso do PSO a atualização acontece se o valor *fitness* da nova posição alcançada pela partícula for menor que a do *pbest*, no caso de problemas de minimização. No cenário multi-objetivo, o *pbest* só é atualizado se a nova posição dominar o *pbest* armazenado; caso contrário, o valor do *pbest* é mantido. Após a atualização da posição e velocidade do enxame, o algoritmo se repete para todas as partículas do enxame, sendo executado até o ponto de parada.

3.3 Variações de Algoritmos Multi-Objetivo

Desde o surgimento da proposta de adaptação do PSO para problemas multi-objetivo, diversas abordagens foram desenvolvidas. Os principais pontos abordados nestes trabalhos estão relacionados a:

1. Como maximizar o número de soluções, e ao mesmo tempo, como eliminar soluções indesejadas no Pareto *front*?

2. Como gerar Pareto *fronts* com qualidade: soluções diversas e uniformemente distribuídas?

3. Como selecionar líderes social e cognitivo adequadamente?

Diante destes questionamentos, são apresentados a seguir 6 algoritmos multi-objetivo inspirados em enxames, que foram desenvolvidos com o intuito de aperfeiçoar o processo de otimização.

3.3.1 MOPSO

O primeiro algoritmo desenvolvido foi o MOPSO, proposto por Coello Coello et al. [29]. A diferença que este algoritmo apresenta com relação ao pseudo-código apresentado no Algoritmo 3.2 é na representação do repositório de soluções não dominadas. O repositório, neste trabalho, é chamado de arquivo externo (AE). Neste AE, o espaço de soluções não dominadas é uma grade adaptativa, sendo reajustado quando uma nova solução é inserida/removida. A grade adaptativa é formada por hipercubos, sendo interpretados como uma região geográfica que contém um certo número de soluções.

Cada hipercubo recebe uma "nota" que depende do número de partículas contidas nele. A nota do hipercubo influencia no processo de seleção do líder social. O hipercubo de maior nota tem mais chances de ser escolhido através de um algoritmo chamado *roulette wheel*. Através dela, é escolhido o hipercubo, e dentro deste, escolhe-se uma partícula de forma aleatória para ser a líder. A nota do hipercubo aumenta à medida que o número de soluções no hipercubo diminui. Isto acontece pois favorece à diversidade.

Este algoritmo é considerado base de todos os demais algoritmos que surgiram. O seu grande problema está no gerenciamento da grade. À medida que novas soluções são inseridas e outras removidas, a grade deve ser atualizada e as notas recalculadas, tornando-se um algoritmo custoso.

3.3.2 m-DNPSO

O m-DNPSO foi proposto por Hu e Eberhart [43] e objetiva a redução do custo computacional do processo de seleção do líder social. O m-DNPSO utilizou o mesmo conceito de AE apresentado no MOPSO, e quando o Pareto *front* possui o certo número de soluções, este processo se torna bastante custoso.

O m-DNPSO tem como característica a resolução de um objetivo a cada passo, após a otimização de um objetivo, otimizar os demais. Os múltiplos objetivos são divididos em dois grupos: f_1 e f_2. f_1 é definido como o objetivo da vizinhança e f_2 como o objetivo de otimização. A seleção destes grupos é arbitrária.

O processo de seleção de líderes acontece da seguinte forma: Primeiramente calcula-se a distância de cada partícula no enxame com relação às soluções no arquivo externo em relação aos objetivos f_1, selecionando-se como vizinhas as m (m é o tamanho da vizinhança) soluções mais próximas, aplicando-se a distancia euclidiana. Finalmente, o *gbest* é escolhido dentre as soluções vizinhas com relação à f_2. Nesta proposta, por mais que haja o custo ao se computar

a distância entre as soluções, este ainda é menor que o processo de seleção apresentado no trabalho anterior.

O grande problema deste trabalho é que o processo de seleção de líderes proposto tende a ser útil apenas no escopo de apenas dois objetivos, além de que a ordem na escolha dos objetivos pode influenciar na performance do algoritmo.

3.3.3 CSS-MOPSO

As principais deficiências encontradas nos algoritmos anteriores são o custo, a convergência prematura e a baixa diversidade. O CSS-MOPSO foi proposto por Chiu [40] e apresenta um método chamado *cross-searching strategy* que favorece a busca local, objetivando uma maior explotação. Este novo método, adicionado ao algoritmo básico multi-objetivo, apresenta uma peculiaridade, que é a ausência do componente cognitivo da Equação 3.1 e a presença de dois componentes sociais. A atualização na equação da velocidade é vista a seguir:

$$\begin{aligned}\vec{v}_i(t+1) = \vec{v}_i(t) + c_1 r_1(\vec{n}_{1i}(t) - \vec{x}_i(t)) + \\ c_2 r_2(\vec{n}_{2i}(t) - \vec{x}_i(t)).\end{aligned} \tag{3.11}$$

Para selecionar-se o $\vec{n_{1i}}(t)$ é necessário definir um ponto *Datum c* que representa a intersecção das linhas perpendiculares referentes às duas soluções extremas do AE. A solução do AE que possuir o menor ângulo θ (ângulo entre a linha que conecta a solução do arquivo externo ao ponto *datum c*) é selecionada como $\vec{n_{1i}}(t)$. A Figura 3.5 (a) ilustra o processo de escolha do $\vec{n_{1i}}(t)$ descrito anteriormente.

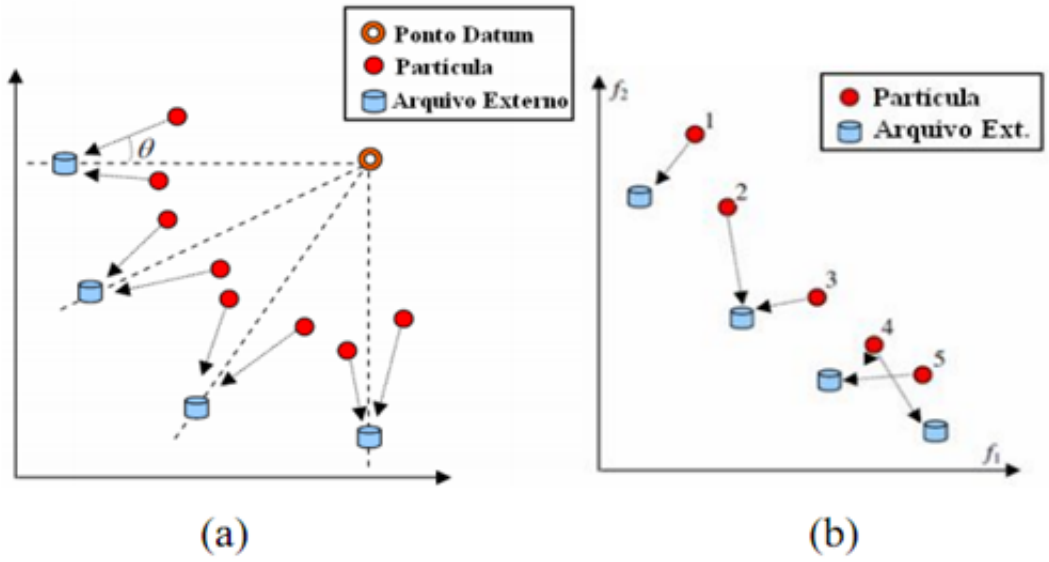

Figura 3.5 Estratégia de escolha do (a) $n_{1i}(t)$ e (b) $n_{2i}(t)$. Figura adaptada de Chiu [40].

A seleção do $\vec{n_{2i}}(t)$ é de acordo com o valor do *fitness* de um objetivo f_i selecionado aleatoriamente. Após ordenar todas as partículas do enxame pelo valor referente ao objetivo f_i,

é atribuído um número serial a cada uma delas. Para todas as partículas com número par, a solução do AE que for mais próxima e que apresentar o valor de *fitness* com relação ao objetivo f_i maior que o da partícula é selecionada como novo $\vec{n_{2i}}(t)$ da partícula. O mesmo princípio é feito para as partículas ímpares, a solução do AE mais próxima cujo *fitness* relacionado ao objetivo f_i for menor que o da partícula será selecionada como $\vec{n_{2i}}(t)$. A Figura 3.5 (b) ilustra a estratégia de escolha do $\vec{n_{2i}}(t)$ para o objetivo f_1.

Este método apresentado alcançou melhoras com relação à diversidade, alcançando novas soluções que antes eram inexistentes no Pareto *front*. Porém, esta estratégia de seleção de líderes sociais também é limitada para problemas com mais de 2 objetivos.

3.3.4 MOPSO-CDLS

Proposto por Tsou et al. [41], o MOPSO-CDLS utiliza um novo mecanismo para a seleção do líder social no arquivo externo. Baseado na proposta de Raquel e Naval Jr. [42], o MOPSO-CDLS utiliza o conceito de *crowding distance* (CD), que é uma estimativa da densidade das regiões do AE. O CD se torna importante na análise de como estão distribuídas as soluções, permitindo uma avaliação do grau de proximidade entre elas. A Figura 3.6 mostra como é realizada a estimativa da densidade. O CD da *i*-ésima solução no AE (representado pelos círculos sólidos) é a média do comprimento do lado do cubóide (caixa pontilhada). As soluções que se localizam em regiões mais populosas possuem CD menor, e, por outro lado, as mais distantes da concentração de soluções possuem CD maior.

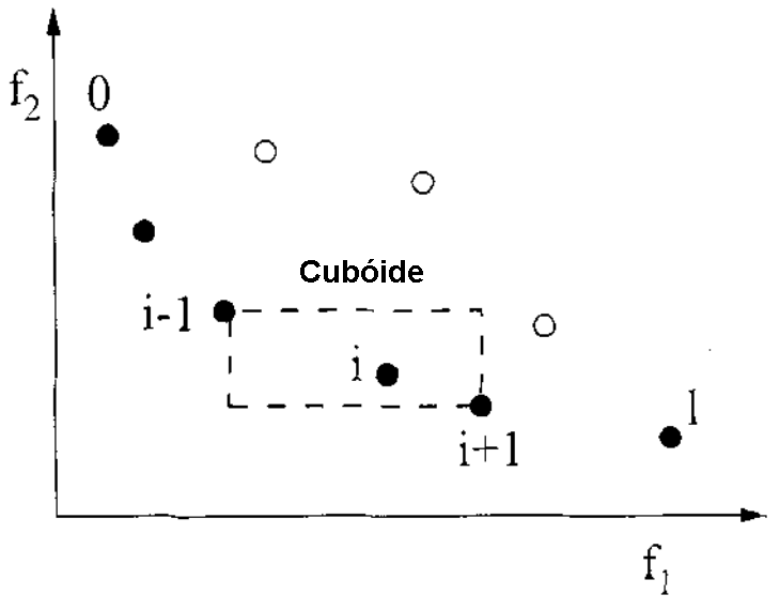

Figura 3.6 Cálculo do *crowding distance* no AE com dois objetivos f_1 e f_2. Os pontos preenchidos são soluções que pertencem ao AE. Figura adaptada de Deb [28].

Como pode-se perceber, este novo mecanismo tornou desnecessária a divisão do AE em grade, reduzindo bastante o custo computacional.

Um dos requisitos considerados importantes para a geração de um bom Pareto *front* é o espalhamento das partículas. Deste modo, as partículas localizadas em regiões populosas devem receber um peso menor na seleção do líder social. A seleção do líder social acontece da seguinte forma: o líder social é aleatoriamente escolhido entre 10% das soluções com maior CD caso a partícula do enxame seja dominada por pelo menos uma das soluções do AE; caso contrário, o líder é escolhido aleatoriamente utilizando todas as soluções do arquivo externo.

A aplicação do mecanismo de CD no processo de seleção do líder social trouxe grande avanço para a geração de Paretos *front* mais diversos e uniformemente distribuídos. Diante disto, novas técnicas passaram a utilizá-lo, propondo mais adaptações no processo de seleção.

3.3.5 MOPSO-CDR

O MOPSO-CDR foi proposto por Santana *et. al* [56] em 2009, visando apresentar melhorias no modo de seleção dos líderes social e cognitivo e também no processo de remoção de soluções do AE. Este algoritmo foi inspirado no MOPSO-CDLS e incorpora uma seleção por roleta baseada em *crowding distance* para selecionar o líder social ($\vec{n}_i$) e prevenir um número excessivo de soluções não dominadas no arquivo externo. Soluções com maior *crowding distance* tem mais chances de serem selecionadas como líder social.

Além disso, o MOPSO-CDR apresenta uma nova forma de atualizar o líder cognitivo ($\vec{p}_i$). O $\vec{p}_i$ de uma partícula é atualizada se a nova posição da partícula domina $\vec{p}_i$. Se a nova posição e o $\vec{p}_i$ forem incomparáveis, o *AE* é utilizado na definição do novo $\vec{p}_i$. O algoritmo busca no *AE* pela solução mais próxima de $\vec{p}_i$ e pela solução mais próxima da nova posição (a distância é calculada através da distância euclidiana). Se a solução proveniente do *AE* relacionada à nova posição possui um CD maior que a solução relacionada à $\vec{p}_i$, a nova posição será escolhida como novo $\vec{p}_i$. Caso contrário, o antigo $\vec{p}_i$ é mantido.

Os resultados obtidos pelo MOPSO-CDR foram bastante interessantes, gerando Pareto *fronts* ainda mais diversos que os do MOPSO-CDLS. Embora bons resultados tenham sido alcançados, este algoritmo não gera Pareto *fronts* com soluções bem uniformes quando comparadas com os resultados obtidos pelo MOPSO-CDLS.

3.3.6 MOPSO-CDRS

O MOPSO-CDRS foi proposto por Miranda e Bastos-Filho [45] em 2011 e é inspirado no MOPSO-CDR. O principal objetivo deste algoritmo é melhorar a diversidade e a uniformidade das soluções de forma balanceada. O MOPSO-CDRS propõe um algoritmo multi-objetivo com dois modos de operação distintos: básico e especiação. O modo básico executa o algoritmo MOPSO-CDR tradicional; e no modo de especiação, o enxame é dividido em sub-enxames. Considerando-se m objetivos, são criados $m+1$ sub-enxames: um sub-enxame executando o MOPSO-CDR e os demais sub-enxames executando o PSO para otimizar cada objetivo. A Figura 3.7 apresenta o modo de especiação considerando-se 2 objetivos f_1 e f_2. Como pode-se ver, o enxame foi dividido em 3 sub-enxames. Um sub-enxame é responsável por executar o MOPSO-CDR levando em consideração os objetivos f_1 e f_2, enquanto um dos sub-enxames é responsável por executar o PSO para otimizar f_1 e o outro para otimizar f_2.

Durante a execução do MOPSO-CDRS, são aplicadas as métricas de qualidade de Pareto

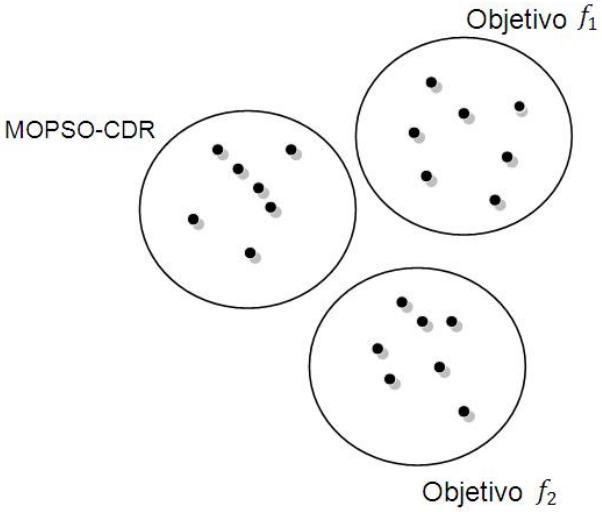

Figura 3.7 MOPSO-CDRS no modo de Especiação com dois objetivos f_1 e f_2.

front, *Spacing* e *Maximum Spread* [69]. Estas métricas são utilizadas para avaliar a qualidade do Pareto *front* com relação à diversidade e distribuição das soluções. As métricas foram aplicadas a cada iteração e o resultado desta aplicação é determinante para a troca de modo operacional do algoritmo (básico ou especiação). Quando o Pareto *front* entra em um processo de estagnação com relação à diversidade, o modo de especiação é ativado passando a operar objetivando uma distribuição mais uniforme das soluções. Quando esta distribuição do Pareto *front* fica estagnada, o modo básico é ativado e o MOPSO-CDR volta a ser executado. O principal objetivo deste algoritmo é proporcionar uma convergência balanceada, gerando um Pareto *front* resultante bem diverso e uniformemente distribuído. O balanceamento proposto pelo MOPSO-CDRS gerou Pareto *fronts* superiores aos apresentados pelo MOPSO-CDR e MOPSO-CDLS tanto em diversidade como em distribuição das soluções.

3.4 Considerações Finais

Neste capítulo foram abordados os conceitos da otimização por enxames de partículas. Primeiramente foram apresentadas a história do PSO, os componentes do algoritmo PSO e a motivação para a utilização de um algoritmo multi-objetivo. Finalmente, foram apresentados a formalização da otimização multi-objetivo, as adaptações necessárias no algoritmo do PSO e seis variações de algoritmos multi-objetivos inspirados em enxame de partículas.

No capítulo seguinte: “Arquitetura híbrida para seleção de parâmetros de SVMs” detalha-se a arquitetura híbrida multi-objetivo que foi implementada, os componentes da arquitetura e suas interações, os desafios encontrados e a implementação da arquitetura.

CAPÍTULO 4

Arquitetura híbrida para seleção de parâmetros de SVMs

Neste capítulo apresenta-se o trabalho desenvolvido, uma arquitetura híbrida multi-objetivo que combina algoritmos de otimização multi-objetivo com meta-aprendizado para solucionar o problema de seleção de parâmetros de SVMs. Foi apresentada na Seção 2.4 do Capítulo 2 uma abordagem híbrida, combinando-se MA com PSO [60]. Foi constatado um grande potencial no trabalho realizado devido ao bom desempenho alcançado em comparação com a utilização tradicional de PSO e de MA na seleção de parâmetros de SVM. Como a abordagem híbrida foi pouco estudada até o momento, decidimos investigar pontos a serem melhorados de modo a tentar aperfeiçoar o processo.

O ponto crucial identificado na abordagem híbrida, que fundamenta este trabalho, é a sua limitação quanto ao número de objetivos levados em consideração. Apenas a minimização da taxa de erro de regressão é utilizada em [60] para definir a qualidade das soluções. De acordo com Narzisi [20], o problema de seleção de parâmetros de SVM é em sua essência um problema multi-objetivo. A estrutura de sistemas de aprendizado supervisionado exige uma combinação adequada entre uma série de objetivos possivelmente conflitantes.

Neste trabalho a arquitetura proposta por Gomes e Prudêncio [60] foi adaptada para o cenário multi-objetivo. Deste modo, a nova arquitetura gera soluções otimizadas levando-se em consideração múltiplos critérios. Para a realização deste trabalho, foi implementado um *framework* de recomendação multi-objetivo, ver Figura 4.1.

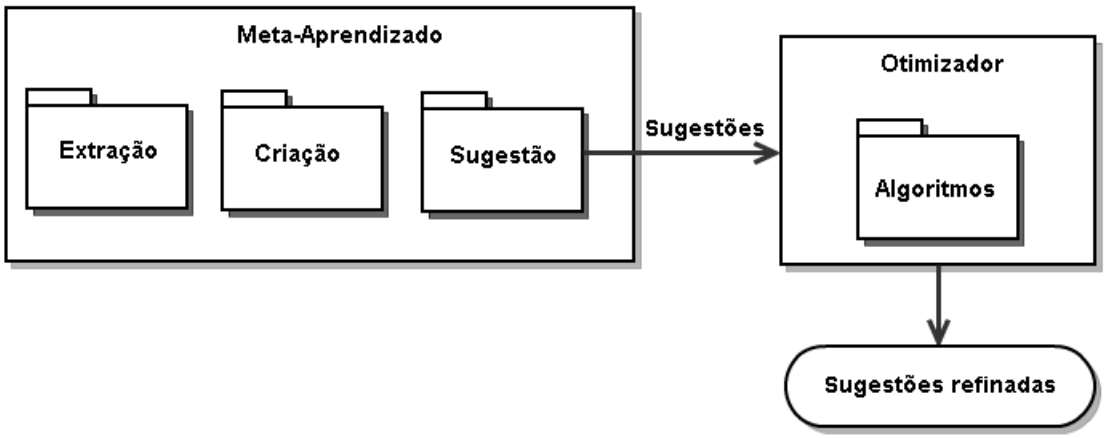

Figura 4.1 *Framework* de recomendação multi-objetivo.

Este *framework* incorpora a implementação do *Meta-Aprendizado*, composto pelos pacotes: *Extração*, responsável pela extração das meta-características; *Criação*, responsável pela criação da meta-base; e *Sugestão*, responsável pela sugestão de valores de parâmetros. Além disso, o *framework* incorpora o *Otimizador* (*engine* de otimização multi-objetivo). Nele foram implementados 6 algoritmos multi-objetivo para a realização dos experimentos. O *Otimizador* possui duas formas de execução de algoritmos: com inicialização aleatória (forma tradicional) e a híbrida, que faz uso dos parâmetros sugeridos pelo *Meta-Aprendizado*.

O objetivo deste trabalho é avaliar se a abordagem híbrida gera melhores soluções quando comparadas a técnicas de otimização multi-objetivo tradicionais. Nos experimentos, realizamos a comparação dos 6 algoritmos implementados com suas respectivas versões híbridas.

4.1 Arquitetura do Trabalho

A arquitetura desenvolvida neste trabalho é baseada em Gomes e Prudêncio [60], que propuseram a combinação de meta-aprendizado com algoritmos baseados em enxames com único objetivo.

Para o cenário multi-objetivo foram necessárias algumas mudanças no processo tradicional de único objetivo. A primeira mudança é com relação ao número de objetivos que vão ser usados para avaliar a solução. A segunda mudança é no processo de sugestão de soluções a partir dos meta-exemplos mais similares. No cenário de único objetivo era feita uma ordenação baseada no desempenho, o mesmo não pode ser feito no contexto multi-objetivo. Finalmente, a terceira mudança é no módulo de busca, onde o algoritmo de otimização passa a ser multi-objetivo.

A Figura 2.8 apresentada no Capítulo 2 foi adaptada para o cenário multi-objetivo, sendo sua estrutura mantida. Inicialmente, há a criação da meta-base. Na arquitetura desenvolvida, os meta-exemplos são criados da mesma forma, porém, agora com múltiplas medidas de desempenho e não apenas uma, como proposta por Gomes e Prudêncio [60]. O meta-aprendiz recupera um número pré-definido de meta-exemplos armazenados na meta-base, selecionados com base na similaridade dos valores de suas meta-características com as do problema de entrada. Deste modo, as configurações de parâmetros dos meta-exemplos mais similares são recomendadas para o módulo de busca, que as utiliza como população inicial. O processo de recomendação de configurações de parâmetros também muda nesta nova arquitetura, pois várias medidas de desempenho devem ser consideradas. O algoritmo de busca multi-objetivo é executado usando as soluções sugeridas pelo meta-aprendiz, gerando novas configurações a serem executadas pela SVM. Ao final do processo, o módulo de busca gera as melhores soluções, fruto do refinamento.

Nas subseções a seguir, cada módulo da arquitetura desenvolvida será apresentado.

4.1.1 Módulo de Busca

O problema de seleção de parâmetros de SVM pode ser definido como um problema de otimização. Considerando o espaço de busca de parâmetros $\mathscr{S}$ e um conjunto de funções objetivo $\vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]$, onde $\vec{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ é o vetor de variáveis de deci-

são (configurações de parâmetros) em $\mathscr{S}$. A tarefa de otimização é encontrar um conjunto de configurações $S^{opt} \subset \mathscr{S}$ considerando o vetor de objetivos $\vec{f}(\vec{x})$.

Em uma técnica de otimização iterativa, o processo de busca começa com um conjunto inicial $S^{ini} \subset \mathscr{S}$ de possíveis soluções que progressivamente deslocam-se em direção a melhores regiões. Dado um problema de aprendizado d, as configurações de parâmetros otimizadas S^{opt} encontradas durante a busca podem ser vistas como uma função dos pontos de busca iniciais:

$$S^{opt} \leftarrow Search(d, S^{ini}) \tag{4.1}$$

Em uma proposta de busca convencional, S^{ini} é inicializado aleatória e uniformemente no $\mathscr{S}$. Idealmente, a técnica de busca deveria ser menos dependente da população inicial. De modo a minimizar esta dependência, o conjunto S^{ini} é recomendado pelo meta-aprendizado.

4.1.2 Meta-Base

A meta-base é derivada de um conjunto de problemas de aprendizado $\mathscr{D}$, que podem ser coletados de um repositório de dados ou de base de dados artificialmente geradas. Formalmente, $\mathscr{M}$ é o conjunto de meta-exemplos derivados de $\mathscr{D}$. Cada meta-exemplo $e_i \in \mathscr{M}$ é relacionado a um problema de aprendizado $d_i \in \mathscr{D}$ e armazena: (1) um vetor de p meta-características $\mathbf{c}_i = (c_i^1, \ldots, c_i^p)$ descrevendo o problema d_i; (2) as configurações de parâmetros mais adequadas $(s_i^1, \ldots, s_i^*) \subset \mathscr{S}$, com seu respectivo desempenho no problema. Dois pontos são considerados na criação de meta-exemplos: (1) as meta-características adotadas para descrever os problemas de aprendizado; (2) a definição das configurações mais adequadas para o problema.

As meta-características formam um conjunto de aspectos descritivos das bases de dados (e.g., o número de atributos, número de *outliers*, número de atributos categóricos, entre outros) que sejam capazes de capturar informação útil para determinar o desempenho dos parâmetros. Diversos trabalhos têm sido desenvolvidos para definição de meta-características, incluindo medidas gerais, estatísticas, de teoria da informação e baseadas em modelo (veja [68, 57, 54]). De acordo com Prudêncio e Ludermir [58], existem algumas formas para se escolher meta-características. Primeiro, meta-características devem ser adequadas para a classe do problema a ser resolvido (por exemplo, classificação ou regressão). Segundo, meta-características não devem ser provenientes de análise subjetiva, como inspeção visual de gráficos.

Finalmente, o conjunto de configurações $(s_i^1, \ldots, s_i^*)$ armazenado no meta-exemplo é definido através da aplicação de um conjunto de parâmetros candidatos $S^C \subset \mathscr{S}$ no problema associado. As configurações mais adequadas em S^C são escolhidas levando-se em consideração o seu desempenho com relação aos objetivos presentes em $\vec{f}(\vec{x})$.

A qualidade das configurações armazenadas depende do número de configurações de parâmetros candidatos em S^C. Considerando um pequeno conjunto de candidatos, a qualidade do meta-aprendizado pode ser prejudicada. Por outro lado, considerando-se um grande número de candidatos, o custo da criação da meta-base aumenta.

4.1.3 Meta-Aprendiz

Após a construção da meta-base, o meta-aprendiz $\mathscr{L}$ é responsável por sugerir configurações de parâmetros adequadas baseadas na descrição do problema em questão. Formalmente, **c** é a descrição do novo problema d e $\mathscr{M}$ é o conjunto de meta-exemplos. O meta-aprendiz gera um conjunto S^{rec} de configurações recomendadas:

$$S^{rec} \leftarrow \mathscr{L}(\mathbf{c}, \mathscr{M}) \tag{4.2}$$

Inicialmente, dada a descrição **c**, o meta-aprendiz seleciona os meta-exemplos mais similares de $\mathscr{M}$. A similaridade dos meta-exemplos é definida em termos dos valores das meta-características. As configurações armazenadas contidas nos meta-exemplos selecionados são inseridas no conjunto a ser recomendado S^{ini}.

O conjunto de configurações de parâmetros recomendado pelo meta-aprendizado é adotado como população inicial do módulo de busca. Deste modo, a estratégia híbrida pode ser formalmente definida como:

$$Hybrid(d, \mathbf{c}, \mathscr{M}) = Search(d, \mathscr{L}(\mathbf{c}, \mathscr{M})) \tag{4.3}$$

Na seção seguinte, será apresentada a implementação da solução híbrida apresentada.

4.2 Implementação da Arquitetura

Neste trabalho, implementamos a solução híbrida para selecionar parâmetros da SVM para problemas de classificação. Embora a implementação e o caso de estudo estejam focados em classificação, enfatizamos que a solução proposta também pode ser aplicada em problemas de regressão.

Neste trabalho, foi utilizado o *framework Scikit Learn* para implementar SVMs [65]. Este *toolbox* reuni implementações de diversos algoritmos presentes na aprendizagem de máquina, sendo todos bem documentados e validados. Dois parâmetros específicos foram considerados: parâmetro γ do kernel RBF e o parâmetro de regularização C.

Nas próximas sub-seções serão detalhadas as implementações de cada módulo da arquitetura da solução.

4.2.1 Módulo de Busca

Foram implementados algoritmos de otimização multi-objetivo, apresentados na Seção 3.3, sendo adaptadas para buscar configurações (γ, C). As funções objetivo escolhidas avaliam a qualidade de desempenho e a complexidade do modelo da SVM ao executar configurações para um dado problema de classificação.

Neste trabalho, foram definidos como funções objetivo a taxa de sucesso ou acerto (TS) na classificação, a forma mais direta de calcular o desempenho do modelo da SVM, e o número de vetores de suporte (NVS), que influencia na complexidade do modelo da SVM [52]. Os valores

de cada objetivo podem ser obtidos através da SVM no experimento de validação cruzada *10-fold*. Desta forma, o objetivo da busca é encontrar configurações γ do kernel RBF e C, não dominadas que maximizem a TS e minimizem NVS para um dado problema de classificação.

Nesta implementação, cada partícula representa uma configuração $x_i = (\gamma, C)$, indicando a posição da partícula no espaço de busca. Cada partícula também possui uma velocidade que indica a direção atual do pássaro na busca. O algoritmo multi-objetivo atualiza a posição (fórmula 3.2) e velocidade (fórmula 3.1) de cada partícula através das fórmulas apresentadas no Capítulo 3, explorando progressivamente as melhores regiões do espaço de busca. Embora seja a mesma equação do PSO com único objetivo, o processo de atualização de $\vec{n}_i(t)$ é diferente. Faz parte do processo seletivo as soluções não dominadas durante a busca (Pareto *front*).

Em nosso trabalho, os algoritmos foram implementados para realizar a busca em um espaço representado por uma grade discreta de configurações de SVM, consistindo de 399 diferentes configurações de parâmetros γ e C. Para isto, os valores contínuos referentes à posição da solução é discretizado para um dos valores contidos na grade. Seguindo as instruções sugeridas em [27], foram considerados como potenciais valores para as configurações escolhidas: γ assumiu 19 valores diferentes (de 2^{-15} a 2^{3}) e C assumiu 21 diferentes valores (de 2^{-5} a 2^{15}), gerando-se $19x21 = 399$ diferentes combinações de valores de parâmetros no espaço de busca.

4.2.2 Meta-Base

A criação da meta-base envolveu 40 bases de dados correspondentes a 40 problemas de classificação distintos, disponíveis para download no site do repositório do *UCI Machine Learning*. A lista de problemas de classificação adotados neste trabalho para a geração de meta-exemplos é apresentada na Tabela 4.1.

Tabela 4.1 Problemas de Classificação Adotados para a Geração de Meta-Exemplos

Balance Scale	Blood	Breast Tissue	Breast-w
Cancer	Colic	Colon	Column-2c
Column-3c	Ecoli	Glass	Haberman
Heart	Heart-Statlog	Hepatitis	Hill Valley
Hypothyroid	Ionosphere	Iris	Kr-vs-kp
Letter	Libras	Lung Cancer	Lymph
Mamography	Optdigits	Parkinson	Pen Digits
Primary Tumor	Prina Diabetes	Red Wine Qual	Segment
Sick	Sonar	Vehicle	Vote
White Wine Qual	Wine	Yeast	Zoo

As bases de dados selecionadas não possuem atributos categóricos, pois o algoritmo da SVM foi concebido para trabalhar com valores numéricos. Os valores faltosos foram substituídos pela média dos valores e os conjuntos de dados tiveram a ordem de seus exemplos alterada de maneira aleatória para minimizar alguma tendência da coleta de dados do conjunto original. Estes problemas correspondem a bases de dados associadas a diferentes domínios de aplicação.

A diversidade de domínios é positiva em nosso contexto, pois as características das bases de dados tendem a ter uma boa variação.

A meta-base é um repositório de meta-exemplos, sendo cada meta-exemplo, como mencionado anteriormente, composto por um vetor de meta-características e pelas configurações de parâmetros mais adequadas juntamente com a respectiva informação de desempenho no problema. As seções seguintes detalham a respeito destes componentes do meta-exemplo.

4.2.2.1 Meta-Características

Neste trabalho, foram utilizadas 8 meta-características para descrever os conjuntos de dados dos problemas de classificação. Estas meta-características foram retiradas do conjunto de características definido em [21]. As meta-características selecionadas são listadas na Tabela 4.2, e divididas em três categorias: Simples, Estatística e Teoria da Informação.

Tabela 4.2 Meta-Características para Problemas de Classificação.

Simples
Número de Exemplos
Número de Atributos
Número de Classes
Estatística
Média da Correlação entre Atributos
Média Geométrica dos Atributos
Skewness
Kurtosis
Teoria da Informação
Entropia de Classe

Os dados do tipo Simples são os já discriminados nas bases de dados de classificação, tais como número de exemplos, atributos e classes. O grupo do tipo Estatística é composto pela média da correlação entre os atributos; a média geométrica dos atributos, que indica a tendência central ou valor típico do conjunto de atributos; *skewness* médio, que mede a assimetria da distribuição dos dados com relação ao eixo central; e o *kurtosis* médio, que mede a dispersão da distribuição dos dados. Finalmente, o grupo de Teoria da Informação é composto pela entropia, que define um grau de incerteza de classificação. No Apêndice A são apresentados os valores de cada meta-característica nos 40 problemas de classificação. As meta-características referentes à categoria Simples foram nomeadas como MC_1, MC_2 e MC_3 respectivamente, já as relacionadas à categoria Estatística foram nomeadas como MC_4, MC_5, MC_6 e MC_7 respectivamente. Finalmente, a meta-característica da categoria Teoria da Informação foi nomeada como MC_8.

4.2.2.2 Informações de Desempenho

Para o cenário multi-objetivo, as configurações de parâmetros de cada meta-exemplo precisam ser avaliadas levando-se em consideração múltiplas perspectivas ou objetivos contidos em $\vec{f}(\vec{x})$. Como mencionando anteriormente, os objetivos escolhidos neste trabalho foram TS e NSV. As sugestões geradas pelo meta-aprendizado são provenientes dos problemas mais similares com base nos dois objetivos, visando gerar um modelo de baixa complexidade e com um bom potencial de acerto de classificação.

4.2.3 Meta-Aprendiz

O meta-aprendiz é responsável por sugerir soluções de problemas similares com relação ao problema de entrada. Neste trabalho, a comparação entre meta-exemplos é realizada através da distância euclidiana entre suas meta-características.

No Capítulo 2, foi citada a combinação de meta-aprendizado com algoritmos inspirados em enxames de único objetivo no problema de seleção de parâmetros. Nesta situação, o processo de sugestão de configurações de parâmetros é feito levando-se em consideração o objetivo escolhido, que é a taxa de erro da SVM na regressão. Deste modo, o melhor conjunto de configurações de parâmetros é composto pelas configurações que possuem as menores taxas de erro. Por outro lado, no cenário em que estamos trabalhando (com múltiplos objetivos), a escolha das configurações de parâmetros não é trivial como o cenário apresentado anteriormente. Pelo fato de existirem diversos critérios a serem considerados para se escolher configurações, a comparação não acontece de forma direta, como na abordagem de única restrição.

Neste trabalho foi utilizado um mecanismo para seleção de configurações em cenários de múltiplos objetivos. Este mecanismo é baseado no mesmo conceito de dominância, visto na Seção 3.2, que é aplicado em algoritmos de otimização multi-objetivo. O primeiro passo é aplicar dominância entre todas as configurações de parâmetros para os meta-exemplos mais similares, gerando-se para cada meta-exemplo um Pareto *front*. O próximo passo é selecionar as configurações a partir dos Pareto *fronts* gerados. Neste mecanismo, ordenam-se as configurações de cada Pareto *front* com relação ao *crowding distance* (veja na sub-seção 3.3.4); e seleciona-se uma configuração aleatoriamente, de cada Pareto *front*, pertencente ao grupo de soluções com maior CD.

O mecanismo desenvolvido seleciona uma solução de cada um dos k problemas mais similares. Embora a estratégia gere um conjunto de configurações através de uma seleção aleatória, são selecionadas configurações baseando-se em CD. Deste modo, apenas configurações não dominadas localizadas em regiões de baixa densidade que poderão ser selecionadas. A vantagem deste mecanismo está na seleção de configurações localizadas em regiões promissoras, considerando-se, ainda, o espalhamento entre as soluções. Este procedimento pode auxiliar na geração de Paretos *front* bem distribuídos desde as iterações iniciais, maximizando a qualidade.

Um ponto crítico existente neste mecanismo é a definição de k, o número de problemas similares a serem selecionados. Se k assume um valor baixo, problemas similares podem ser descartados. Por outro lado, caso k assuma um valor alto, é possível que problemas não similares sejam considerados, podendo impactar no processo de busca.

4.3 Considerações Finais

Neste capítulo foi apresentado o trabalho desenvolvido: “Arquitetura Híbrida para Otimização Multi-Objetivo de SVMs”. Inicialmente apresentamos a motivação e o *framework* de recomendação multi-objetivo, tornando possível a realização desta pesquisa. Além disso, a arquitetura e a implementação do trabalho desenvolvido foram detalhados, discutindo-se cada componente: busca, meta-base e meta-aprendiz.

O próximo capítulo: "Experimentos e Resultados"apresenta todo o arranjo experimental: configuração dos parâmetros, métricas de avaliação dos algoritmos e metodologia do experimento; e os resultados obtidos.

CAPÍTULO 5

Experimentos e Resultados

Este trabalho apresenta uma alternativa híbrida para o problema de seleção de parâmetros de SVM, combinado-se algoritmos de otimização e busca com meta-aprendizado. Objetivando validar a hipótese considerada, ou seja que a combinação de meta-aprendizado com algoritmos multi-objetivo é uma alternativa promissora para a seleção de parâmetros de SVM, iremos apresentar o arranjo experimental e os resultados obtidos.

5.1 Arranjo Experimental

O experimento objetiva verificar o desempenho da abordagem desenvolvida com relação ao problema de seleção de parâmetros de SVM. Para tal, foram implementados 6 algoritmos de otimização multi-objetivo (apresentados na Seção 3.3 do Capítulo 3). Cada algoritmo foi ajustado para o problema em questão, e executado na forma tradicional (algoritmo puro) e com meta-aprendizado (algoritmo híbrido), resultando em 12 simulações, 6 simulações referentes à aplicação dos algoritmos em sua forma tradicional (sem meta-aprendizado) e 6 simulações referentes à aplicação dos algoritmos híbridos (com meta-aprendizado).

5.1.1 Configuração dos Parâmetros

Cada algoritmo apresenta seu próprio conjunto de parâmetros. No caso do MOPSO, a taxa de mutação é $0,5$, o número de divisões para o grid adaptativo é 30 e o fator de inércia diminui linearmente de $0,4$ a $0,0$ [29]. No MOPSO-CDLS o fator de inércia diminui linearmente de $0,9$ a $0,4$ [41]. No m-DNPSO o valor de $m = 10$ e o fator de inércia é gerado aleatoriamente em cada iteração no intervalo $[0,5;1,0]$ [56]. O CSS-MOPSO adota como desvio padrão para a mutação Gaussiana o valor $0,01$ e o fator de inércia diminui linearmente de $0,9$ a $0,4$ [40]. No MOPSO-CDR é utilizado uma taxa de mutação de $0,05$ e o fator de inércia diminui linearmente de $0,9$ a $0,4$. O MOPSO-CDRS apresenta a mesma configuração do MOPSO-CDR, porém, pelo fato de haver a análise estatística quanto à melhora significativa, em relação às métricas *Spacing* e *Maximum Spreading*, do Pareto *front*, adota-se como limite de saturação $0,1\%$. As constantes de aceleração cognitiva e social igual a $1,49$, quando aplicadas.

Além dos algoritmos, o parâmetro k, referente ao tamanho da população, também foi definido com os valores $k = 5$ e 10. Foram adotadas 10 iterações como ponto de parada da execução do algoritmo.

5.1.2 Métricas de Avaliação dos Algoritmos

O experimento envolve a aplicação de algoritmos multi-objetivo tradicionais e híbridas. Deste modo, foram necessárias métricas para calcular a qualidade das configurações resultantes (Pareto *front*) de cada algoritmo. Neste trabalho foram utilizadas quatro métricas: *Hypervolume* (HV), *Spacing* (S), *Maximum Spread* (MS) e *Coverage* (CV). Cada uma destas métricas avalia diferentes aspectos do Pareto *front* obtido.

A métrica *Hypervolume* foi proposta por Zitzler e Thiele [69] e é definida como o hipervolume no espaço de objetivos coberto pelo Pareto *front* obtido ($\mathscr{P}^*$). Para um problema de otimização com w objetivos, HV é definida por:

$$HV = \left\{ \bigcup_i a_i \,|\, s_i \in \mathscr{P}^* \right\}, \tag{5.1}$$

onde s_i ($i = 1,2,...,n$) é uma solução não dominada do Pareto *front* ($\mathscr{P}^*$), n é o número de soluções no Pareto *front* e a_i é o *hypervolume* do hipercubo delimitado pela posição da solução s_i no espaço de objetivos e a origem.

A métrica *Spacing* foi proposta por Schott [70]. S estima a diversidade do Pareto *front* obtido. S é calculada pela computação da distância relativa entre as soluções adjacentes do Pareto *front*:

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2}, \tag{5.2}$$

onde n é o número de soluções não dominadas, d_i é a distância entre as soluções não dominadas adjacentes para a solução s_i e $\overline{d}$ é a distância média entre as soluções não dominadas adjacentes. $S = 0$ significa que todas as soluções do Pareto *front* são igualmente espaçadas. Deste modo, valores de S que tendem a zero são preferidos.

O *Maximum Spread* foi proposto por Zitzler et al. [71] e calcula a extensão máxima coberta pelo Pareto *front*. MS é computada usando-se a Equação 5.3:

$$MS = \sqrt{\sum_{m=1}^{k} (max_{i=1}^{n} f_m^i - min_{i=1}^{n} f_m^i)^2}, \tag{5.3}$$

onde n é o número de soluções no Pareto *front* e k é o número de objetivos. Valores de MS maiores indicam maior diversidade.

A métrica *coverage* foi proposta por Zitzler *et. al* [71]. CV é calculada através da Equação 5.4:

$$CV(A,B) = \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|}, \tag{5.4}$$

onde A e B são dois Pareto *fronts*. O valor $CV(A,B) = 1$ significa que todas as soluções em B são dominadas por A. Por outro lado, $CV(A,B) = 0$ significa que nenhuma das soluções em B é dominada por A. Ambos $CV(A,B)$ e $CV(B,A)$ tem que ser calculados, pois $CV(A,B)$ não é

necessariamente igual a $1 - CV(B,A)$. Se $0 < CV(A,B) < 1$ e $0 < CV(B,A) < 1$, então nem A domina totalmente B, nem B domina totalmente A. Esta métrica é interessante, pois permite a comparação entre Pareto *fronts* gerados por algoritmos diferentes, baseando-se na dominância entre suas soluções.

5.1.3 Metodologia do Experimento

Para a realização dos experimentos foi aplicada a medida de avaliação do erro de generalização chamada *leave-one-out*. Esta medida é um caso especial de validação cruzada. Uma regra de decisão é construída com base nos dados de treinamento. O conjunto de treinamento é composto por todas as instâncias disponíveis menos um e o teste é realizado utilizando a instância removida [77]. Esse procedimento é repetido até que cada instância do conjunto de treinamento seja testada (utilizando diferentes regras de decisão).

A cada passo do *leave-one-out*, um meta-exemplo é retirado para ser utilizado (como novo problema) pelo algoritmo de busca e os 39 meta-exemplos restantes são mantidos para serem selecionados pelo meta-aprendiz. Um número k de meta-exemplos mais similares é selecionado, sendo escolhida uma configuração de cada um deles. O conjunto k de configurações é sugerido como população inicial para o algoritmo de otimização e busca multi-objetivo. O algoritmo, então, otimiza as configurações da SVM levando em consideração o problema retirado. A cada iteração um *Pareto Front* é formado, e para mensurar sua qualidade, aplicamos métricas (vistas na Sub-seção 5.1.2). Este procedimento é repetido 30 vezes por questões de confiabilidade.

Para avaliar o desempenho da nossa abordagem, os resultados foram analisados através de duas perspectivas. A primeira, *Perspectiva 1*, é com relação à média, por iteração, dos valores associados à qualidade das configurações (levando em consideração os 40 problemas de classificação). Este tipo de análise fornece uma informação importante, que é o comportamento médio da convergência do Pareto *front* ao longo das iterações.

A segunda análise, *Perspectiva 2*, é com relação ao número de vitórias de cada algoritmo (levando em consideração os 40 problemas de classificação) por iteração. O número de vitórias está relacionado à qualidade (ver as métricas de qualidade na seção 5.1.2) das configurações durante a otimização, comparando-se os resultados obtidos pelos algoritmos tradicionais com os respectivos algoritmos híbridos.

5.2 Resultados

De modo a analisar os resultados adequadamente, foi realizada uma análise estatística. Como os dados não seguem uma distribuição normal (após aplicação do teste Kolmogorov-Smirnov), foi aplicado o teste de Wilcoxon para testar as hipóteses com 95% de confiança. Todas as análises a seguir usaram esta metodologia.

Inicialmente são apresentados os resultados da *Perspectiva 1*, observando-se todos os algoritmos em cada métrica de qualidade adotada. As Figuras 5.1 e 5.2 apresentam os resultados com relação ao *Spacing*. Esta métrica de qualidade deve ter seu valor minimizado, de modo a garantir um Pareto *front* bem distribuído e equidistante.

Como pode-se observar na Figura 5.1, todas as abordagens híbridas geraram Pareto *fronts* com valores de *spacing* mais baixos ao final da simulação. Percebe-se também que o meta-aprendizado gerou boas soluções iniciais para os algoritmos, contribuindo para o bom início da execução dos algoritmos (Figura 5.1 A, C, D e E). O resultado apresentado pelo algoritmo MOPSO-CDR, em sua versão híbrida, não foi melhor que a sua versão tradicional, no início da simulação, devido a sua característica nativa que é valorizar o espalhamento entre as partículas. O algoritmo MOPSO-CDRS, extensão do MOPSO-CDR, balanceia o comportamento de espalhamento e espaçamento das soluções; deste modo, observa-se uma melhoria, no início da simulação, com relação ao MOPSO-CDR.

Na Figura 5.2, nota-se que o aumento do número de partículas favoreceu o desempenho tanto dos algoritmos híbridos quanto dos tradicionais, havendo uma redução significativa nos valores de *spacing* desde as primeiras iterações. Isto é devido à maior diversidade e interação entre as partículas. Os algoritmos híbridos apresentaram resultados ainda melhores devido ao aumento do número de partículas sugeridas como população inicial, fazendo com que soluções que seriam encontradas posteriormente já estivessem contidas no enxame.

Embora seja visível a contribuição do meta-aprendizado para todos os algoritmos, fez-se necessária a aplicação do teste de Wilcoxon, garantindo-se, assim, com 95% de confiança, que os resultados de todas as abordagens híbridas são melhores que sua versão tradicional, tanto para 5 como para 10 partículas levando-se em consideração toda a simulação.

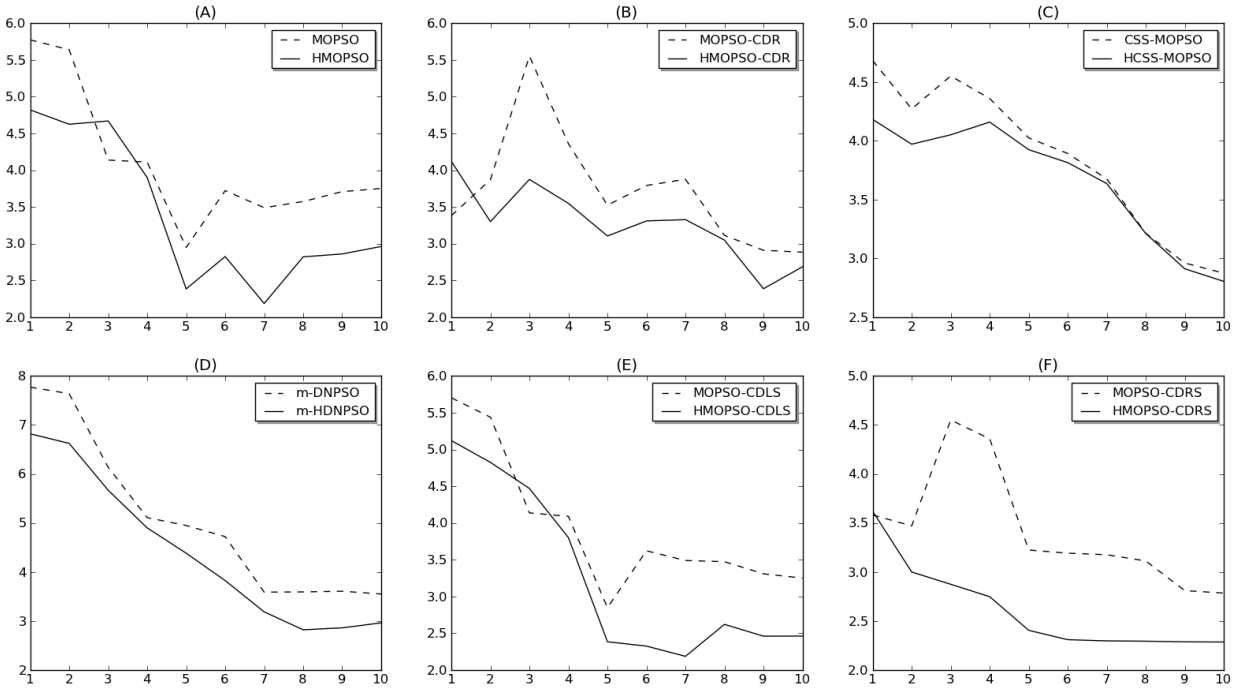

Figura 5.1 Média de *Spacing* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

As Figuras 5.3 e 5.4 apresentam os resultados com relação ao *hypervolume*. Esta métrica deve ter seu valor maximizado, pois representa a área (ou volume dependendo do número de

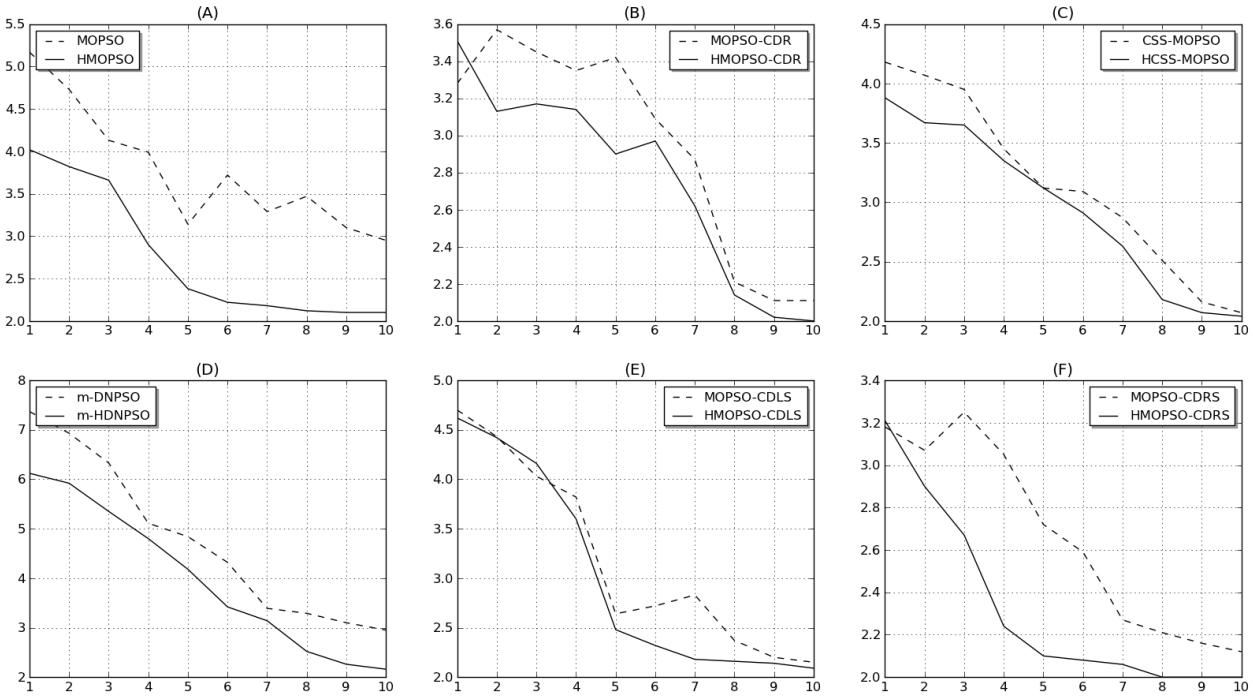

Figura 5.2 Média de *Spacing* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

objetivos) formada pelas soluções do Pareto *front*.

Os resultados apresentados na Figura 5.3 mostram que as soluções sugeridas pelo MA potencializaram o desempenho dos algoritmos, tornando-os capazes de encontrar soluções mais interessantes, aumentando a qualidade do Pareto *front* com relação à métrica em questão. Levando em consideração toda a simulação, todos os algoritmos híbridos atingiram, com 95% de confiança, melhores resultados que suas versões tradicionais. Dentre os algoritmos avaliados, percebe-se que o MOPSO-CDR e o CSS-MOPSO, a partir da sétima iteração, apresentam similaridade com relação aos respectivos algoritmos híbridos (ver Figura 5.3 B e C). Deste modo, foram aplicados testes estatísticos para avaliar o nível de similaridade entre as distribuições, da sétima à última iteração, sendo constatado que o HMOPSO-CDR e o HCSS-MOPSO são considerados similares à sua versão não híbrida.

A Figura 5.4 mostra a melhora dos valores de *hypervolume* com o aumento do número de partículas, para todos os algoritmos híbridos e tradicionais. Na simulação com 5 partículas, dois algoritmos não atingiram superioridade com 95% de confiança. No cenário com 10 partículas, o HMOPSO-CDR passou a prevalecer com 95% de confiança e o HCSS-MOPSO, mesmo obtendo resultados melhores, continuou sendo considerado similar à versão tradicional (ver Figura 5.4 B e C).

As Figuras 5.5 e 5.6 apresentam os resultados com relação ao *Max. Spread*. Na Figura 5.5, levando-se em consideração a primeira iteração de todos os algoritmos, percebe-se que a sugestão de soluções realizada pelo MA proporcionou um bom início para todos. Apesar desta vantagem, a natureza dos algoritmos MOPSO e m-DNPSO (ambos não possuem mecanismos

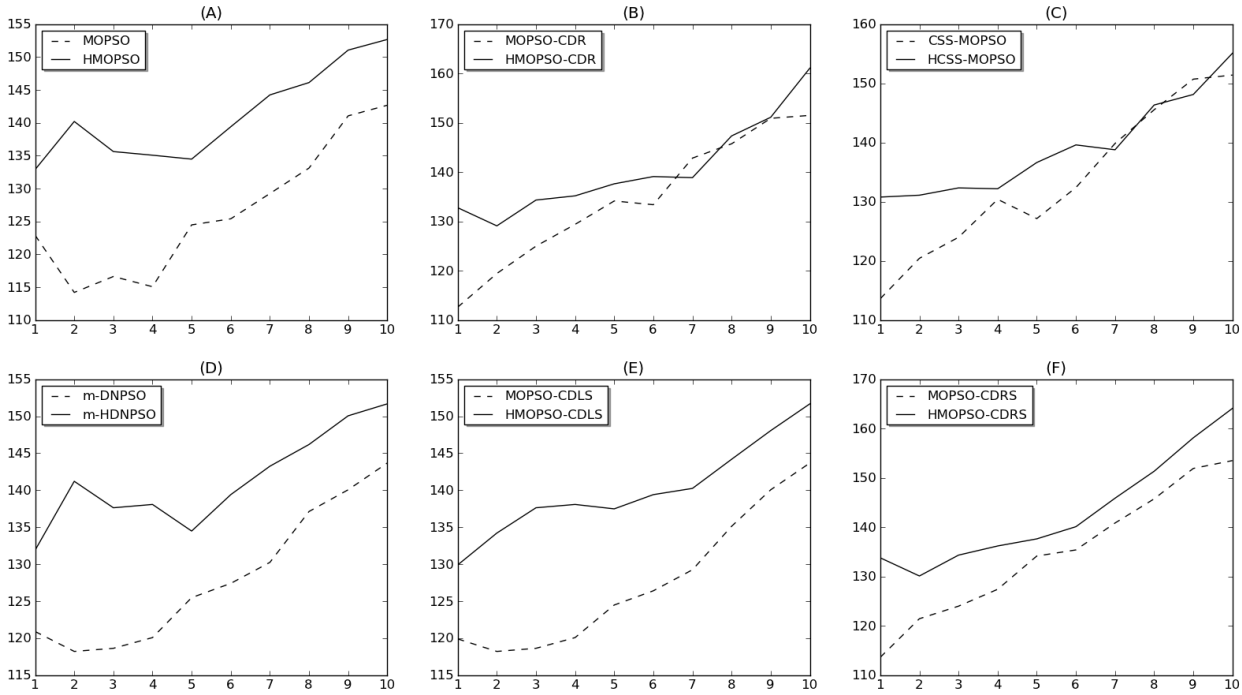

Figura 5.3 Média de *hypervolume* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

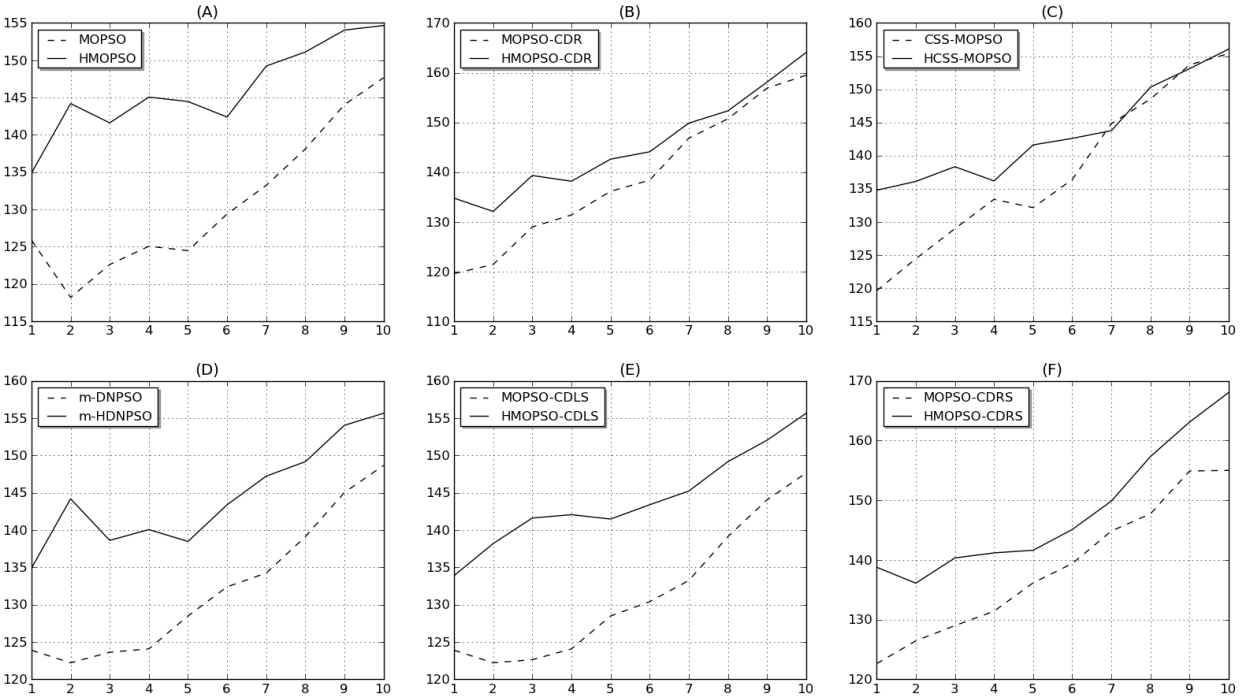

Figura 5.4 Média de *hypervolume* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

de diversidade) não favoreceu a convergência para um Pareto *front* com bom espalhamento nas primeiras iterações. Sendo assim, as duas abordagens híbridas são consideradas iguais aos respectivos algoritmos tradicionais, considerando-se a simulação como um todo. Por outro lado, os resultados apresentados na Figura 5.5 B, C, E e F mostram que as sugestões iniciais do MA e mecanismos de diversidade, nativos dos algoritmos em questão, favoreceram o maior espalhamento das soluções nas iterações iniciais. Este alto grau de espalhamento inicial gerou um comportamento descendente nas curvas dos resultados em questão; este fenômeno é explicado pela variedade de bases de dados de diferentes naturezas. Cada base de dados possui um conjunto de melhores valores de parâmetros específico, podendo estar agrupados das mais diversas formas possíveis. Deste modo, os mecanismos de diversidade podem influenciar na escolha das soluções que devem pertencer ao Pareto *front* resultante.

Na Figura 5.6, percebe-se a melhora do desempenho dos algoritmos HMOPSO e m-HDNPSO com relação à simulação como um todo. No cenário com 5 partículas, a ausência de mecanismos de diversidade dificultou a convergência. Com a adição de mais partículas, os dois algoritmos alcançaram melhores resultados que suas versões tradicionais em mais iterações, sendo superiores com 95% de confiança da primeira à sétima iteração. Com relação aos demais algoritmos híbridos, houve melhoras nas últimas iterações. Na simulação com 5 partículas havia um decrescimento abrupto, sendo estabilizado nas iterações finais. Na simulação com 10 partículas, o decrescimento aconteceu de forma suave, atingindo, nas iterações finais, resultados melhores.

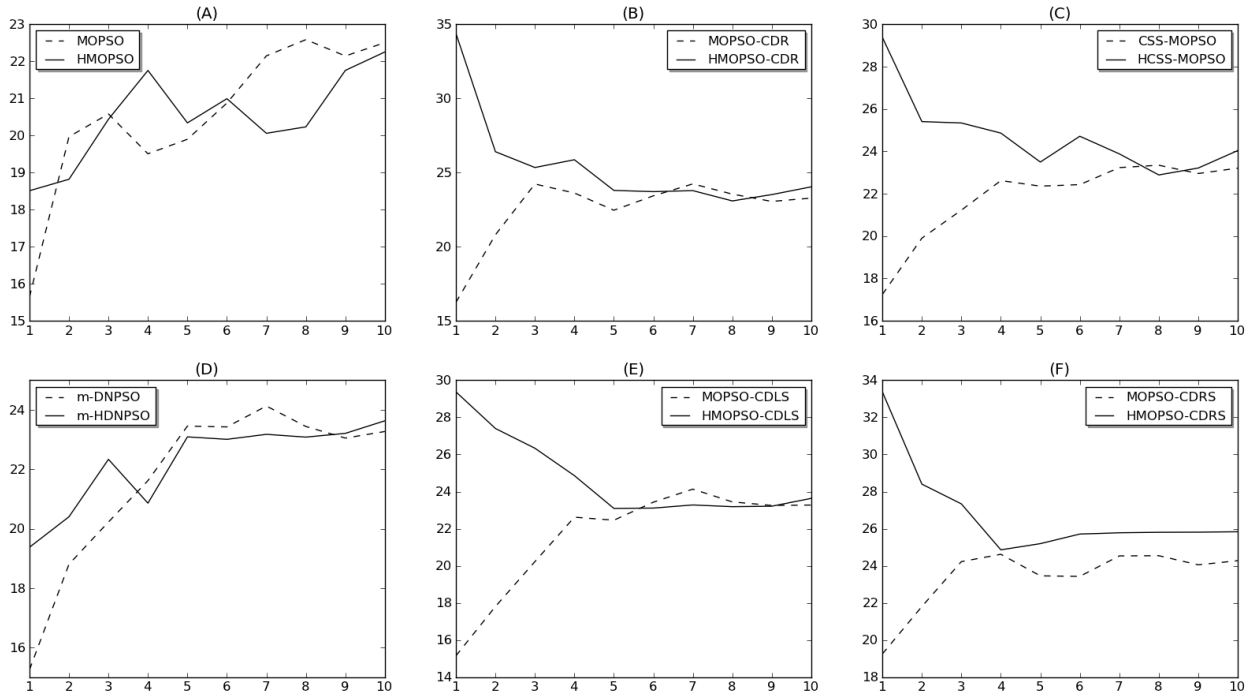

Figura 5.5 Média de *Max. Spread* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

As Figuras 5.7 e 5.8 apresentam os resultados com relação ao *Coverage*. Esta métrica é

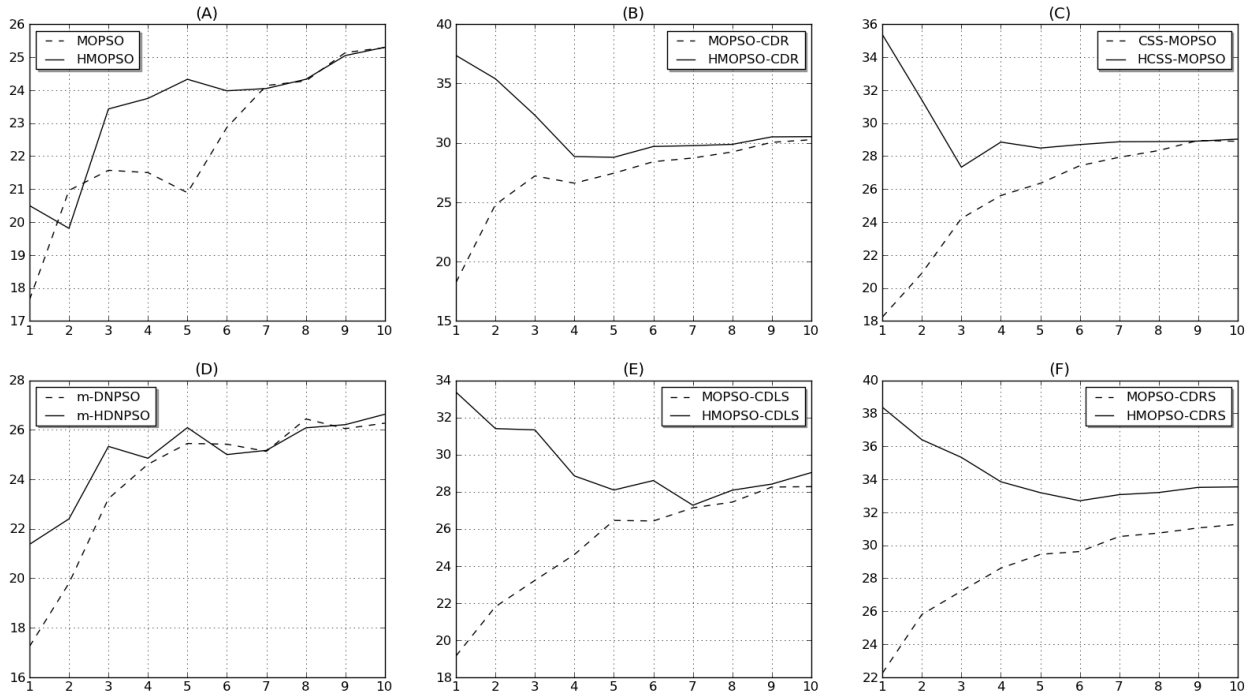

Figura 5.6 Média de *Max. Spread* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

muito importante, pois informa o percentual de dominância entre Pareto *fronts*. Os resultados apresentados mostram a média do percentual de dominância por iteração dos algoritmos.

Como pode-se ver na Figura 5.7, os algoritmos híbridos geraram Pareto *fronts* de maior qualidade, apresentando elevado percentual de dominância desde as primeiras iterações. Os algoritmos HMOPSO, HMOPSO-CDR e HMOPSO-CDRS alcançaram 100% de dominância, ao final da simulação, com relação às respectivas abordagens tradicionais.

Os resultados apresentados na Figura 5.8 seguem um padrão parecido com o que foi discutido anteriormente. A principal diferença é o aumento no percentual de dominância nas primeiras iterações. O HMOPSO, HMOPSO-CDR, m-HDNPSO apresentaram um aumento médio de $3,6\%$, o HCSS-MOPSO apresentou um aumento médio de $2,6\%$, o HMOPSO-CDLS apresentou um aumento médio de $2,3\%$ e o HMOPSO-CDRS apresentou um aumento médio de 4%. Todas as médias são referente às 5 primeiras iterações.

Os resultados apresentados no gráfico da média envolvem 40 problemas de classificação de natureza diversa, podendo impactar na amostragem final. Este formato impossibilita uma análise individual, por problema. Diante da limitação do gráfico da média, são apresentados os resultados na *Perspectiva 2*, que retrata o número de vitórias que o algoritmo alcançou levando em consideração todos os problemas, por iteração. Essa perspectiva torna a análise independente da natureza do problema.

As Figuras 5.9 e 5.10 apresentam os resultados com relação ao *Spacing*. Os resultados apresentados na Figura 5.9 deixam clara a superioridade dos algoritmos híbridos para a métrica em questão. Todos os algoritmos híbridos venceram em pelo menos 20 dos 40 problemas de clas-

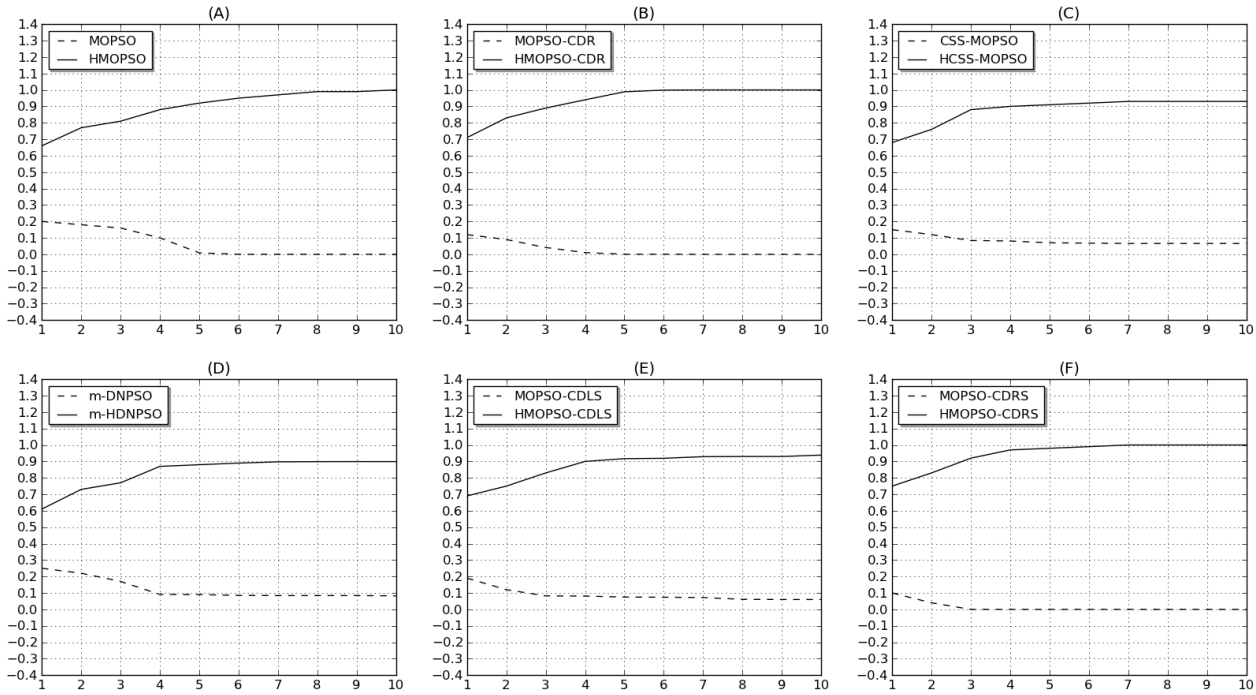

Figura 5.7 Média de *Coverage* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

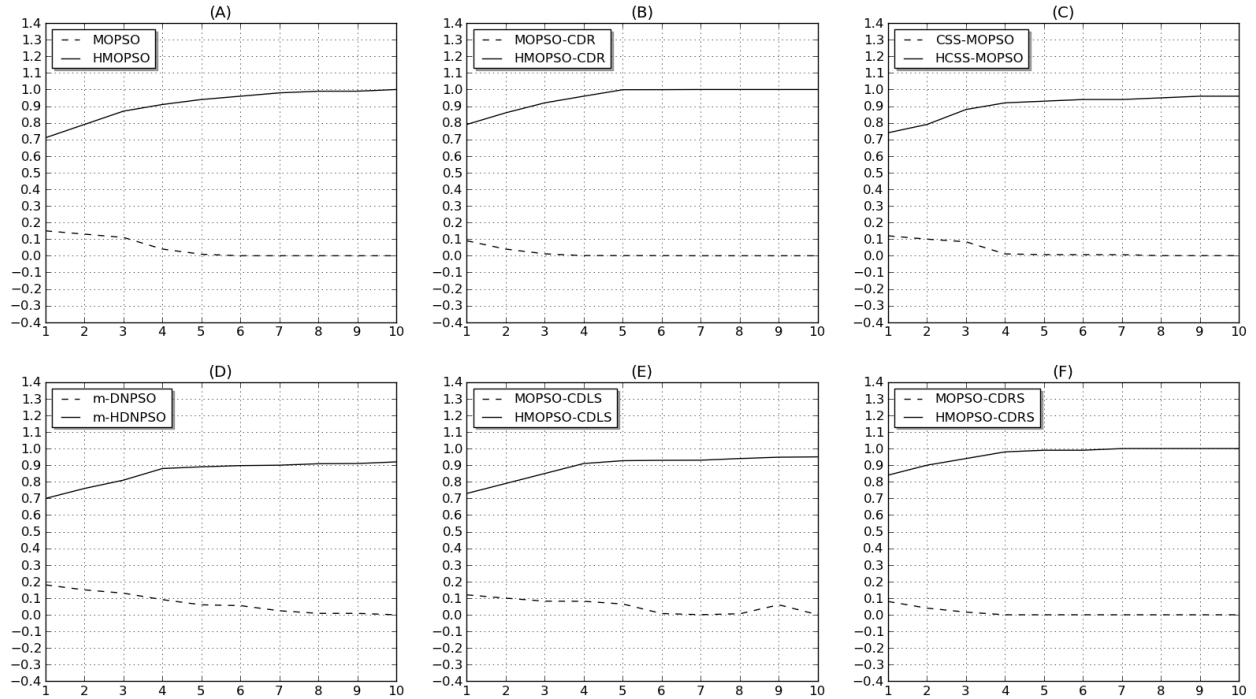

Figura 5.8 Média de *Coverage* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

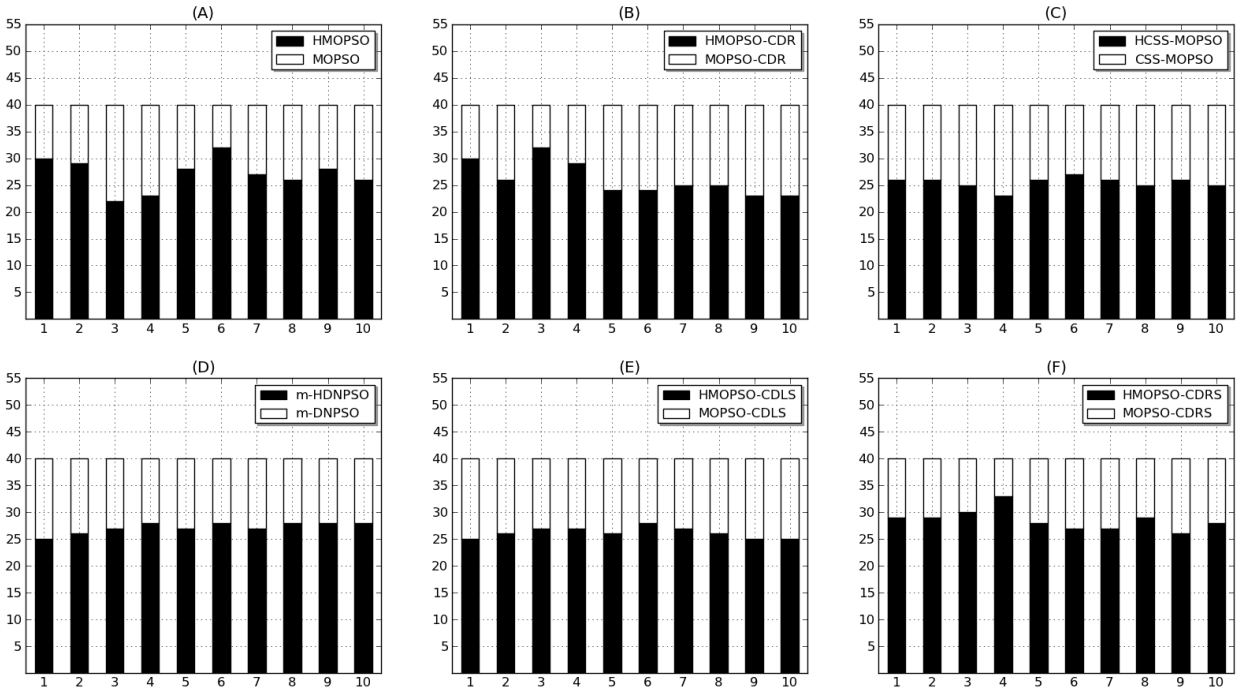

Figura 5.9 Número de vitórias em *Spacing* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

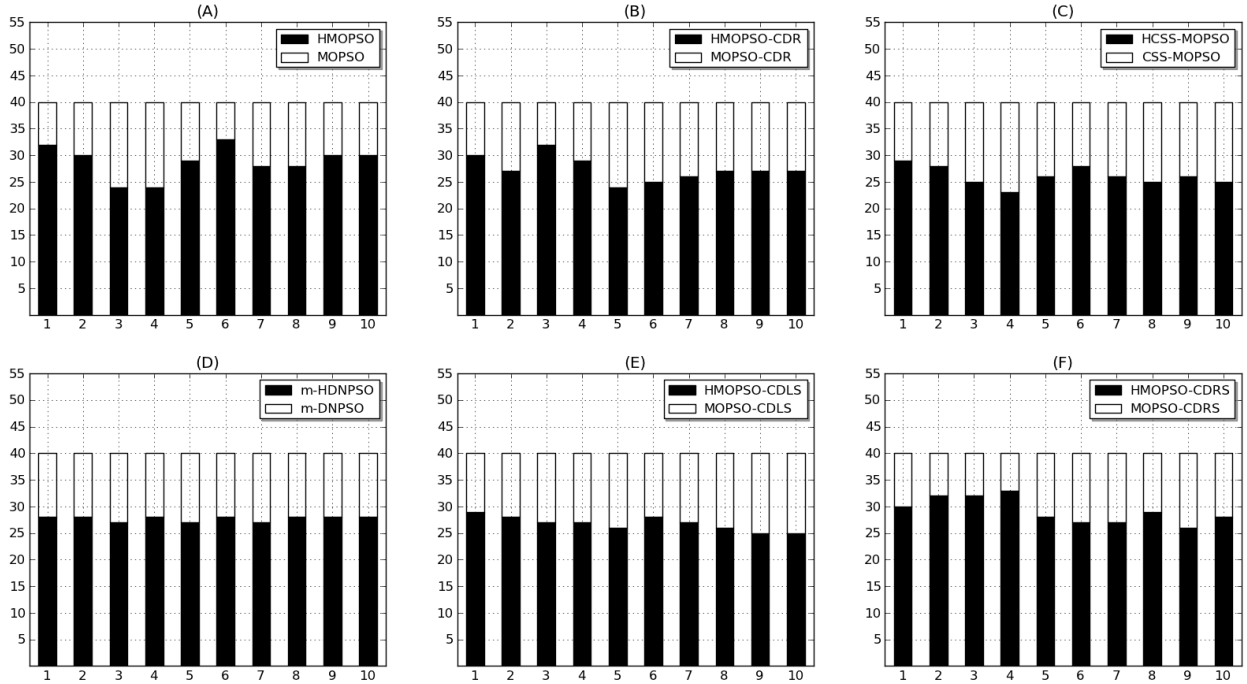

Figura 5.10 Número de vitórias em *Spacing* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

sificação em todas as iterações. O algoritmo HMOPSO-CDRS (ver Figura 5.9 F) apresentou o maior número de vitórias em comparação aos demais, chegando a vencer em até 33 problemas de classificação.

O aumento do número de partículas sugeridas pelo MA favoreceu os algoritmos híbridos, tornando-os vencedores em problemas não vencidos na simulação com 5 partículas (ver Figura 5.10). Levando em consideração toda a simulação, os algoritmos híbridos conseguiram vencer de 2 a 5 problemas, por iteração, não vencidos no cenário anterior.

As Figuras 5.11 e 5.12 apresentam os resultados com relação ao *hypervolume*. O número de vitórias dos algoritmos híbridos também é muito expressivo com relação ao *hypervolume* para 5 e 10 partículas. A média de vitórias no cenário com 5 partículas, envolvendo todos os algoritmos, é de $24,59$ problemas, sendo o HMOPSO-CDRS o algoritmo com maior média de vitórias: $26,9$ problemas. Levando em consideração todos os algoritmos híbridos, pelos menos 21 vitórias foram alcançadas, sendo 30 o número de vitórias mais expressivo.

O aumento do número de partículas torna os resultados ainda mais expressivos. Os algoritmos HMOPSO, HMOPSO-CDR, HCSS-MOPSO, m-HDNPSO, HMOPSO-CDLS e HMOPSO-CDRS venceram, em média, 27, $29,4$, $26,4$, $28,2$, $29,3$ e $30,5$ problemas de classificação, respectivamente. Estes resultados, aumentaram a média de vitórias, envolvendo todos os problemas, de $24,59$ para $28,51$ problemas.

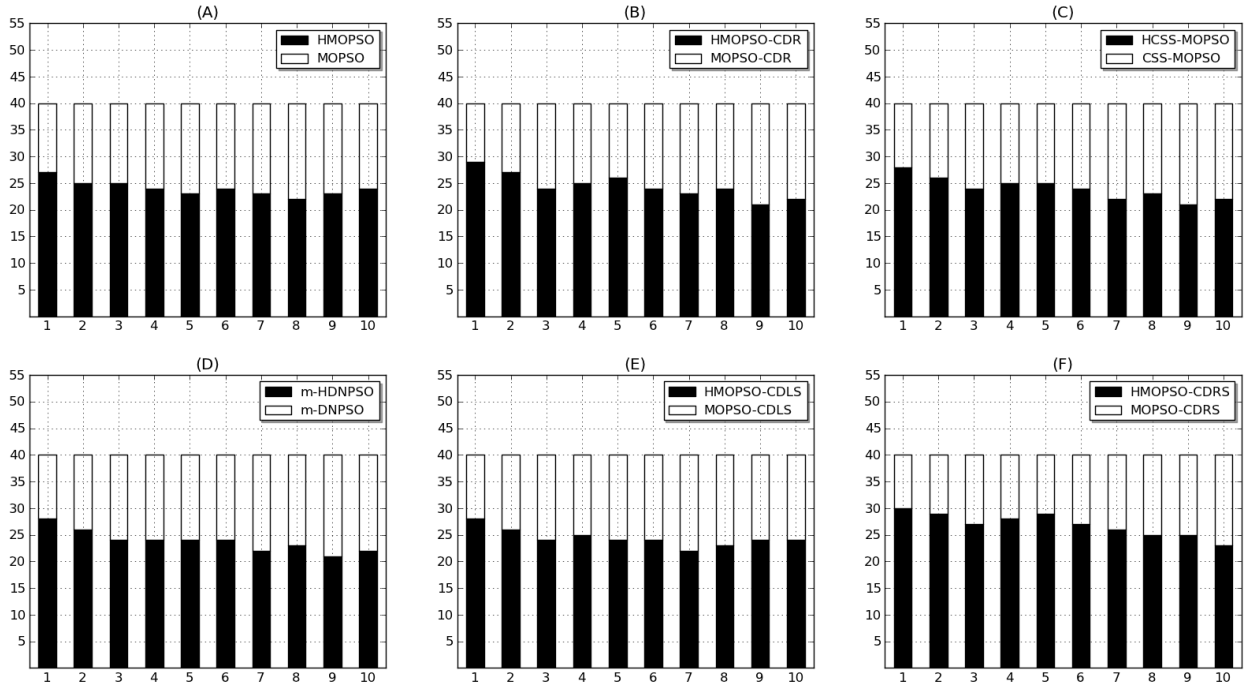

Figura 5.11 Número de vítorias em *hypervolume* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

As Figuras 5.13 e 5.14 apresentam os resultados com relação ao *max. spread*. No cenário de 5 partículas, inicialmente, vê-se maior número de vitórias por parte dos algoritmos híbridos, corroborando com os resultados apresentados na Figura 5.5. Os resultados obtidos pelos algo-

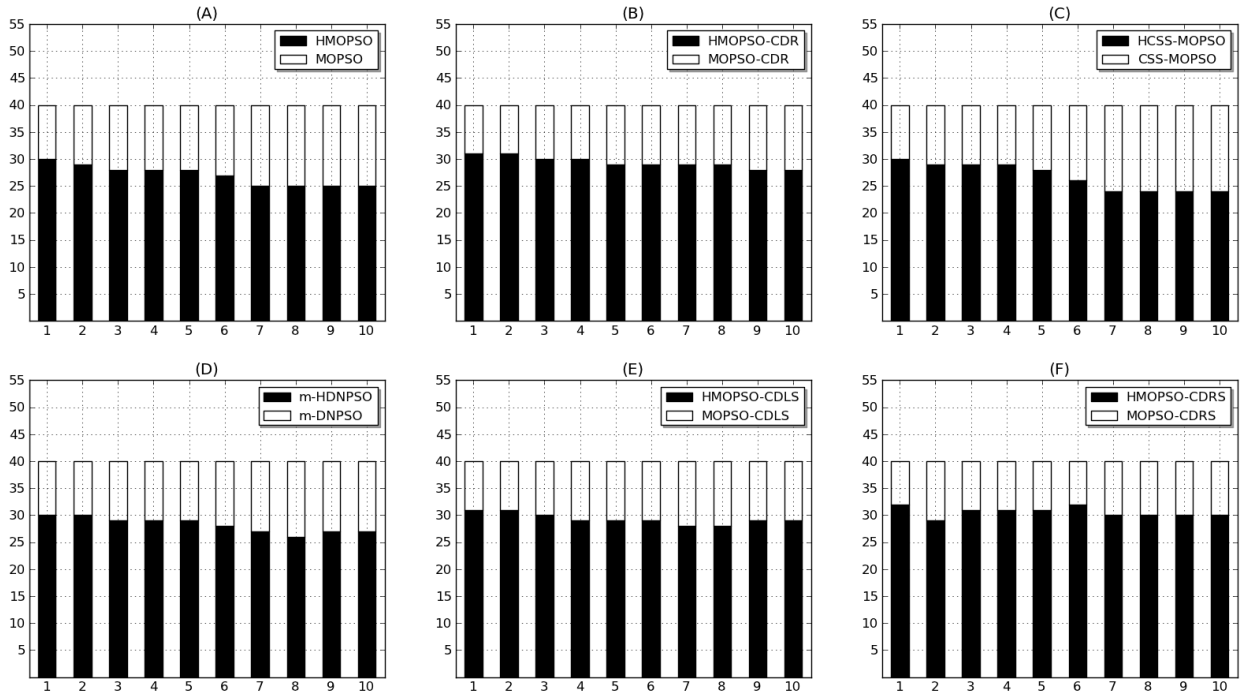

Figura 5.12 Número de vítorias em *hypervolume* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

ritmos HMOPSO e m-HDNPSO foram considerados iguais no gráfico de média. Já com a perspectiva de vitórias por problema, estes algoritmos apresentaram maior número de vitórias em 7 das 10 iterações. Os algoritmos híbridos restantes alcançaram resultados ainda melhores, não sendo derrotados em nenhuma iteração. O HMOPSO-CDR, HCSS-MOPSO, HMOPSO-CDLS e o HMOPSO-CDRS venceram respectivamente 67%, $62,5\%$, $61,5\%$ e $67,5\%$ dos problemas nas 5 primeiras iterações.

Os resultados apresentados na Figura 5.14 mostram que o aumento do número de partículas impactou no número de vitórias dos algoritmos híbridos. Os algoritmos HMOPSO e m-HDNPSO apresentaram melhor desempenho e venceram em todas as iterações. O HMOPSO-CDR, HCSS-MOPSO, HMOPSO-CDLS e o HMOPSO-CDRS venceram $70,5\%$, $71,75\%$, $67,5\%$ e $74,5\%$ dos problemas em todas as iterações, respectivamente.

As Figuras 5.15 e 5.16 apresentam os resultados com relação ao *coverage*. Na Figura 5.15 são apresentados os resultados da simulação com 5 partículas. Como pode-se ver, a sugestão de soluções favoreceu a geração de Pareto *fronts* com mais qualidade para a maioria dos problemas de classificação envolvidos. Dentre os algoritmos híbridos, 3 deles (ver Figura 5.15 A, B e F) alcançaram a dominância total em todos os problemas envolvidos, ao final das iterações. Além disso, o número de vitórias dentre todos os algoritmos híbridos é alta desde as primeiras iterações, proporcionando melhora e refinamento do Pareto *front* nas iterações seguintes.

Na Figura 5.16 são apresentados os resultados da simulação com 10 partículas. Como pode-se ver, o maior número de partículas melhorou os Pareto *fronts* gerados pelos algoritmos híbridos no início da simulação.

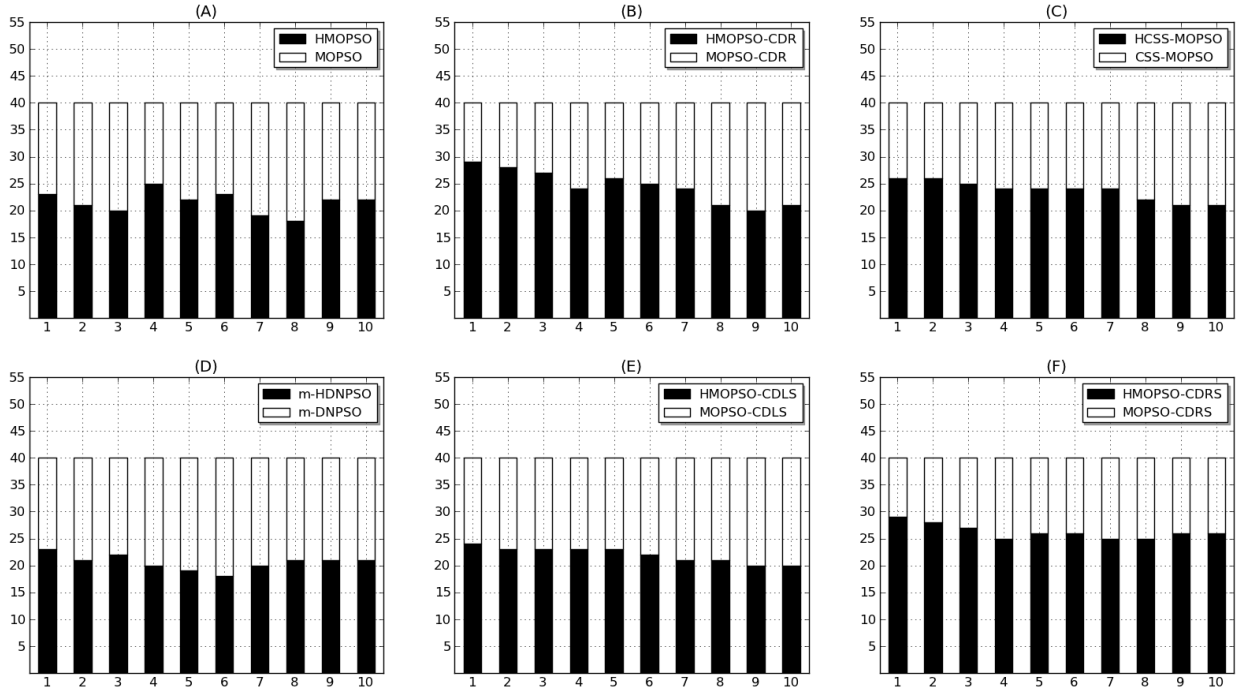

Figura 5.13 Número de vítorias em *Max. Spread* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

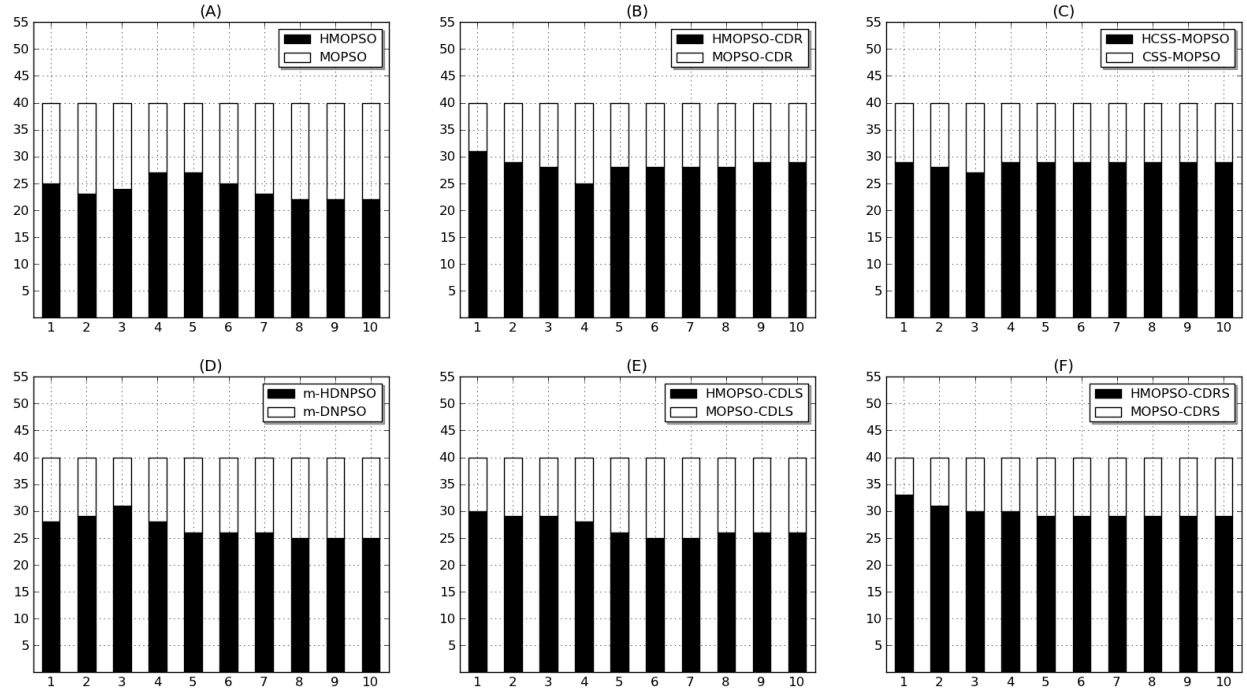

Figura 5.14 Número de vítorias em *Max. Spread* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

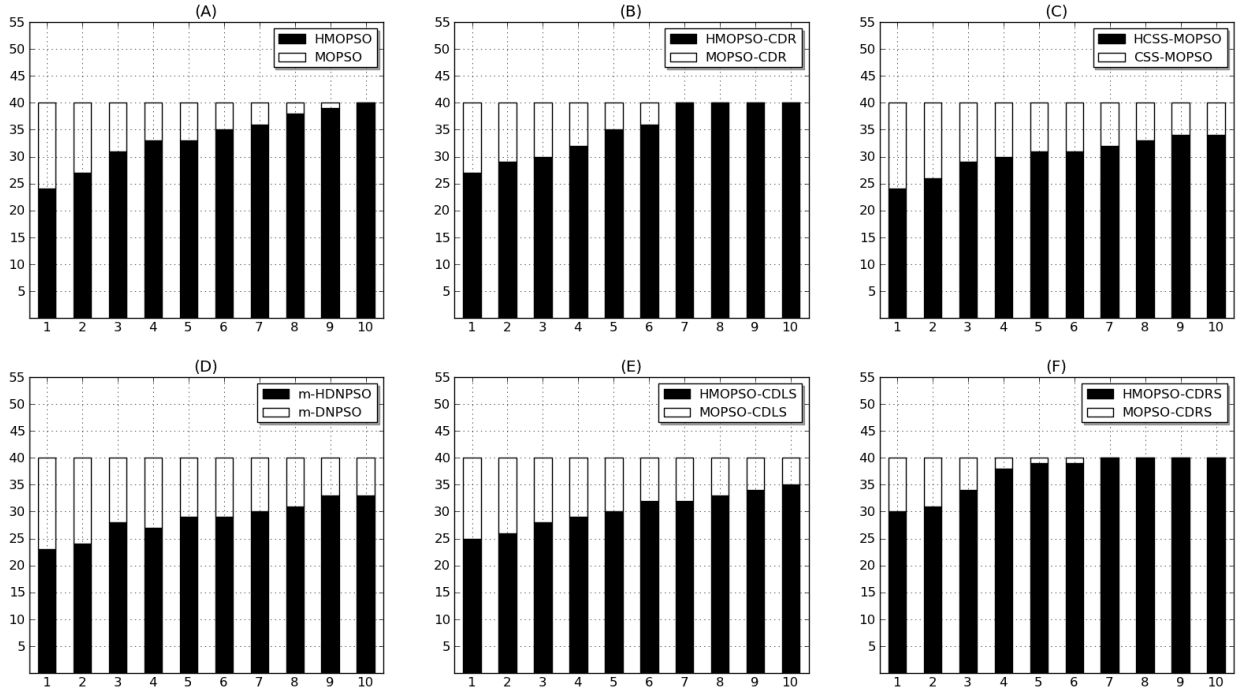

Figura 5.15 Número de vítorias em *Coverage* x Iteração, adotando-se 5 partículas e levando em consideração os 40 problemas de classificação.

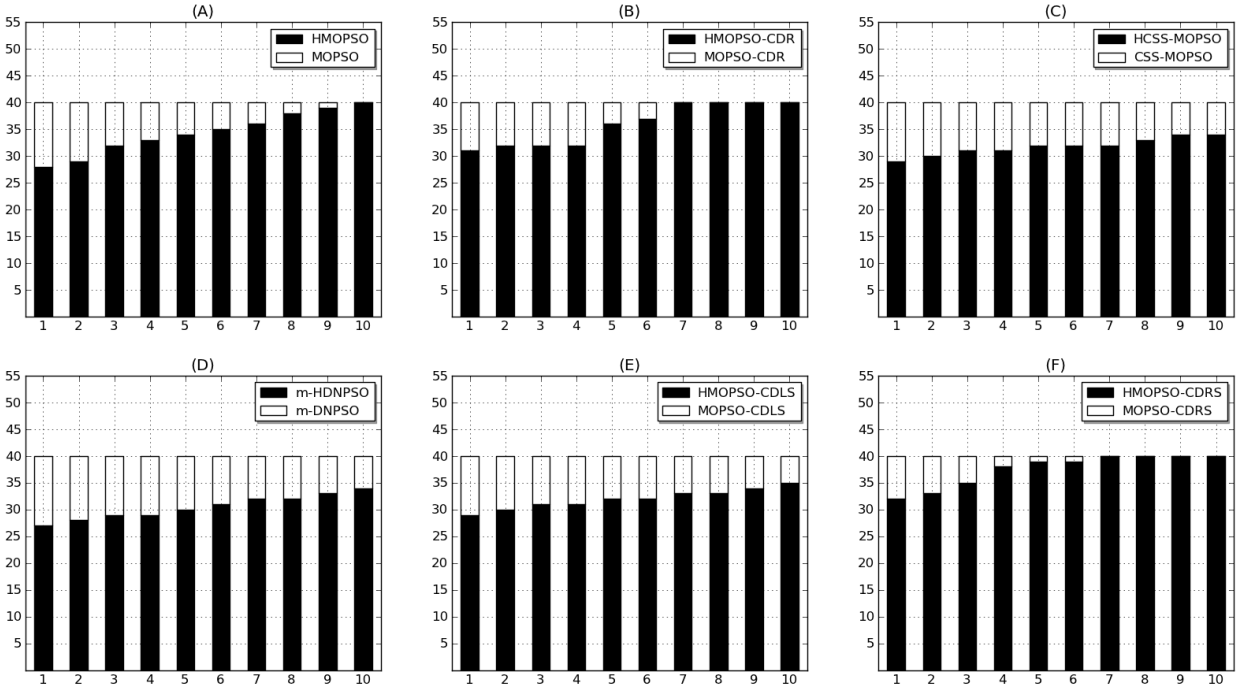

Figura 5.16 Número de vítorias em *Coverage* x Iteração, adotando-se 10 partículas e levando em consideração os 40 problemas de classificação.

Todos os resultados apresentados neste trabalho levam em consideração a qualidade do Pareto *front*. Em casos em que há a necessidade de escolha de uma das soluções dos Pareto *fronts* gerados, existem estratégias baseadas em *ranking* [78] que auxiliam na escolha de soluções.

5.3 Considerações Finais

Neste capítulo foi descrito o arranjo experimental adotado no trabalho: Configuração dos parâmetros, métricas de avaliação e a metodologia do experimento. Além disso, foram apresentados os resultados obtidos, frutos da comparação entre algoritmos utilizando meta-aprendizado e os algoritmos tradicionais. A comparação foi baseada na aplicação de 4 métricas de qualidade sobre os Pareto *fronts* gerados por cada algoritmo. De acordo com os resultados apresentados, ficou comprovada estatisticamente a potencialidade da arquitetura desenvolvida, ou seja, MA é uma alternativa eficaz para auxiliar no processo de otimização de configurações de parâmetros de SVM.

No capítulo seguinte apresentamos os desafios encontrados, nossas contribuições e trabalhos futuros.

CAPÍTULO 6

Conclusão

O trabalho desenvolvido foi motivado pelo problema apresentado pela SVM, que é a influência da escolha dos valores dos parâmetros no seu desempenho. Em capítulos anteriores, foram apresentadas diversas alternativas para este problema, entre elas os algoritmos híbridos que combinam meta-aprendizado com técnicas de otimização. Os algoritmos híbridos alcançaram melhorando os resultados, superando as demais alternativas. Mesmo que os resultados tenham sido expressivos, os trabalhos relacionados tratavam o problema de seleção de parâmetros de forma inadequada, considerando apenas um único objetivo. Esta limitação inspirou o trabalho desenvolvido, que é uma arquitetura híbrida para otimização multi-objetivo de SVMs.

Os resultados atingidos foram animadores e abriram caminho para futuras contribuições. Espera-se que com este trabalho novas pesquisas envolvendo abordagens híbridas possam ser realizadas. E que este trabalho seja fonte de inspiração e alvo de novas adaptações e melhorias.

A seguir apresentamos um resumo das principais contribuições do nosso trabalho durante o mestrado:

- Escolha de objetivos relevantes a serem considerados para a definição do modelo da SVM.
- Mecanismo para seleção de configurações de parâmetros levando em consideração múltiplos fatores (objetivos).
- Adaptação da arquitetura híbrida, desenvolvida por [60], para otimização multi-objetivo de SVMs.
- Implementação de um *framework* de recomendação multi-objetivo.

Durante a pesquisa, alguns artigos foram escritos, avaliados e publicados com o objetivo de validar o trabalho realizado durante a dissertação.

- Publicação em International Joint Conference on Artificial Neural Networks - IJCNN, 2012, Brisbane. (Multi-objective Optimization and Meta-learning for SVM Parameter Selection. Miranda, P. B. C. ; Prudêncio, R. B. C. ; Carvalho, A. C. P. L. F. ; Soares, C. Proceedings of the International Joint Conference on Artificial Neural Networks, 2012. p. 117-124).
- Publicação em International Conference on Computational Science and Its Applications - ICCSA, 2012, Salvador. (An Experimental Study of the Combination of Meta-Learning

with Particle Swarm Algorithms for SVM Parameter Selection. Miranda, P. B. C. ; Prudêncio, R. B. C. ; Carvalho, A. C. P. L. F. ; Soares, C.. Lecture Notes in Computer Science, 2012. v. 7335. p. 562-575).

- Publicação em IEEE Conference on Systems, Man, and Cybernetics - SMC, 2012, Seoul. (Combining a Multi-Objective Optimization Approach with Meta-Learning for SVM Parameter Selection. Miranda, P. B. C. ; Prudêncio, R. B. C. ; Carvalho, A. C. P. L. F. ; Soares, C.).
- Publicação em Brazilian Symposium on Neural Network - SBRN, 2012, Curitiba. (Combining Meta-Learning with Multi-Objective Particle Swarm Algorithms for SVM Parameter Selection: An Experimental Analysis. Miranda, P. B. C. ; Prudêncio, R. B. C. ; Carvalho, A. C. P. L. F. ; Soares, C.. Proceedings of the Brazilian Symposium on Neural Network, 2012.)
- Publicação em International Journal of Organizational and Collective Intelligence - IJOCI, 2013. (Miranda, P. B. C. ; Prudêncio, R. B. C. ; Carvalho, A. C. P. L. F. ; Soares, C.. Combining a Multi-Objective Optimization Approach with Meta-Learning for SVM Parameter Selection).

A publicação feita à conferência SMC 2012: *"Combining a Multi-Objective Optimization Approach with Meta-Learning for SVM Parameter Selection"* foi premiada como *Best Conference Paper Award*.

6.1 Limitações e trabalhos futuros

O trabalho desenvolvido obteve resultados expressivos, porém diversos pontos precisam ser melhorados e pesquisados. A seguir, são listados possíveis trabalhos futuros:

- Construção de uma meta-base com mais meta-exemplos e dados melhor pré-processados.
- Pesquisar outras métricas de avaliação multi-objetivo.
- Pesquisar mais meta-características para auxiliar no cálculo de similaridade.
- Investigar outros meios para calcular a similaridade entre meta-exemplos, como técnicas de agrupamento.
- Aplicar a arquitetura híbrida multi-objetivo para otimização de outros problemas de aprendizado, como regressão.
- Adaptar a arquitetura para problemas de aprendizado com dados categóricos.
- Pesquisar mais objetivos relevantes que possam favorecer a construção de um melhor modelo de SVM.
- Utilizar outros algoritmos de otimização.

Além dos itens listados, alguns pontos relacionados à parte arquitetural precisam ser estudados e aperfeiçoados, como o processo de seleção de configurações de parâmetros e o modo de interação entre o meta-aprendizado e os algoritmos de otimização.

Seleção de Configurações de Parâmetros O processo de seleção de configurações de parâmetros é baseado no desempenho das configurações presentes na meta-base. Quando o desempenho é expresso através de um único valor (i.e. taxa de acerto na classificação), com uma simples ordenação, as soluções já são definidas. Porém, em cenários onde o desempenho é medido através de múltiplos valores, esta tarefa se torna mais complexa e ainda não apresenta soluções validadas na literatura. Neste trabalho foi utilizada um mecanismo inspirado na dominância e *crowding distance*, provenientes da área de otimização.

Modo de interação com algoritmos meta-heurísticos O meta-aprendizado identifica o conjunto de meta-exemplos mais similar e sugere um conjunto de configurações de parâmetros como entrada para o algoritmo meta-heurístico. Este tipo de abordagem tendencia o processo a operar nas regiões sugeridas pelo meta-aprendizado. Em casos de problemas simples, onde as soluções ótimas estão próximas, não haveria dificuldades. Por outro lado, em problemas mais complexos, onde as soluções ótimas podem estar distantes umas das outras, o algoritmo desconsidera certas regiões, tendenciando a busca.

Para o estágio de **Seleção de Configurações de Parâmetros** visamos pesquisar/desenvolver mecanismos de seleção de soluções para cenários de múltiplos valores/objetivos. Para o estágio **Modo de interação com algoritmos meta-heurísticos**, acreditamos que estratégias dinâmicas de sugestão de soluções para algoritmos, onde as sugestões não devem necessariamente ser geradas como entrada (antes da execução) para o processo de otimização e busca, mas também durante o processo de busca, pode favorecer ainda mais na geração de Pareto *fronts* de maior qualidade.

6.2 Considerações finais

O projeto automático para seleção de parâmetros de SVM é um tema importante que ainda precisa de um estudo aprofundado. Nesta dissertação foi apresentada uma arquitetura híbrida, combinando-se MA com algoritmos de otimização, para cenários multi-objetivo.

APÊNDICE A

Valores das Meta-características

Tabela A.1 Valores das 8 meta-características para os 40 problemas de classificação.

Problema	MC_1	MC_2	MC_3	MC_4	MC_5	MC_6	MC_7	MC_8
Balance Scale	625	4	3	0.75	2.605	0.0	-1.3	-1328.312
Blood	748	4	2	2.777e-06	234.437	2.258	10.148	-2900.463
Breast-w	699	9	2	1.566e-22	2.241	1.476	1.669	-2624.004
Breast Tissue	106	9	6	0.340	1658.37	0.922	0.342	-20.264
Cancer	699	9	2	1.566e-22	2.235	1.478	1.671	-2624.004
Colic	368	27	2	0.219	25884.130	1.4513	17.488	-1214.710
Colon	62	2000	2	0.019	327.300	1.479	3.074	-124.492
Column-2c	310	6	2	1.736e-06	48.298	1.185	7.586	-987.188
Column-3c	310	6	3	1.736e-06	48.298	1.185	7.586	-535.096
Ecoli	336	7	8	0.029	0.274	3.589	51.187	-123.525
Glass	214	9	6	0.123	10.862	1.640	9.650	-93.073
Haberman	306	3	2	0.129	38.031	1.064	3.267	-975.300
Heart	187	22	2	0.207	0.0	0.686	-1.236	-572.310
Heart Statlog	270	13	2	0.039	44.998	0.424	0.090	-821.574
Hepatitis	155	19	2	0.255	7.279	0.553	1.304	-425.774
Hill Valley	606	100	2	0.0	327.103	2.970	9.444	-2186.036
Hypothyroid	3772	29	4	0.148	0.0	5.418	167.590	-8867.558
Ionosphere	351	34	2	0.157	0.014	0.007	-0.856	-1138.992
Iris	150	4	3	0.045	3.221	0.0667	-0.765	-192.0
Kr-vs-kp	3196	36	2	0.099	0.046	4.315	111.021	-15399.192
Letter	9877	16	26	0.0	0.0	1.596	1.697	-2399.754
Libras	360	90	15	0.182	0.458	-0.186	-0.715	0.0
Lung Cancer	32	56	3	0.577	1.572	0.164	1.244	-17.509
Lymph	148	18	4	0.179	0.300	0.748	2.355	-145.041
Mamography	961	5	2	0.015	12.078	4.303	135.936	-3797.256
Optdigits	1797	64	10	0.984	0.0	4.310	104.603	-728.299
Parkinson	197	22	2	0.343	0.0	1.525	11.976	-567.953
Pen Digits	3498	16	10	1.103e-07	0.0	0.665	-0.726	-1961.714
Primary Tumor	339	17	21	0.193	0.0	1.545	3.994	-8.0
Prina Diabetes	768	8	2	0.130	3.980	0.528	2.749	-2938.302
Red Wine Qual	1599	11	6	0.001	6.888	1.681	8.279	-1679.059
Segment	2310	19	7	0.233	0.0	3.144	36.250	-1827.459
Sick	3772	29	2	0.081	0.0	7.781	201.812	-20143.097
Sonar	208	60	2	0.084	0.224	0.960	2.193	-586.052
Vehicle	846	18	4	0.0	113.838	1.039	5.115	-1200.228
Vote	435	16	2	0.034	0.0	-0.103	-1.824	-1475.307
White Wine Qual	4898	11	10	0.0	0.0	0.594	0.869	-3373.888
Wine	177	13	3	0.071	64.280	0.346	-0.026	-248.752
Yeast	1484	8	10	0.105	0.245	2.909	28.560	-685.719
Zoo	101	16	7	0.074	0.0	0.394	-0.118	-13.609

Referências Bibliográficas

[1] A. H. Simon. Razão nos assuntos humanos. Stanford: Stanford University Press, 1983.

[2] Bernhard E. Boser, Isabelle M. Guyon e Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In Haussler, David (editor). In 5th Annual ACM Workshop on COLT, pages 144152, Pittsburgh, PA, 1992. ACM Press.

[3] S. Haykin. Redes Neurais: princípios e prática. Trad. Paulo Martins Engel. 2. ed. Porto Alegre: Bookman, p. 349-381, 2001.

[4] A. C. Lorena e A. C. P. L. F Carvalho. Uma introdução às Máquinas de Vetor de Suporte. Revista de Informática Teórica e Aplicada, vol. 14, 2007.

[5] J.-Y Wang. Application of support vector machines in bioinformatics. Dissertação (Mestrado). National Taiwan University - Computer Science and Information Engineering. 2002.

[6] H. Qiao, X. Zhang, B. Zhang e J. Keane. Face recognition using SVM decomposition methods, Intelligent Robots and Systems. In Proceedings 2004 IEEE/RSJ International Conference on , vol .2, p. 2015-2020, 2004.

[7] D. Cui e D. Curry. Prediction in Marketing Using the Support Vector Machine. Marketing Science Fall, vol. 24, p. 595-615, 2005.

[8] C. Soares, P. Brazdil e P. Kuba. A Meta-Learning Method to Select the Kernel Width in Support Vector Regression. Machine Learning, vol. 54(3), p. 195-209, 2004.

[9] N. Cristianini e J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2000.

[10] B. C. Lovell e C. J. Walder. Business Applications and Computational Intelligence. Idea Group Publishing, 2006.

[11] T. Jaakkola e D. Haussler. Probabilistic kernel regression models. In Proceedings of the Seventh Workshop on AI and Statistics. San Francisco: Morgan Kaufman, 1999.

[12] V. Vapnik. Statistical learning theory. Chichester, UK: Wiley, 1998-a.

[13] V. Vapnik. Statistical learning theory. New York: John Wiley e Sons, 1998-b.

[14] T. Joachims. Estimating the generalization performance of a SVM efficiently (LS VIII-Report 25). Universitat Dortmund, Germany, 1999.

[15] V. Vapnik. The nature of statistical learning theory. Berlin: Springer, 1995.

[16] V. Vapnik e O. Chapelle. Bounds on Error Expectation for Support Vector Machines. Neural Comput., vol. 12, p. 2013-2036, 2000.

[17] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In International joint Conference on artificial intelligence, vol. 14, p. 1137-1145, 1995.

[18] N. Cristianini, C. Campbell, e J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. *NIPS*, p. 204-210, 1998.

[19] S. Ali e K. Smith-Miles. On optimal degree selection for polynomial kernel with support vector machines: Theoretical and empirical investigations. *KES Journal*, vol. 1(1), p. 1-18, 2007.

[20] G. Narzisi. An Experimental Multi-Objective Study of the SVM Model Selection problem. Disponível em: http://cims.nyu.edu/ gn387/multSVM.pdf, Acesso em: 20/04/2011.

[21] P. Brazdil e R. Henery. Analysis of results. In Machine Learning, Neural and Statistical Classification, D. Michie et al., eds. Ellis Horwood, New York. Chapter 10, 1994.

[22] S. Lessmann, R. Stahlbock e S. Crone. Genetic algorithms for support vector machine model selection. *International Joint Conference on Neural Networks*, p. 3063–3069, 2006.

[23] F. Friedrichs e C. Igel. Evolutionary tuning of multiple svm parameters. In Neurocomputing, vol. 64, p. 107-117, 2005.

[24] A. Lorena e A. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. In Neurocomputing, vol. 71(16-18), p. 3326-3334, 2008.

[25] S.S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. In IEEE Transactions on Neural Networks, vol. 13, p. 1225-1229, 2002.

[26] C.-W Hsu, C.-C Chang e C.-J Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[27] P. Kuba, P. Brazdil, C. Soares e A. Woznica. Exploiting sampling and meta-learning for parameter setting of support vector machines. In F. Garijo, J. Riquelme, e M. Toro (Eds.), Proceedings of the Workshop de Mineração de Dados e Aprendizagem do IBERAMIA 2002, p. 217-225, 2002.

[28] K. Deb, A. Pratap, S. Agarwal e T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. In IEEE Transactions on Evolutionary Computation, vol. 6, p. 182-197, 2000.

[29] C. Coello Coello, A. Carlos e M. Lechuga. MOPSO: a proposal for multi-objective particle swarm optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, p. 1051-1056, 2002.

[30] N. Ayat, M. Cheriet e C. Suen. Automatic model selection for the optimization of SVM kernels. Pattern Recognition, vol. 38(10), p. 1733-1745, 2005.

[31] B. de Souza, A. de Carvalho, and R. Ishii. Multiclass svm model selection using particle swarm optimization. In Sixth International Conference on Hybrid Intelligent Systems, p.31, 2006.

[32] J. Kennedy e R. C. Eberhart. Particle swarm optimization. In Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 1, p. 1942-1948, 1995.

[33] G. Cawley. Model selection for support vector machines via adaptive step-size tabu search. In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, vol. 28(8), p. 434-437, 2001.

[34] B. Scholkopf, C. J. C. Burges e A. J. Smola. Advances in Kernel Methods: Support Vector Learning. Cambridge: MIT Press, 1999.

[35] A. R. G. Lima. Máquinas de Vetores Suporte na Classicação de Impressões Digitais. Dissertação (Mestrado) | Universidade Federal do Ceará, Fortaleza, 2002.

[36] B. L. Milenova, J. S. Yarmus e M. M. Campos. SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines. In Proceedings of the 31ST VLDB Conference, Trondheim, Norway, 2005.

[37] M. M. Millonas. Swarms, phase transitions, and collective intelligence. In C. G. Langton, Ed., Artificial Life III. Addison Wesley, Reading, MA. 1994.

[38] P. Pomeroy. An Introduction to Particle Swarm Optimization. Disponível em: http://www.adaptiveview.com/articles/ipsoprnt.html. Acesso em: 20 de abril de 2011.

[39] N. Nedja, L. dos Santos Coelho e L. de Macedo Mourelle. Multi-Objective Swarm Intelligent Systems. Studies in Computational Intelligence. In Springer, vol. 261, 2009.

[40] Chiu, S. Y. Cross-Searching Strategy for Multi-Objective Particle Swarm Optimization. In IEEE Congress on Evolutionary Computation, p. 3135-3141, 2007.

[41] C. Tsou, S. Chang e P. Lai. Using Crowding Distance to Improve Multi-Objective PSO with Local Search. In Swarm Intelligence, Focus on Ant and Particle Swarm Optimization - InTech, p. 77-86, 2007.

[42] C. R. Raquel e P. C. Naval Jr.. An Effective Use of Crowding Distance in Multi-Objective Particle Swarm Optimization. In Proceedings of the Conference on Genetic and Evolutionary Computation, New York, USA, p. 257-264, 2005.

[43] X. Hu, R. Eberhart, e Y. Shi. Particle Swarm With Extended Memory For Multi-Objective Opitmization. In Swarm Intellingence Symposium, p. 193-197, 2003.

[44] C. A. C. Coello, G. B. Lamont e D. A. Van Veldhuizen. Evolutionary Algorithms for Solving Multi-Objective Problems. In Genetic and Evolutionary Computation, New York, 2007.

[45] C. J. A. Bastos-Filho e P. B. C. Miranda. Multi-Objective Particle Swarm Optimization using speciation. In IEEE Symposium on Swarm Intelligence (SIS), p. 1-6, 2011.

[46] Y. Shi e R. C. Eberhart. Parameter selection in particle swarm optimization. In Proceedings of Seventh Annual Conference on Evolutionary Programming, New York, p. 591-600, 1998.

[47] D. F. Reyes e S. A. Rojas. Adapting multiple kernel parameters for support vector machines using genetic algorithms. In IEEE Congress on Evolutionary Computation, vol. 1, p. 626-631, 2005.

[48] L. Rendell e H. Cho. Empirical learning as a function of concept character. In Machine Learning, vol. 5, p. 267-298, 1990.

[49] D. Aha. Generalizing from case studies: A case study. In Proceedings of the 9th International Conference on Machine Learning, p. 110, 1992.

[50] T. Glasmachers e C. Igel. Gradient-based adaptation of general Gaussian kernels. In Neural Computing, vol. 17(10), p. 2099-2105, 2005.

[51] C. Igel. Multi-objective Model Selection for Support Vector Machines. In Proceedings of the 3rd Int. Conf. on Evolutionary Muilt Criterion Optimzation, p. 534-546, 2005.

[52] C. Igel e T. Suttorp Multi-objective optimization of support vector machines. In Springer-Verlag, Multi-objective Machine Learning Studies in Computational Intelligence, vol. 16, p. 199-220, 2006.

[53] Y. Jin e B. Sendhoff. Pareto-Based Multi-Objective Machine Learning: An Overview and Case Studies. In IEEE Transactions on Systems, Man and Cybernetics: Part C, vol. 38(3), p. 397-415, 2008.

[54] S. Ali e K. A. Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, vol. 70(1-3), p. 173–186, 2006.

[55] M. Reif, F. Shafait e Andreas Dengel. Meta-learning for evolutionary parameter optimization of classifiers. In Machine Learning, vol. 87(3), p. 357-380, 2012.

[56] R. A. Santana, M. R. Pontes e C. J. A. Bastos-Filho. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, p. 237-242, 2009.

[57] P. Brazdil, C. Giraud-Carrier, C. Soares e R. Vilalta. Metalearning: Applications to Data Mining. In Cognitive Technologies, Springer, 2009.

[58] R.B.C. Prudêncio e T.B. Ludermir. Meta-learning approaches to selecting time series models. In Neurocomputing, vol. 61, p. 121-137, 2004.

[59] S. Ali e K. A. Smith. Matching svm kernel's suitability to data characteristics using tree by fuzzy c-means clustering. In Design and application of hybrid intelligent systems, Ajith Abraham, Mario Köppen, and Katrin Franke (Eds.). IOS Press, Amsterdam, The Netherlands, The Netherlands, p. 553-562, 2003.

[60] T. A. F. Gomes, R. B. C. Prudêncio, C. Soares, A. L. D. Rossi e A. C. P. L. F. Carvalho. Combining meta-learning and search techniques to select parameters for support vector machines. In Neurocomputing, vol. 75(1), p. 3-13, 2012.

[61] S. L. Louis e J. McDonnel. Learning with Case-Injected Genetic Algorithms. In IEEE Transactions on Evolutionary Computation, vol. 8(4), p. 316-328, 2004.

[62] K. Deb e D. Kalyanmoy. Multi-Objective Optimization Using Evolutionary Algorithms. John Willey and Sons, First Ed., 2001.

[63] S. Louis e J. McDonnell. Learning with case-injected genetic algorithms. IEEE Transactions on Evolutionary Computation, vol. 8(4), p. 316-328, 2004.

[64] S. S. Keerthi e C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. In Journal Neural Computation, vol. 15 (7), p. 1667-1689, 2003.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay. Scikit-Learn: A Toolkit of Machine Learning in Python. In Journal of Machine Learning Research, vol. 12, p. 2825-2830, 2011.

[66] WEKA, The University of Waikato, Disponível em: http://www.cs.waikato.ac.nz/ml/weka/. Acesso em: 20 de abril de 2011.

[67] A. Frank e A. Asuncion. UCI Machine Learning Repository, Disponível em: http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

[68] Kate A. Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. In Journal ACM Computing Surveys vol. 41(1), n. 6, 2005.

[69] E. Zitzler e L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In Conference on Parallel Problem Solving from Nature Algorithms, vol. 5, p. 292-301, 1998.

[70] J. R. Schott. Fault tolerant design using single and multi-criteria genetic algorithms. Dissertação de Mestrado. Department of Aeronautics and Astrounatics, Massachusettes Institute of Technology, Bosto, MA, 1995.

[71] E. Zitzler, K. Deb e L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. In Evolutionary Computation Journal, vol. 8(2), p. 125-148, 2000.

[72] M. Takaki, D. Cavalcanti, R. Gheyi, J. Iyoda, M. d'Amorim e R. B. Prudêncio. A comparative study of randomized constraint solvers for random-symbolic testing. In Proceedings of the Nasa Formal Methods Symposium, p. 56-65, 2008.

[73] Q. Zhang, G. Shan, X. Duan e Z. Zhang. Parameters Optimization of Support Vector Machine Based on Simulated Annealing and Genetic Algorithm. In Proceedings of the IEEE International Conference on Robotics and Biomemetics, p. 1203-1206, 2009.

[74] MN. Kapp, R. Sabourin e P. Maupin. A PSO-Based Framework for Dynamic SVM Model Selection. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, p. 1227-1234, 2009.

[75] Y. Zhang. Evolutionary Computation Based Automatic SVM Model Selection. In Proceedings of the Fourth International Conference on Natural Computation, vol. 2, p. 66-70, 2008.

[76] I. Mierswa. Controlling overfitting with multi-objective support vector machines. In Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, New York, NY, USA, p. 1830-1837, 2007.

[77] K. Fukunaga e D.M. Hummels. Leave-one-out procedures for nonparametric error estimates. In IEEE Trans. Pattern Anal Machine Intell, vol. 11(4), p. 421-423, 1989.

[78] C. Dwork, R. Kumar, M. Naor e D. Sivakumar. Rank aggregation methods for the web. In Proceedings of the 10th International World Wide Web Conferences, pp. 613-622, 2011.