UNIVERSIDADE FEDERAL DO PARANÁ

DIOGO SCHWERZ DE LUCENA

COMPARAÇÃO DE META-HEURÍSTICAS NA OTIMIZAÇÃO DE SVM-MULTICLASSES APLICADAS À IDENTIFICAÇÃO DE TAREFAS MOTORAS

CURITIBA
2014

DIOGO SCHWERZ DE LUCENA

COMPARAÇÃO DE META-HEURÍSTICAS NA OTIMIZAÇÃO DE SVM-MULTICLASSES APLICADAS À IDENTIFICAÇÃO DE TAREFAS MOTORAS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Área de Concentração: Sistemas Eletrônicos, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Leandro dos Santos Coelho

CURITIBA
2014

AGRADECIMENTOS

Ao professor Leandro pelos ensinamentos e orientações – neste trabalho e na vida. Por sua confiança, amizade e suporte através dos anos. Pela sua dedicação e amor à pesquisa, que me serviram de inspiração para muitas decisões em minha vida.

Aos professores Eduardo Parente, Maria Teresinha Arns, Giselle Ferrari, Roberto Zanetti e Gideon Villar Leandro pelas palavras de estímulo e pelas valiosas contribuições dadas.

Ao meu amigo José Francisco pela disposição na revisão do trabalho e por todas as melhorias propostas.

À minha família de coração, Rejane, Antônio, Leopoldo e Mariana, por suas palavras de incentivo e abraços de ternura nos momentos difíceis.

À minha noiva Mariana, pela sua compreensão sempre - pelo consolo, afeto, carinho, companheirismo e amor em todas as horas. Também à Márcia, Eduardo, Henrique e Neida, que estiveram sempre presentes nos momentos que precisei.

Aos meus pais, Vanda e Kiko, e meus irmãos, Marcelo e Gustavo, que desde o princípio me apoiam e, mesmo de longe, me impulsionam a sempre ir além.

RESUMO

As interfaces cérebro-computador são sistemas que possibilitam ao usuário controlar componentes externos através de sua atividade cerebral. As aplicações desse tipo de sistema mostraram a capacidade de controlar próteses, cadeiras de rodas e *software* voltados para a comunicação, o que pode significar a melhoria na qualidade de vida e independência de pessoas desabilitadas. O uso de sinais de eletroencefalografia viabiliza a utilização de interfaces cérebro-computador sem intervenção cirúrgica ou riscos ao paciente. Estas interfaces são compostas por várias etapas, sendo uma das mais importantes o algoritmo de classificação de dados. Um dos algoritmos de classificação de dados mais difundidos na literatura é a máquina de vetores de suporte que, apesar de possuir caraterísticas positivas em sua aplicação, é sensível aos parâmetros de controle. Este trabalho busca avaliar a eficiência da utilização de meta-heurísticas na seleção destes parâmetros no contexto da aplicação da máquina de vetores de suporte à interface cérebro-computador. A máquina de vetores de suporte foi originalmente apresentada para utilização em problemas de duas classes, neste trabalho, entretanto, ela é aplicada em problemas de quatro classes, sendo necessária a expansão do algoritmo para esta tarefa. Para tanto, são utilizados os dois métodos mais conhecidos na literatura, denominado um-contra-todos e todos-contra-todos. Além destes um terceiro método é proposto (chamado de agrupamento por similaridade). No contexto de otimização dos parâmetros de controle, são utilizadas versões do algoritmo de evolução diferencial, otimização por enxame de partículas, colônia artificial de abelhas e busca por retropropagação. Também é proposto um algoritmo adaptativo híbrido entre o algoritmo de evolução diferencial e a busca por retropropagação. As técnicas são utilizadas – e posteriormente comparadas – em um sistema de interface cérebro-computador simples, com métodos de pré-processamento e extração de caraterísticas conhecidos na literatura, aplicadas a uma base de dados disponibilizada para a competição de interfaces cérebro-computador de Berlim. Os resultados mostram que a escolha do algoritmo de otimização da máquina de vetores de suporte tem grande influência na eficiência da interface cérebro-computador, fazendo dessa forma, a escolha deste algoritmo um passo importante no desenvolvimento desta tecnologia.

Palavras-chave: Meta-heurísticas. Otimização. Máquinas de vetores de suporte. Interface cérebro-computador.

ABSTRACT

Brain-computer interfaces are systems that allow users to control external components through brain activity. The applications of these systems showed the possibility to control prostheses, wheelchairs and software for communication, which means improvement in quality of life and independence of disabled people. The use of electroencephalography signals enables the use of brain-computer interfaces without surgery or any potential risk to the patient. These interfaces are composed of several steps and one of the most important is the classification algorithm. One of the most widely used data classification algorithms in the literature is the support vector machine which, despite possessing positive features in its application, is sensitive to control parameters. This work aims to evaluate the efficiency of using meta-heuristic strategies in the selection and tuning of these parameters, when support vector machine is applied to brain-computer interface. Support vector machine was originally presented to be used in two-class problems. In this work, the machine will be applied to four-class problems; an expansion of the algorithm for this task is needed. In this direction, two of the most popular methods in the literature were used, called one-versus-all and all-versus-all; furthermore, a third method is proposed (called grouping by similarity). In the context of control parameters optimization, different versions of differential evolution algorithm were applied as well as particle swarm optimization, artificial bee colony and backtracking search. A hybrid adaptive algorithm is proposing in this work, using differential evolution algorithm and backtracking search. The techniques are applied and compared in a simplistic brain-computer interface with well know methods for pre-processing and features extraction, which is applied to an electroencephalograph database available online for the Berlin Competition in brain-computer interfaces. The results show that the selection of the optimization algorithm for support vector machine has an important influence on the efficiency of brain-computer interfaces, thus making the selection of the right algorithm is an important step in developing this technology.

Key-words: Meta-heuristics. Optimization. Support Vector Machine. Brain-computer interface.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

ABC	-	Colônia Artificial de Abelhas
BCI	-	Interface Cérebro-Computador
BMI	-	Interface Cérebro-Máquina
BSA	-	Algoritmo de Busca por Retropropagação
CLPSO	-	Otimização por Enxame de Partículas com Aprendizagem Abrangente
CODE	-	Evolução Diferencial Composta
CSP	-	Padrão Espacial Comum
DE	-	Evolução Diferencial
EEG	-	Eletroencefalografia
EOG	-	Eletro-oculografia
ERD	-	Dessincronização Relacionada a Evento
ERS	-	Sincronização Relacionada a Evento
HEDRA	-	Algoritmo Híbrido de Evolução Diferencial e Retropropagação Adaptativa
JADE	-	Evolução Diferencial Adaptativa
PSO	-	Otimização por Enxame de Partículas
RBF	-	Função de Base Radial
SV	-	Vetor de Suporte
SVM	-	Máquina de Vetores de Suporte
TP	-	Tamanho da População

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

O cérebro humano é um órgão complexo, que nos permite pensar, mover, sentir, ver, ouvir, cheirar, entre outros, controlando o corpo e recebendo, analisando e guardando informações externas (memória). O cérebro produz sinais elétricos e reações químicas, que juntos são chamados de atividade cerebral, que fazem a comunicação das diferentes partes do corpo (SALADIN, 2001).

Doenças e danos neurológicos frequentemente resultam em perda permanente de funções motoras. Em muitos casos, a incapacidade é acentuada de modo que indivíduos não são aptos a se alimentarem e a se comunicarem (ANDERSON, 2004). Apesar de intervenções cirúrgicas e farmacológicas atualmente terem a capacidade de restaurar nervos e promover recuperação do sistema nervoso periférico, a maioria das deficiências no sistema nervoso central ainda não possuem tratamentos eficientes.

Buscando aumentar a qualidade de vida destes pacientes, novas tecnologias têm sido estudadas, entre elas está a interface cérebro-máquina (BMI – do inglês *Brain-Machine Interface*), que também é conhecida como prótese neural (as duas nomenclaturas serão usadas intermutavelmente neste trabalho). BMIs podem ser definidos como qualquer sistema que monitora a atividade cerebral e é capaz de transformar intenções de um indivíduo em comandos para um aparelho externo. Isso implica que, mesmo se uma atividade mental resultar em sinais sendo enviados para músculos, coluna vertebral, nervos periféricos e/ou o sistema nervoso autônomo e suas saídas, um BMI não vai usar nenhum desses sistemas, ele definirá as tarefas a serem executadas baseadas exclusivamente nos sinais oriundos do cérebro (GILJA *et al.*, 2011).

Duas classes principais de BMIs são definidas por Lebedev e Nicolelis (2006): não invasivos e invasivos. Os sistemas invasivos obtém o sinal neural diretamente do cérebro, fazendo uso de microeletrodos inseridos no córtex cerebral. Em alguns casos, apenas um eletrodo é utilizado e, sendo posicionado perto de um neurônio específico, é capaz de ler a atividade cerebral daquela célula. Porém, a tecnologia mais atual e

utilizada é o sensor matriz chamado *Utah*, que possui 100 hastes de silicone estreitas com 1,5 milímetro de comprimento, conforme mostrado na (FIGURA 1).

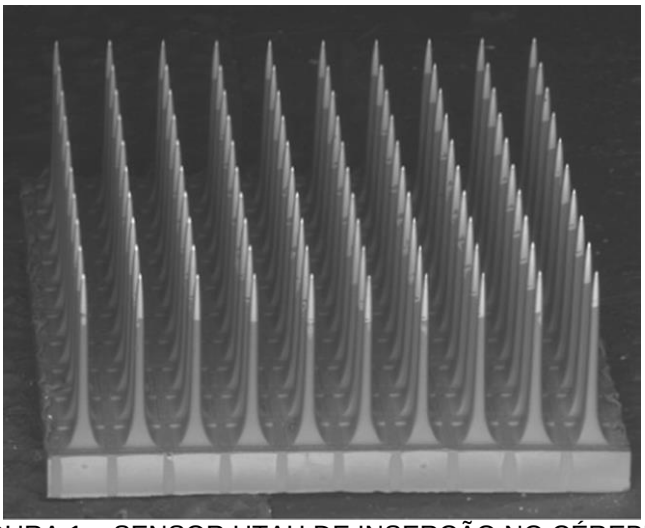

FIGURA 1 – SENSOR UTAH DE INSERÇÃO NO CÉREBRO
FONTE: Bhandari *et al.* (2010)

Os primeiros resultados apresentados de um braço robótico sendo controlado pela mente, foram realizados por Chapin *et al.* (1999) em que um rato foi treinado para mover um braço robótico em uma dimensão para obter água. Já Wessberg *et al.* (2000) obtiveram resultados similares com macacos controlando um braço robótico em uma e também em três dimensões. Finalmente Velliste *et al.* (2008) conseguiram que um macaco se alimentasse usando uma próstese neural, sendo este um dos trabalhos mais importantes para demonstrar as reais capacidades dos BMIs e afirmar o seu potencial de aplicação.

Em seres humanos, os primeiros testes foram feitos por Kennedy e Bakay (1998), em pacientes com esclerose lateral amiotrófica severa, doença que causa a degeneração progressiva dos neurônios motores (dos membros superiores e inferiores) acarretando em redução na capacidade motora e diminuição da força muscular. Nos experimentos, eles foram capazes de produzir sinais binários, ligando e desligando indicadores com a mente. Mais recentemente, Hochberg *et al.* (2012) conseguiram, usando o sensor *Utah*, com que uma tetraplégica, ou seja, uma mulher com as quatro extremidades e o tronco paralisados fosse capaz de tomar café de uma garrafa usando

um braço robótico, como o mostrado na (FIGURA 2) controlado pela sua atividade cerebral.

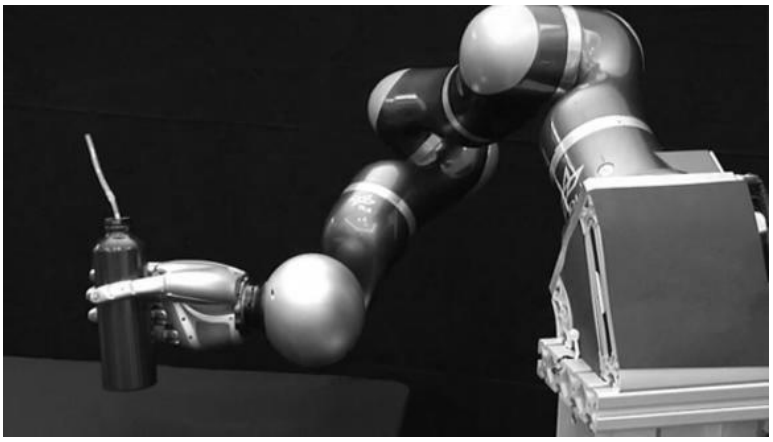

FIGURA 2 – ROBÔ USADO POR TETRAPLÉGICA ATRAVÉS DE BMI PARA TOMAR CAFÉ
FONTE: Hochberg *et al*. (2012)

Entretanto, como é mostrado por Yang (2009), devido à necessidade de cirurgia e aos riscos intrínsecos a esta para implantação do sensor no cérebro humano – além dos custos associados ao procedimento – os estudos neste tipo de tecnologia ainda são reduzidos, quando comparados às técnicas não invasivas.

Lebedev e Nicolelis (2006) mostram que sistemas não invasivos exploram os eletroencefalogramas (EEGs) para controlar computadores ou outros dispositivos. Estes sinais são obtidos através de eletrodos conectados a cabeça do paciente, desta forma não é necessário expô-los aos riscos de uma cirurgia cerebral. Pesquisas recentes (PFURTSCHELLER *et al.,* 2003; CINCOTTI *et al.* 2008; REBSAMEN *et al.,* 2010) têm demonstrado que esta abordagem pode ser utilizada com sucesso para soluções práticas, como controle de cursor de computadores, comunicação em geral, controle de cadeiras de rodas, entre outros. O lado negativo desta abordagem é que uma vez que os sinais neurais têm sua magnitude reduzida conforme se distanciam dos neurônios, a recuperação de funções motoras e o controle de prósteses artificiais ainda não foram provados possíveis, pois muita informação necessária para esta tarefa é perdida ou atenuadas antes chegar aos sensores.

1.2 CONTEXTUALIZAÇÃO

Nicolas-Alonso e Gomez-Gil (2012) fizeram uma revisão dos sistemas BCIs (do inglês *Brain-Computer Interface*) baseados em EEG que foram apresentados na literatura. Neste estudo, é colocado que estes sistemas são divididos em quatro classes principais para suporte ao usuário, que são (de acordo com suas finalidades): comunicação, restauração motora, controle de ambiente e locomoção.

Na área de comunicação, é colocado que a maioria das aplicações fazem uso de potenciais evocados, que ocorrem no cérebro devido a estímulos visuais ou auditivos, de forma que potenciais elétricos diferenciados podem ser identificados no EEG quando o elemento que o usuário deseja é destacado na tela. Para isso, Farwell e Donchin (1988) utilizaram matrizes 6x6 com os caracteres do alfabeto, como mostrado na (FIGURA 3), para obter uma velocidade de digitação de 2 caracteres por minuto.

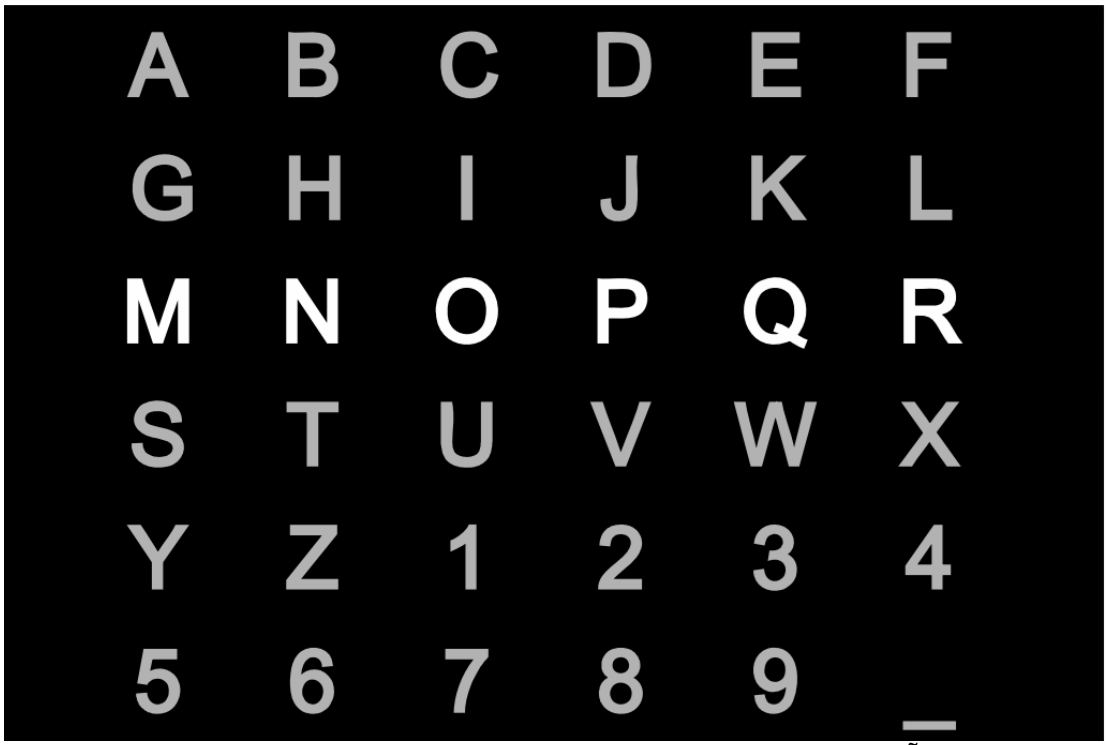

FIGURA 3 – MATRIZ DE CARACTERES PARA COMUNICAÇÃO
FONTE: Farwell e Donchin (1988)

Furdea *et al.* (2009) utilizaram um método similar, porém com uso de áudio, buscando atingir pessoas com dificuldades visuais. Também utilizando esses potenciais

evocados, Karim *et al.* (2006) e Bensch *et al.* (2007) propuseram uma navegação na Internet através de navegadores adaptados a esses sistemas.

Na restauração motora, Pfurtscheller *et al.* (2003) desenvolveram uma aplicação em que um paciente tetraplégico foi capaz de controlar uma garra robótica para pegar um cilindro através da imaginação dos movimentos. Já Muller-Putz e Pfurtscheller (2008) utilizaram quatro luzes piscando em diferentes frequências em que o usuário fixava o olhar para cada uma delas indicando um comando diferente para um braço robótico.

O controle de ambiente ajuda usuários com dificuldades motoras a interagirem com o ambiente a sua volta, tais como televisões, cortinas e luzes. Cincotti *et al.* (2008) fizeram a utilização da imaginação motora para devolver a pacientes com paralisia total a capacidade de executar essas operações. Para pacientes com algumas capacidades motoras ou sonoras ainda ativas, foram também utilizados controles e comandos de voz para aumentar sua autonomia.

Na área de locomoção, Tanaka, Matsunaga e Wang (2005) apresentaram a primeira cadeira de rodas controlada apenas por sinais de EEG. Neste caso, o usuário utiliza a imaginação de movimentos dos braços ou pernas para escolher o movimento para um dos quatro quadrantes em torno da cadeira. Os testes realizados com 6 pacientes saudáveis foram bem sucedidos. Posteriormente, utilizando também o potencial evocado, Rebsamen *et al.* (2010) apresentaram diversos caminhos ao usuário e este focaria em um deles para a sua escolha. Mais atualmente, cadeiras inteligentes com controle dividido foram apresentadas, tal como em Phillips *et al.* (2007), em que a identificação de passagens, detecção de colisão e de objetos são feitas através de sensores ultrassônicos instalados na cadeira.

Todos esses sistemas tem em comum sua arquitetura geral de funcionamento. Um sistema BCI passa pelas etapas de aquisição de sinais, pré-processamento, extração de caraterísticas, classificação dos sinais e aplicação na interface de saída. Esse sistema geral é apresentado na (FIGURA 4).

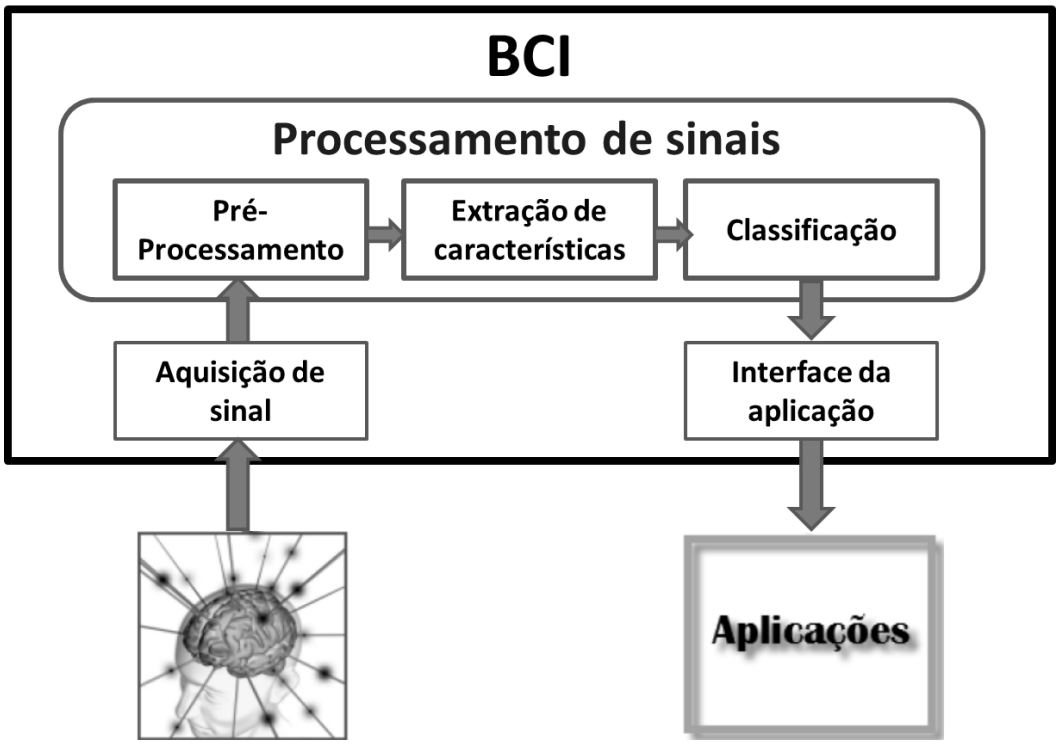

FIGURA 4 – METODOLOGIA GERAL DE SISTEMAS BCI
FONTE: Adaptado de Yang (2009)

O sistema pode ser descrito como:

1. A aquisição dos sinais envolve a leitura dos dados de EEG dos eletrodos posicionados em locais específicos na cabeça, que servem como entrada do sistema BCI. O número de eletrodos, posição e tipo dos mesmos pode mudar de sistema para sistema;
2. O pré-processamento consiste na filtragem dos sinais com relação às frequências de interesse e redução de ruídos de leitura, processos para redução ou remoção de artefatos - como o piscar do olho ou movimentos dos eletrodos - que podem influenciar nos sinais lidos. Nesta etapa também podem ser feitas filtragens espaciais, para aumentar a amplitude e facilitar a identificação das características desejadas para os eletrodos de interesse;
3. A extração de caraterísticas é uma dos pontos mais importantes de um sistema BCI. Aqui, são feitas análises espectrais, medições de amplitude e potência no domínio do tempo e da frequência, entre outras operações. Para obter caraterísticas dos sinais de EEG a fim de facilitar a

identificação/separação de diferentes comandos cerebrais dados pelo usuário;

4. A classificação, que é a etapa de maior interesse neste trabalho, tem como entradas as caraterísticas extraídas no passo anterior. Usando um conjunto de treinamento, busca-se um sistema matemático capaz de identificar novas entradas de características como pertencendo a uma ou outra classe de pensamentos. Nesta etapa, é proposto neste trabalho o estudo da eficiência de diferentes algoritmos evolutivos para a otimização do sistema de classificação;
5. A interface de aplicação faz a tradução de uma classe de comando selecionado pelo classificador e o transforma em um sinal para uma aplicação externa, tal como o deslocamento do cursor em um computador, o movimento de uma cadeira de rodas ou a ação em um braço robótico.

Tratando dos algoritmos de classificação, Lotte *et al*. (2007) fizeram uma vasta revisão dos algoritmos presentes na literatura aplicados à BCI. Eles concluíram que a máquina de vetores de suporte (SVM, do inglês *support vector machine*) é o método mais utilizado e também o que apresenta melhores resultados. Os autores concluem que razão para isso é a capacidade de regularização das SVMs e também sua simplicidade. Os autores ainda mostram que, apesar das SVMs serem mais lentas que alguns outros classificadores, elas podem ser utilizadas para aplicações online.

Neste trabalho são utilizadas SVMs para várias classes (multiclasses), que é um método de aplicação da SVM clássica para duas classes. Souza *et al.* (2006) mostram que em qualquer aplicação da SVM os parâmetros de ajuste da mesma têm grande impacto em sua performance e complexidade. Na literatura, muitas estratégias são apresentadas para definir os parâmetros das SVMs para duas classes, porém Souza *et al.* (2006) expõem que pouco é visto para a otimização dos parâmetros SVMs multiclasses. Os trabalhos apresentados neste escopo utilizam algoritmos evolutivos, tais como em Souza *et al.* (2006), Saini, Aggrawal e Kumar (2009) e Samadzadegan, Soleymani e Abbaspour (2010).

Os algoritmos evolutivos se diferem dos métodos tradicionais de otimização fazendo o uso de uma população de possíveis soluções que é distribuída pelo universo de buscas e evolve conforme o algoritmo executa suas iterações. Em cada uma dessas iterações o algoritmo gera uma nova população através da mutação de seus atuais elementos e do cruzamento dos mesmos, depois, com algum método de competição entre toda a nova população e a antiga, os indivíduos mais fracos e com soluções menos satisfatórias são removidos do processo de evolução. Segundo Fogel (1994), os algoritmos evolutivos são mais robustos e alcançam melhor balanço entre exploração global e local no espaço de busca quando aplicados a problemas reais do que as técnicas tradicionais de otimização.

Neste trabalho, são utilizados algoritmos evolutivos tradicionais da literatura, entretanto, na atualidade diversos autores estudam a implementação de técnicas mais avançadas, com a incorporação de conceitos da mecânica quântica e da teoria do caos, como mostrados por Coelho e Mariani (2008), Coelho (2008) e Sabat, Coelho e Abraham (2009). Estudos foram feitos de forma aprofundada sobre os principais algoritmos, mostrando suas potencialidades e deficiências, por Binitha e Sathya (2012) e Das *et al.* (2011).

1.3 OBJETIVO

Apesar dos algoritmos evolutivos terem mostrado sua capacidade na otimização de parâmetros de SVMs para duas classes e para multiclasses, pouco é visto na literatura sobre a utilização dessas técnicas aplicadas em conjunto em BCIs. Em especial, não existem comparações entre o desempenho dos principais algoritmos evolutivos neste contexto.

Este trabalho propõe, utilizando dados conhecidos na literatura, comparar sete algoritmos evolutivos dentre os principais da atualidade no problema de otimização dos parâmetros de SVMs multiclasses aplicadas a sistemas BCIs. Para isso serão utilizados oito algoritmos e, através de comparações estatísticas, guiar futuros pesquisadores quanto as qualidades e fraquezas de cada um nesta aplicação.

Além dos sete algoritmos obtidos na literatura, é proposto um novo algoritmo através da hibridização das técnicas que obtiverem melhores resultados na otimização do sistema proposto. Desta forma, almeja-se a obtenção de um algoritmo ainda mais adaptado para o problema e, assim, obtenha resultados superiores aos demais algoritmos.

Em termos das estratégias multiclasses das SVMs são utilizadas as duas técnicas mais difundidas na literatura. Além destas, ainda é proposto um terceiro algoritmo, com o objetivo, mais uma vez, de buscar uma técnica mais adaptada ao problema tratado.

Estes diferentes métodos serão todos comparados a fim de obter a melhor combinação de estratégias. Com esta combinação, busca-se obter resultados comparáveis àqueles dos principais institutos de pesquisa da área.

2 O CÉREBRO HUMANO

O encéfalo humano, através de sinais elétricos e reações químicas, nos dá a capacidade de pensar, sentir, ouvir, entre outros (SALADIN, 2001). Pesando aproximadamente 1,4 kg, o encéfalo pode ser divido em três componentes principais: o cérebro, o cerebelo e o tronco cerebral, como são mostrados na (FIGURA 5). Segundo Purves *et al.* (2004), o cérebro, que preenche maior parte do crânio, é responsável pelas lembranças, resolução de problemas, pensamentos e sentimentos. O cerebelo tem como função o planejamento de movimentos, o equilíbrio e a postura corporal. Finalmente, o tronco cerebral é um condutor de impulsos nervosos, motores e sensoriais, que conectam o cérebro e a medula espinhal.

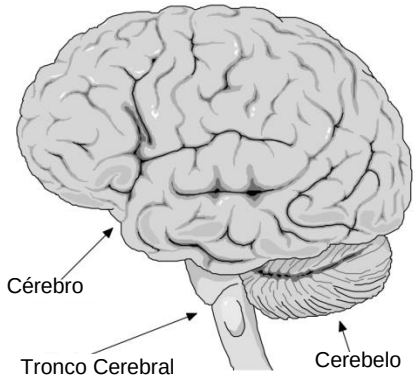

FIGURA 5 – ENCÉFALO HUMANO
FONTE: Adaptado de Gray (2002)

O cérebro pode então ser dividido em dois hemisférios similares, o direito e o esquerdo. O hemisfério esquerdo é responsável por processar os sinais sensoriais e controlar o movimento do lado direito do corpo. Analogamente, o hemisfério direito tem tal responsabilidade sobre o lado esquerdo do corpo. Desta forma, é dito que o cérebro controla as informações de forma contralateral (FORSLUND, 2003).

A camada mais externa do cérebro, com espessura entre dois e quatro milímetros, é o córtex cerebral, que é responsável pelo processamento da memória, dos movimentos, da atenção, da consciência, da linguagem e da percepção sensorial. Sendo esta então a parte mais importante a ser estudada quando se trata de BCIs. O córtex cerebral pode ser dividido em quatro lóbulos e posteriormente em diversas áreas

(FIGURA 6) com cada área sendo conectada a diferentes funções exercidas pelo cérebro, tal como definido no (QUADRO 1).

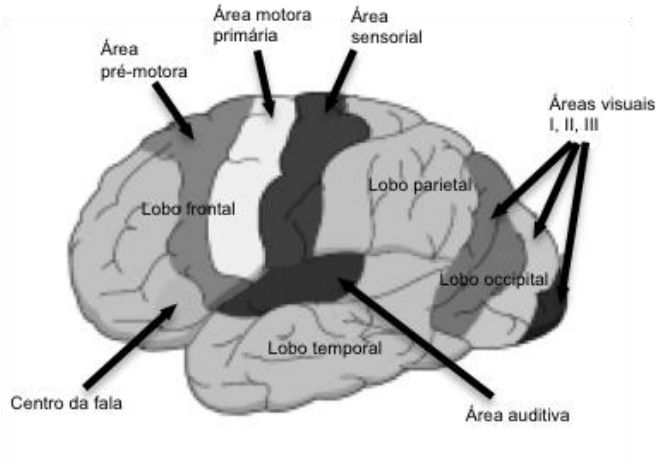

FIGURA 6 – AS DIFERENTES ÁREAS DO CÓRTEX CEREBRAL
FONTE: Adaptado de Nudo (2003)

Área Cortical	Função
Área de associação auditiva	Processamento complexo da informação auditiva
Córtex auditivo	Detecção da qualidade do som (intensidade e tom)
Centro da fala	Produção e articulação da fala
Córtex pré-frontal	Resolução de problemas, emoção, pensamento complexo
Centro associação motora	Coordenação de movimentos complexos
Córtex motor primário	Iniciação de movimento voluntário
Córtex somatosensorial primário	Recebe informação tátil do corpo
Área de associação sensorial	Processamento de informação multisensorial
Área de associação visual	Processamento complexo de informação visual
Área de Wernicke	Processamento complexo de informação visual

QUADRO 1 – ÁREAS CORTICAIS DO CÉREBRO E A SUAS FUNÇÕES
FONTE: Adaptado de Nudo (2003)

Para este trabalho é dado maior enfoque ao córtex motor, responsável pelo controle do movimento voluntário das regiões do corpo. Mais uma vez, essa sub-região do cérebro pode ser subdividida de tal forma que pode ser obtido um mapeamento do córtex motor com relação a cada parte do corpo, apresentado por Penfield e Rasmussen (1950), tal mapeamento pode ser visto na (FIGURA 7).

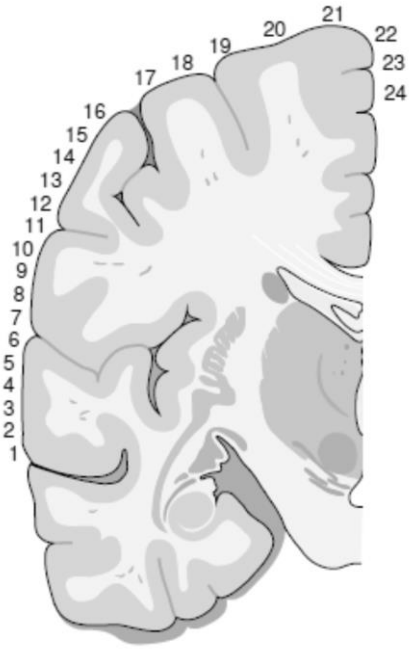

FIGURA 7 – PARTES DO CÓRTEX MOTOR
FONTE: Adaptado de Forslund (2003)

Forslund (2003) mostra que o córtex cerebral, incluindo o córtex motor, é constituído por mais de 100 bilhões de células nervosas, chamadas de neurônios. Os neurônios compartilham das mesmas características e têm os mesmos componentes como outras células das demais partes do corpo, entretanto o aspecto eletroquímico relacionado ao neurônio o diferencia, permitindo trocar mensagens por longas distâncias através de sinais elétricos que são transferidos de um para o outro. Um neurônio possui três partes básicas, como mostrado na (FIGURA 8). Smith (2004) também explica que existem diferentes tipos de neurônios no córtex cerebral, porém todos seguem, de certa forma, a mesma estrutura:

- O corpo celular é a parte principal do neurônio, possuindo todos os componentes necessários de uma célula, tais como o núcleo, o retículo endoplasmático e o ribossomo (responsáveis pela produção de proteínas) e a mitocôndria (para geração de energia);
- O axônio é o canal de saída dos sinais elétricos vindos do corpo do neurônio, em que suas terminações passarão os sinais a outro neurônio ou a um músculo. Um axônio pode chegar a medir um metro ou mais de comprimento;

- Os dendritos fazem a recepção de estímulos nervosos do ambiente ou de outros neurônios, transmitindo-os para o corpo celular. Cada neurônio pode ter de alguns poucos dendritos até centenas de milhares. Os dendritos têm a importante característica de se alterarem com o tempo, quebrando conexões e criando novas, conforme necessário.

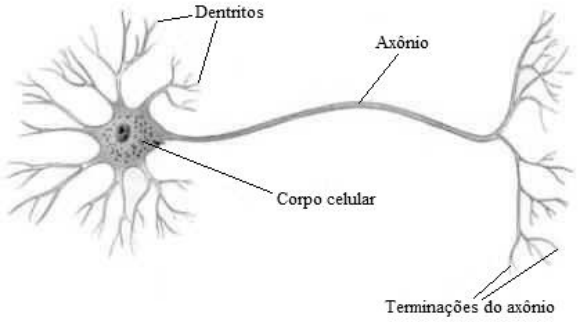

FIGURA 8 – NEURÔNIO E SUAS PARTES BÁSICAS
FONTE: Smith (2004)

Cada um desses neurônios pode estar conectado com até dez mil outros neurônios, o que pode dar uma ideia do grau de complexidade que a rede neural possui. Porém, a conexão entre as terminações do axônio de um neurônio e dos dendritos do seguinte, não existem de forma física. Entre os dois, há um espaço muito pequeno, chamado de fenda sináptica. É por essa fenda, que através de uma reação química, a sinapse ocorre, transmitindo então sinais elétricos de um neurônio para o outro (FORSLUND, 2003).

2.1 ELETROENCEFALOGRAMA

O físico inglês Richard Caton, em 1875, conectou eletrodos diretamente ao córtex cerebral de gatos e macacos – e usando apenas um galvanômetro com lupas - mostrou pela primeira vez a capacidade da geração de potencial elétrico no cérebro. Sem a existência de amplificadores eletrônicos e com o pouco conhecimento do assunto na época, seus resultados foram considerados incríveis, inspirando muitos outros pesquisadores para esta área. Porém, foi apenas em 1929 que o psiquiatra alemão Hans Berger anunciou que era possível inferir e apresentar de forma gráfica as correntes

elétricas geradas pelo cérebro humano. Ainda mostrou que o comportamento da geração de potencial variava conforme a atividade e estado mental do paciente; e principalmente, tudo isso sem a abertura do crânio. Tal método foi batizado de eletroencefalograma, o EEG (FORSLUND, 2003).

Após anos de estudos e desenvolvimento da tecnologia para extração e compreensão do EEG algumas características foram observadas, em especial com relação à frequência e à amplitude dos sinais lidos no EEG. Foi verificado, por exemplo, que a amplitude do sinal de EEG lido enquanto uma pessoa está dormindo profundamente é maior do que o de alguém totalmente acordado. Isso ocorre porque, o EEG não lê o sinal apenas de um neurônio, mas sim da soma de um grande conjunto de neurônios ativos no mesmo instante de tempo. Sendo assim, a amplitude do sinal do EEG é de alguma forma menos dependente da quantidade de atividade cerebral e mais condicionado a sincronia dos mesmos - que é maior quando o cérebro está descansando, o corpo está relaxado e os olhos estão fechados, de forma que os sinais sensoriais, muitas vezes, nem alcançam o córtex cerebral.

Lehtonen (2002) mostra que nesse mesmo sentido foram percebidos diferentes ritmos, ou frequências, de atividade cerebral na leitura do EEG conforme a atividade que um determinado indivíduo está executando. As frequências até hoje observadas são: gama, beta, alfa, teta e delta (FIGURA 9).

2.1.1 Artefatos

Muitas das mudanças de potencial elétrico visto no sinal de EEG são oriundas de diferentes fontes, além da atividade cerebral. Tais mudanças são chamadas de artefatos e podem ter origem em questões técnicas do equipamento, no ambiente onde o EEG está sendo coletado ou em questões fisiológicas (SMITH, 2004).

Como artefatos técnicos, podem ser citadas as interferências causadas por equipamentos elétricos ao redor do sistema de medição conectados a rede elétrica (geralmente na frequência de 50 ou 60 Hz) e também questões relacionadas aos próprios eletrodos utilizados, que quando conectados de forma errônea ou em condições de uso precárias podem ter alterações em sua impedância, afetando as leituras elétricas.

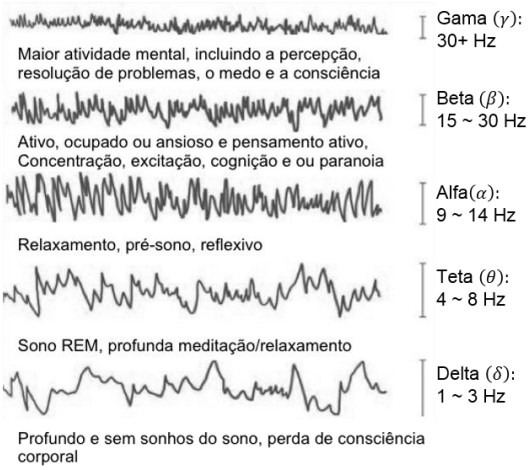

FIGURA 9 – RITMOS DE ONDAS CEREBRAIS E ATIVIDADES RELACIONADAS
FONTE: Adaptado de Lehtonen (2002)

Já como artefatos fisiológicos encontram-se os sinais de eletro-oculografia (EOG, do inglês *electrooculography*), em que o movimento e piscar dos olhos afetam os sinais de EEG; os sinais de eletromiografia, que são causados pela tensão muscular – em especial nos músculos relacionados com o pescoço, testa e sistema mastigatório; os artefatos também podem estar conectados com o sistema cardíaco, a respiração e outros; o suor da pessoa em que o sinal está sendo lido também pode alterar a impedância dos eletrodos e movimentos da pessoa também podem alterar os sinais lidos.

Smith (2004) ainda coloca que muitos dos tipos de artefatos apresentados podem ser evitados no momento da leitura, tal como impedir a movimentação durante a leitura do EEG, evitar movimentos oculares e piscadas, deixar o paciente confortável e seguro para evitar suor e tensão musculares, etc. Para os artefatos que permanecem, algoritmos podem ser aplicados para reduzir seus efeitos no sinal de EEG.

2.1.2 Colocação dos Eletrodos

Forslund (2003) expõe que uma importante questão a ser abordada é o posicionamento dos eletrodos na cabeça. Sabendo que o processamento de cada tipo de atividade ocorre em uma área diferente do córtex cerebral, a distribuição dos eletrodos deve ser feita de tal forma que todas essas áreas sejam contempladas na leitura das

atividades cerebrais. Para tanto, um sistema padrão de localização dos eletrodos foi definido e nomeado de sistema internacional 10-20 de colocação de eletrodos. Os valores “10” e “20” presentes no nome representam a distância entre eletrodos vizinhos, sendo 10% ou 20% da distância total do lado esquerdo ao lado direito do crânio ou a distância total da parte frontal à traseira, como pode ser visto na (FIGURA 10).

Os pontos de referência para as medições do crânio (e dessa forma do sistema 10-20) são, para o lado direito e esquerdo, os lóbulos das orelhas direita e esquerda, respectivamente, para a parte frontal é o násio, que é o ponto entre a testa e o nariz na altura dos olhos, e na parte traseira é o ínion, que é um osso sobressalente na base do crânio no meio da nuca (FORSLUND, 2003).

Para referenciar os eletrodos, um sistema de nomenclatura também foi criado. Iniciando pela letra representando o lóbulo em que está colocado, podendo ser *F*, *Fp*, *T*, *C*, *P* e *O*, que representam o lóbulo frontal, frontal polar, temporal, central, parietal e occipital, respectivamente. Em seguida um número (ou a letra “*z*”) representando o hemisfério da colocação e sua posição em relação ao centro, sendo os números ímpares para o hemisfério esquerdo e os números pares para o hemisfério direito. Ainda, quanto maior o número, mais distante do centro está o eletrodo. A letra “*z*” representa exatamente os eletrodos posicionados na linha central (FIGURA 10).

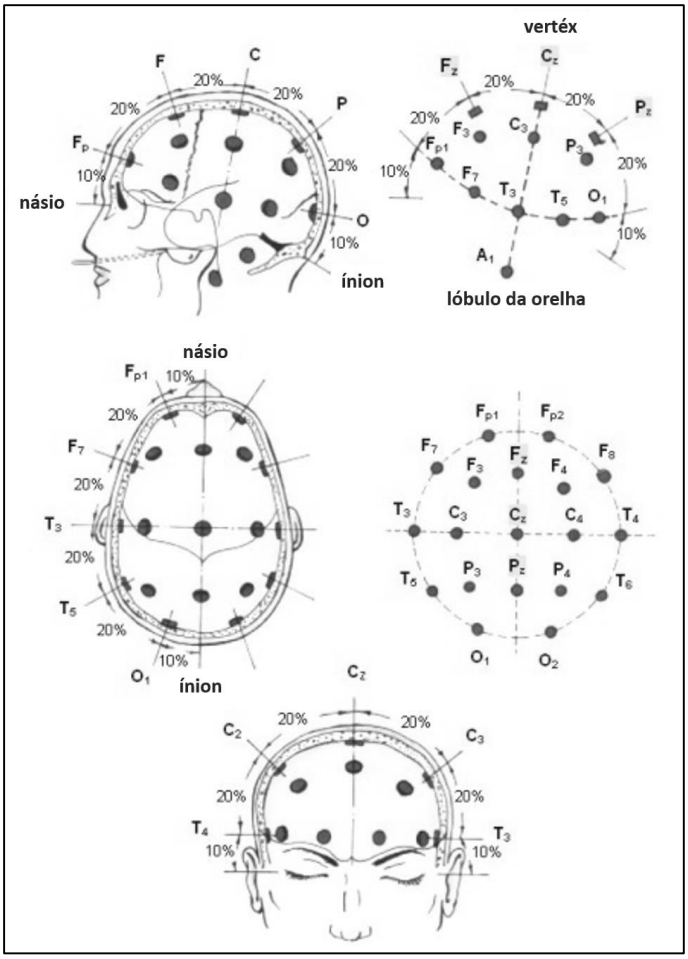

FIGURA 10 – SISTEMA 10-20 DE COLOCAÇÃO DOS ELETRODOS PARA LEITURA DE SINAL EEG
FONTE: Forslund (2003)

2.2 DESSINCRONIZAÇÃO E SINCRONIZAÇÃO RELACIONADA A EVENTOS

A dessincronização relacionada a evento (ERD, do inglês *event-related desynchronization*) é a atenuação da amplitude de um determinado ritmo de EEG no acontecimento de um evento específico. Já a sincronização relacionada a evento (ERS, do inglês *event-related synchronization*) é definida, opostamente, como o fortalecimento da amplitude de um ritmo cerebral para aquele evento (PETERS; PFURTSCHELLER; FLYVBJERG, 2001).

Peters, Pfurtscheller e Flyvbjerg (2001) ainda mostram que para fazer essa medição de ERD e ERS, é calculada a potência presente na frequência de interesse (o conceito de potência da frequência é explicado com mais detalhes na descrição dos

extratores de características). Quando ERD e ERS estão sendo avaliadas para a imaginação de movimentos motores, a frequência que deve ser observada no caso da ERD é chamada de μ, e é definida entre 8 e 12 Hz. Já para a ERS é a frequência β, definida entre 15 e 30 Hz. Os estudos mostram que essa variação de atenuação e magnificação de frequências na imaginação de movimentos ocorrem nas mesmas regiões do cérebro que a execução do movimento real, como mostrado na (FIGURA 11).

FIGURA 11 – ATIVAÇÃO DO CÉREBRO PARA IMAGINAÇÃO MOTORA
FONTE: Pfurtscheller *et al.* (2000)

A utilização das potências de banda, em especial nas bandas μ e β, possibilitam a identificação e classificação da imaginação motora.

3 MÁQUINA DE VETORES DE SUPORTE

Em aprendizagem de máquinas e reconhecimento de padrões, classificação se refere à tarefa de assignar a um conjunto de dados de entrada, chamado de vetor de características, uma classe à qual o mesmo deve pertencer. Neste trabalho, é utilizado um sistema de classificação supervisionado, em que primeiramente, utiliza-se um conjunto de dados cujas classes são previamente conhecidas, sendo este conjunto chamado de dados de treinamento. Após o sistema ser treinado para o conjunto de treinamento, este é testado e avaliado para o conjunto de testes ou para a aplicação de interesse. Para este fim, neste trabalho é utilizada a máquina de vetores de suporte (SVM, do inglês, *support vector machine*).

3.1 MÁQUINA DE VETORES DE SUPORTE LINEARES COM MARGENS RÍGIDAS

Seja *M* o número de entradas m-dimensionais x_i $(i = 1, 2, \dots, M)$ pertencentes à Classe 1 ou Classe 2, em que são associados os valores de $y_i = +1$ para a Classe 1 e $y_i = -1$ para a Classe 2, o conjunto de dados é dito linearmente separável se os padrões das diferentes classes pertencentes a ele podem ser separados por pelo menos um hiperplano. Porém, como pode ser visto no exemplo da (FIGURA 12), para dados linearmente separáveis, existem infinitos hiperplanos capazes de separar corretamente esses dados (SHILTON, 2006). A escolha deste hiperplano tem consequências diretas na capacidade de generalização do sistema de reconhecimento. Vapnik (1998) definiu uma forma de escolher o hiperplano a ser utilizado de tal forma que se maximiza a sua margem, em que margem é definida como a distância entre os pontos mais próximos de cada uma das classes ao hiperplano, como é visto na (FIGURA 13).

Shilton (2006) ainda expõe que é intuitivamente claro que o método proposto por Vapnik define o hiperplano ótimo a fim de maximizar a capacidade de generalização do sistema se as amostradas utilizadas forem boas representantes daquelas classes como um todo. Um aprofundamento matemático neste tópico é apresentado por Burges (1998).

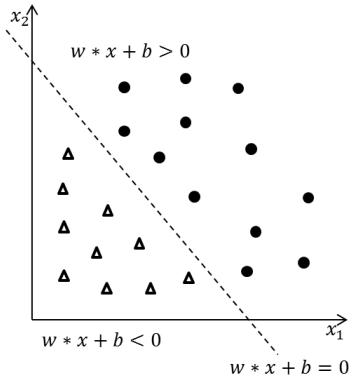

FIGURA 12 – EXEMPLO DE CONJUNTO DE DADOS LINEARMENTE SEPARÁVEL
FONTE: Adaptado de Burges (1998)

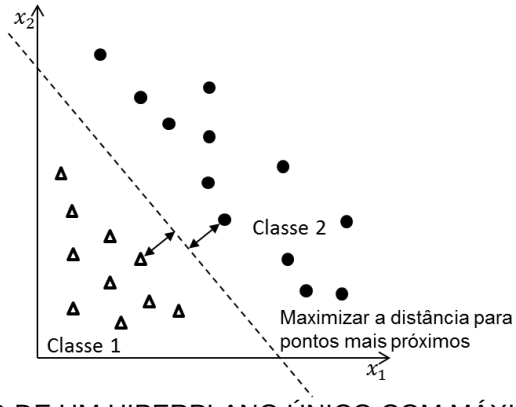

FIGURA 13 – DEFINIÇÃO DE UM HIPERPLANO ÚNICO COM MÁXIMA DISTÂNCIA PARA OS PONTOS MAIS PRÓXIMOS DAS DUAS CLASSES
FONTE: Adaptado de Burges (1998)

Burges (1998) mostra que para a determinação do hiperplano ótimo, considerando a premissa de que os dados são linearmente separáveis, escalam-se os valores de *w* e *b*, para que os pontos mais próximos ao hiperplano de cada uma das classes satisfaçam a $|w \cdot x + b = 1|$. Sendo esta a representação canônica do hiperplano a ser utilizada no decorrer das demonstrações para determinação do hiperplano ótimo.

Com esta descrição, define-se a equação 1 como classificador linear que separa o conjunto dado com uma margem positiva. Analisando este sistema, observa-se que não existem pontos entre os planos $w \cdot x + b = 0$ e $w \cdot x + b = \pm 1$, sendo assim, a distância entre qualquer ponto de uma das classes e o hiperplano de separação, será maior ou igual à distância entre os planos $w \cdot x + b = 0$ e $|w \cdot x + b| = 1$. Sendo as SVMs de margem rígidas então definidas conforme a equação 1,

$$w \cdot x + b \begin{cases} \geq +1, & \text{para } y_i = +1 \\ \leq -1, & \text{para } y_i = -1 \end{cases} \quad (1)$$

onde *w* é um vetor m-dimensional e *b* é o termo compensador. Em que os valores de +1 e -1 no lado direito da equação poderiam ser qualquer constante a com $a > 0$ e $-a$, respectivamente. Porém, ao dividir os dois lados da inequação por a, volta-se à original. A equação 1 também pode ser representada de forma equivalente como na equação 2 (BURGES, 1998), tal que,

$$y_i(w \cdot x + b) \geq 1 \text{ para } i = 1,2, \dots M \quad (2)$$

Considerando os pontos x_1 e x_2 sobre os hiperplanos $w \cdot x + b = -1$ e $w \cdot x + b = +1$, respectivamente, ou seja, os valores mais próximos ao hiperplano de separação de cada uma das classes, e considerando que x_1 intercepta a reta perpendicular a x_2 como apresentado na (FIGURA 14), define-se a equação 3,

$$\begin{cases} w \cdot x_1 + b = -1 \\ w \cdot x_1 + b = +1 \end{cases} \quad (3)$$

A partir da equação 3, pode-se obter a equação 4.

$$w \cdot (x_1 + x_2) = 2 \quad (4)$$

Sendo $\|\cdot\|$ a norma de um vetor e, sabendo que *w* e $x_2 - x_1$ são ortogonais ao hiperplano separador e, desta forma, paralelos entre si, obtêm-se a equação:

$$|w \cdot (x_1 + x_2)| = \|w\| \times \|x_2 - x_1\| \quad (5)$$

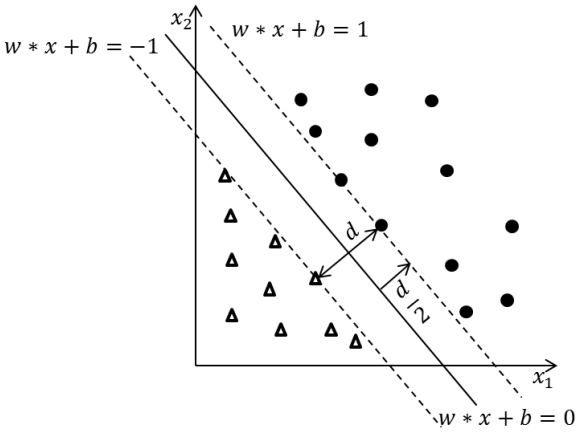

FIGURA 14 – DISTÂNCIA ENTRE HIPERPLANOS
FONTE: Adaptado de Burges (1998)

Ao substituir a equação 4 em 5, obtém-se a equação 6, tal que,

$$\|x_2 - x_1\| = \frac{2}{\|w\|} \tag{6}$$

Dessa forma, verifica-se que a distância entre os planos de $w \cdot x + b = -1$ e $w \cdot x + b = +1$ é $\frac{2}{\|w\|}$, uma vez que essa pode é a mesma da distância entre os pontos x_1 e x_2, dada por $\|x_2 - x_1\|$.

Burges (1998) ainda mostra que se pode também definir que a distância entre o plano de divisão ótimo $w \cdot x + b = 0$ e ambos os planos $w \cdot x + b = -1$ e $w \cdot x + b = +1$ é $\frac{1}{\|w\|}$, sendo esse o tamanho da margem. Uma vez que se busca maximizar o valor da margem – ou seja, minimizar o valor de $\|w\|$ – o hiperplano de divisão ótimo é definido pelos valores de *w* e *b* que satisfaçam o sistema de otimização definido na equação:

$$\begin{gathered} Minimizar\ \|w\|^2 \\ Sujeito\ \text{à}: y_i(w \cdot x_i + b) - 1 \geq 0, \text{para}\ i = 1,2,..m \end{gathered} \tag{7}$$

Sendo este um problema clássico para otimização por programação quadrática. Para tanto, a função Lagrangiana conforme apresentado na equação 8 é definida em termos de *w* e *b*.

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (w \cdot x_i + b) - 1) \quad (8)$$

Em que α são os vetores *multiplicadores de Lagrange*. Agora, busca-se minimizar a equação 8 e maximizar os valores de α. Tal que, os valores ótimos para esta otimização podem ser obtidos pelas equações 9 e 10, tal que:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \quad (9)$$

$$\frac{\partial L}{\partial w} = w = \sum_{i=1}^{n} \alpha_i y_i \, x_i \quad (10)$$

Substituindo a equação 8 com os resultados obtidos nas equações 9 e 10, é definido o problema conhecido como *dual*, apresentado na equação 11, tal que,

$$Maximizar \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$
$$Sujeito \text{ à: } \begin{cases} \alpha_i \geq 0, i = 1,2, \dots mn \\ \sum_{j=1}^{n} \alpha_i y_i = 0 \end{cases} \quad (11)$$

Ao obter o valor ótimo de α, denominado α^*, equação 8, é possível calcular o valor w^* (valor ótimo de w) através da substituição de α por α^* na equação 10. Com os valores encontrados para w^* nesta substituição, b^* pode ser calculado pela equação 12. Sendo $x_{Classe1}$ e $x_{Classe2}$ os padrões de treinamento da Classe 1 e da Classe 2, respectivamente.

$$b^* = -\frac{1}{2}[\max(w^* \cdot x_{Classe1}) + \min(w^* \cdot x_{Classe2})] \quad (12)$$

Através dos equacionamentos demonstrados, obtêm-se os valores necessários para determinação da SVM - w^*, b^* e α^* - ou seja, o treinamento da mesma. É importante notar que o valor de α^* assume valores positivos para todos os elementos que estão sobre a margem (ou seja, a uma distância $\frac{1}{\|w\|}$ do hiperplano ótimo). Estes valores são denominados vetores de suporte (SV, do inglês *support vector)* e são exclusivamente responsáveis pela determinação do hiperplano ótimo. Dessa forma, pode-se definir w^* como na equação:

$$w^* = \sum_{x_i \in SV} \alpha_i^* y_i x_i \quad (13)$$

Para todos os elementos cuja distância é maior que a margem, α^* é nulo; caracterizando a propriedade chamada de esparsividade da solução. Finalmente, com os valores de w^*, b^* e α^* obtidos pelo treinamento da SVM, um novo padrão x_{new} pode ser classificado (C) através da equação 14, tal que,

$$C(x_{new}) = sign\left(\sum_{x_i \in SV} \alpha_i^* y_i x_i \cdot x_{new} + b^*\right) \quad (14)$$

Em que $sign(\cdot)$ é definido pela equação 15. Dessa forma, a classificação de um novo padrão exige apenas ao produto interno entre o novo padrão e os SVs.

$$sign(k) = \begin{cases} -1, & k < 0 \\ +1, & k \geq 0 \end{cases} \quad (15)$$

3.2 MÁQUINA DE VETORES DE SUPORTE LINEARES COM MARGENS FLEXÍVEIS

Na abordagem da SVM feita anteriormente, considerava-se que não havia nenhum exemplo entre o hiperplano ótimo definido por $w \cdot x + b = 0$ e os hiperplanos paralelos a ele que contém o exemplo mais próximo a este hiperplano ótimo, definidos por $w \cdot x + b = +1$ e $w \cdot x + b = -1$.

Porém, em problemas reais, tal situação raramente ocorre, principalmente devido a ruídos nos dados ou na leitura dos mesmos. Tal não linearidade também pode ser intrínseca ao problema de classificação (SHILTON, 2006). Um exemplo de um caso de dados não linearmente separáveis pode ser visto na (FIGURA 15).

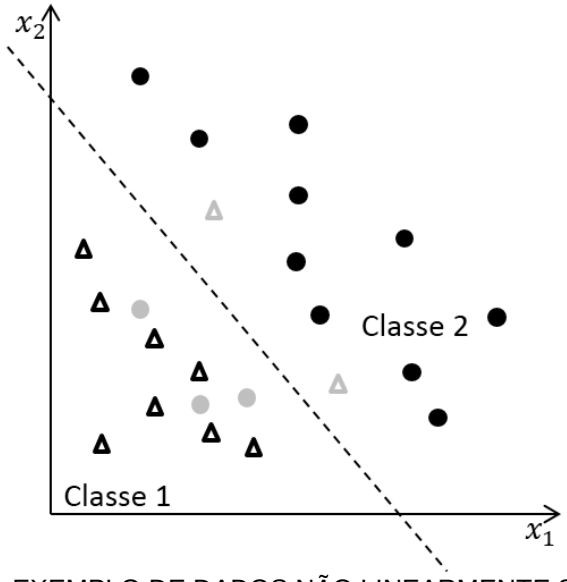

FIGURA 15 – EXEMPLO DE DADOS NÃO LINEARMENTE SEPARÁVEIS
FONTE: Adaptado de Shilton (2006)

Shilton (2006) mostra que independente do motivo da inseparabilidade dos dados por um hiperplano, uma adaptação as SVMs com margens rígidas foi apresentada. Nesta adaptação, é proposta a introdução do conceito de relaxamento, de forma que alguns erros de classificação são permitidos, suavizando assim as restrições impostas para a determinação do hiperplano ótimo. Desta forma, foram adicionadas variáveis de relaxamento na formulação do problema. Para que erros de classificação sejam aceitos, define-se a inequação conforme a equação 16.

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \text{ para } i = 1,2, \dots M \tag{16}$$

Com $\xi_i > 0$. Quando $\xi_i > 1$ a *i-ésima* inequação é violada em comparação com a inequação correspondente para o caso de margem rígida. Dessa forma, o problema de otimização pode ser definido conforme a equação:

$$\begin{gathered} minimizar\ \|w\|^2 + c\sum_{i=1}^{n}\xi_i \\ sujeito\ a: \begin{cases} y_i(w \cdot x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \end{gathered} \tag{17}$$

Em que *c* é uma constante real positiva. Dessa forma, a Lagrangiana deve ser considerada como na equação 18, como mostrado a seguir:

$$L(w, b, \xi, \alpha, \gamma) = \|w\|^2 + c\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\gamma_i\xi_i \tag{18}$$

Em que os multiplicadores de Lagrange $\alpha \geq 0$ e $\gamma \geq 0$, sendo este segundo conjunto necessário pelas novas variáveis de relaxamento ξ. Com tal definição, podem-se obter as equações 19, 20 e 21, tal que,

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{19}$$

$$\frac{\partial L}{\partial w} = w = \sum_{i=1}^{n}\alpha_i y_i\, x_i \tag{20}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq c, i = 1,2, \dots, m \tag{21}$$

E, finalmente, substituindo as equações 19, 20 e 21 em 18, obtém-se o problema *dual* que pode ser resolvido pelo método de programação quadrática como apresentado na equação 22, tal que,

$$Maximizar \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$
$$Sujeito\ \text{à}: \begin{cases} 0 \leq \alpha_i \leq c, i = 1,2,\dots,m \\ \sum_{j=1}^{n} \alpha_i y_i = 0 \end{cases} \quad (22)$$

Em comparação para o caso de margem rígida, apenas a restrição quanto ao limite superior de α é modificado. A partir desta otimização os valores de α^* são obtidos. Os valores de w^* e b^* são obtidos da mesma forma como apresentado para o problema de margem rígida - assim como a classificação de novos padrões. O valor de ς_i determina a posição do padrão *i* com relação à margem e ao hiperplano ótimo e pode ser calculado conforme a equação 23, como mostrado a seguir:

$$\varsigma_i(w,b) = \begin{cases} 0, & y_i = +1\ e\ w \cdot x_i + b \geq +1 \\ 1 - w \cdot x_i + b, & y_i = +1\ e\ w \cdot x_i + b < +1 \\ 0, & y_i = -1\ e\ w \cdot x_i + b \leq -1 \\ 1 + w \cdot x_i + b, & y_i = -1\ e\ w \cdot x_i + b > -1 \end{cases} \quad (23)$$

Dessa forma, o valor de ς_i pode significar três posições distintas, também mostradas na (FIGURA 16):

a) $\varsigma_i = 0$, então o padrão *i* foi classificado corretamente;
b) $0 < \varsigma_i \leq 1$, significa que o padrão *i* foi corretamente classificado, mas sua distância para o hiperplano ótimo é menor que a margem;
c) $\varsigma_i > 1$, o padrão *i* foi classificado erroneamente.

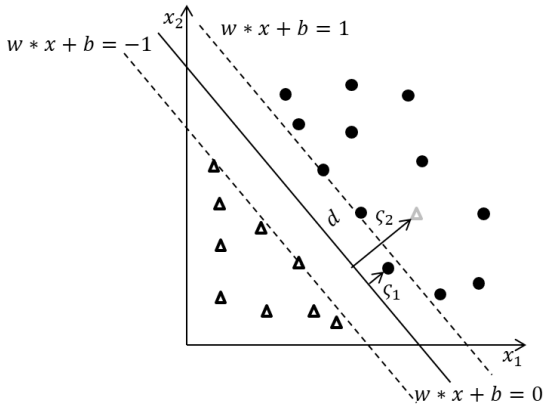

FIGURA 16 – EXEMPLO DOS VALORES DE CSI
FONTE: Adaptado de Shilton (2006)

Shilton (2006) mostra que para o treinamento das SVMs com margem flexível, têm-se os mesmos objetivos da SVM com margem rígida. Busca-se maximizar a margem entre o hiperplano ótimo e os padrões mais próximos do mesmo e zerar os erros de classificação. Quando os erros de classificação não podem ser zerados, busca-se então reduzir ao máximo o número de erros de classificação e de padrões no interior da margem. Para tanto, combinam-se estes valores em uma equação para posterior minimização, como a equação 24, em que *C* é uma constante que define o peso para o relaxamento, ou seja, aceitar mais ou menos erro.

$$\varepsilon(w, b) = \|w\|^2 + C \sum_{i=1}^{n} \varsigma_i(w, b) \tag{24}$$

3.3 SVM NÃO LINEAR

As SVMs lineares descritas nas seções anteriores são muito simples e, para alguns casos, bastante eficientes. Porém, sua utilização é restrita, uma vez que em diversas situações um hiperplano não é capaz de separar os dados de entrada de forma satisfatória. Para isso, um mapeamento para o chamado espaço de características é feito, de forma a transformar os dados em linearmente separáveis (ou próximo disso) novamente, como mostrado na (FIGURA 17).

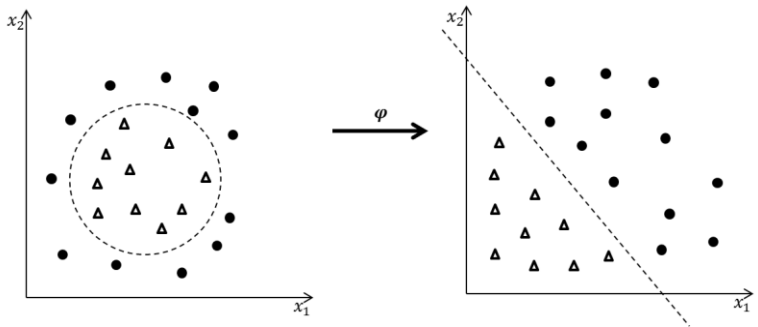

FIGURA 17 – TRANSFORMAÇÃO DO CONJUNTO DE DADOS NO ESPACO DE ENTRADA PARA O ESPACO DE CARACTERÍSTICAS
FONTE: Adaptado de Shilton (2006)

Usando um vetor de funções não lineares $\phi(x) = \left(\phi_1(x), \phi_2(x), \ldots, \phi_l(x)\right)$ que mapeia o vetor de entrada *m*-dimensional *x* para o espaço de características *l*-dimensional, onde a separação linear é feita pelo hiperplano mostrado na equação 25. Em que w é um vetor *l*-dimensional e *b* é o termo compensador.

$$w \cdot \phi(x) + b = 0 \tag{25}$$

Devido ao modo como as SVMs são definidas, é necessário a execução do produto interno entre dois padrões no espaço de características. Tal operação pode ser computacionalmente custosa, uma vez que o espaço de caraterísticas pode assumir um número de dimensões muito alto (algumas vezes infinitas dimensões, o que faria a transformação impossível). Para contornar este problema, o chamado truque de *kernel* (do inglês, *kernel trick*) é utilizado, em que os padrões não são tratados explicitamente no espaço de características.

O que se busca é calcular o valor do produto interno entre dois padrões de entrada x_1 e x_2, porém sem ter que calcular seus respectivos mapeamentos no espaço de características. Dessa forma, define-se uma função *kernel* como $K(x_1, x_2)$ que resultará no produto interno de x_1 e x_2 em algum espaço de características, ou seja, $\phi(x_1) \cdot \phi(x_2)$, sem calcular os valores de $\phi(x_1)$ ou de $\phi(x_2)$ – de fato, não é nem mesmo necessário saber qual é o espaço de características é $\phi(\cdot)$.

Para se definir um *kernel* as chamadas condições de Mercer devem ser satisfeitas. Essas condições garantem que um determinado *kernel* $K(x_1, x_2)$ de fato

representa o produto interno entre $\phi(x_1)$ e $\phi(x_2)$ em algum espaço de características ϕ. As condições de Mercer são representadas na equação 26, para todo *M, x_i* e *h_i*, tal que *M* é um número natural e h pertence aos reais.

$$\sum_{i,j=1}^{M} h_i h_j K(x_i, x_j) = \left(\sum_{j=1}^{M} h_j \phi^T(x_j)\right)\left(\sum_{j=1}^{M} h_j \phi(x_j)\right) \geq 0 \tag{26}$$

Usando a função *kernel,* o problema *dual* pode ser definido de acordo com a equação 27, tal que,

$$\begin{aligned} &Maximizar \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &Sujeito\ \text{à}: \begin{cases} 0 \leq \alpha_i \leq c, i = 1,2, \ldots, m \\ \sum_{j=1}^{n} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{27}$$

Ao obter o valor ótimo de α, denominado α^*, pela equação 27, é possível calcular o valor w^* (valor ótimo de *w*) através da substituição de α por α^* do mesmo modo como feito na SVM linear. O valor de b^* também pode ser calculado do mesmo modo. Para a classificação de um novo padrão x_{new}, a equação 28 é aplicada, tal que,

$$C(x_{new}) = sign\left(\sum_{x_i \in SV} \alpha_i^* y_i K(x_i, x_{new}) + b^*\right) \tag{28}$$

Apesar de qualquer função poder ser usada como *kernel*, uma vez que satisfaça a condições de Mercer, algumas funções são utilizadas com maior frequência nas aplicações usando SVMs. Essas funções são apresentadas a seguir, em que os valores de d na *kernel* polinomial, σ na função de base radial (*RBF,* do inglês *Radial Basis Function)* e β_0 e β_1 no sigmoidal são parâmetros que o usuário deve definir ao utilizar a SVM.

a) Linear: $K(x_i, x_j) = x_i \cdot x_j$;

b) Polinomial de grau *d*: $K(x_i, x_j) = (\tau + x_i \cdot x_j)^d$;

c) RBF: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$;

d) Sigmoidal: $K(x_i, x_j) = \tanh(\beta_0 x_i \cdot x_j + \beta_1)$.

Alguns pontos que devem ser considerados com relação aos *kernels* apresentados, são:

1. Na *kernel* polinomial as funções de mapeamento ϕ também são polinomiais e sua complexidade aumenta conforme o grau do polinômio é elevado;
2. A *kernel* RBF representa um espaço de características de infinitas dimensões, em que quase todas as formas de mapeamento podem ser implementados por essa função. Por meio dessa função, também pode ser definida a rede neural do tipo RBF, tal que o número de funções de base radial é igual ao número de SVs e o centro delas é igual aos valores dos multiplicadores de Lagrange de cada um desses SVs;
3. A sigmoidal pode explicitar uma rede neural do tipo *perceptron* multicamadas, sendo o número de SVs referente à quantidade de neurônios na camada intermediária e os valores dos multiplicadores de Lagrange aos vetores de *bias*.

A capacidade de uma SVM ter uma boa generalização assim como a sua complexidade (número de SVs) dependem dos valores selecionados para os parâmetros e do próprio *kernel* escolhido, assim como a constante *c* – relacionada ao relaxamento da SVM. Algumas técnicas foram desenvolvidas para a seleção destes parâmetros, tais como a regra *span*, a regra de *Jaakola* e *Haussler* e a regra proposta por Chapelle *et al.* (2002).

3.4 SVM PARA MÚLTIPLAS CLASSES

Conforme pode ser observado na elaboração matemática das SVMs, elas são originalmente utilizadas para a classificação de duas classes distintas. Entretanto, muitas aplicações reais necessitam da classificação de um número maior de grupos. Para aplicação nesses problemas, chamados de classificação de múltiplas classes, foram criados métodos para estender as SVMs para este tipo de problema. Duas formas de decomposição são comumente utilizadas para esta tarefa: “um-contra-todos” e “todos-contra-todos”. Estes dois métodos são mais utilizados devido à simplicidade de implementação e de compreensão, além de apresentarem resultados similares ou superiores a métodos mais complexos apresentados na literatura. Estes dois métodos serão cobertos mais a fundo nas subseções 3.4.1 e 3.4.2.

Na subseção 3.4.3, entretanto, é proposto um terceiro método, que busca unir as qualidade dos dois métodos anteriormente apresentados e, paralelamente, suprimir as carências dos mesmos.

Todos os métodos utilizarão como problema teste os dados conforme apresentado na (FIGURA 18), que foram gerados através de distribuições normais no plano – ou seja, para duas caraterísticas de entrada.

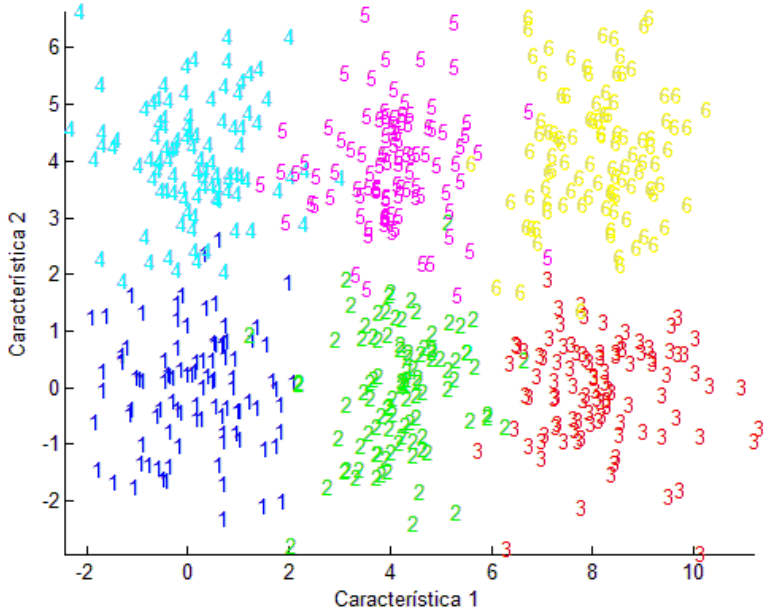

FIGURA 18 – PROBLEMA DE CLASSIFICAÇÃO TESTE
FONTE: O autor (2014)

3.4.1 Método “um-contra-todos”

Weston e Watkins (1998) mostram que o método “um-contra-todos” consiste na criação de *N* SVMs, sendo *N* o número total de classes do problema. Para o treinamento de cada uma das SVMs, a classe respectiva a essa SVM é fixada e todas as demais são consideradas como a outra classe do problema. Essa metodologia é exemplificada na (FIGURA 20), que mostra a classificação de saída de cada uma das *N* SVMs geradas para o problema teste.

Ao final, para a classificação de um novo indivíduo, este deve passar por todas as SVMs e, finalmente, a SVM que apresentar o valor máximo de classificação, será definida como o valor de saída. Na (FIGURA 19) mostra-se o resultado final obtido para a classificação do problema multiclasses (os elementos em vermelho representam valores classificados erroneamente).

Um ponto fraco do método “um-contra-todos” é que este utiliza toda a população de treinamento em cada uma das SVMs. Desta forma, a velocidade de treinamento das SVMs é baixa e o custo computacional elevado.

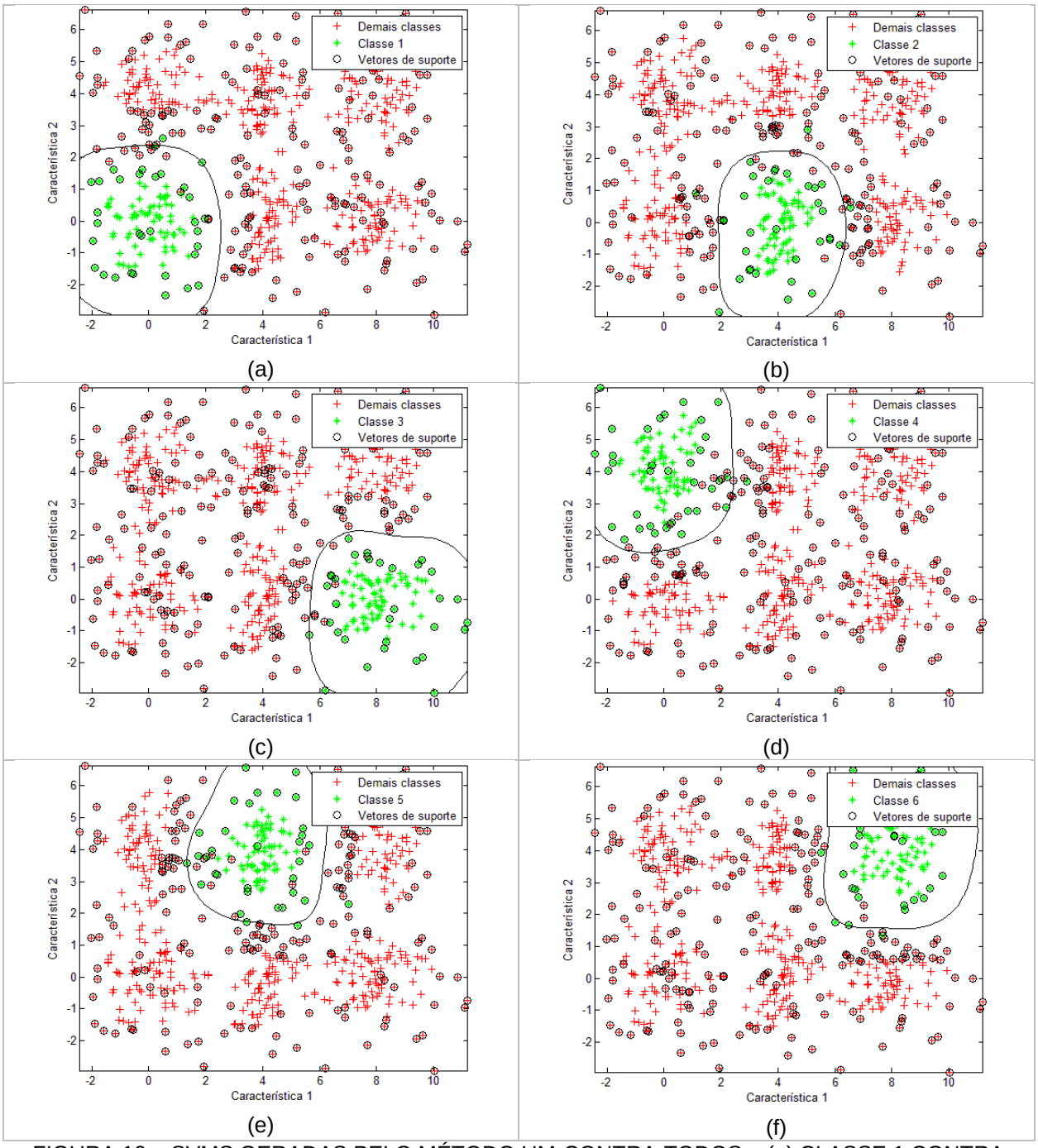

FIGURA 19 – SVMS GERADAS PELO MÉTODO UM-CONTRA-TODOS – (a) CLASSE 1 CONTRA TODAS. (b) CLASSE 2 CONTRA TODAS (c) CLASSE 3 CONTRA TODAS (d) CLASSE 4 CONTRA TODAS (e) CLASSE 5 CONTRA TODAS (f) CLASSE 6 CONTRA TODAS.
FONTE: O autor (2014)

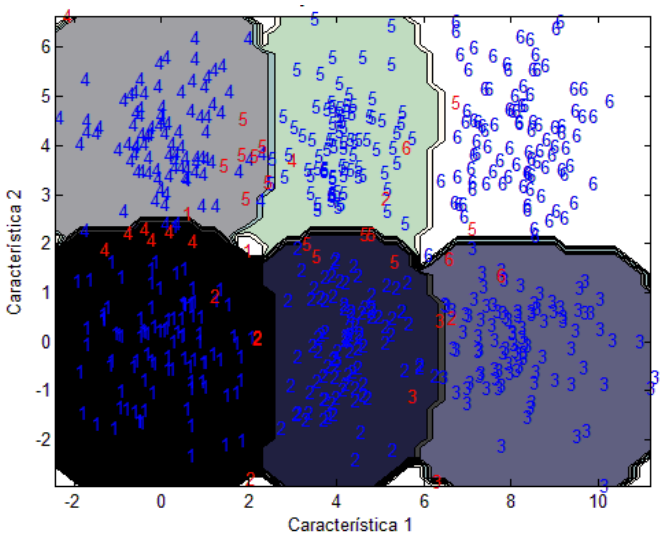

FIGURA 20 – RESULTADO FINAL DAS CLASSES PARA O MÉTODO UM-CONTRA-TODOS
FONTE: O autor (2014)

3.4.2 Método “todos-contra-todos”

Este método, proposto por Friedman (1996), faz uso de $N \cdot (N-1)/2$ SVMs de classificação binária. Cada uma dessas SVMs faz a classificação entre duas das classes presentes no problema. A (FIGURA 22) mostra os resultados das SVMs geradas para o problema exemplo – neste caso foram utilizadas apenas 4 classes, pois a utilização das 6 classes apresentadas anteriormente geraria 15 SVMs, o que dificultaria a exibição dos resultados.

Para a classificação de um novo padrão, o mesmo passa por todas as SVMs e em cada uma delas receberá um “voto” para classificá-lo a uma das duas classes. Ao final, ele será considerado como pertencente da classe que recebeu mais “votos” durante o processo. A (FIGURA 21) mostra o resultado final obtido para o problema teste (os elementos em vermelho representam valores classificados erroneamente).

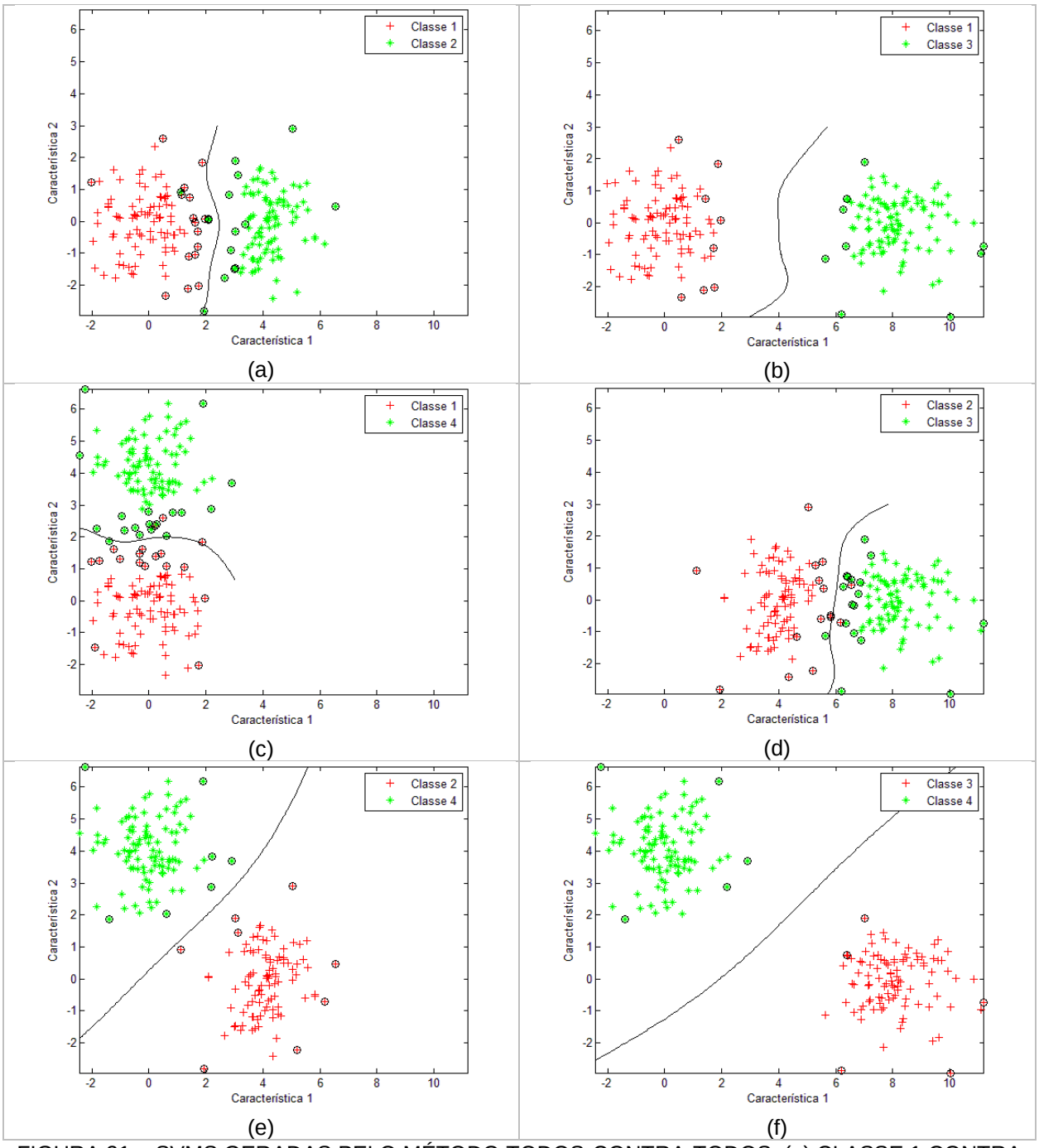

FIGURA 21 – SVMS GERADAS PELO MÉTODO TODOS-CONTRA-TODOS. (a) CLASSE 1 CONTRA CLASSE 2. (b) CLASSE 1 CONTRA CLASSE 3. (c) CLASSE 1 CONTRA CLASSE 4. (d) CLASSE 2 CONTRA CLASSE 3. (e) CLASSE 2 CONTRA CLASSE 4 (f) CLASSE 3 CONTRA CLASSE 4
FONTE: O autor (2014)

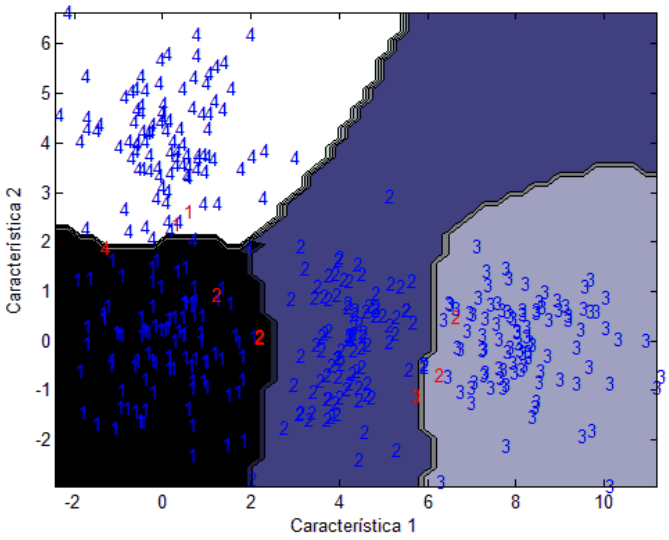

FIGURA 22 – RESULTADO FINAL DAS CLASSES PARA O MÉTODO TODOS-CONTRA-TODOS
FONTE: O autor (2014)

A maior fraqueza do método é o número de SVMs. As SVMs utilizam, neste caso, apenas um subconjunto da população de treinamento para definição dos vetores de suporte, o que acelera o processo de treinamento das mesmas. Em compensação, sabendo que para cada SVM existem parâmetros para seres definidos, a quantidade elevada de SVMs pode tornar esse processo altamente complexo.

3.4.3 Método de agrupamento por similaridade

Este é o método novo proposto neste trabalho. Sendo fora do conhecimento do autor a formulação deste método ou outro método similar na literatura. O método de agrupamento por similaridade consiste no agrupamento de classes de forma sistemática, usando como referência para o agrupamento a maior similaridade entre duas classes – sendo tal avaliação feita através da distância euclidiana entre representantes destas classes. Neste método são utilizadas $N - 1$ SVMs, sendo N o número de classes do problema.

O processo se inicia com a definição do representante de cada uma das classes, que é feito através do cálculo do centro de massa de todos os padrões de cada classe. Em seguida buscam-se os pares de classes que tem seus representantes mais próximos.

Cada um desses agrupamentos de duas classes gera uma SVM, que será responsável pela classificação dos padrões dessas classes.

As classes pertencentes ao mesmo grupo se tornam agora uma classe e o processo se reinicia. A única diferença é que, ao invés de calcular novos representantes, os representantes das classes que foram agrupadas são mantidos e a distância entre as novas classes é a média das distâncias de todos os representantes de uma classe para todos os da outra. Esse processo é exemplificado no problema teste da (FIGURA 24).

Para a classificação de um novo padrão, segue-se a avaliação contrária do método de treinamento, que para o problema teste, se chega às classes apresentadas na (FIGURA 25). Para o problema teste, a árvore da (FIGURA 23) seria utilizada na criação e execução da SVM.

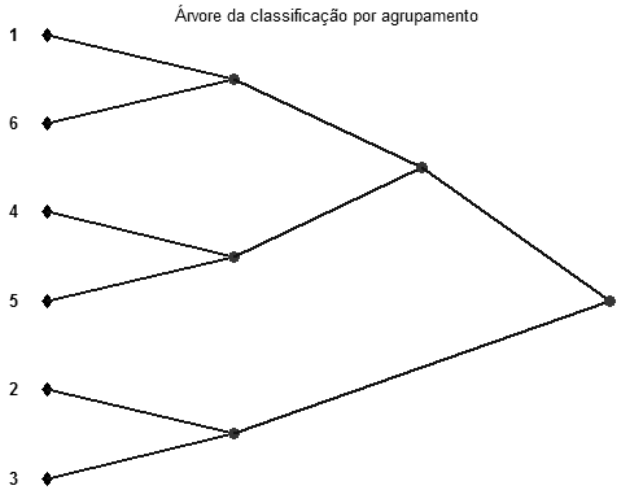

FIGURA 23 – ÁRVORE GERADA PARA O PROBLEMA TESTE
FONTE: O autor (2014)

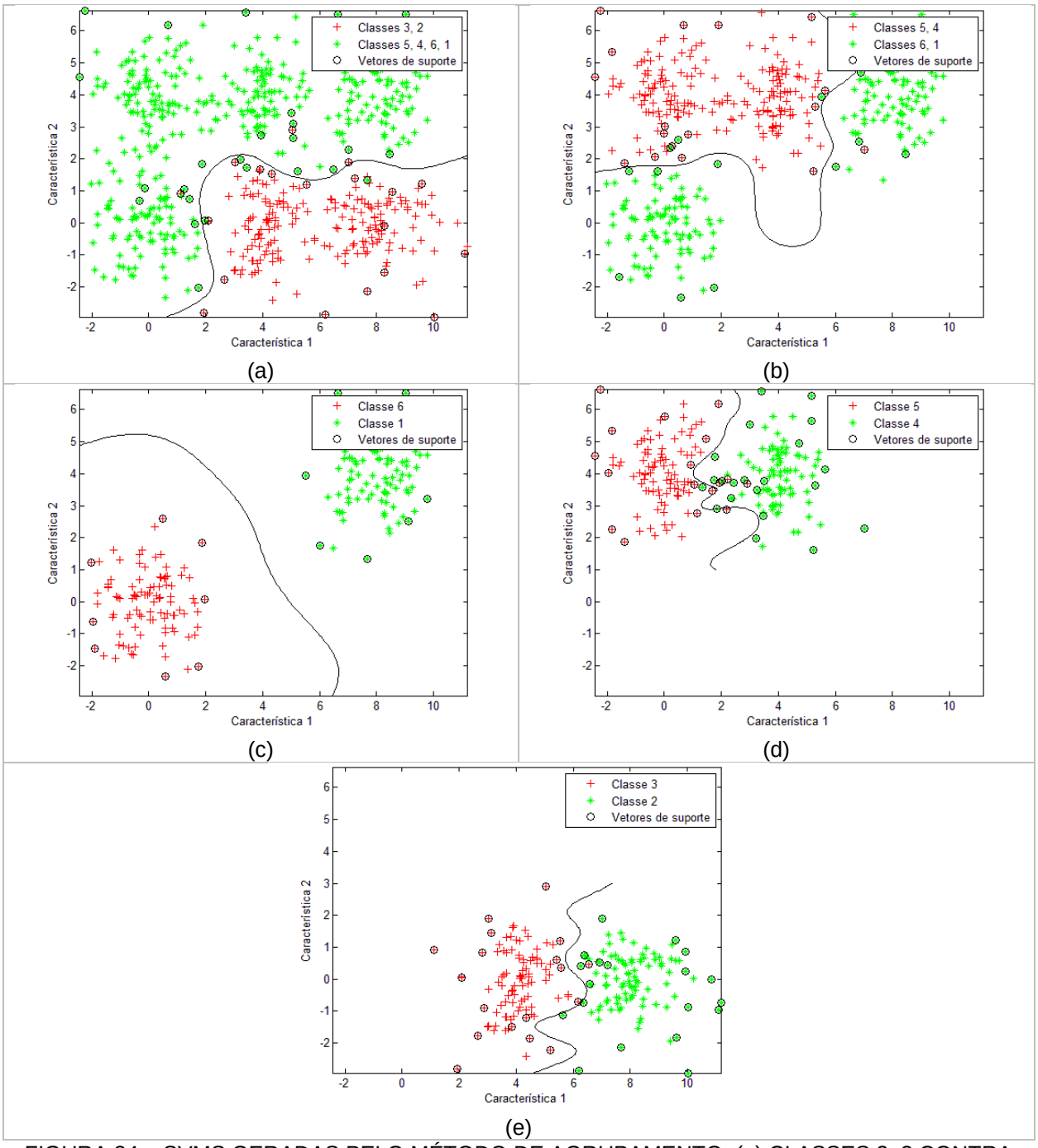

FIGURA 24 – SVMS GERADAS PELO MÉTODO DE AGRUPAMENTO. (a) CLASSES 2, 3 CONTRA CLASSES 4, 5, 6, 1 (b) CLASSES 5, 4 CONTRA CLASSES 6, 1 (c) CLASSE 6 CONTRA CLASSE 1 (d) CLASSE 5 CONTRA CLASSE 4 (e) CLASSE 3 CONTRA CLASSE 2
FONTE: O autor (2014)

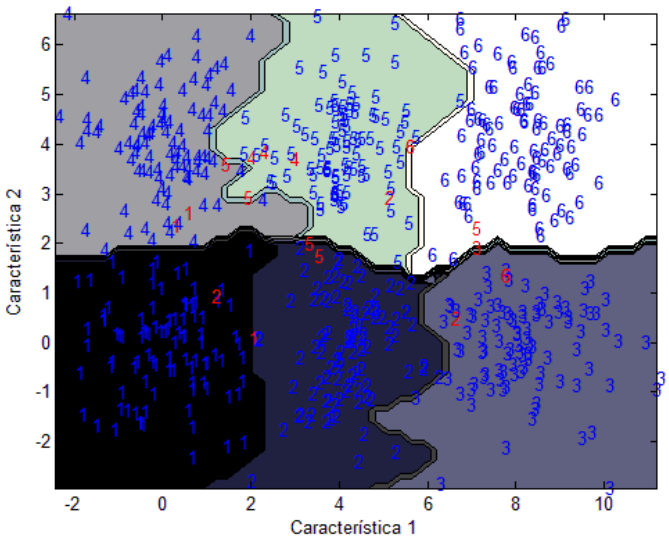

FIGURA 25 – RESULTADO FINAL DAS CLASSES PARA O MÉTODO DE AGRUPAMENTO
FONTE: O autor (2014)

Uma vez que com este método não é necessária a avaliação de todas as SVMs para classificar um novo padrão, o número máximo de SVMs a serem avaliadas é $\lceil \log(N) \rceil$ SVMs, em que $\lceil x \rceil$ é o teto de x – ou seja, o menor inteiro maior que x.

Ainda sugere-se como sistema de agrupamento, ao invés do centro de massa das características de cada classe, a utilização de um determinado número de padrões obtidos de forma aleatória de cada uma das classes ou a utilização do conhecimento prévio do usuário quanto a similaridades naturais das classes do problema. Nota-se que, apesar deste método adicionar a necessidade do cálculo das distâncias euclidianas, ele usa um número menor de SVMs do que os outros métodos.

Ainda pode-se averiguar que todas, exceto uma das SVMs criadas, utilizam um subconjunto da população de treinamento, reduzindo a complexidade e tempo computacional para treinamento das SVMs. Outra vantagem observada é que com este método, ao contrário dos outros apresentados, o número de vetores de suporte não é a soma de número de vetores de suporte das diversas máquinas criadas e sim a soma do número de vetores de suporte do galho com maior número de vetores de suporte – apresentando dessa forma, potencial para reduzir a complexidade da máquina final.

4 COMPUTAÇÃO EVOLUTIVA

Os cientistas e engenheiros de todas as áreas frequentemente tem que resolver problemas de busca e otimização, que significa encontrar a solução com melhor resultado ao problema, respeitando certas restrições e flexibilidades do mesmo.

Quando se fala de otimização de sistemas, é buscado o melhor conjunto de valores para os parâmetros do sistema com os quais seu desempenho será o melhor possível seguindo certas condições dadas.

Os parâmetros de performance do sistema são geralmente representados por um vetor como $\vec{X} = [x_1, x_2, x_3, \dots, x_D]$, no qual *D* é a dimensão do problema. Para parâmetros reais, como o próprio nome implica, cada parâmetro x_i é um valor real. Para mensurar a qualidade de uma determinada solução, uma função objetivo é elaborada e relacionada com o sistema em questão. Desta forma, a tarefa de otimização é encontrar a melhor solução $\overrightarrow{X^*}$, a qual minimiza a função $f(\vec{X})(f : \Omega \subseteq \Re^D \rightarrow \Re)$, desta forma $f(\overrightarrow{X^*}) < f(\vec{X})$ para todo $\vec{X} \in \Omega$, no qual Ω é um conjunto não vazio e extenso de *D* dimensões de valores servindo como domínio da busca. Quando trata-se de um problema sem restrições $\Omega = \Re^D$. E sabendo que $\text{máximo}\{f(\vec{X})\} = -mínimo\{f(\vec{X})\}$, garante-se a generalização do uso da função de minimização para qualquer problema (HUANG, 2006).

Em geral, problemas reais de engenharia são considerados de alta complexidade, pois possuem múltiplos mínimos locais impossibilitando o uso de técnicas de otimização tradicional, que não são capazes convergir para o mínimo global quando encontram um mínimo local no caminho. Um mínimo local $f_l = f(\vec{X}_l)$ pode ser definido como $\exists \varepsilon > 0 \; \forall \vec{X} \in \Omega: \|\vec{X} - \vec{X}_l\| < \varepsilon \Rightarrow f_l < f(\vec{X})$, em que $\|\cdot\|$ indica qualquer medida de distância entre os vetores (DAS; SUGANTHAN, 2010), como é mostrado na (FIGURA 26).

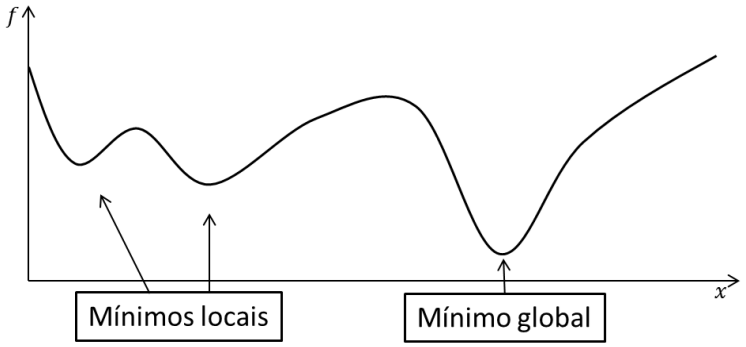

FIGURA 26 – EXEMPLO DE MÍNIMOS LOCAIS E MÍNIMO GLOBAL
FONTE: O autor (2014)

Bengoetxea (2002) mostra que as técnicas de otimização podem ser classificadas primeiramente em completas ou heurísticas. Técnicas completas examinam todo o espaço de busca para encontrar o melhor resultado, fazendo com que seu custo computacional para problemas reais seja inviável. As técnicas heurísticas, em contrapartida, se concentram apenas em parte desse espaço de buscas seguindo um determinado algoritmo.

Essas estratégias heurísticas são então dividas em determinísticas e não-determinísticas. As técnicas determinísticas são aquelas que, sob as mesmas condições, vão sempre obter o mesmo resultado final quando executadas; estas técnicas, geralmente, tendem a ficar presas em mínimos locais. Já as técnicas não determinísticas evitam esta situação pelo uso de componentes aleatórios, o que pode causar a obtenção de diferentes soluções para diferentes execuções mesmo quando sob as mesmas condições (BENGOETXEA, 2002).

Finalmente, os algoritmos heurísticos e não-determinísticos podem ser divididos em duas classes, os algoritmos não populacionais e os populacionais. Os não populacionais, por exemplo, o arrefecimento simulado (do inglês, *simulated annealing)*, possuem apenas uma solução por iteração e a mesma é modificada a fim de alcançar o melhor resultado. Já os algoritmos populacionais usam um conjunto de soluções, chamadas de indivíduos, que juntos buscam dentro do espaço de buscas a melhor solução para o problema. Duas classes dos algoritmos populacionais são utilizadas neste trabalho: os algoritmos evolutivos e os algoritmos de inteligência de partículas.

Cada uma das duas classes de algoritmos populacionais possui vantagens e desvantagens, ficando fora no escopo deste projeto tal estudo. Em contrapartida, a utilização de algoritmos das duas classes é feita para avaliar essas caraterísticas no problema estudado. Os algoritmos que são utilizados neste trabalho são a evolução diferencial (DE, do inglês *differential evolution*), Colônia Artificial de Abelhas (ABC, do inglês *Artificial Bee Colony*), Otimização por Enxame de Partículas (PSO, do inglês *Particle Swarm Optimization*) e Algoritmo de Busca por Retropropagação (BSA, do inglês *Backtracking Search Algorithm*).

4.1 ALGORITMO DE EVOLUÇÃO DIFERENCIAL

A DE clássica transcorre através de quatro passos principais, que entram em ciclo como mostra a (FIGURA 27): inicialização dos vetores, mutação, cruzamento e seleção. Esses passos serão agora descritos detalhadamente.

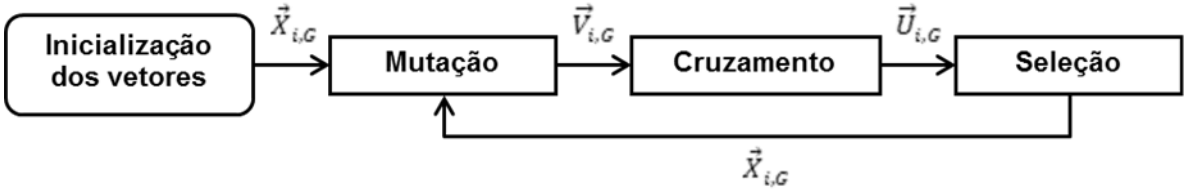

FIGURA 27 – PASSOS E FLUXO DO ALGORITMO DE EVOLUÇÃO DIFERENCIAL
FONTE: Adaptado de Das e Suganthan (2010)

A DE procura um ótimo global dentro de um espaço de buscas real de *D* dimensões $\Re^D$. Para inicializar sua busca uma população inicial aleatória é comumente usada. A população inicial é formada por TP (tamanho da população) vetores de *D* dimensões compostos de valores reais. Cada vetor, também conhecido como cromossomo ou genoma, forma uma solução candidata para o problema de otimização multidimensional. Sabendo que o DE é evoluído através de gerações, serão denotadas as gerações subsequentes por $G = 0,1, \dots, G_{max}$, sendo G_{max} o máximo de iterações permitida para o algoritmo, sendo este um valor de entrada da DE clássica.

Uma vez que os vetores devem mudar conforme as gerações são passadas, a notação apresentada na equação 29 é usada para fazer a distinção entre cada vetor de cada geração, tal que

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}] \quad (29)$$

sendo $\vec{X}_{i,G}$ o *i-ésimo* vetor da geração *G*.

Para cada um dos parâmetros do sistema, geralmente, há um intervalo de valores que este pode assumir. Esta restrição é devido ao fato de que os parâmetros são conectados a componentes ou medidas físicas e tem barreiras naturais (por exemplo, peso, massa, comprimento, etc. são medidas que não podem assumir valores negativos). Porém há casos em que este intervalo se limita a auxiliar na geração da população inicial, não sendo então uma restrição do parâmetro para as gerações subsequentes.

A população inicial, $G = 0$, deve cobrir o máximo possível do intervalo determinado. Para tanto é feita uma distribuição uniforme dos indivíduos dentro do espaço de buscas levando em consideração as restrições do sistema.

Os limites mínimos e máximos para cada parâmetro são definidos como: $\vec{X}_{min} = \{x_{1,min}, x_{2,min}, x_{3,min}, \dots, x_{D,min}\}$ e $\vec{X}_{max} = \{x_{1,max}, x_{2,max}, x_{3,max}, \dots, x_{D,max}\}$, respectivamente. Desta forma, conforme a equação 30 é feita a inicialização do *j-ésimo* componente do *i-ésimo* vetor.

$$x_{i,0} = x_{j,min} + rand(0,1) * \left(x_{j,max} - x_{j,\min}\right) \quad (30)$$

Sendo $rand(0,1)$ um valor aleatório com distribuição uniforme gerado para atender ao intervalo [0,1], independente para cada elemento de cada vetor.

A maior diferença entre o DE e os demais algoritmos evolutivos está exatamente na implementação de seu operador de mutação. A mutação no DE utiliza a diferença entre vetores existentes na população para determinar ambos a direção e o sentido da variação aplicada a um terceiro vetor. O módulo desta variação vai depender da

diferença entre os primeiros vetores e também de um parâmetro de entrada do algoritmo F, que é um fator de multiplicação tipicamente definido tal que $0.4 \leq F \leq 1$ (CHIANG; LEE; HEH, 2010). O vetor que é perturbado na operação de mutação é chamado de vetor alvo, a diferença obtida entre os outros vetores e usada para perturbar o vetor alvo é chamada de vetor doador e, finalmente, o resultado da combinação do vetor alvo e do vetor doador é o vetor experimental.

Diferentes métodos de mutação foram propostos pelos criadores do DE (PRICE, 1999; PRICE; STORN, 2005) e muitos outros podem ser encontrados na literatura atual. Para diferir entre estes diferentes esquemas uma metodologia de nomeação foi criada, sendo então generalizada pela forma *DE/x/y/z*, em que *DE* representa o algoritmo de evolução diferencial, *x* é o vetor alvo utilizado, *y* é o número de diferenças entre vetores usados na criação do vetor doador e, por último, *z* representa o tipo de cruzamento utilizado.

Cinco foram os métodos de mutação propostos por (PRICE; STORN, 2005) (note que na nomenclatura a última parte, que é referente ao método de cruzamento, é omitida, pois este será introduzido posteriormente e os esquemas de mutação apresentados são aplicáveis a todos):

$$\mathrm{DE/aleatório/1}: \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F * (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \tag{31}$$

$$\mathrm{DE/melhor/1}: \vec{V}_{i,G} = \vec{X}_{melhor,G} + F * (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{32}$$

$$\begin{aligned} \mathrm{DE/alvo-para-melhor/1}: \vec{V}_{i,G} \\ = \vec{X}_{i,G} + F * (\vec{X}_{melhor,G} - \vec{X}_{i,G}) + F * (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \end{aligned} \tag{33}$$

$$\mathrm{DE/melhor/2}: \vec{V}_{i,G} = \vec{X}_{melhor,G} + F * (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) + F * (\vec{X}_{r_3^i,G} - \vec{X}_{r_4^i,G}) \tag{34}$$

$$\mathrm{DE/aleatório/2}: \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F * (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) + F * (\vec{X}_{r_4^i,G} - \vec{X}_{r_5^i,G}) \tag{35}$$

Os índices $r_1^i, r_2^i,\ r_3^i,\ r_4^i$ e r_5^i são mutuamente exclusivos e inteiros, randomicamente selecionados, de forma que $1 \leq r_1^i, r_2^i,\ r_3^i,\ r_4^i$ e $r_5^i \leq TP$ e ainda todos devem ser diferentes do índice de base i. O vetor $\vec{X}_{melhor,G}$ é a solução que possui melhor resultado na função objetivo (ou seja, o menor quando for um problema de minimização) na população na geração G.

Pesquisas foram feitas por Mezura-Monte, Velazquez-Reyes e Coello Coello (2005) buscando definir quais métodos de mutação são mais eficientes. Eles testaram e compararam de forma empírica oito métodos de mutação diferentes (incluindo os cinco apresentados nas equações 31 a 35) em treze problemas de benchmark distintos. O experimento de Mezura-Monte, Velazquez-Reyes e Coello Coello (2005), indicou que o método $\mathrm{DE/melhor/1}$ foi o mais competitivo, não importando o tipo de problema, quando comparada a acurácia final e a robusteza do algoritmo.

Com intuito de aumentar o potencial de diversidade da população, um operador de cruzamento é utilizando após o processo de mutação. Na mutação o vetor doador troca componentes com o vetor alvo $\vec{X}_{i,G}$, formando o vetor experimental chamado $\vec{V}_{i,G} = \left[v_{1,i,G}, v_{2,i,G}, v_{3,i,G}, \ldots, v_{D,i,G}\right]$.

No cruzamento também existem o vetor alvo e o vetor doador, porém aqui os vetores alvos são os da população da geração atual, enquanto os vetores doadores são os vetores experimentais obtidos no processo de mutação anteriormente descrita.

Como mostrado por Price e Storn (2005), na família dos DEs dois tipos de cruzamento são utilizados: exponencial (ou modulo de dois pontos) e binomial (ou uniforme).

No cruzamento exponencial, primeiro é selecionado um número inteiro n de forma aleatória, de modo que $1 \leq n \leq D$, que será o ponto de início das trocas de componentes entre o vetor alvo e o vetor doador. Então, outro número inteiro L, tal que $1 \leq L \leq D$. O valor L denota o número de componentes do vetor doador que de fato contribuirão com o vetor alvo e é definido conforme o pseudocódigo:

$L = 0;$

$Faça\ \{$

$\quad L = L + 1;$

$\}\ Enquanto\big((rand(0,1) \leq Cr) E\ (L \leq D)\big).$

sendo $rand(0,1)$ um valor aleatório com distribuição uniforme, tal que $0 \leq rand(0,1) \leq 1$, Cr é a probabilidade de crossover que aparece como um parâmetro do DE assim como F, de tal forma que $0 \leq Cr \leq 1$.

Com n e L definidos, a equação 36 mostra como o novo vetor experimental é obtido.

$$u_{j,i,G} = v_{j,i,G} \text{ para } j = n, n+1, \dots, \text{mínimo}(n + L - 1, D)$$
$$u_{j,i,G} = x_{j,i,G} \text{ para todos os outros } j \tag{36}$$

Para cada vetor doador, um novo conjunto de n e L deve ser calculado como mostrado acima.

Em contra partida, o cruzamento binomial é executado em cada uma das D variáveis sempre que um valor aleatoriamente gerado entre 0 e 1 for menor ou igual a C_r. Assim, os parâmetros oriundos do vetor doador apresentam uma distribuição similar à binomial. A equação 37 mostra como o novo vetor experimental é gerado.

$$f(x) = \begin{cases} v_{j,i,G}, & \text{se } (rand(0,1) \leq C_r \text{ ou } j = j_{rand}) \\ x_{j,i,G}, & \text{caso contrário} \end{cases} \tag{37}$$

Como anteriormente, $rand(0,1)$ um valor aleatório com distribuição uniforme, tal que $0 \leq rand(0,1) \leq 1$, que é recalculado para cada *j-ésimo* componente do *i-ésimo* vetor de parâmetros. O $j_{rand} \in [1,2,3, \dots, D]$ é escolhido randomicamente, garantindo assim que pelo menos um componente de $\vec{V}_{i,G}$ vai estar presente em $\vec{U}_{i,G}$, e é reiniciado para cada diferente vetor doador.

Pode-se observar que em um espaço de buscas de 2 dimensões, três possíveis vetores experimentais podem resultar do cruzamento do vetor doador com o vetor alvo. Esses vetores experimentais são:

1) $\vec{U}_{i,G} = \vec{V}_{i,G}$, tal que ambos os componentes de $\vec{U}_{i,G}$ são provenientes de $\vec{V}_{i,G}$.
2) $\overrightarrow{U'}_{i,G}$, no qual o primeiro componente $(j = 1)$ vem de $\vec{V}_{i,G}$ e o segundo de $(j = 2)$ de $\vec{X}_{i,G}$.
3) $\overrightarrow{U''}_{i,G}$, sendo o primeiro componente $(j = 1)$ de $\vec{X}_{i,G}$ e $(j = 2)$ de $\vec{V}_{i,G}$.

Para manter o tamanho da população constante nas gerações ulteriores, o último passo do algoritmo chamado seleção para determinar se vetor alvo ou vetor experimental vai sobreviver para a próxima geração, ou seja, em $G = G + 1$. A equação 38 descreve a operação de seleção, tal que

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} \; se \; f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} \; se \; f(\vec{U}_{i,G}) > f(\vec{X}_{i,G}) \end{cases} \tag{38}$$

em que $\vec{U}_{i,G}$ é o vetor experimental, $\vec{X}_{i,G}$ é a população atual e $f(\vec{X})$ sendo a função objetivo a ser minimizada.

Assim, se o novo vetor experimental traz uma função objetivo menor ou igual que a do vetor alvo, ele substituirá o vetor alvo na próxima geração; caso contrário, o vetor alvo é mantido. Dessa forma, a nova população ou irá melhorar (com respeito à função objetivo a ser minimizada) ou continuará com a mesma qualidade, nunca piorando.

Pode-se ver na equação 38 que o vetor alvo é substituído pelo vetor experimental mesmo que os dois possuam mesmo valor de função objetivo – característica essa que possibilita os vetores do DE a moverem-se mesmo em funções com perfil planar através das gerações.

Colocando então todos os passos juntos, é possível elaborar um pseudocódigo da DE clássica que é mostrado no (QUADRO 2).

Existem três principais parâmetros para serem controlados no DE: o fator de escala do processo de mutação F, a probabilidade de cruzamento C_r e o tamanho da

população TP. Alguns estudos foram feitos relacionados aos parâmetros da DE a fim de melhorar o seu desempenho. Storn e Price (1995), indicaram que um valor razoável para TP pode ser escolhido entre $5 * D$ e $10 * D$ (sendo D o número de dimensões do problema a ser tratado). Eles também apontam que um bom ponto de partida para F é 0.5 e seu intervalo será, normalmente, entre $0{,}4$ e 1.

INÍCIO: Algoritmo DE	
1.	Define todos os valores de controle do DE: fator de escala F, probabilidade de cruzamento C_r e tamanho da população TP.
2	Faz o contador de gerações $G = 0$ e inicializa a população de TP indivíduos $P_G = \{\vec{X}_{1,G}, \vec{X}_{2,G}, \vec{X}_{3,G}, \dots, \vec{X}_{TP,G}\}$ com $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}]$ e cada indivíduo distribuído no intervalo $[\vec{X}_{min}, \vec{X}_{max}]$, sabendo que $\vec{X}_{min} = \{x_{1,min}, x_{2,min}, x_{3,min}, \dots, x_{D,min}\}$ e $\vec{X}_{max} = \{x_{1,max}, x_{2,max}, x_{3,max}, \dots, x_{D,max}\}$ com $i = [1,2,3, \dots, TP]$.
2.	Para G = 1 até Máximo de Ciclos
3.	Para i = 1 até *TP*
	Gera um vetor experimental da mutação $\vec{V}_{i,G} = [v_{1,i,G}, v_{2,i,G}, v_{3,i,G}, \dots, v_{D,i,G}]$ correspondente ao *i-ésimo* vetor alvo $\vec{X}_{i,G}$ através de um dos métodos de mutação apresentados
	Cria um vetor experimental do cruzamento $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, \dots, u_{D,i,G}]$ para o *i-ésimo* vetor alvo $\vec{X}_{i,G}$:
	Executa o cruzamento
	Avalia os novos indivíduos
	Compara os resultados dos novos indivíduos com os usados para sua criação e mantém-se o melhor entre eles
4.	Fim para
12.	Fim para
FIM	

QUADRO 2 – PSEUDOCÓDIGO DE CLÁSSICA
FONTE: O autor (2014)

O parâmetro C_r controla quantos parâmetros são esperados para serem alterados em um membro da população. Para baixos valores poucos parâmetros serão alterados em cada geração, o que causa um movimento ortogonal dos elementos com relação aos eixos coordenados. Em contrapartida, altos valores de C_r (próximos de 1) causam variações em todas as direções, impossibilitando passos ortogonais aos eixos. A (FIGURA 28), mostrada por Das e Suganthan (2010), apresenta o resultado das variações de C_r (com $C_r = 0$, $C_r = 0{,}5$ e $C_r = 1$) com $TP = 10$ e um número de gerações igual a 200, sem usar o processo de seleção.

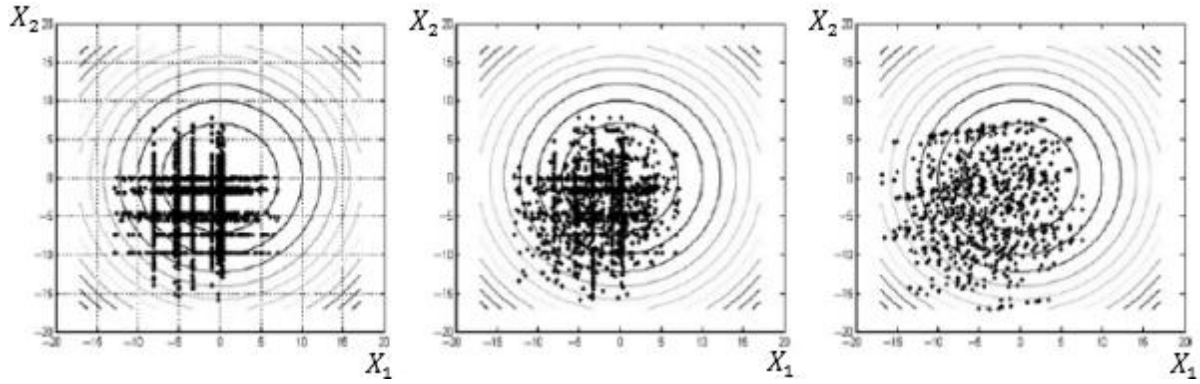

FIGURA 28 – INDIVÍDUOS PARA DIFERENTES C_r(DA ESQUERDA PARA A DIREITA $C_r = 0$, $C_r = 0{,}5$ E $C_r = 1$)
FONTE: Das e Suganthan (2010)

Também foram feitas avaliações dos diversos grupos de parâmetros nas funções de Rosenbrock e Rastrigin. Seus resultados mostraram que a capacidade de convergência ao ótimo global e a velocidade com que essa é executada é muito sensível a escolha dos parâmetros TP, F e C_r. Ele ainda diz que uma escolha plausível de TP é entre $3 * D$ e $8 * D$, o fator de escala da mutação F de $0{,}6$ e a probabilidade de mutação C_r entre $0{,}3$ e $0{,}9$ inclusive. Já Ronkkonen, Kukkonen e Price (2005) mostram que tipicamente $0{,}4 < F < 0{,}95$, sendo que $F = 0{,}9$ é uma boa primeira escolha. Eles também dizem que $0 < C_r < 0{,}2$ quando a função é separável e $0{,}9 < C_r < 1$ quando os parâmetros da função são dependentes.

Das e Suganthan (2010), comentam que, na literatura, muitas diferentes formas de controle dos parâmetros do DE são apresentados e isto pode dificultar a escolha da DE para resolução de problemas reais. Além disso, a maioria desses valores propostos não possuem justificativas experimentais suficientes para serem aceitas como verdade.

4.2 ALGORITMO DE EVOLUÇÃO DIFERENCIAL ADAPTATIVA

Zhang e Sanderson (2009) propõem uma modificação da DE em relação ao método de mutação e também sugere um modelo de adaptação dos parâmetros da DE, chamada de evolução diferencial adaptativa (JADE, do inglês *Adaptive Differential Evolution).*

No contexto de mutação, em (ZHANG; SANDERSON, 2009) é proposto o método DE/alvo para p-melhores, que é definido conforme a equação 39, tal que,

$$v_{i,g} = x_{i,g} + F_i * \left[(x^p_{melhor,g} - x_{i,g}) + (x_{r_1,g} - x_{r_2,g}) \right] \quad (39)$$

em que $x^p_{melhor,g}$ é escolhido aleatoriamente entre as $p\%$ melhores soluções da população atual. O valor de F_i no algoritmo JADE é recriado a cada iteração, conforme um processo de adaptação que será explicado posteriormente. Esta proposta de mutação é vista como uma generalização do método DE/alvo para melhor/1 anteriormente apresentado, porém os resultados obtidos são superiores devido ao aumento da capacidade de exploração do algoritmo, como é demonstrado por Zhang e Sanderson (2009).

A cada iteração no processo de otimização, os parâmetros F e CR são gerados para cada um dos membros da população.

O valor de CR segue uma distribuição normal, com média μ_{CR} e desvio padrão 0,1. O valor de μ_{CR} é calculado através da média aritmética dos CRs que obtiveram resultados bem sucedidos (ou seja, o vetor após o cruzamento foi melhor do que o vetor que o gerou), como é definido na equação 40, assim como

$$\mu_{CR} = (1 - c) * \mu_{CR} + c * média(S_{CR}) \quad (40)$$

em que c é uma constante positiva entre 0 e 1, S_{CR} é o conjunto de CRs bem sucedidos. O valor de μ_{CR} é inicializado em 0,5. Após a geração dos novos valores de CR, eles são truncados para valores entre 0 e 1.

O valor de F segue uma distribuição de Cauchy, com parâmetro de locação igual μ_F e escala igual a 0,1. O valor de F também é gerado independentemente para cada membro da população e os membros bem sucedidos são guardados em S_F. A atualização de μ_F é feita conforme a equação 41, tal que,

$$\mu_F = (1 - c) * \mu_F + c * médiaL(S_F) \quad (41)$$

em que $médiaL$ é a média de Lehmer, que é definida como:

$$médiaL(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (42)$$

Ao final da geração dos valores de F, os valores são truncados em 1, para valores maiores que 1, ou são recriados, para valores menores que 0.

Zhang e Sanderson (2009) explicam com detalhes a razão de utilizar diferentes sistemas de distribuição e média para cada um dos parâmetros. O pseudocódigo de JADE, que apresenta em detalhes como o algoritmo é desenvolvido, é mostrado no (QUADRO 3).

```
INÍCIO: Algoritmo JADE
Inicializa a população X_0 e define os valores μ_CR = 0,5 e μ_F = 0,5
Para t = 1 até Máximo de Iterações
    Gera CR_i = randn_i(μ_CR; 0,1), para todo i = 1, 2, 3, ..., TP
    Gera 1/3 de F_i = rand_i(0; 1,2) e os demais F_i = randn(μ_F; 0,1) para i = 1, 2, 3, ..., TP
    S_F = ∅ e S_CR = ∅
    Para i = 1 até TP
        Escolhe aleatoriamente x^P_melhor,t entre os P% melhores candidatos
        Escolhe aleatoriamente r_1 ≠ r_2 ≠ i do conjunto {1, 2, 3, ..., TP}
        v_i,t = x_i,t + F_i * (x^P_melhor,t − x_i,t) + F_i * (x_r1,t − x_r2,t)
        Gera j_rand = randint_i(1; D)
        Para j = 1 até D
            Se j = j_rand e rand_j(0; 1) ≤ CR_i então
                u_j,i,t = v_j,i,t
            Senão
                u_j,i,t = x_j,i,t
            Fim se
        Fim para
        Se f(x_i,t) ≤ f(u_i,t) então
            x_i,t+1 = x_i,t
        Senão
            x_i,t+1 = u_i,t; S_F = S_F ∪ F_i; S_CR = S_CR ∪ CR_i
        Fim se
    Fim para
    μ_F = (1 − c) * μ_F + c * médiaL(S_F)
    μ_CR = (1 − c) * μ_CR + c * media(S_CR)
Fim para
FIM
```

QUADRO 3 – PSEUDOCÓDIGO JADE
FONTE: O autor (2014)

4.3 ALGORITMO DE EVOLUÇÃO DIFERENCIAL COMPOSTA

Wang, Cai e Zhang (2011) mostram que a maioria dos algoritmos DE propostos utilizam apenas um método de mutação e um conjunto de parâmetros. Os autores ainda defendem que esta estratégia reduz a habilidade geral de busca do algoritmo. Quando existe maior conhecimento prévio do problema a ser resolvido e investigações sobre o mesmo foram feitas, a seleção de apenas um valor é interessante. Entretanto, para grande parte das abordagens, em que não se tem informação prévia sobre o problema, estas estratégias se tornam menos eficazes que a utilização de diversos métodos e conjuntos de parâmetros.

Em consideração a estas afirmações, os autores propuseram um algoritmo chamado de Evolução Diferencial Composta (CODE, do inglês *Composite Differential Evolution*), que tem como ideia principal a seleção aleatória do método de mutação e dos parâmetros utilizados nas operações do algoritmo.

A cada iteração do algoritmo, para cada um dos indivíduos, são selecionados de forma aleatória e independente do grupo de estratégias candidatas – composta por três métodos de mutação (aleatório/1/bin, aleatório/2/bin e atual-para-aleatório/1) – e do grupo de parâmetros candidatos – composto por três conjuntos de configurações ([$F = 1{,}0$; $CR = 0{,}1$], [$F = 1{,}0$; $CR = 0{,}9$] e [$F = 0{,}8$; $CR = 0{,}2$]) – os valores a serem utilizados. Estas estratégias e valores de parâmetros são, segundos os autores, comumente utilizados na solução de diferentes problemas apresentados na literatura. Wang, Cai e Zhang (2011) defendem que estas modificações no algoritmo DE deixam o mesmo mais robusto, capaz de obter bons resultados independente de conhecimento prévio do problema.

O (QUADRO 4) mostra o pseudocódigo para a implementação do algoritmo CODE.

INÍCIO: Algoritmo CODE

1. Define o grupo de estratégias (GE): {“aleatório/1/bin”, “aleatório/2/bin”, “atual-para-aleatório/1”} em que NGE é o número de estratégias no grupo
2. Define o grupo de parâmetros (GP): {(F = 1,0; CR = 0,1); (F = 1,0; CR = 0,9); (F = 0,8; CR = 0,2)} em que NGP é o número de pares de parâmetros no grupo
3. Inicializa a população X_0
4. Para t = 1 até Máximo de Iterações
5. Para i = 1 até TP
6. Escolhe aleatoriamente $r_1 \neq r_2 \neq i$ do conjunto {1, 2, 3, ..., TP}
7. Para *k* = 1 até o NGE
8. F_k e CR_k são definidos através da seleção aleatória de um par de parâmetro do GP
9. $u_{i,t_k} =$ Mutação e cruzamento de $x_{i,t}$ conforme a k-ésima estratégia no GE
10. Fim para
11. $u_{i,t}$ é definido como o melhor valor entre u_{i,t_k} para *k*={1, 2, 3, ..., NGE} com relação a função objetivo
12. Se $f(x_{i,t}) \leq f(u_{i,t})$ então
13. $x_{i,t+1} = x_{i,t}$
14. Senão
15. $x_{i,t+1} = u_{i,t}$
16. Fim se
17. Fim para
18. Fim para

FIM

QUADRO 4 – PSEUDOCÓDIGO CODE
FONTE: O autor (2014)

4.4 ALGORITMO DE COLÔNIA ARTIFICIAL DE ABELHAS

O algoritmo de Colônia Artificial de Abelhas (ABC, do inglês *Artificial Bee Colony*) foi proposto por Karaboga (2005) e usa o conceito da inteligência coletiva das abelhas, mais especificamente sua forma de interação na busca por fontes de comida. Karaboga (2005) coloca que existem três componentes essenciais sobre as abelhas que devem ser citados: as fontes de alimento, as abelhas operárias e as abelhas não-operárias, tal que:

- As fontes de alimentos são a principal procura das abelhas. Porém, dentre as diversas fontes existentes as abelhas se concentram naquelas que tem maior rentabilidade, ou seja, a mais rica em energia e com maior facilidade para extração do alimento;

- As abelhas operárias são relacionadas com uma fonte de alimento específica na qual estão associadas naquele momento. Elas guardam as informações da sua fonte de alimento em específica, tais como a rentabilidade, direção e distância da colmeia. Essa informação é compartilhada com o resto da colônia com certa frequência;
- As abelhas não-operárias podem ser classificadas em dois novos grupos. As abelhas exploradoras, que fazem buscas aleatórias em torno da colmeia em busca de novas fontes de alimentos. Quando estas abelhas exploradoras encontram uma nova fonte de alimento satisfatória, elas se tornam operárias da respectiva fonte de alimento encontrada. O segundo grupo são as abelhas observadoras, que permanecem na colmeia e analisam a informação compartilhada pelas abelhas operárias sobre as fontes de comida. Ao identificar uma região proveitosa, a mesma vai à busca de comida e se torna uma abelha operária da fonte de alimento escolhida.

A troca de informações sobre as fontes de alimentos é um dos pontos mais importantes da inteligência de enxame das abelhas. Essa troca de informação é feita através de uma dança em forma de zig-zag que as abelhas fazem com relação ao seu conhecimento sobre a fonte de alimento. A dança aponta a direção da fonte de alimento com relação à colmeia e a distância da mesma, tal que quanto mais longa for a dança, mais distante está a fonte de alimento, como mostrado na (FIGURA 29). A probabilidade de uma abelha fazer a dança é maior conforme a maior qualidade da sua fonte de alimento.

A dança é feita em um local específico, chamado de área de dança, onde também estão as abelhas observadoras, que obtém as informações com relação às fontes de alimentos disponíveis. Uma vez que a probabilidade de uma abelha fazer a dança aumenta conforme a qualidade da fonte de alimento, as chances de as abelhas observadoras verem estas danças também são maiores, o que faz com que áreas com melhores fontes de alimento sejam mais exploradas.

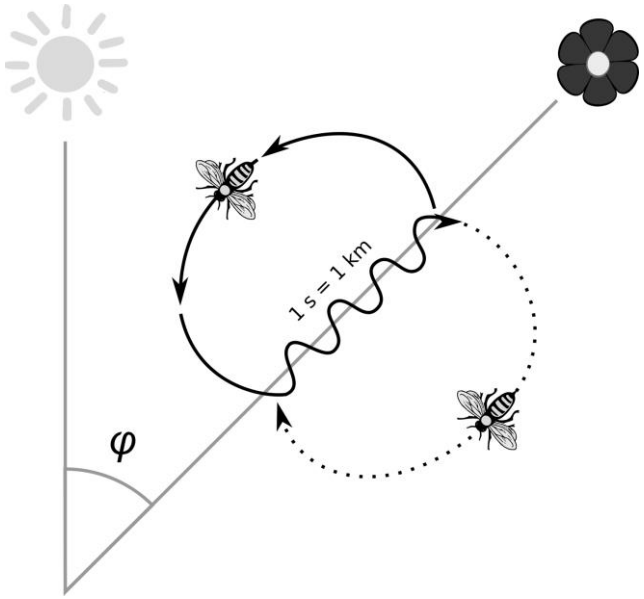

FIGURA 29 – DANÇA DAS ABELHAS USADA PARA COMUNICAÇÃO SOBRE QUALIDADE E POSIÇÃO DAS FONTES DE ALIMENTO
FONTE: Adaptado de Karaboga (2005)

O algoritmo ABC utiliza esses conceitos para buscar a solução ótima para problemas de otimização de múltiplas dimensões. No algoritmo, como descrito por Karaboga (2005), são utilizados os três tipos de abelhas: operárias, observadoras e exploradoras.

O algoritmo é iniciado com todas as abelhas como exploradoras, para uma primeira busca aleatória no espaço de buscas, dessa forma, são encontradas as primeiras fontes de alimento. A partir das próximas iterações, as abelhas se dividem – metade delas se torna operárias e a outra metade será de observadoras. Cada fonte de comida tem a si relacionada apenas uma abelha operária – ou seja, o número de abelhas operárias é igual ao número de fontes de alimento ao entorno da colmeia.

A cada ciclo, as abelhas operárias trazem uma quantidade de néctar da fonte de alimento, que representa a qualidade da solução. Essa quantidade de néctar é utilizada para calcular a probabilidade de uma abelha observadora utilizar a informação daquela fonte de alimento ou não.

Quando a região de uma fonte de alimento tem seu néctar esgotado, ou seja, após algumas visitas a fonte de alimento, a abelha operária não consegue trazer néctar, a abelha operária se desvencilha desta fonte de alimento e se torna uma exploradora,

para encontrar uma nova fonte de alimento ainda não explorada. O pseudocódigo apresentado no (QUADRO 5) mostra o procedimento para a otimização utilizada pelo algoritmo ABC.

INÍCIO: Algoritmo ABC	
1.	Inicializa as fontes de alimento X_0 com valores aleatórios sobre o espaço de buscas Avalia a população de fontes de alimentos e atribui os valores a f
2.	Para Ciclo = 1 até Máximo de Ciclos
3.	Gera nova solução (posição da fonte de comida) $v_{i,j}$ na vizinhança de $x_{i,j}$ para as abelhas operárias usando a formula $v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j})$ – em que k é uma solução na vizinhança de i e ϕ é um valor aleatório entre -1 e 1.
4.	Avalia os membros de v
5.	Compara os pares v e x e mantenha os com melhores resultados
6.	Calcula os valores de probabilidades P_i, $P_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i}$, para cada solução x_i através dos seus valores de *fitness*, que podem ser calculados tal que: $fitness_i = \begin{cases} 1 + abs(f_i), & f_i < 0 \\ \frac{1}{1+f_i}, & f_i \geq 0 \end{cases}$ – o valor de P_i é normalizado entre 0 e 1
7.	Seleciona as abelhas observadoras em x de acordo com a probabilidade P para produzir novas soluções (ou seja, novas posições de comida) v como no passo 3.
8.	Avalia os membros de v
9.	Compara os pares v e x e mantenha os com melhores resultados
10.	Determina as fontes de alimento para abandonar, que são aquelas que não melhoraram seus resultados em um determinado número de ciclos, e cria novas soluções aleatórias para substituí-las
11.	Armazena a melhor fonte de alimento encontrada até então
12.	Fim para
FIM	

QUADRO 5 – PSEUDOCÓDIGO ABC
FONTE: O autor (2014)

4.5 OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS COM APRENDIZAGEM ABRANGENTE

A otimização por enxame de partículas (PSO, do inglês *Particle Swarm Optimization*) é um algoritmo evolucionário baseado na inteligência de enxames. O primeiro PSO, proposto por Kennedy e Eberhart (1995), foi desenvolvido com objetivo de simular a procura de comida feita pelos pássaros, que encontram comida pela cooperação com outros pássaros em sua vizinhança. Simplisticamente, pode-se

observar que ao redor de fontes de alimentos migalhas do mesmo estão presentes. Conforme se afasta da fonte, essas migalhas vão se tornando menores. Os pássaros fazem uso dessa informação e se comunicam de tal forma que referenciam seu movimento em direção ao pássaro que encontrou o maior pedaço de comida para continuarem sua busca por alimento.

No algoritmo PSO os pássaros passam a ser chamados de partículas. Cada uma dessas partículas é avaliada conforme a sua qualidade, que é definida como *fitness*. As partículas se movimentam no espaço de buscas, onde sua velocidade e posição são atualizadas pelo algoritmo de forma iterativa para busca do ponto ótimo. As equações de cálculo de velocidade e de posição são as partes principais do algoritmo e são originalmente definidas como nas equações 43 e 44, tal que,

$$V_{i,d} = V_{i,d} + c_1 * rand_{i,d} * \left(melhorP_{i,d} - x_{i,d}\right) + c_2 * rand_{i,d} * \left(melhorG_d - x_{i,d}\right) \quad (43)$$

$$x_{i,d} = x_{i,d} + V_{i,d} \quad (44)$$

onde i assume valores de um até o número total de partículas e d de um ao número de dimensões do problema a ser tratado; c_1 e c_2 são as constantes de aceleração, que junto com o valor de $rand$ (que representa um valor aleatório entre zero e um), representam uma aceleração estocástica que levam a partícula em direção de $melhorP$ e $melhorG$; $melhorP_i$ representa a posição com maior qualidade que a partícula i já esteve e $melhorG$ é a melhor posição que qualquer partícula já ocupou; x_i é um vetor do tamanho do número de dimensões do problema representando a posição da $iésima$ partícula no espaço de buscas; V_i é um vetor da mesma dimensão de x_i com as velocidades no mesmo em cada uma das dimensões do problema.

As equações 43 e 44 são executadas em cada uma das iterações, de forma que a atualização da velocidade e posição das partículas é feita de forma suave, evitando convergência precoce das partículas em direção àquela com maior qualidade. A equação 43 pode ser dividida em três componentes principais, como mostrado por Shi e Eberhart (1991). A primeira parte da equação tem relação ao momento, em que a velocidade atual

da partícula é introduzida a fim evitar a convergência precoce e troca abrupta de posição e velocidade como discutido anteriormente. A segunda parte tem relação à cognição, em que cada partícula usa seu próprio conhecimento adquirido durante o processo de otimização para contribuir na definição da sua posição. Finalmente, a terceira parte da equação é chamada de social, em que o conhecimento adquirido por todo o grupo, no caso a melhor posição já encontrada por qualquer partícula, é usada para guiar o cálculo da nova velocidade da partícula em questão.

A velocidade das partículas é limitada a um valor máximo e mínimo, $+V_{máx}$ e $-V_{máx}$, respectivamente. A definição do valor $V_{máx}$ é importante, pois quando o mesmo é pequeno demais o algoritmo tende a ficar preso em mínimos locais, já quando é muito grande, as partículas podem passar por cima de regiões de boas soluções sem avaliá-las.

O algoritmo PSO é visto como de simples conceito, fácil implementação e computacionalmente eficiente. Como os outros algoritmos evolucionários, o PSO é baseado em população, a qual é inicializada aleatoriamente, e o comportamento das partículas depende da sua interação com os demais membros da população. Shi e Eberhart (2001) afirmam que o que diferencia a PSO dos demais algoritmos é o uso da memória de cada uma das partículas individualmente para o processo de otimização. Angeline (1998) analisa que, em comparação com os demais algoritmos evolucionários, a versão original do PSO é mais rápida na convergência inicial da população, porém mais lento no refinamento da mesma.

Buscando melhorar o refinamento e evitar o problema convergência para mínimos locais, como mostrado por Liang *et al.* (2006), uma variante do algoritmo PSO, chamada Otimização por Enxame de Partículas com Aprendizagem Abrangente (CLPSO, do inglês *Comprehensive Learning Particle Swarm Optimization*) foi proposto por Liang *et al.* (2006) e apresentou resultados superiores as demais versões do algoritmo. O CLPSO faz uso de uma nova estratégia para atualização da velocidade das partículas, como na equação 45.

$$V_{i,d} = \omega \times V_{i,d} + c \times rand \times \left(melhorP_{I_{i,d},d} - x_{i,d}\right) \tag{45}$$

Nesta estratégia, um valor de inércia ω é utilizado, o qual é atualizado a cada iteração; o valor de $rand$ é obtido através de uma distribuição aleatória e uniforme entre zero e um; c é uma constante definida pelo usuário. Nesta estratégia, utiliza-se tanto a melhor posição visitada pela própria partícula ou das outras partículas da população. As outras partículas que serão utilizadas são escolhidas através de um torneio, ou seja, pegam-se duas partículas aleatórias da população e usa-se a melhor posição já encontrada pela partícula que naquele ponto apresentou maior qualidade. Este torneio é feito para cada uma das dimensões do problema. A decisão se para cada determinada dimensão será utilizado o ganhador do torneio ou a própria partícula em questão é feito de forma aleatória, conforme a variável de probabilidade de aprendizado, chamada Pc, que é definida conforme a equação 46, em que tp é o tamanho da população, tal que,

$$Pc_i = 0{,}5 * \frac{\exp\left(\frac{10 * (i-1)}{tp-1}\right) - 1}{\exp(10) - 1} \tag{46}$$

O processo de otimização do algoritmo CLPSO pode ser visto no pseudocódigo apresentado no (QUADRO 6). Seus resultados, como reportados por Liang *et al.* (2006) são significativamente melhores que os do PSO original e das variações apresentadas na literatura.

4.6 ALGORITMO DE BUSCAS POR RETROPROPAGAÇÃO

O algoritmo de buscas por retropropagação (BSA, do inglês *Backtracking Search Algorithm*) usa a filosofia do retorno de um grupo social de criaturas vivas em intervalos aleatórios a áreas de caça que foram anteriormente descobertas como satisfatórias fontes de alimento, como mostrado por Civicioglu (2013). Pouco é descrito pelo autor sobre como cada etapa do processo de evolução se relaciona com a inspiração biológica do mesmo, entretanto, os resultados obtidos são altamente competitivos quando comparados com os resultados dos algoritmos atuais da literatura.

INÍCIO: Algoritmo CLPSO

1. Inicializa a população de posições X_0 e suas respectivas velocidades V_0 aleatoriamente
2. Avalia população em f; $melhorFP$ (melhor da partícula) é definido como f e $melhorP = X_0$
3. Baseado em f defina $melhorFG$ (melhor global) e $melhorG$ sendo a melhor partícula em X_0
4. Define o valor de $ContadorSemMelhoras = 0$
5. Para t = 1 até Máximo de Iterações faça
6. Calcula valor de $\omega = \omega_0 \times \frac{(\omega_0 - \omega_1) \times t}{Máximo\ de\ Iterações}$, com $\omega_0 = 0{,}9$ e $\omega_1 = 0{,}4$
7. Para *i* = 1 até o Tamanho da População faça
8. Se $ContadorSemMelhoras_i \geq m$ (em que m é um valor definido pelo usuário) então
9. Para cada dimensão d
10. Se $rand(0;1) \leq Pc_i$ então
11. $I_1 = randint(1; Tamanho\ da\ população)$; $I_2 = randint(1; Tamanho\ da\ população)$
12. Se $f_{I_1} \leq f_{I_2}$ então
13. $I_{i,d} = I_1$
14. Senão
15. $I_{i,d} = I_2$
16. Fim se
17. Senão
18. $I_{i,d} = i$
19. Fim se
20. Fim para
21. $ContadorSemMelhoras_i = 0$
22. Fim se
23. Para cada dimensão d
24. $V_{i,d} = \omega \times V_{i,d} + c \times rand \times \left(melhorP_{I_{i,d},d} - x_{i,d}\right)$ - caso $V_{i,d}$ esteja fora do intervalo permitido, é atribuído ao mesmo a velocidade permitida mais próxima
25. $x_{i,d} = x_{i,d} + V_{i,d}$
26. Fim para
27. Calcula o valor de f_i para x_i
38. Se $f_i < melhorFP_i$ então
29. $melhorFP_i = f_i$; $melhorP_i = x_i$; $ContadorSemMelhoras_i = 0$
30. Se $f_i < melhorFG$ então
31. $melhorFG = f_i$; $melhorG = x_i$
32. Fim se
33. Senão
34. $ContadorSemMelhoras_i = ContadorSemMelhoras_i + 1$
35. Fim se
36. Fim para
37. Fim para

FIM

QUADRO 6 – PSEUDOCÓDIGO CLPSO
FONTE: O autor (2014)

O BSA possui cinco passos principais: inicialização, seleção I, mutação, cruzamento e seleção II. A inicialização é similar à maioria dos algoritmos evolutivos, é feita de forma aleatória e uniforme por todas as dimensões do problema.

A seleção I trabalha com a chamada população antiga, P_{antiga}, que inicialmente é definida de forma aleatória, como a população inicial. Em cada iteração do processo evolutivo, a população antiga tem 50% de chance de ser substituída pela população atual. E, ao final da seleção I, a população antiga é embaralhada de forma aleatória (em termos de seus indivíduos, não nas dimensões do problema). O processo da mutação utiliza a população atual e a antiga, fazendo uma combinação entre as duas, conforme a equação 47, em que *F* controla a amplitude da busca, de forma que

$$P_{mutante} = P + F * \left(P_{antiga} - P\right) \tag{47}$$

Os autores recomendam a utilização de $F = 3 * randn(0; 1)$, tal que $randn(a; b)$ é um valor aleatório regido por uma distribuição normal com média a e desvio padrão b. A utilização de P_{antiga} neste processo de mutação faz com que o algoritmo use, de forma parcial, as vantagens da experiência de gerações passadas.

O processo de cruzamento ocorre alterando a população mutante, $P_{mutante}$. Para esta execução, duas etapas são processadas. Primeiro gera-se uma matriz binária de mapeamento, $mapa$, que é inicializada com todos os valores iguais a 1. A alteração dos valores deste mapeamento pode ocorrer de duas formas, com 50% de chance para cada uma delas a cada iteração. A primeira faz uso de um parâmetro definido pelo usuário, $TaxaMistura$, em que para cada indivíduo da população um número igual a $\lceil TaxaMistura * rand(0; 1) * D \rceil$ escolhidos de forma aleatória são transformados para 0 ($\lceil \cdot \rceil$ é a função teto e D é o número de dimensões do problema). O segundo modo transforma apenas um dos valores de $mapa$ para cada indivíduo de 1 para 0, sendo a posição escolhida de forma aleatória. Com a matriz $mapa$ montada, $P_{mutante}$ pode ser modificada, de forma que em todos os valores de $P_{mutante}$ em que seu correspondente em $mapa$ for igual a 1, será substituído pelo valor correspondente na população atual P.

Finalmente, a seleção II substitui na população atual P os valores em que $P_{mutante}$ obteve melhores resultados que a população atual. Os melhores valores da

geração atual são salvos para apresentação. Todo esse processo é apresentado no pseudo-algoritmo do (QUADRO 7).

4.7 ALGORITMO HÍBRIDO DE EVOLUÇÃO DIFERENCIAL E RETROPROPAGAÇÃO ADAPTATIVAS

O algoritmo híbrido de evolução diferencial e retropropagação adaptativas (HEDRA) está sendo proposto nesse trabalho buscando mesclar, em uma única solução, algumas técnicas que têm obtido sucesso em diversas aplicações apresentadas na literatura. Este algoritmo utiliza as estratégias de atualização do algoritmo DE (como apresentado na seção 4.1) e do BSA (como apresentado na seção 4.6). Uma vantagem desta estratégia é a ausência de parâmetro a serem definidos pelo usuário. Através de estratégias de adaptação baseadas no algoritmo JADE (apresentado na seção 4.2).

A seleção do número de membros da população que será atualizada com uma ou outra estratégia é adaptativa e dinâmica, utilizando a eficiência de cada um dos métodos no decorrer das iterações. Dessa forma, quando uma técnica tem uma eficiência superior a outra para um determinado problema, o algoritmo é capaz de se adaptar e assim dar prioridade a essa abordagem – característica que também é favorável no processo de otimização de um problema único, uma vez que um algoritmo pode ter mais facilidade de encontrar soluções em uma região de busca que o outro.

A população é inicializada de forma aleatória, como as demais técnicas. Os valores médios dos parâmetros das duas técnicas (DE e BSA) são inicializados – esses valores médios serão atualizados no processo de otimização, se adequando a necessidade do problema. Ainda é definida a porcentagem de seleção que vai definir a proporção dos membros da população que é atribuída para cada uma das técnicas. Essa porcentagem é atualizada a cada iteração usando a memória de taxa de sucesso, que guarda a taxa de sucesso de cada um dos métodos para as últimas iterações (o tamanho da memória depende do número máximo de iterações, sendo esse valor definido em 5% do número máximo de iterações).

INÍCIO: Algoritmo BSA

1. Inicializa a população P e P_{antiga} com valores aleatórios
2. Avalia a população P e atribui a f
3. Para $ciclo$ = 1 até $Máximo\ de\ ciclos$
4. Gera $a = rand(0;1)$ e $b = rand(0;1)$
5. Se a>b então
6. $P_{antiga} = P$
7. Fim se
8. Embaralha a população P_{antiga} (processo de permutação)
9. $P_{mutante} = P + 3 * randn(0;1) * (P_{antiga} - P)$
10. Cria $Mapa$ de tamanho TP por D com todos os valores iguais a 1
11. Gera $c = rand(0;1)$ e $d = rand(0;1)$
12. Se c>d então
13. Para i = 1 até TP
14. $u = permutação([1:D])$
15. $mapa_{i,u_{1:\lfloor TaxaMistura*rand(0;1)*D \rfloor}} = 0$
16. Fim para
17. Senão
18. Para i = 1 até TP
19. $mapa_{i,randint(1;D)} = 0$
20. Fim para
21. Fim se
22. Para i = 1 até TP
23. Para j = 1 até D
24. Se $mapa_{i,j} = 1$ então
25. $P_{mutante\,i,j} = P_{i,j}$
26. Fim se
27. Fim para
28. Fim para
29. Corrige os valores de $P_{mutante}$ caso tenha algum valor fora dos limites permitidos
30. Avalia a população $P_{mutante}$ e atribui a $f_{mutante}$
31. Para os indivíduos da população em que $f_{mutante}$ é melhor que f, substitui-se os valores de f por $f_{mutante}$ e os valores de P por $P_{mutante}$
32. Fim para

FIM

QUADRO 7 – PSEUDOCÓDIGO BSA
FONTE: O autor (2014)

Dentro do processo de iteração e evolução do algoritmo, a cada iteração a população é dividida em duas subpopulações, conforme o valor da porcentagem de seleção, em que cada uma das técnicas de otimização será utilizada para evoluir os

membros de sua subpopulação. Depois de executados os passos de otimização, os novos membros são avaliados conforme através da função a ser minimizada. Com esta avaliação, são selecionados os valores que foram bem sucedidos, ou seja, que os novos indivíduos gerados são melhores do que os seus predecessores. Os valores dos parâmetros utilizados para gerar os indivíduos bem sucedidos serão utilizados para atualizar os valores médios que, por sua vez, serão utilizados para gerar os novos parâmetros para a próxima iteração.

Utilizando a taxa de indivíduos bem sucedidos nas últimas iterações (5% do número máximo de iterações), a porcentagem de membros da população que será utilizado em cada um dos métodos é atualizada. O (QUADRO 8) apresenta o processo de evolução do algoritmo HEDRA.

INÍCIO: Algoritmo HEDRA

1. Inicializa a população X_0
2. Define os valores $\mu_{CR_{DE}} = 0{,}5$, $\mu_{F_{DE}} = 0{,}5$, $\mu_{F_{BSA}} = 3$ e $\mu_{M_{BSA}} = 0{,}9$
3. Atribui a PS (porcentagem de seleção) 50% para cada algoritmo
4. Define a memória TS (taxa de sucesso) nula para ambos os algoritmos
5. **Para** t = 1 até Máximo de Iterações
6. Gera $CR_i = randn_i(\mu_{CR}; 0{,}1)$, para todo $i = 1, 2, 3, \ldots, TP$
7. Escolhe aleatoriamente $x^P_{melhor,t}$ entre os P% melhores candidatos (para DE)
8. Armazena população em P_{antiga} (para BSA)
9. Através da PS, define quais membros da população vão ser atualizados por cada um dos algoritmos (DE ou BSA)
10. $S_{F_{DE}} = \emptyset \; e S_{CR_{DE}} = \emptyset$
11. $S_{F_{BSA}} = \emptyset \; e S_{D_{BSA}} = \emptyset$
12. Para cada membro da população designado com o algoritmo DE
13. Gere $F_{DE} = randn(\mu_{F_{DE}}; 0{,}1)$ e $CR_{DE} = randn(\mu_{CR_{DE}}; 0{,}1)$
14. Execute os passos de atualização de indivíduos da DE
15. Fim para
16. Para cada membro da população designado com o algoritmo BSA
17. Gere $F_{BSA} = randn(\mu_{F_{BSA}}; 0{,}3)$ e $D_{BSA} = randn(\mu_{D_{BSA}}; 0{,}1)$
18. Execute os passos de atualização de indivíduos do BSA
19. Fim para
20. Avalia os novos membros gerados
21. Armazena os valores F_{DE} e CR_{DE} bem sucedidos de DE em $S_{F_{DE}}$ e $S_{CR_{DE}}$, respectivamente
22. Armazena os valores F_{BSA} e D_{BSA} bem sucedidos de BSA em $S_{F_{BSA}}$ e $S_{D_{BSA}}$, respectivamente
23. $\mu_{F_{DE}} = (1 - 0{,}1) * \mu_{F_{DE}} + 0{,}1 * L(S_{F_{DE}})$
24. $\mu_{F_{BSA}} = (1 - 0{,}1) * \mu_{F_{BSA}} + 0{,}1 * L(S_{F_{BSA}})$
25. $\mu_{CR_{DE}} = (1 - 0{,}1) * \mu_{CR_{DE}} + 0{,}1 * media(S_{CR_{DE}})$
26. $\mu_{D_{BSA}} = (1 - 0{,}1) * \mu_{D_{BSA}} + 0{,}1 * media(S_{D_{BSA}})$
27. Armazena em TS a taxa de sucesso de cada uma das técnicas (DE e BSA)
28. Se algum valor em TS é mais velho que 5% do Máximo de Iterações então
29. Apaga valores mais velhos que 5% do Máximo de Iterações
30. Fim se
31. Calcula novo valor de PS com relação à TS
32. Fim para

FIM

QUADRO 8 – PSEUDOCÓDIGO HEDRA
FONTE: O autor (2014)

5 MÉTODOS DE AVALIAÇÃO DOS ALGORITMOS

Derrac *et al.* (2011) mostram que devido à origem aleatória dos algoritmos evolucionários, é comum que na avaliação dos mesmos em relação às funções de interesse, estes sejam rodados diversas vezes, em que cada uma destas execuções é chamada de experimento, e ao fim obtém-se uma tabela com o valor médio, desvio padrão, valor máximo e valor mínimo obtido para cada uma das funções utilizando cada um dos algoritmos. Esta tabela é por fim utilizada para a comparação dos algoritmos e avaliação do desempenho de cada um.

A fim de fazer uma avaliação estatística mais aprofundada, também é utilizado o teste de Wilcoxon, que é um procedimento não paramétrico para aplicação em teste de hipótese envolvendo dois conjuntos de dados (DERRAC *et al.*, 2011), sendo a hipótese nula H_0, neste caso, que os conjuntos de dados não são significativamente diferentes. Este teste é capaz de definir se os dois conjuntos de entradas são estatisticamente diferentes um do outro, que no caso dos algoritmos evolucionários, o teste poderá mostrar, em comparações por pares, quais algoritmos tiveram melhor desempenho no conjunto de problemas proposto.

Usando como exemplo dois conjuntos de dados A e B com n elementos em cada. Define-se então d como a diferença ordenada entre os valores de A e B, tal que $d = A - B$, e d é ordenado de forma crescente, que são posteriormente ranqueados com valores de 1 até n. Finalmente, R^+ é definido como a soma dos ranques para os valores em que d é positivo e R^- como a soma dos ranques para os valores com d negativo, tal como mostrado nas equações 48 e 49.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{48}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{49}$$

Em que $i = 1, 2, 3, \dots, n$. No caso da comparação dos resultados dos algoritmos evolutivos, não busca-se somente avaliar se os conjuntos de dados A e B são significativamente diferentes e sim se um conjunto é superior ao outro. Desta forma, busca-se definir se o conjunto A é estatisticamente melhor que o conjunto B (ou seja, se a curva de resultados de A está deslocada para a esquerda da curva de B). Para tanto, é necessário comparar valor de R^+ encontrado na equação 48, se R^+ for menor do que um valor tabelado, como mostrado em (DERRAC *et al.*, 2011), então o conjunto de dados A é estatisticamente melhor que B – considerando problemas de minimização.

5.1 VALIDAÇÃO DOS ALGORITMOS EM FUNÇÕES DE TESTE

A fim de comparar os algoritmos apresentados, estes são aplicados em 8 problemas teste do conjunto apresentado e utilizado por Suganthan *et al.* (2005), distribuídos nas classes de problemas: unimodal, multimodal simples e multimodal rotacionado, conforme mostrado por Suganthan *et al.* (2005). Os problemas a serem tratados são apresentados no (QUADRO 9), em que D é a dimensão do problema.

Nota-se que as funções são dadas em z, porém a variável utilizada de entrada será dada por x, então é feito o deslocamento que todas as funções utilizam, dessa forma $z = x - bias$, em que *bias* são valores definidos a fim de mudar a posição da solução ótima, que na maioria desses problemas fica exatamente no meio do espaço de buscas e na mesma posição para todas as dimensões. Os valores do *bias* e maiores detalhes dessas funções são dados em Suganthan *et al.* (2005). Para a função *Weierstrass rotacionada*, a variável de entrada x, além de deslocada é rotacionada, tal que $z = x * M - bias$, em que a matriz de rotação M também é definida em Suganthan *et al.* (2005). A (FIGURA 30) e a (FIGURA 31) mostram o mapeamento das 8 funções apresentadas anteriormente para D igual a 2.

Função	Definição	Características	Espaço de buscas e valor ótimo
Esfera	$f(z) = \sum_{i=1}^{D} z_i^2$	• Unimodal • Deslocada • Separável • Escalável	$z \in [-100;\ 100]^D$ $f_{ótimo} = 0$
Schwefel	$f(z) = \sum_{i=1}^{D} \left(\sum_{i=1}^{D} z_j \right)^2$	• Unimodal • Deslocada • Não separável • Escalável	$z \in [-100;\ 100]^D$ $f_{ótimo} = 0$
Elíptica condicionada	$f(z) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} * z_i^2$	• Unimodal • Deslocada • Não separável • Escalável	$z \in [-100;\ 100]^D$ $f_{ótimo} = 0$
Rosenbrock	$f(z) = \sum_{i=1}^{D-1} (100 * (z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$	• Multimodal • Deslocada • Não separável • Escalável	$z \in [-100;\ 100]^D$ $f_{ótimo} = 0$
Rastrigin	$f(z) = \sum_{i=1}^{D} (z_i^2 - 10 * \cos(2\pi z_i) + 10)$	• Multimodal • Deslocada • Separável • Escalável	$z \in [-5;\ 5]^D$ $f_{ótimo} = 0$
Weierstrass rotacionada	$f(z) = \sum_{i=1}^{D} \left(\sum_{k=0}^{20} \left[0.5^{\mathrm{k}} \cos\left(2\pi 3^k (z_i + 0.5)\right) \right] \right) - D \sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k 0.5)]$	• Multimodal • Deslocada • Não separável • Escalável • Rotacionada	$z \in [-0{,}5;\ 0{,}5]^D$ $f_{ótimo} = 0$
Rastrigin rotacionada	$f(z) = \sum_{i=1}^{D} (z_i^2 - 10 * \cos(2\pi z_i) + 10)$	• Multimodal • Deslocada • Não separável • Escalável • Rotacionada	$z \in [-5;\ 5]^D$ $f_{ótimo} = 0$
Ackley	$f(z) = -20e^{-0{,}2 * \sqrt{\frac{1}{D} * \sum_{i=1}^{D} z_i^2}} - e^{\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi z_i)} + 20 + e$	• Multimodal • Rotacionada • Deslocada • Não separável • Escalável • Ótimo global no limite da função	$z \in [-40;\ 40]^D$ $f_{ótimo} = 0$

QUADRO 9 – DEFINIÇÃO DAS FUNÇÕES *BENCHMARK*
FONTE: O autor (2014)

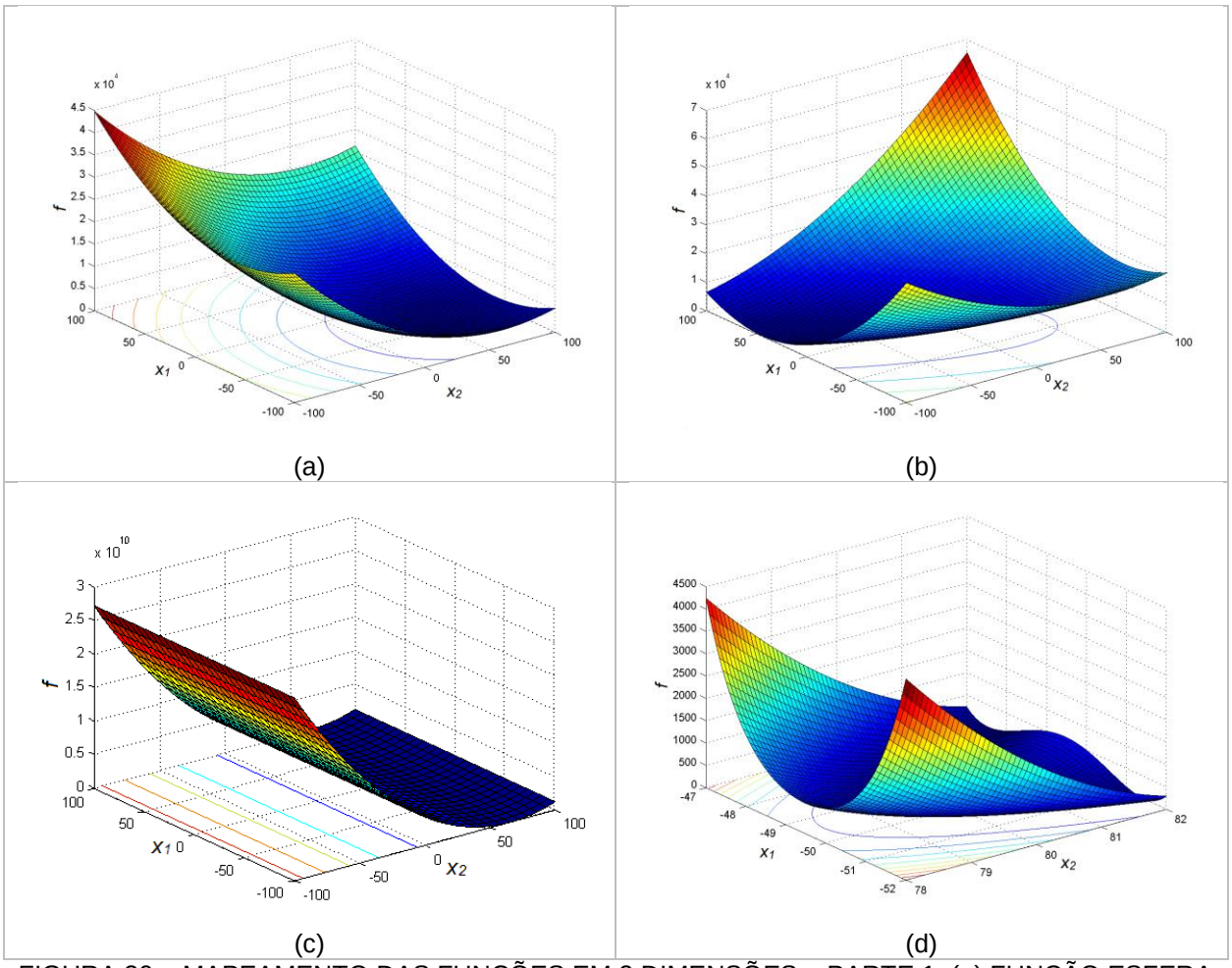

FIGURA 30 – MAPEAMENTO DAS FUNÇÕES EM 2 DIMENSÕES – PARTE 1. (a) FUNÇÃO ESFERA (b) FUNÇÃO SCHWEFEL (c) FUNÇÃO ELIPTICA CONDICIONADA (d) FUNÇÃO ROSENBROCK
FONTE: Suganthan *et al.* (2005)

Oito algoritmos, dentre os apresentados anteriormente, foram selecionados para utilização neste trabalho, sendo duas variantes da DE e uma do ABC, CLPSO, BSA, JADE, CODE e HEDRA.

Devido à origem heurística dos algoritmos, é necessária a execução de diversos experimentos a fim de obter dados estatísticos suficientes para comparação dos mesmos e verificação de sua convergência para os problemas utilizados. Desta forma, cada algoritmo foi executado 50 vezes para cada um dos problemas de benchmark. Para cada execução dos algoritmos foi definido um tamanho de população igual a 100 e máximo de iteração (ou gerações) igual a 1000. A fim de não privilegiar nenhum método, também foi definido um número máximo de execuções da função objetivo igual a 100.000. Todos os problemas de benchmark foram definidos para 30 dimensões.

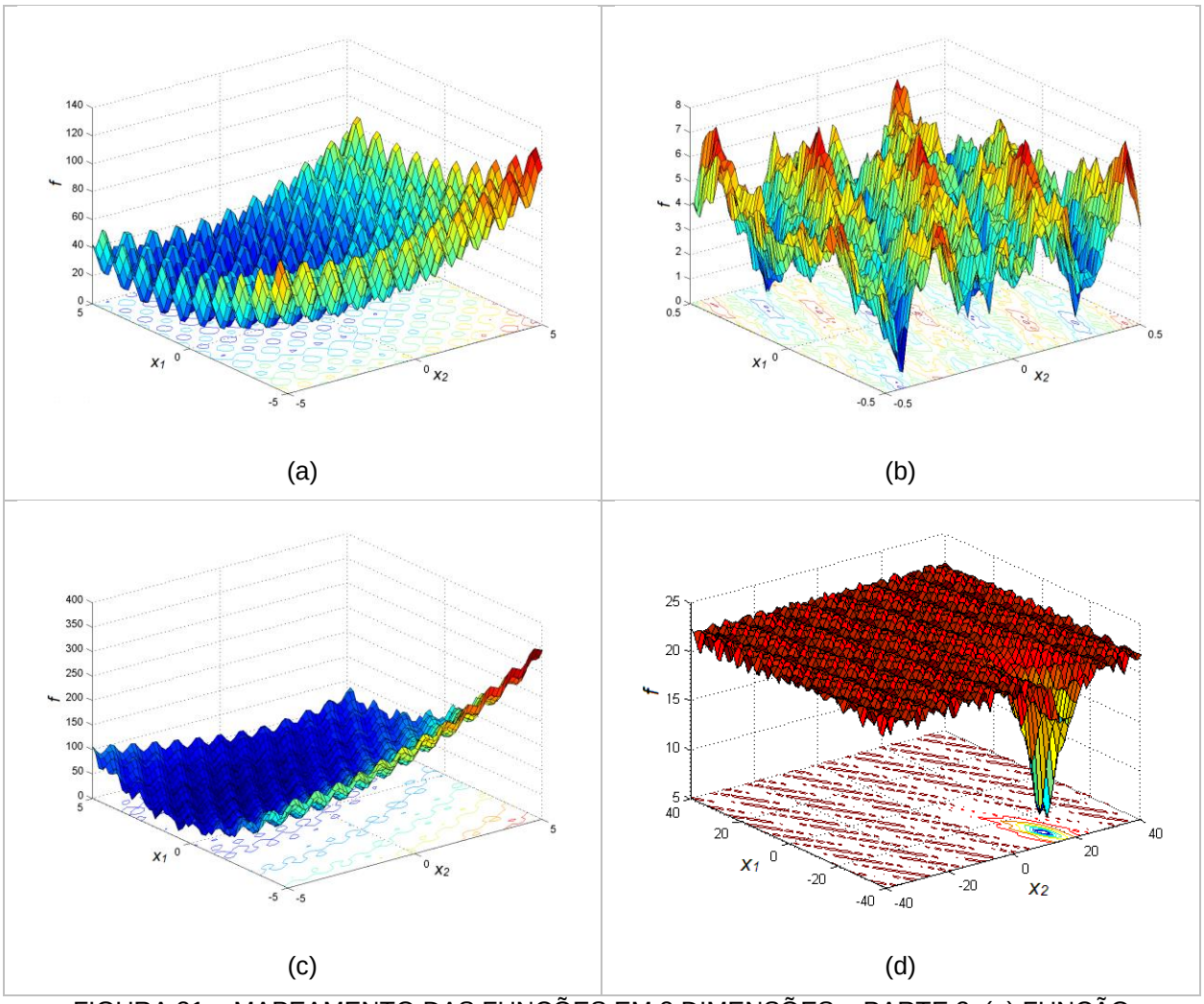

FIGURA 31 – MAPEAMENTO DAS FUNÇÕES EM 2 DIMENSÕES – PARTE 2. (a) FUNÇÃO RASTRIGIN (b) FUNÇÃO WEIERSTRASS ROTACIONADA (c) FUNÇÃO RASTRIGIN ROTACIONADA (d) FUNÇÃO ACKLEY
FONTE: Suganthan *et al.* (2005)

Cada algoritmo apresentado possui diferentes parâmetros a serem definidos, que podem ter grandes influências na capacidade de otimização de cada um. Sendo assim, são utilizados os valores sugeridos pelos autores dos mesmos, os quais, em geral, são obtidos de forma empírica tal que o algoritmo seja capaz de otimizar satisfatoriamente a maior diversidade de problemas possível.

A primeira variante da DE, que aqui é chamada de DE1, usa o método de mutação mais básica - $\mathrm{DE/aleatório/1}$. Conforme Price e Storn (2005), o valor de F utilizado é 0,5 e o de $\mathrm{C_r}$ é 0,9.

A segunda variante da DE, chamada de DE2, usa outro método de mutação bastante difundido, o $\mathrm{DE/melhor/1}$. Assim como na DE1, os valores de F e $\mathrm{C_r}$ são 0,5 e 0,9, respectivamente.

No caso do algoritmo ABC, o número de abelhas operárias adotadas é igual a metade da população total (ou seja, 50), as demais são abelhas não operárias. As fontes de alimentos são abandonadas quando após 10 iterações não há melhora no resultado da abelha que está explorando aquela fonte.

Para o CLPSO o valor de m foi definido como 10, a constante c como 1,49445 e os valores de ω_0 e ω_1 iguais a 0,4 e 0,9, respectivamente. A velocidade máxima das partículas foi definida como 20% da diferença entre o valor máximo e mínimo para cada dimensão (a velocidade mínima é igual à simétrica negativa da velocidade máxima).

Para o algoritmo BSA o único parâmetro a ser definido é a taxa de mistura, que foi definida conforme proposto por Civicioglu (2013) igual a 1.

Os algoritmos JADE, CODE e HEDRA não possuem parâmetros para serem definidos além daqueles valores já fixados no desenvolvimento dos algoritmos.

A (TABELA 1) mostra os resultados obtidos após os 50 experimentos para cada um dos algoritmos em cada uma das funções benchmark. Na tabela são apresentadas as médias aritméticas do melhor valor encontrado em cada experimento, assim como os valores máximos e mínimos e os desvios padrões. Os valores em negrito representam os melhores resultados alcançados para o problema em questão.

É possível notar pela (TABELA 1) que um algoritmo que obtém os melhores resultados para um problema, pode acabar sendo o de piores resultados para outro. Por exemplo, a DE1 alcançou um dos melhores resultados nas avaliações para a função *Schwefel*, em contrapartida, obteve a pior média e piores valores máximos e mínimos na função *Weierstrass rotacionada*. Mesmo quando se limita a um tipo de problema específico, por exemplos os unimodais (*Esfera* e *Schwefel*), nota-se que não é o mesmo algoritmo que obtém os melhores resultados – sendo neste caso os melhores resultados pertencentes ao JADE e ao HEDRA, respectivamente.

Buscando uma avaliação estatística mais aprofundada, foi aplicado o teste de Wilcoxon nos resultados obtidos. Sendo que para o caso de 8 funções, como executado nesta validação, o valor encontrado para R^+, calculado conforme a equação 48, deve ser

menor que 6 para que um algoritmo seja considerado estatisticamente melhor que o outro para o dado conjunto de problemas teste.

TABELA 1 – RESULTADOS OBTIDOS PARA OS PROBLEMAS TESTE DE OTIMIZAÇÃO

		DE1	DE2	ABC	CLPSO	BSA	JADE	CODE	HEDRA
Esfera	Média	6,16E-08	1,96E-03	1,06E-11	1,21E+00	4,89E-03	**0,00E+00**	3,36E-08	1,96E-29
	Mínimo	4,95E-09	5,72E-04	9,97E-13	6,19E-01	7,70E-04	**0,00E+00**	1,30E-08	**0,00E+00**
	Máximo	2,21E-07	5,63E-03	3,49E-11	1,95E+00	2,26E-02	**0,00E+00**	6,26E-08	4,99E-28
	Desv. Padrão	5,06E-08	1,09E-03	9,72E-12	3,22E-01	4,55E-03	**0,00E+00**	1,29E-08	7,70E-29
Schwefel	Média	1,94E+01	2,89E+02	1,61E+04	1,80E+04	9,92E+03	5,55E-07	3,05E-01	2,66E-08
	Mínimo	2,87E+00	1,50E+02	6,95E+03	1,22E+04	6,93E+03	4,93E-09	4,52E-02	3,37E-10
	Máximo	8,72E+01	5,43E+02	2,48E+04	2,31E+04	1,28E+04	5,33E-06	9,73E-01	1,81E-07
	Desv. Padrão	1,34E+01	1,02E+02	3,54E+03	2,27E+03	1,44E+03	1,05E-06	2,24E-01	3,90E-08
Elíptica condicionada	Média	7,12E-05	2,22E+00	2,99E-08	3,10E+03	1,02E+01	2,88E-24	2,76E-05	**0,00E+00**
	Mínimo	1,01E-05	5,43E-01	9,38E-10	1,60E+03	9,56E-01	**0,00E+00**	1,28E-05	**0,00E+00**
	Máximo	3,53E-04	7,76E+00	1,74E-07	5,16E+03	6,55E+01	7,94E-23	6,40E-05	**0,00E+00**
	Desv. Padrão	6,71E-05	1,36E+00	2,79E-08	8,67E+02	1,10E+01	1,17E-23	1,02E-05	**0,00E+00**
Rosenbrock	Média	3,79E+01	6,24E+01	1,21E+01	2,44E+04	2,27E+02	2,41E+01	2,69E+01	1,99E+01
	Mínimo	1,70E+01	2,68E+01	2,64E-01	8,27E+03	1,41E+02	2,25E-03	2,34E+01	6,20E+00
	Máximo	1,48E+02	3,71E+02	3,44E+01	4,07E+04	4,00E+02	1,18E+02	8,84E+01	1,19E+02
	Desv. Padrão	3,21E+01	6,94E+01	9,81E+00	7,55E+03	5,18E+01	3,68E+01	8,99E+00	2,32E+01
Rastrigin	Média	1,80E+02	1,85E+02	6,70E-02	1,48E+01	8,66E-05	2,68E+01	1,73E+01	2,28E-02
	Mínimo	1,49E+02	1,63E+02	1,11E-08	1,06E+01	2,54E-05	2,01E+01	1,40E+01	2,13E-03
	Máximo	2,03E+02	2,06E+02	9,98E-01	1,97E+01	2,44E-04	3,67E+01	2,00E+01	6,44E-02
	Desv. Padrão	1,32E+01	1,16E+01	2,39E-01	1,96E+00	4,38E-05	3,66E+00	1,50E+00	1,29E-02
Weierstrass rotacionada	Média	4,04E+01	4,03E+01	3,92E+01	3,21E+01	3,09E+01	2,77E+01	3,36E+01	2,91E+01
	Mínimo	3,75E+01	3,68E+01	3,61E+01	2,93E+01	2,86E+01	2,36E+01	2,94E+01	2,22E+01
	Máximo	4,22E+01	4,26E+01	4,13E+01	3,46E+01	3,31E+01	3,04E+01	3,63E+01	3,35E+01
	Desv. Padrão	1,06E+00	1,46E+00	1,30E+00	1,24E+00	1,14E+00	1,45E+00	1,59E+00	1,93E+00
Rastrigin rotacionada	Média	2,04E+02	2,12E+02	3,65E+02	2,19E+02	1,35E+02	5,41E+01	1,64E+02	6,34E+01
	Mínimo	1,78E+02	1,74E+02	1,87E+02	1,68E+02	1,09E+02	3,67E+01	1,23E+02	4,13E+01
	Máximo	2,25E+02	2,33E+02	4,32E+02	2,48E+02	1,75E+02	7,36E+01	1,85E+02	9,01E+01
	Desv. Padrão	1,26E+01	1,27E+01	4,98E+01	1,56E+01	1,60E+01	8,13E+00	1,43E+01	8,46E+00
Ackley	Média	2,10E+01	2,10E+01	2,10E+01	2,10E+01	2,10E+01	2,10E+01	2,09E+01	2,10E+01
	Mínimo	2,08E+01	2,08E+01	2,08E+01	2,08E+01	2,08E+01	2,06E+01	2,08E+01	2,09E+01
	Máximo	2,11E+01	2,11E+01	2,11E+01	2,11E+01	2,11E+01	2,11E+01	2,10E+01	2,11E+01
	Desv. Padrão	5,78E-02	5,64E-02	4,85E-02	6,74E-02	5,39E-02	8,22E-02	5,35E-02	4,51E-02

FONTE: O autor (2014)

A (TABELA 2) mostra o teste de Wilcoxon nos algoritmos testados. Para cada célula da tabela deve ser executado o teste sobre o algoritmo da linha contra o da coluna. Ou seja, se para uma determinada célula o valor apresentado é menor do que 6, significa

que o algoritmo referente àquela linha é estatisticamente melhor que o algoritmo referente àquela coluna. Em todas as células que apresentam tais valores estão preenchidas com cinza escuro. Já as células preenchidas com o cinza claro indicam os algoritmos da linha que foram superiores aos da coluna, porém não o suficiente para serem classificados como superiores pelo teste de Wilcoxon.

TABELA 2 – TESTE DE WILCOXON PARA OS PROBLEMAS TESTE DE OTIMIZAÇÃO

	DE1	DE2	ABC	CLPSO	BSA	JADE	CODE	HEDRA
DE1	=	4	22	8	16	36	36	33
DE2	32	=	22	9	18	36	36	35
ABC	14	14	=	8	24	25	22	28
CLPSO	28	27	28	=	36	32	29	36
BSA	20	18	12	0	=	30	22	31
JADE	0	0	11	4	6	=	10	19
CODE	0	0	14	7	14	26	=	33
HEDRA	3	1	8	0	5	17	3	=

FONTE: O autor (2014)

Através da (TABELA 2) pode se observar que nenhum dos algoritmos foi estatisticamente superior a todos os demais, entretanto é possível notar que os algoritmos JADE, CODE e HEDRA obtiveram resultados superiores para este conjunto de problemas.

As (FIGURA 32) e (FIGURA 33) ainda apresentam a média dos melhores valores obtidos para cada função através dos experimentos. Mais uma vez, não é possível obter um padrão que determine se algum dos algoritmos tem convergência mais rápida ou não comparada aos demais. Ao contrário, o formato da curva de otimização dos algoritmos é bastante similar.

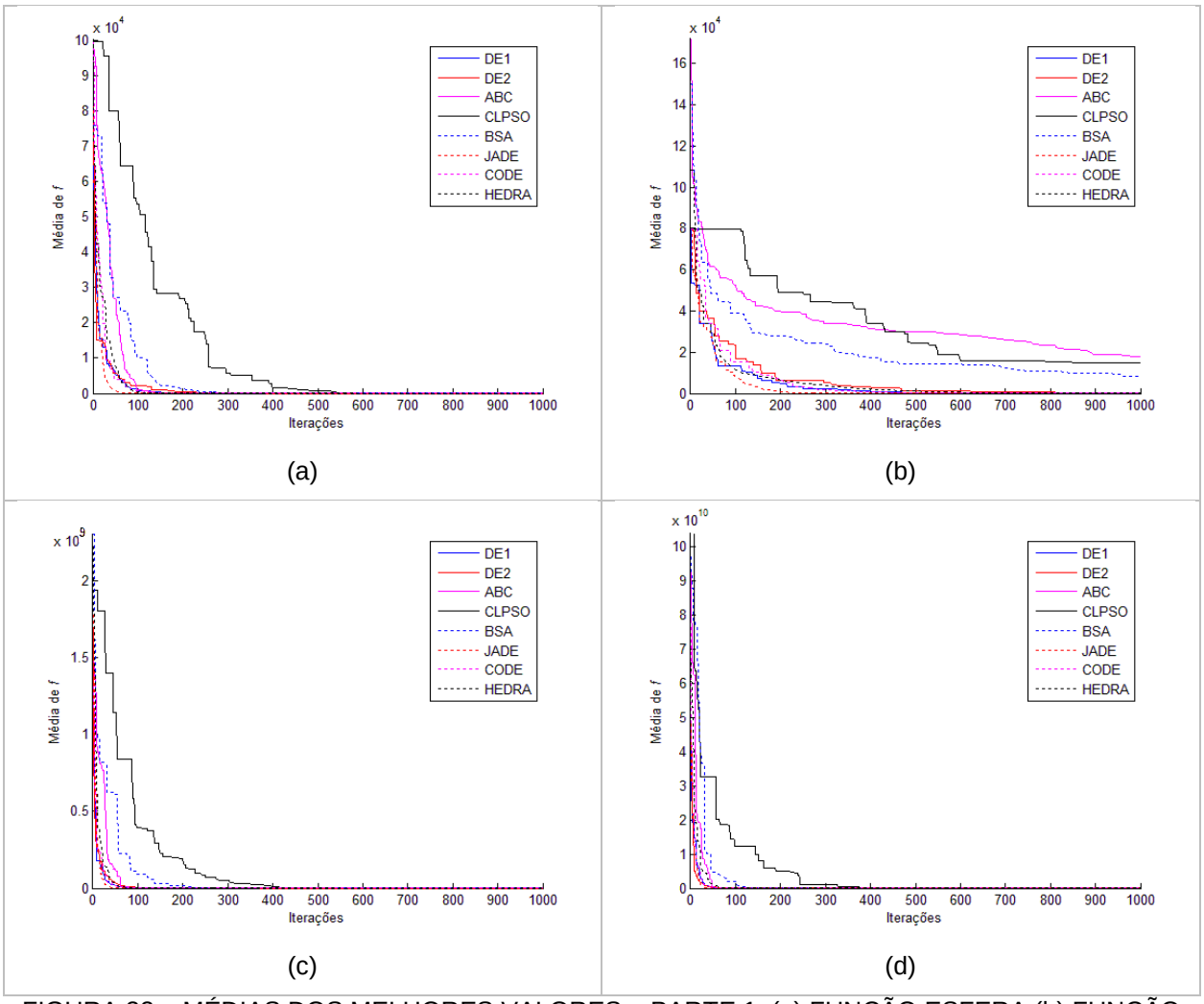

FIGURA 32 – MÉDIAS DOS MELHORES VALORES – PARTE 1. (a) FUNÇÃO ESFERA (b) FUNÇÃO SCHWEFEL (c) FUNÇÃO ELIPTICA CONDICIONADA (d) FUNÇÃO ROSENBROCK
FONTE: O autor (2014)

Este tipo de resultado mostra a dificuldade da escolha do algoritmo a ser utilizado em problemas reais sem a prévia avaliação das opções disponíveis para o problema a ser tratado especificamente. Sendo assim, e adicionando que todos os algoritmos obtiveram resultados satisfatórios para os problemas propostos – de alta complexidade e grande número de variáveis de otimização – todos eles são utilizados neste trabalho.

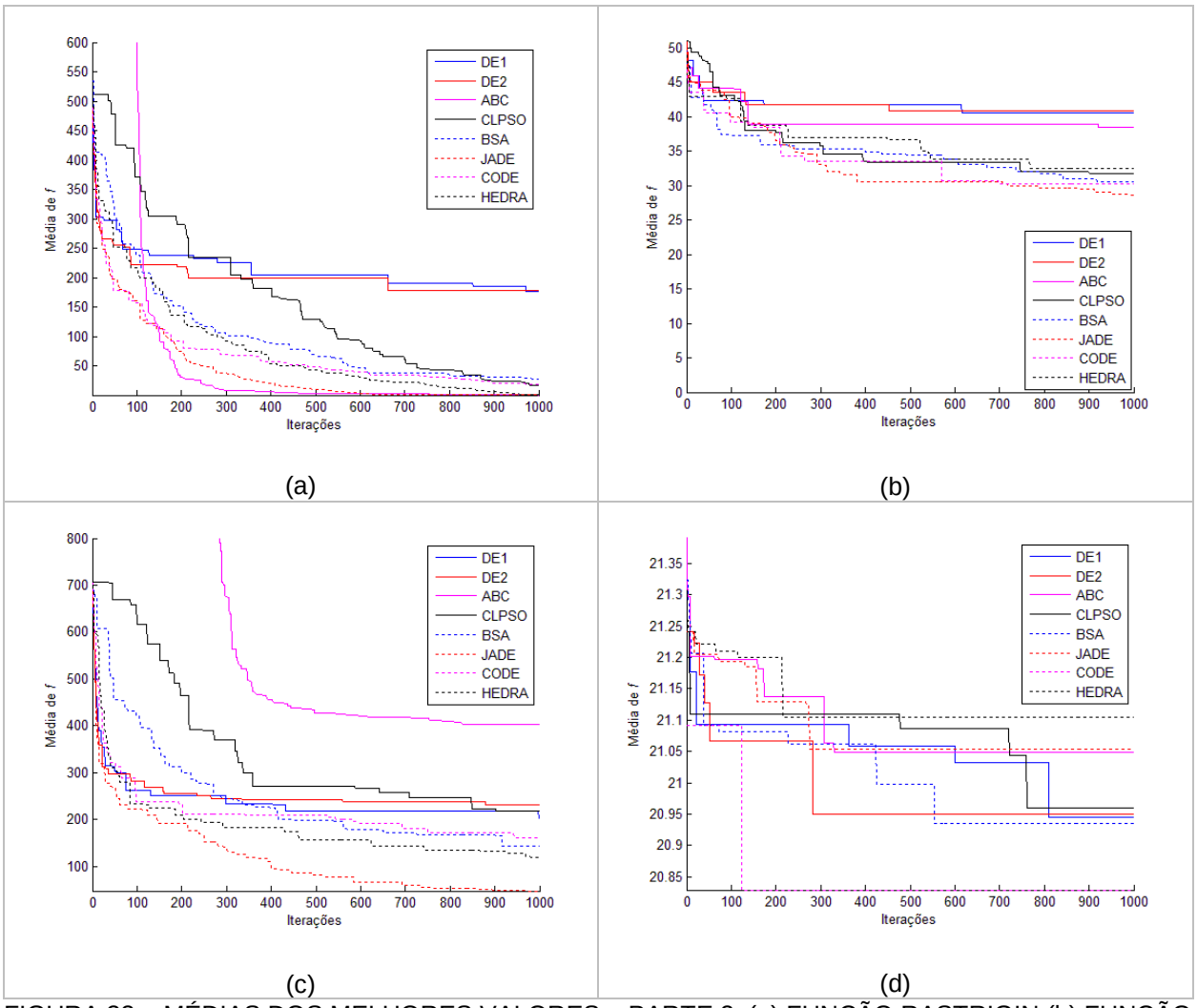

FIGURA 33 – MÉDIAS DOS MELHORES VALORES – PARTE 2. (a) FUNÇÃO RASTRIGIN (b) FUNÇÃO WEIERSTRASS ROTACIONADA (c) FUNÇÃO RASTRIGIN ROTACIONADA (d) FUNÇÃO ACKLEY
FONTE: O autor (2014)

6 EXTRATORES DE CARACTERÍSTICAS

Extração de caraterísticas é o processo de mapear os dados originais em caraterísticas mais eficientes para uma posterior classificação (FUKUNAGA, 1990). Dois métodos para extração de características são utilizados neste trabalho: padrão espacial comum (CSP do inglês, *common spatial pattern*) e potência de banda. A escolha dos métodos foi feita devido aos resultados utilizando os mesmos como mostrado por Lee *et al.* (2005).

6.1 PADRÃO ESPACIAL COMUM

O CSP é uma técnica para análise de múltiplos canais de EEG. O método se baseia na projeção do sinal em um subespaço em que as diferenças dos sinais são salientadas e as similaridades atenuadas. Para um sinal puro de EEG dado por X, busca-se a matriz W que fará a transformação de X para X_{CSP}, tal que,

$$X_{CSP} = W_T * X \tag{50}$$

Para obter a matriz W, os N canais de EEG como linhas e as T amostras de tempo como colunas, obtém-se X, na forma matricial, como mostrado na equação:

$$X = \begin{bmatrix} x_1(1) & \cdots & x_1(T) \\ \vdots & \ddots & \vdots \\ x_N(1) & \cdots & x_N(T) \end{bmatrix} \tag{51}$$

Yang (2009) mostra que X é completamente caracterizada por sua função de distribuição, mas ele ainda afirma que a dificuldade e a complexidade computacional para determinar essa função são muito altas, sendo mais vantajosa a utilização de uma estimativa de distribuição para os cálculos subsequentes. Sendo assim, a matriz de covariância é utilizada para indicar a dispersão da distribuição do sinal, como mostrado na equação 52, tal como:

$$\Sigma = E\{(X - M) * (X - M)'\} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,N} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,N} \end{bmatrix} \quad (52)$$

em que $E\{\cdot\}$ é o operador de expectativa matemática, em que cada um de seus valores é definido conforme a equação 53.

$$c_{i,j} = \frac{1}{T} * \sum_{t=1}^{T} \Big((x_i(t) - M_i) * (x_j(t) - M_j) \Big) \quad (53)$$

Tal que M é uma matriz de N linhas e T colunas como mostrado na equação 48 e seus elementos são definidos pela média dos valores de X na linha referente ao canal em questão, tal que,

$$M = \begin{bmatrix} M_1 & \cdots & M_1 \\ \vdots & \ddots & \vdots \\ M_N & \cdots & M_N \end{bmatrix} \quad (54)$$

$$M_i = \frac{1}{T} * \sum_{t=1}^{T} \big(x_i(t) \big) \quad (55)$$

Sendo assim, a matriz de covariância é simétrica, em que os valores da diagonal principal são as variâncias individuais de cada canal e os valores fora da diagonal são as covariâncias dos pares de canais referentes àquele elemento (FUKUNAGA, 1990).

Qualquer matriz pode ser vista como uma transformação linear. O autovetor da matriz transformada é a única propriedade que somente muda em escala durante essa transformação (o valor desta mudança depende do seu correspondente autovalor). Sendo assim, os autovetores podem ser visto como os componentes básicos em um sistema e seus autovalores a importância de cada componente deste vetor (YANG, 2009).

Para a matriz N por N de covariância Σ encontrada, os autovalores λ e os autovetores ν são obtidos pela resolução do sistema linear de equações dado por

$(\Sigma - \lambda I) * v = 0$, com I sendo a matriz identidade. Os autovetores com os maiores correspondentes autovalores são chamados de componentes principais de Σ. Os componentes principais descrevem a variação espacial dominante para uma dada atividade cerebral, sendo assim, entende-se que o sinal de EEG capturado para diferentes classes de pensamento apresentará autovetores diferentes e para a mesma classe, valores similares.

Ao calcular a média de todas as matrizes de covariância de cada classe (aqui serão utilizadas apenas duas classes para simplificação), obtém as matrizes R_1 e R_2. Ao fim, obtém-se a matriz W através da resolução do sistema apresentado na equação 56.

$$\begin{aligned} \Lambda_1 &= W' * R_1 * W \\ \Lambda_2 &= W' * R_2 * W \\ \Lambda_1 + \Lambda_2 &= I \end{aligned} \tag{56}$$

Uma descrição matemática mais aprofundada da resolução para W é dada em (YANG, 2009).

Yang (2009) ainda mostra que a matriz de transformação W pode ser utilizada para extração de características do sinal, como é utilizado neste trabalho, em que apenas algumas linhas da matriz W são utilizadas (por exemplo, as duas primeiras e as duas últimas, pois são os que mostrarão as maiores diferenças entre as classes) ou no pré-processamento do sinal, sendo necessária a posterior extração de caraterísticas através de outros métodos.

6.2 POTÊNCIA DE BANDA

Existem diferentes métodos para extração de potência de banda de um sinal. Para o método que é utilizado neste trabalho, o sinal de entrada $x(t)$ passa por um filtro Butterworth passa-banda de resposta infinita ao impulse de quarta ordem, gerando o sinal de saída filtrado $x_{filtrado}(t)$.

Cada amostra do sinal resultante $x_{filtrado}(t)$, que só possui as componentes da frequência desejada, é elevado a potência 2, gerando o sinal $p(t)$ conforme a equação 57, tal que,

$$p(t) = x^2_{filtrado(t)} \quad (57)$$

Sobre o sinal $p(t)$ é utilizada uma janela de suavização, que aqui é definida com o tamanho w, de forma que o sinal de saída é obtido através da média das amostras nessa janela, como mostrado na equação 58, de forma que

$$p(t) = \frac{1}{w} * \sum_{k=0}^{w} p(t-k) \quad (58)$$

Sendo assim, a potência de banda para o instante t é dado pela média das últimas w amostras. Em geral, o valor final da potência de banda é dado pelo logaritmo natural de p, pois esta operação pode aumentar a separabilidade das características obtidas.

Nos sinais de EEG, para cada instante de tempo e cada um dos canais utilizados, um valor de potência de banda é dado para cada uma das bandas de interesse.

7 METODOLOGIA

Neste capítulo são apresentadas a base de dados e a metodologia aplicada neste trabalho.

7.1 BASE DE DADOS (COMPETIÇÃO BCI DE BERLIM)

Competição BCI de Berlim é uma competição aberta e tem o objetivo de avaliar diversas abordagens utilizadas em sistemas BCI. Essas técnicas são aplicadas a base de dados comuns, dando a possibilidade de comparação mais justa e confiável do que as apresentadas em artigos, onde cada autor faz uso de diferentes problemas e tipos de dados para avaliar seus resultados.

Quatro edições da competição foram organizadas, com a primeira no ano de 2000 e a última em 2008. Diferentes bases de dados eram disponibilizadas em cada competição, onde cada uma apresentava uma problemática diferente a ser tratada. Uma primeira base de dados para treinamento era disponibilizada, em que a classificação de cada experimento também era fornecida – esta era utilizada pelos competidores para criar seus sistemas e treiná-los – e, posteriormente, uma segunda base de dados seguindo a mesma configuração da primeira, porém sem a classificação dos experimentos, era fornecida – aqui os competidores utilizavam os sistemas treinados com a primeira base de dados para obter as classificações e enviavam os resultados para os organizadores da competição, que poderiam compará-las aos valores corretos para obter métricas de comparação dos sistemas.

O problema que é tratado neste trabalho é oriundo da Competição BCI de Berlim IV, fazendo uso da base de dados IIa, que está disponível de forma livre online. Como é mostrado por Brunner *et al.* (2008), esta base de dados consiste no EEG lido de nove voluntários, que executaram quatro tarefas movimento imaginário diferentes: movimento da mão esquerda, mão direita, ambos os pés e língua. Duas sessões foram utilizadas, cada uma em um dia, sendo a primeira definida como a base de dados de treino e a segunda como a base de dados de avaliação. Cada sessão teve 6 execuções com

intervalos curtos entre elas. Cada execução continha 12 experimentos para cada classe, totalizando 48 experimentos em cada execução e 288 experimentos em cada sessão.

A aquisição de dados ocorreu com os voluntários sentados em poltronas confortáveis em frente à tela de um computador. Ao iniciar cada experimento, uma cruz para fixação era apresentada na tela e um som de aviso era soado. Após dois segundos de execução, um sinal no formato de uma flecha (para cima, para baixo ou para os lados, cada um correspondendo a uma das quatro classes) era apresentado na tela para que o voluntário se preparasse e iniciasse a imaginação do movimento determinado pelo sinal, este sinal permanecia na tela por 1,25 segundos. O voluntário, entretanto, era solicitado para manter a imaginação do movimento por 3 segundos, quando o mesmo era avisado que podia relaxar. Nenhuma resposta era dada ao voluntário com relação aos dados coletados. A (FIGURA 34) mostra o esquema de tempo utilizado nas gravações.

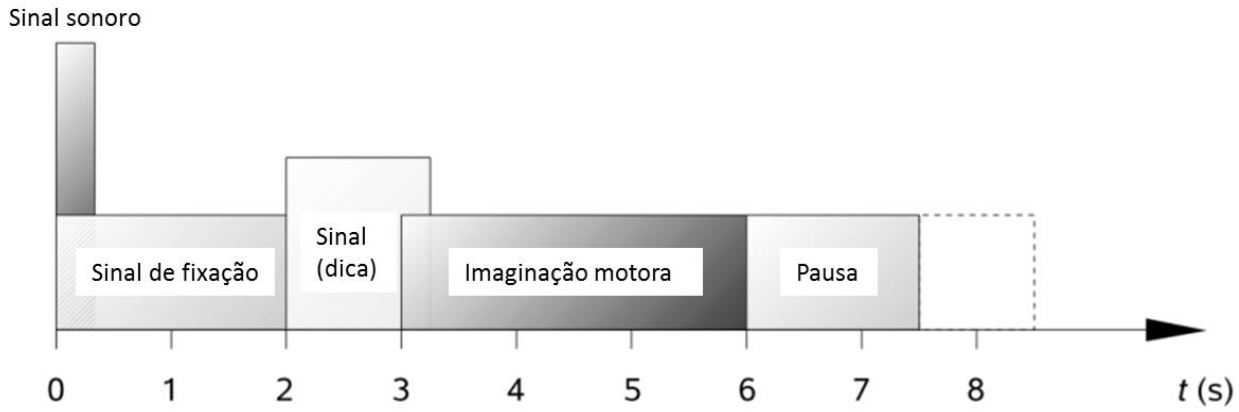

FIGURA 34 – PROTOCOLO DE GRAVAÇÃO DOS SINAIS
FONTE: Adaptado de Brunner *et al.* (2008)

Neste sistema, foram utilizados 22 eletrodos, com uma distância de 3,5 cm entre um e outro, e foram montados conforme o sistema 10-20 mostrado na (FIGURA 35). A leitura foi feita com amplificadores de sensibilidade de 100 μV a uma frequência de 250 Hz e filtrada entre 0.5 Hz e 100 Hz. Também foi feita a filtragem do sinal de 50 Hz para remover ruídos oriundos do sistema de energia. A (FIGURA 35) também mostra a montagem de eletrodos para leitura de EOG – com mesma frequência e esquema de filtragem do EEG, porém a sensibilidade do amplificador foi definida para 1 mV – sendo

que os sinais vindo destes canais podem apenas ser utilizados para remoção de artefatos e não devem ser usados no esquema de classificação.

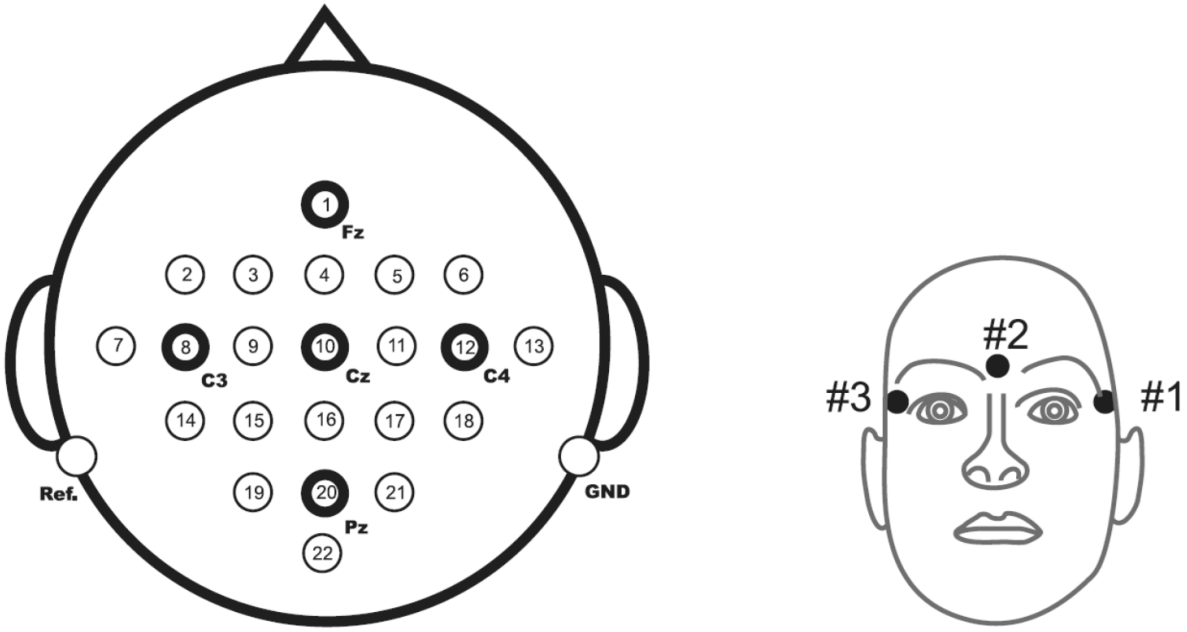

FIGURA 35 – ESQUEMA DE COLOCAÇÃO DOS ELETRODOS DA BASE DE DADOS UTILIZADA
FONTE Brunner *et al.* (2008)

Para avaliação dos resultados, cada participante devia providenciar uma saída de classificação (de 1 a 4) para cada amostra de EEG da fase em que o voluntário estava executando a imaginação do movimento. O coeficiente Kappa de Cohen foi utilizado como métrica, conforme é mostrado na equação 59, tal que,

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{59}$$

em que p_0 é a acurácia do resultado com relação à classificação perfeita e p_e é a probabilidade de acerto ao acaso de cada uma das amostras, ou seja, neste caso igual a 0,25. Sendo assim, o valor de κ igual à zero mostra que o resultado obtido não tem correlação com o problema obtido (ou seja, as classes foram selecionadas de forma

aleatória), já quando κ se aproxima de 1, significa um classificador próximo a perfeição para o problema.

Todos os algoritmos devem ser causais, ou seja, as saídas de classificação só podem fazer uso dos sinais de entrada até o momento atual ou das entradas passadas. Isso faz com que o sistema de classificação funcione de forma pseudo-online.

A (TABELA 3) mostra os valores κ obtidos pelos participantes da competição em 2008, apresentando os valores obtidos para cada um dos voluntários e a média final, que foi utilizada para a classificação.

TABELA 3 – RESULTADOS DA COMPETIÇAO BBCI 2008

#	Autor	κ médio	Voluntário 1	2	3	4	5	6	7	8	9
1	Kai Keng Ang	0,57	0,68	**0,42**	**0,75**	**0,48**	**0,40**	**0,27**	**0,77**	**0,75**	0,61
2	Liu Guangquan	0,52	**0,69**	0,34	0,71	0,44	0,16	0,21	0,66	0,73	**0,69**
3	Wei Song	0,31	0,38	0,18	0,48	0,33	0,07	0,14	0,29	0,49	0,44
4	Damien Coyle	0,30	0,46	0,25	0,65	0,31	0,12	0,07	0,00	0,46	0,42
5	Jin Wu	0,29	0,41	0,17	0,38	0,25	0,06	0,16	0,34	0,45	0,37

FONTE Brunner *et al.* (2008)

Cada um dos participantes apresentados na (TABELA 3) utilizaram diferentes métodos de pré-processamento, extração de características e classificadores. Segundo apresentado por Bunner *et al.* (2008), conforme sua colocação final na competição:

Posição 1. Foi utilizado um filtro passabanda *Chebyshev2* para remoção de artefatos. Para a extração de características foi proposto o algoritmo *Filter Bank Common Spatial Pattern* e para classificação, o algoritmo *Naïve Bayes Parzen Window;*

Posição 2. A remoção de artefatos foi feito através de técnicas passabanda de quinta ordem. Uma técnica de CSP modificada foi utilizada para extração de características – e posteriormente foi aplicado o método de análise de discriminação linear de Fisher para redução da dimensão. Finalmente, um classificador *Bayesian* foi utilizado para obter o resultado final;

Posição 3. No pré-processamento foi feito a redução de frequência e filtragem através de um passabanda linear. Através de um algoritmo recursivo, foram eliminados canais menos influentes ao problema. A extração de

características foi feita pelo algoritmo CSP. Para classificação foi utilizada a SVM um-contra-todos;

Posição 4. O pré-processamento foi feito através de um algoritmo de auto-organização de rede neural *fuzzy*. O sinal ainda passou por um filtro passa-banda e pelo algoritmo CSP para redução de dimensionalidade. Para extração de características foi utilizado, novamente, o CSP. Para a classificação os autores apresentaram três alternativas, uma utilizando o algoritmo de análise de discriminação linear e outras duas utilizando SVM – um-contra-todos e todos-contra-todos;

Posição 5. Pré-processamento através da redução de frequência, seleção dos canais em torno dos eletrodos C3 e C4 e filtragem passabanda. A extração de características foi feita através do algoritmo CSP. Para a classificação foi utilizado um mistura de SVMs um-contra-todos e todos-contra-todos através de hierarquias.

7.2 APLICAÇÃO DA METODOLOGIA

Alterando a método de um sistema BCI básico mostrado no início deste trabalho para incorporar também o sistema de otimização, chega-se a (FIGURA 36).

7.2.1 Aquisição de sinal

Como mostrado, a aquisição de sinais foi feita através de 22 canais de EEG e mais 3 canais de EOG em 9 voluntários. Duas sessões foram executadas em dias separados – a primeira usada para treinamento e a segunda para teste. É importante ressaltar que a gravação dos dados em dias diferentes aumenta a dificuldade do problema, uma vez que variações de impedância dos eletrodos, diferente posicionamento dos mesmos e até variação nas interferências da rede podem variar de forma significativa de um dia para o outro. Além destes pontos, ainda há o lado emocional e motivacional dos voluntários, que podem estar diferentes na nova sessão de gravações e modificar expressivamente os sinais lidos. Um estudo aprofundado nas variâncias

encontradas para diferentes experimentos, sessões e usuários foi feito por Roijendijk (2009).

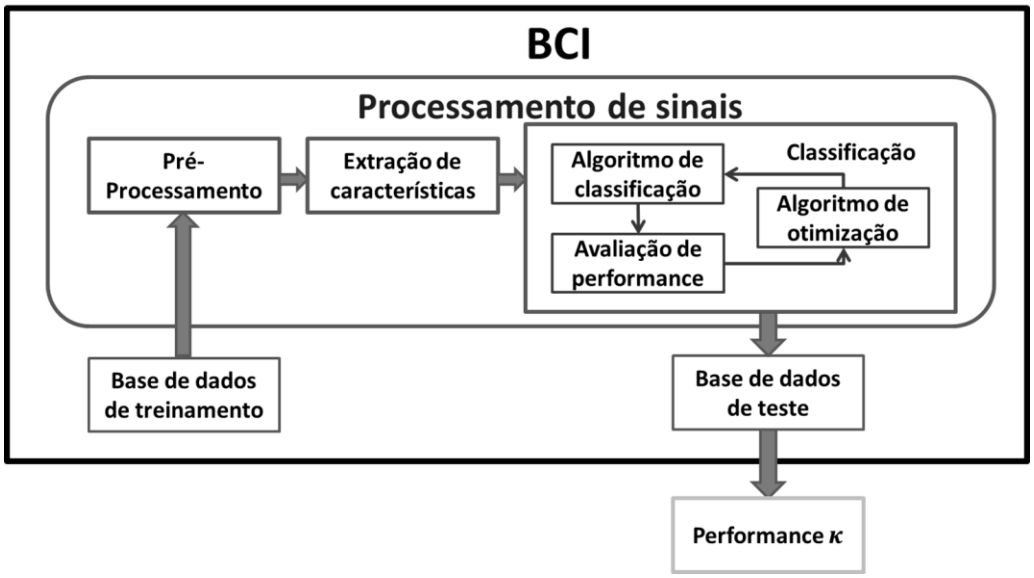

FIGURA 36 – MODELO DE SISTEMA BCI UTILIZADO INCLUINDO ETAPA DE OTIMIZAÇÃO DO ALGORITMO DE CLASSIFICAÇÃO
FONTE O autor (2014)

7.2.2 Pré-processamento

Durante a aquisição de dados, o primeiro passo do pré-processamento já é executado, em que o sinal é filtrado em 50 Hz (frequência da rede Europeia, onde os sinais foram gravados) utilizando um filtro *notch*. Além deste filtro, também foi utilizado um sistema para remoção de artefatos relacionados ao EOG. Para esta tarefa, foi utilizado o método proposto por Schlögl *et al.* (2007) que usa auto covariância dos sinais EOG e covariância cruzada entre o sinal EEG e EOG para remover os artefatos de forma automática. Schlögl *et al.* (2007) mostram que foram capazes de reduzir em 80% dos artefatos com esta técnica. O método está implementado na *toolbox BioSig*, disponível para *Matlab*.

Ainda no pré-processamento foi feito redução da frequência do sinal EEG, que era de 250 Hz e foi transformada para 100 Hz, estratégia para reduzir a quantidade de amostras e, dessa forma, o tempo de execução dos demais elementos do BCI.

7.2.3 Extração de características

Os extratores de características a serem utilizados são a potência de banda e o CSP, como apresentado no capítulo 5. Para a potência de banda, dois intervalos de frequência foram utilizados – entre 10 e 12 Hz (frequência μ) e 16 e 24 Hz (frequência β), dessa forma, para cada instante de tempo 44 características (1 para cada frequência de banda para cada um dos 22 canais de EEG) são extraídas e utilizadas. No caso do CSP, são utilizados os dois primeiros e os dois últimos principais componentes para cada classe, totalizando 16 características para cada instante de tempo.

São testados os dois extratores de características de forma individual e combinados.

7.2.4 Classificação

Neste ponto, foram adicionados dois blocos quando comparado com o fluxograma do BCI básico. O classificador usa parte dos dados de treinamento não somente para a criação do sistema de classificação, mas também para a otimização dos parâmetros desse sistema. Dessa forma, a classificação foi subdividida em três novos blocos: SVM, performance κ e algoritmo evolutivo:

- Para classificação das características extraídas em uma das classes (movimento imaginado) são utilizados os três métodos de SVM multiclasses apresentados anteriormente. Os métodos são comparados com relação à sua performance e número de vetores de suporte em cada uma das metodologias. Neste ponto, são utilizados 70% das amostras de treinamento, sendo que o restante é usado para avaliação da performance dos parâmetros definidos.
- O método de performance κ mede a qualidade dos resultados encontrados pela SVM, como explicado anteriormente. Aqui, serão utilizados os 30% restantes dos dados de treinamento.
- Para otimização dos parâmetros da SVM são utilizados os oito algoritmos evolutivos mostrados no capítulo 4 - DE1, DE2, ABC, CLPSO, BSA, JADE, CODE e HEDRA – sendo que os algoritmos buscam minimizar a porcentagem

de erros de classificação da SVM multiclasses através da variação dos parâmetros da SVM em questão. A execução utilizando cada um dos algoritmos é apresentada de forma individual e comparada através de métodos estatísticos.

7.2.5 Aplicação de teste

Finalmente, os melhores valores de parâmetros da SVM multiclasses encontradas no processo de otimização são utilizados para classificar a base de dados de teste. O valor de performance κ é calculado e comparado com o valor encontrado pelas demais abordagens.

8 RESULTADOS

Como apresentado no capítulo anterior, o sistema é executado sob nove configurações diferentes. Cada um dos três tipos de SVM para multiclasses apresentado é testado usando apenas as características extraídas através do CSP, somente com a potência de banda e finalmente com uma combinação das características extraídas pelos dois métodos. Para cada uma dessas combinações, os 8 algoritmos foram utilizados para otimizar os parâmetros das SVMs.

Vale notar que, dependendo do método de SVM multiclasses utilizado, o número de parâmetros a serem selecionados é diferente. Considerando as quatro classes: braço direito, braço esquerdo, ambas as pernas e a língua, o método um-contra-todos usará 4 SVMs para fazer a classificação, sendo assim, 8 parâmetros (para cada SVM um valor de σ do kernel e um para a constante c), o método um contra todos usará 6 SVMs – resultando em 12 parâmetros – e o método proposto de agrupamento por similaridade, usará 3 SVMs – com apenas 6 parâmetros a serem otimizados. As variáveis relacionadas ao valor de σ tiveram os espaços de buscas definido para $]0, 1000]$ e para as constantes c os espaços de busca são $]0, 100]$.

Mais uma vez, os parâmetros dos algoritmos seguirão as orientações dos seus autores originais. A primeira variante da DE, DE1, usando o método de mutação $\mathrm{DE/aleatório/1.}$, valor de F igual a 0,5 e C_r igual a 0,9. A segunda variante da DE, DE2, método de mutação e valores de F e C_r igual aos de DE1. No algoritmo ABC, o número de abelhas operárias adotadas é igual a metade da população total, as demais são abelhas não operárias. As fontes de alimentos são abandonadas quando após 10 iterações não há melhora no resultado da abelha que está explorando aquela fonte. Para o CLPSO o valor de m foi definido como 10, a constante c como 1,49445 e os valores de ω_0 e ω_1 iguais a 0,4 e 0,9, respectivamente. A velocidade máxima das partículas foi definida como 20% da diferença entre o valor máximo e mínimo para cada dimensão. O algoritmo BSA tem a taxa de mistura igual a 1. Os algoritmos JADE, CODE e HEDRA não possuem parâmetros para serem definidos.

Devido à origem heurística dos algoritmos, cada um foi rodado 50 vezes para cada um dos testes. A população foi definida em 10 indivíduos e o número máximo de

iterações para 100 (com um número total de avaliação da função objetivo igual a 1000). Os valores do número de execuções, tamanho da população, número máximo de iterações e número máximo de avaliações foram reduzidos comparados aos problemas de *benchmark* devido ao maior custo computacional para avaliação dos indivíduos.

A apresentação dos resultados é feita usando o valor obtido na função de aptidão, ou seja, a porcentagem de amostras classificadas erroneamente – dessa forma, quanto menor o valor obtido, melhor o índice de classificação.

O (TABELA 4) mostra os resultados obtidos para a classificação com os três métodos de SVM multiclasses utilizando como dados de entrada da SVM apenas as característica extraídas pela potência de banda. Como definido anteriormente, são utilizados 70% dos dados de entrada para treinamento da SVM e 30% para avaliação da performance da mesma.

É importante ressaltar que os resultados não são apresentados de forma individual por voluntário e sim uma média de todos os valores. O sistema com melhor resultado é aplicado na base de dados teste – em que o valor κ final, é então calculado com discriminação entre os voluntários.

Para facilitar a apresentação dos resultados, são utilizadas as nomenclaturas de SVM 1-vs-Tds para o método um-contra-todos, SVM Tds-vs-Tds para o método todos-contra-todos e SVM Agrup para o agrupamento por similaridade.

TABELA 4 – RESULTADOS PARA CARACTERÍSTICA POTÊNCIA DE BANDA

		Algoritmo							
Método	**Métrica**	**DE1**	**DE2**	**ABC**	**CLPSO**	**BSA**	**JADE**	**CODE**	**HEDRA**
SVM 1-vs-Tds	*Média*	0,328	0,342	0,352	0,368	0,306	0,334	0,336	**0,298**
	Mínimo	0,301	0,296	0,315	0,301	0,279	0,282	0,309	**0,274**
	Máximo	0,379	0,399	0,425	0,416	0,361	**0,355**	0,387	0,371
	Desvio Padrão	0,040	0,045	0,041	0,050	0,029	0,022	**0,017**	0,028
	Média de SVs	929	901	965	976	**890**	920	897	901
SVM Tds-vs-Tds	*Média*	0,357	0,331	0,339	0,351	0,303	**0,280**	0,319	0,304
	Mínimo	0,325	0,310	0,290	0,323	0,283	**0,265**	0,287	0,275
	Máximo	0,422	0,370	0,404	0,395	**0,349**	0,363	0,401	0,377
	Desvio Padrão	0,036	0,030	0,029	**0,020**	0,029	0,025	0,027	0,024
	Média de SVs	1250	1231	1301	1311	**1130**	1193	1201	1134
SVM Agrup	*Média*	0,311	0,322	0,322	0,321	0,293	**0,264**	0,302	0,280
	Mínimo	0,291	0,294	0,281	0,297	0,248	**0,231**	0,283	0,246
	Máximo	0,350	0,379	0,342	0,352	0,350	**0,311**	0,336	0,325
	Desvio Padrão	0,021	0,026	0,023	0,019	0,045	0,030	0,019	**0,014**
	Média de SVs	801	812	723	837	702	796	**698**	728

FONTE: O autor (2014)

É possível notar pelo quadro que a escolha do algoritmo evolutivo correto pode ter alto impacto na performance de classificação. Em especial, os algoritmos JADE e HEDRA obtiveram os melhores resultados para todos os métodos de SVM utilizados. O método de SVM multiclasses também tem impacto no resultado final, em que o método proposto neste trabalho obteve melhor performance e menor complexidade (número de SVs) que as demais. A (FIGURA 37) apresenta a evolução da média dos melhores resultados dos algoritmos através das iterações.

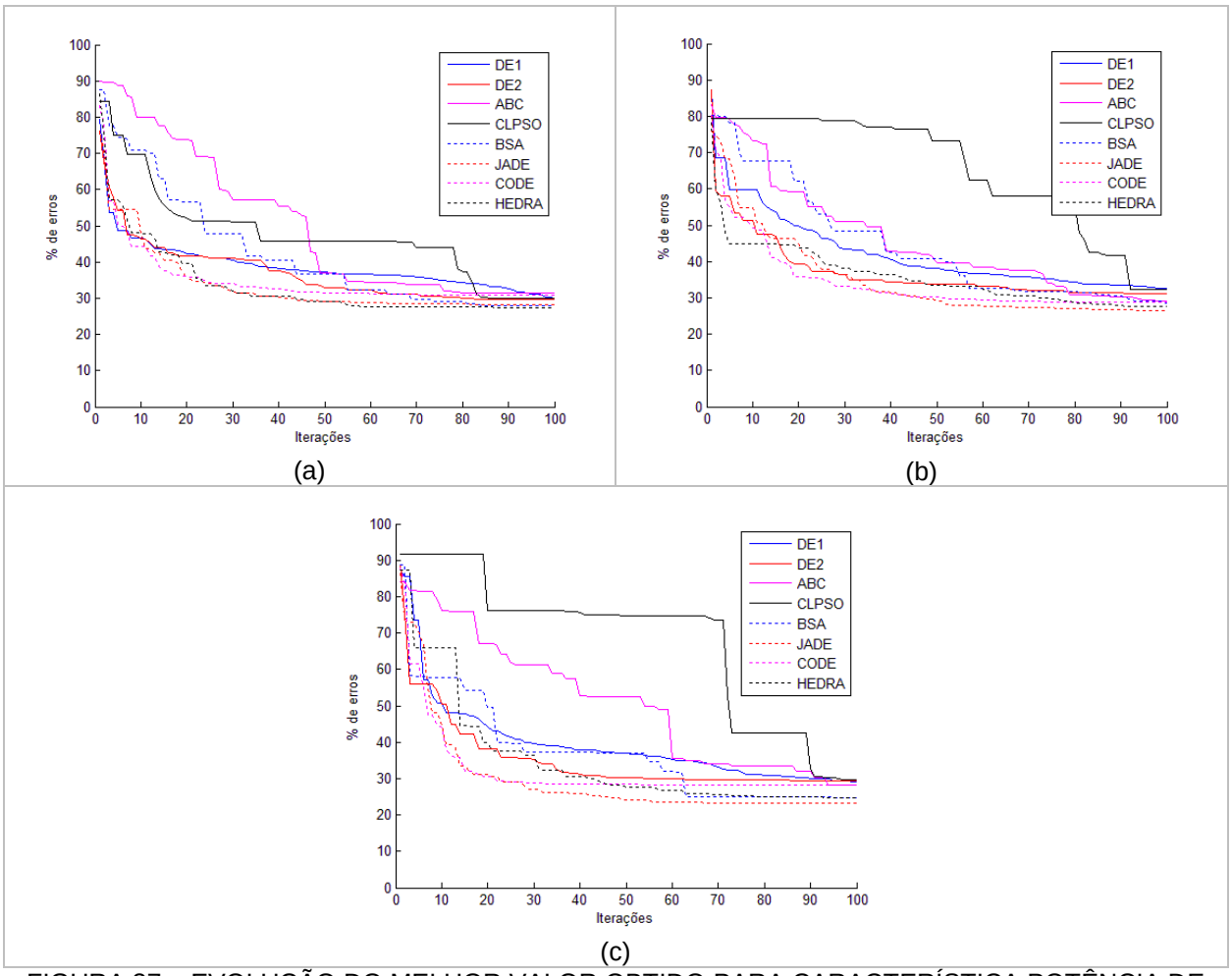

FIGURA 37 – EVOLUÇÃO DO MELHOR VALOR OBTIDO PARA CARACTERÍSTICA POTÊNCIA DE BANDA (a) SVM 1-VS-TDS (b) SVM TDS-VS-TDS (c) SVM AGROUP
FONTE: O autor (2014)

Com exceção do algoritmo CLPSO e ABC, todos os demais algoritmos tiveram curvas similares de convergência. Entretanto, como já indicado anteriormente, o algoritmos JADE e HEDRA obtiveram os melhores resultados entre todos.

O (TABELA 5) mostra as mesmas execuções sendo feitas com a extração de características CSP.

TABELA 5 – RESULTADOS PARA CARATERÍSTICAS CSP

		Algoritmo							
Método	**Métrica**	**DE1**	**DE2**	**ABC**	**CLPSO**	**BSA**	**JADE**	**CODE**	**HEDRA**
SVM 1-vs-Tds	*Média*	0,232	0,273	0,321	0,345	0,226	**0,207**	0,229	0,221
	Mínimo	0,231	0,218	0,267	0,319	0,203	0,161	0,159	**0,146**
	Máximo	0,299	0,370	0,420	0,378	0,241	0,292	0,264	**0,230**
	Desvio Padrão	0,025	0,039	0,052	0,025	0,030	0,043	0,034	**0,013**
	Média de SVs	940	939	883	920	**870**	923	912	930
SVM Tds-vs-Tds	*Média*	0,269	0,259	0,252	0,329	**0,213**	0,232	0,246	0,231
	Mínimo	0,198	0,205	0,252	0,256	**0,175**	0,203	0,214	0,202
	Máximo	0,301	0,354	**0,241**	0,401	0,278	0,318	0,295	0,306
	Desvio Padrão	0,040	0,056	**0,007**	0,045	0,053	0,038	0,016	0,031
	Média de SVs	**1056**	1199	1150	1146	1231	1233	1098	1123
SVM Agrup	*Média*	0,248	0,235	0,282	0,335	0,234	0,237	0,261	**0,222**
	Mínimo	0,218	0,225	0,233	0,278	0,204	0,199	0,209	**0,189**
	Máximo	**0,267**	0,290	0,305	0,418	0,352	0,286	0,300	0,347
	Desvio Padrão	0,024	0,044	0,030	0,055	0,062	0,026	**0,021**	0,046
	Média de SVs	789	789	809	804	**786**	792	812	815

FONTE: O autor (2014)

Os resultados obtidos foram significativamente superiores àqueles alcançados utilizando a potência de banda. Mesmo os piores resultados encontrados utilizando a extração de características por CSP são superiores aos melhores encontrados através da potência de banda, sendo assim, fica claro que a definição de um sistema de extração de características de qualidade também é um passo muito importante para obtenção de melhores resultados dos sistemas BCIs.

Mais uma vez, os algoritmos JADE e HEDRA obtiveram bons resultados, assim como o BSA. Entretanto, com a extração de caraterísticas CSP o impacto da mudança de método SVM multiclasses foi reduzida, em que todos apresentaram performance similar. Apenas a complexidade da máquina final obtida continua sendo menor para o algoritmo proposto por agrupamento de similaridade.

A (FIGURA 38) mostra a média dos melhores valores encontrados através das iterações.

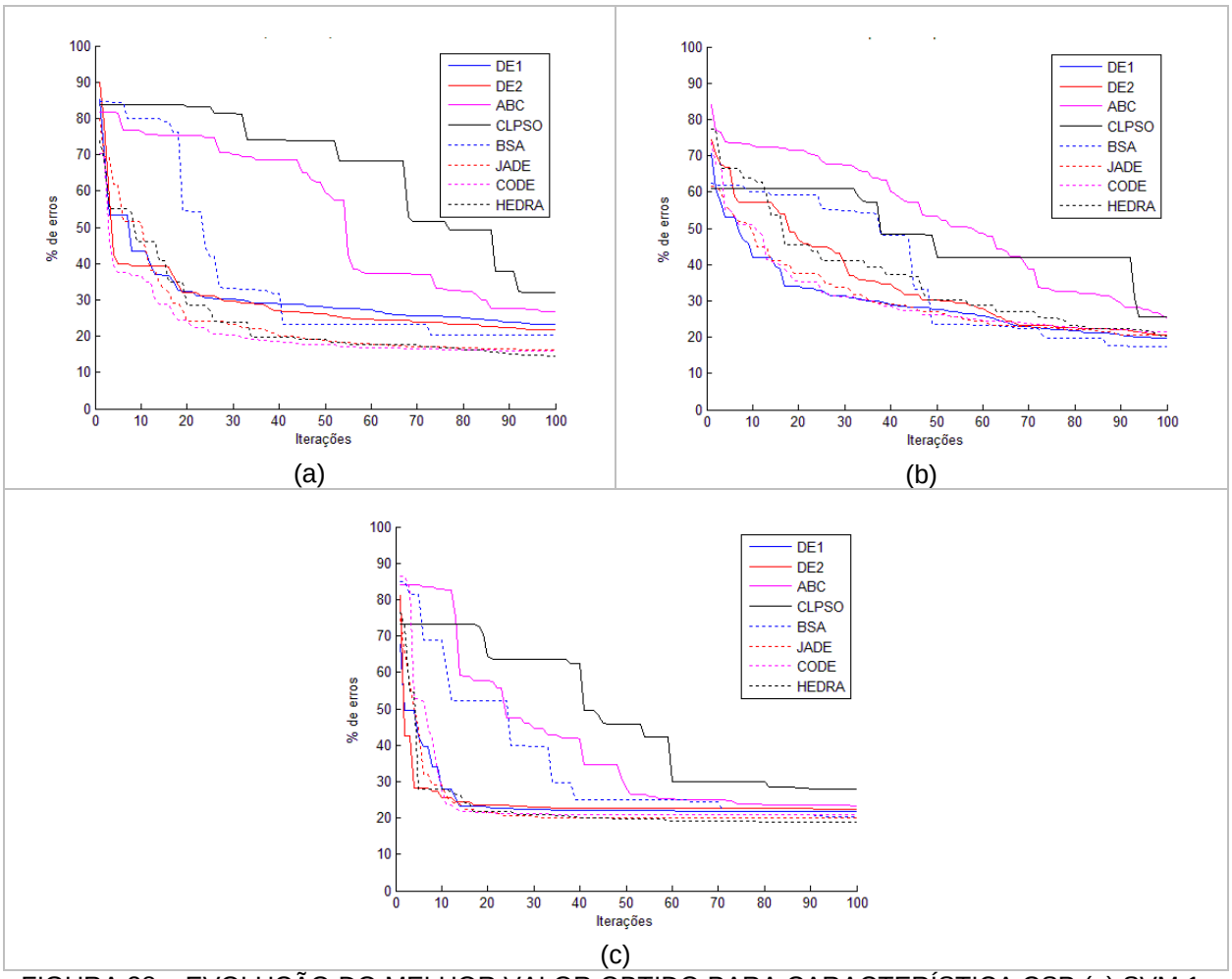

FIGURA 38 – EVOLUÇÃO DO MELHOR VALOR OBTIDO PARA CARACTERÍSTICA CSP (a) SVM 1-VS-TDS (b) SVM TDS-VS-TDS (c) SVM AGROUP
FONTE: O autor (2014)

Os algoritmos seguiram o mesmo padrão de evolução, salvos os algoritmos CLPSO e ABC, que mais uma vez apresentaram uma evolução mais lenta até chegar ao seu valor final.

Finalmente, o (TABELA 6) mostra os resultados para a combinação dos dois métodos de extração de características.

TABELA 6 – RESULTADOS PARA OS DOIS MÉTODOS DE EXTRAÇÃO DE CARATERÍSTICAS COMBINADOS

		Algoritmo							
Método	**Métrica**	**DE1**	**DE2**	**ABC**	**CLPSO**	**BSA**	**JADE**	**CODE**	**HEDRA**
SVM 1-vs-Tds	*Média*	0,292	0,315	0,345	0,368	0,289	0,298	0,316	**0,286**
	Mínimo	**0,230**	0,261	0,276	0,332	0,290	0,279	0,290	0,244
	Máximo	0,417	0,439	0,407	0,426	**0,338**	0,388	0,401	0,362
	Desvio Padrão	0,080	0,070	0,052	0,040	0,033	0,039	0,033	**0,028**
	Média de SVs	927	882	928	897	**854**	894	923	894
SVM Tds-vs-Tds	*Média*	0,282	0,328	0,345	0,396	0,284	0,282	0,343	**0,280**
	Mínimo	0,249	0,235	0,303	0,370	**0,210**	0,235	0,302	0,253
	Máximo	**0,311**	0,435	0,419	0,463	0,361	0,388	0,413	0,338
	Desvio Padrão	0,044	0,067	0,053	0,039	0,047	0,033	0,027	**0,020**
	Média de SVs	1233	1050	1142	1135	1142	1221	**1023**	1287
SVM Agrup	*Média*	0,333	0,302	0,325	0,335	0,317	0,294	0,288	**0,270**
	Mínimo	**0,213**	0,253	0,253	0,294	0,253	0,226	0,252	0,232
	Máximo	0,362	**0,340**	0,363	0,378	0,380	0,348	0,343	0,375
	Desvio Padrão	0,053	0,032	0,036	0,036	0,050	0,028	**0,025**	0,033
	Média de SVs	760	793	**758**	768	808	799	818	782

FONTE: O autor (2014)

Os resultados obtidos com ambos os métodos de extração de caraterísticas foi inferior à utilização individual dos mesmos. Isso pode ocorrer por aumentar de forma agressiva a quantidade de características por amostra de dado. Apesar de o algoritmo HEDRA ter obtido bons resultados novamente, seus resultados foram comparáveis com aqueles encontrados apenas com o pior dos métodos de extração de características, mostrando que o uso das duas características em conjunto não é vantajoso, principalmente se for levado em consideração os custos computacionais para estas operações.

A (FIGURA 39) mostra a evolução dos algoritmos através das iterações.

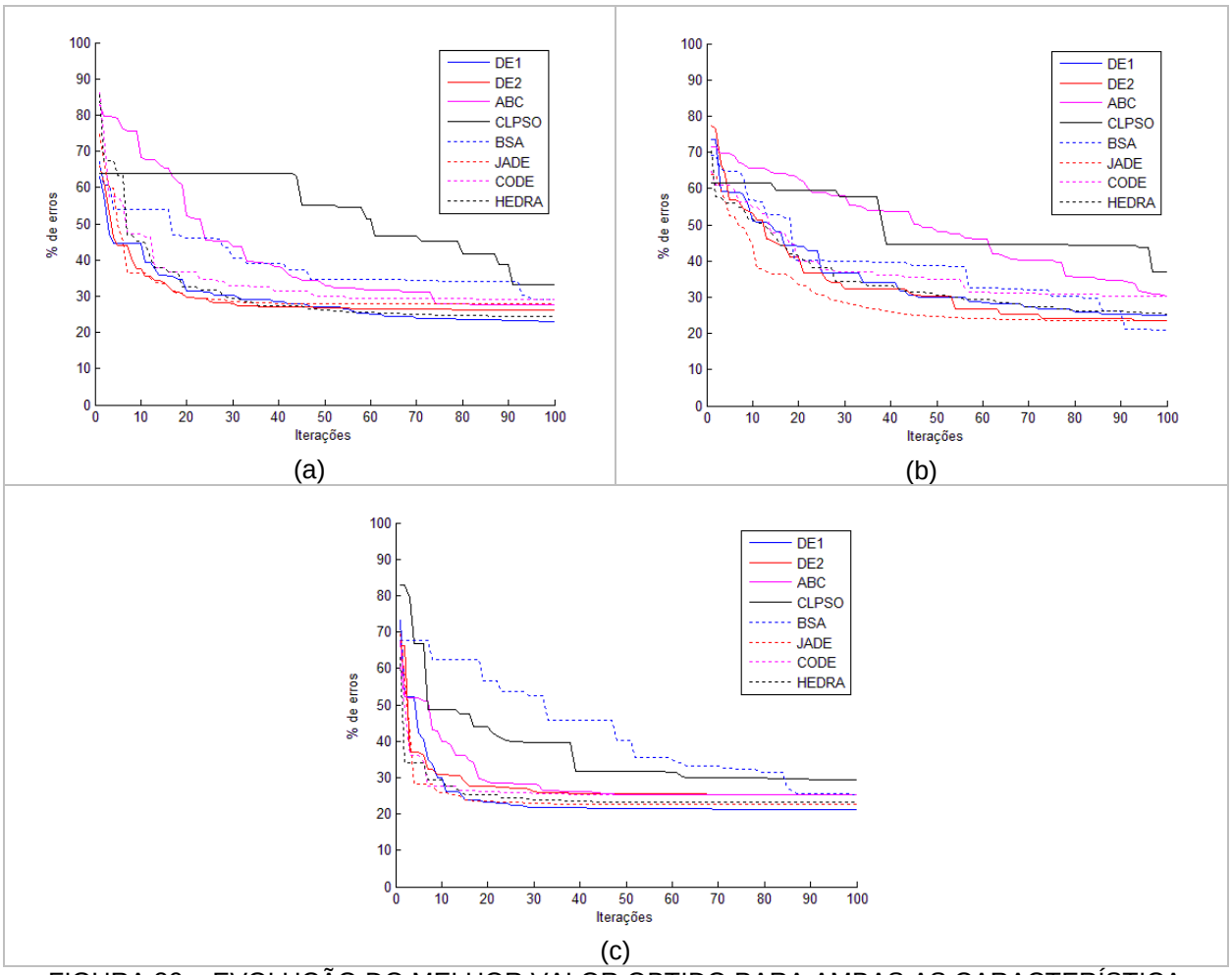

FIGURA 39 – EVOLUÇÃO DO MELHOR VALOR OBTIDO PARA AMBAS AS CARACTERÍSTICA COMBINADAS (a) SVM 1-VS-TDS (b) SVM TDS-VS-TDS (c) SVM AGROUP
FONTE: O autor (2014)

Como mostrado, os resultados foram inferiores ao uso de CSP e potência de bandas individualmente. É possível ver que, em geral, os algoritmos já estão estabilizados, ou seja, o aumento do número de iterações provavelmente não traria resultados superiores aos já apresentados.

Após os resultados obtidos nas técnicas de extração de caraterísticas apresentados e nos métodos SVM multiclasses, o teste de Wilcoxon foi executado para comparação estatística dos algoritmos. O (TABELA 7) apresenta os valores de ranking R^+ conforme apresentado no capítulo 4. Como, neste caso, o número de problemas no conjunto é igual a 9, o valor de R^+ deve ser menor que 8 para um algoritmo ser considerado estatisticamente melhor que outro. As células em cinza escuro mostram que

o algoritmo da referida linha é estatisticamente superior ao da coluna, para este conjunto de problemas – as células cinza claro são para os algoritmos que obtiveram resultados melhores, mas os resultados não são estatisticamente conclusivos.

TABELA 7 – TESTE DE WILCOXON PARA O SISTEMA BCI

	DE1	DE2	ABC	CLPSO	BSA	JADE	CODE	HEDRA
DE1	=	13	6	1	44	42	30	45
DE2	32	=	6	1	45	45	42	45
ABC	39	39	=	1	45	45	45	45
CLPSO	44	44	44	=	45	45	45	45
BSA	1	0	0	0	=	28	2	32
JADE	3	0	0	0	17	=	2	27
CODE	15	3	0	0	43	43	=	45
HEDRA	0	0	0	0	13	18	0	=

FONTE: O autor (2014)

É notável que os algoritmos JADE e HEDRA obtiveram resultados superiores aos demais algoritmos. Entretanto, os dois algoritmos não chegaram a resultados que possam ser considerados superiores um ao outro.

Por fim, o melhor resultado encontrado através de todas as combinações, foi utilizando o método de SVM um-contra-todos, com seus parâmetros otimizados pelo algoritmo proposto HEDRA, e usando a extração de caraterísticas pelo método CSP. Ao utilizar os valores encontrados nessa combinação à base de dados de teste e comparar com os resultados da competição este método teria ficado em terceiro lugar, como é mostrado no (TABELA 8).

TABELA 8 – RESULTADO DA COMPETIÇÃO BCI DE BERLIM 2008 INCLUINDO O MÉTODO PROPOSTO

#	Autor	κ médio	Voluntário								
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	Kai Keng Ang	0,57	0,68	**0,42**	**0,75**	0,48	**0,40**	**0,27**	**0,77**	**0,75**	0,61
2	Liu Guangquan	0,52	**0,69**	0,34	0,71	0,44	0,16	0,21	0,66	0,73	**0,69**
3	*Método proposto*	*0,50*	*0,62*	*0,35*	*0,68*	***0,49***	*0,33*	*0,19*	*0,53*	*0,63*	*0,65*
4	Wei Song	0,31	0,38	0,18	0,48	0,33	0,07	0,14	0,29	0,49	0,44
5	Damien Coyle	0,30	0,46	0,25	0,65	0,31	0,12	0,07	0,00	0,46	0,42
6	Jin Wu	0,29	0,41	0,17	0,38	0,25	0,06	0,16	0,34	0,45	0,37

FONTE: Brunner *et al.* (2008)

9 CONCLUSÃO

Este trabalho fez a comparação de técnicas meta-heurísticas de otimização na seleção dos parâmetros de controle de máquinas de vetores de suporte multiclasses aplicadas em sistemas de interface cérebro-computador. A interface cérebro-computador é um dos sistemas mais pesquisados na atualidade, uma vez que sua evolução pode ter grande impacto na qualidade de vida de milhões de pessoas.

Neste trabalho foi utilizado um sistema de interface cérebro-máquina simples que, entretanto, utiliza ferramentas de processamento de dados dos principais sistemas apresentados na literatura. Fazendo uso de bases de dados oriundas de uma competição internacional – a qual possui uma importante característica de ter sido gravada em dias diferentes (fato que normalmente não ocorre nos demais trabalhos apresentados na literatura), o que faz os resultados se aproximarem com os que os usuários vão encontrar em sistemas de interface cérebro-computador na realidade.

Um dos passos mais importantes nas interfaces cérebro-máquina é o sistema de classificação de tarefas. Para tanto, foi utilizada a máquina de vetores de suporte, que vem sido utilizada para esta e outras aplicações com alto grau de rendimento e performance. Uma vez que a base de dados utilizada possui quatro classes de pensamentos distintas, foi necessária a aplicação da máquina de vetores de suporte para multiclasses. Para tanto foram comparados dois métodos bastante difundidos na literatura. Ainda foi apresentada um terceiro método de máquina de vetores de suporte multiclasses.

A máquina de vetores de suporte multiclasses proposta não obteve resultados superiores aos das estratégias já existentes, porém obteve soluções comparáveis com relação ao desempenho e com complexidade reduzida. Considerando que este tipo de aplicação será usado potencialmente por sistemas embarcados, essa qualidade pode ser determinante.

No contexto das técnicas meta-heurísticas, foram utilizados quatro métodos conhecidos na literatura e algumas de suas versões melhoradas. Para comparação dos algoritmos, estes foram rodados com cinquenta experimentos para cada problema. Foram utilizadas tabelas com síntese destes experimentos, com os valores máximos,

mínimos, médios e desvio padrão. Com o objetivo de comparar essas técnicas com maior rigor estatístico, foi aplicado o teste de Wilcoxon, que é capaz de mostrar se um algoritmo é, ou não, estatisticamente superior a outro para um determinado conjunto de problemas.

Além das técnicas originadas da literatura, um algoritmo meta-heurístico foi proposto fazendo a hibridização de duas outras técnicas. Um importante fator deste algoritmo proposto é que o mesmo não exige a definição de nenhum parâmetro de controle pelo usuário, todos os parâmetros utilizados pelo algoritmo são adaptados dinamicamente pelo mesmo a fim de melhor se adequar ao problema a ser tratado. Em geral, este tipo de estratégia pode ter convergência lenta, entretanto, os dois algoritmos utilizados na hibridização são robustos com relação a seus parâmetros, o que supri esta deficiência.

Após a execução de todos os algoritmos no sistema de interface cérebro-computador utilizando as diferentes técnicas multiclasses da máquina de vetores de suporte e alguns extratores de características apresentados na literatura, as tabelas de comparação e os resultados do teste de Wilcoxon mostraram que os algoritmos HEDRA e o JADE obtiveram melhores resultados, com uma pequena superioridade pelo algoritmo proposto HEDRA para o conjunto de problemas testado. Entretanto, Wolpert e Macready (1997) com o teorema de *No Free Lunch* e Chen *et al.* (2009) com o teorema *Optimal Contraction* mostram que é impossível dizer que uma meta-heurística é melhor que todas as outra para todos os problemas - dessa forma, a conclusão de superioridade do algoritmo HEDRA é restrita para o conjunto de problemas tratado neste trabalho.

Os resultados também mostraram que outros componentes têm importantes impactos no sistema de interface cérebro-computador, tal como a escolha do método de extração de características.

O sistema com melhores resultados foi comparado aos valores apresentados na competição internacional de Berlim, de forma que o sistema apresentado nesse trabalho teria atingido a terceira posição na classificação final. Neste ponto é importante ressaltar que os concorrentes, que participaram da competição, fazem parte de grandes grupos de pesquisa que tem como principal objetivo o estudo deste tipo de sistemas, de forma que a obtenção de resultados comparáveis aos encontrados por esses grupos é muito

significante. Espera-se que os resultados obtidos neste trabalho possam ser utilizados em futuros trabalhos no desenvolvimento dessa importante tecnologia.

REFERÊNCIAS

ANDERSON, K. D. Targeting recovery: Priorities of the spinal cord-injured population. **Journal of Neurotrauma**, v. 1, p. 1371-1383, 2004.

ANGELINE, P. J. Using selection to improve particle swarm optimization **Proceedings of the IEEE Congress on Evolutionary Computation.** p. 84-89, 1998.

BENGOETXEA, E.Inexact Graph Matching Using Estimation of Distribution Algorithm . 259 f. Tese (Doutorado em Engenharia da Computação) Escola Nacional Superior de Telecomunicações, Paris, 2002.

BENSCH, M.; KARIM, A.A.; MELLINGER, J.; HINTERBERGER, T.; TANGERMANN, M.; BOGDAN, M.; ROSENSTIEL, W.; BIRBAUMER, N. Nessi: An EEG-controlled web browser for severely paralyzed patients. **Computation Intelligence and Neuroscience***,* 2007.

BHANDARI, R.; NEGI, S.; SOLZBACHER, F. Wafer-scale fabrication of penetrating neural microelectrode arrays. **Biomedical Microdevices,** v. 12, p. 797–807, 2010.

BINITHA, S.; SATHYA, S. S. A Survey of Bio inspired OptimizationAlgorithms, **International Journal of Soft Computing and Engineering**, v. 2, n. 2, p. 137 – 151, 2012.

BRUNNER, C.; LEEB, R.; MULLER-PUTZ, G. R.; SCHLOGL, A.; PFURTSCHELLER, G. Graz data set A. BCI Competition 2008, 2008. Disponível em: < http://www.bbci.de/competition/iv/desc_2a.pdf>. Acesso em: 20/03/2014

BURGES, C.J.C. A tutorial on support vector machines for pattern recognition. **Data Mining Knowledge Discovery**, v. 2, p. 121–167, 1998.

CHAPELLE , O; VAPNIK, V; BOUSQUET, O; MUKHERJEE, S. Choosing Multiple Parameters for Support Vector Machines. **Machine Learning,** v. 46, n. 1-3, p. 131-159, 2002.

CHAPIN, J. K.; MOXON, K. A.; MARKOWITZ, R. S.; NICOLELIS, M. A. L. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. **Nature Neuroscience**, v. 2, p. 664–670, 1999.

CHEN, J.; XIN, B.; PENG, Z.; DOU, L.; ZHANG, J. Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization. **IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans** v. 39, n. 3, p. 680-691, 2009.

CHIANG, C.W.; LEE, W.P.; HEH, J.S. A 2-Opt based differential evolution for global optimization. **Applied Soft Computing**, v.10, p.1200–1207, 2010.

CINCOTTI, F.; MATTIA, D.; ALOISE, F.; BUFALARI, S.; SCHALK, G.; ORIOLO, G.; CHERUBINI, A.; MARCIANI, M.G.; BABILONI, F. Non-invasive brain-computer interface system: Towards its application as assistive technology. **Brain Research Bulletin**, v.75, p. 796–803, 2008.

CIVICIOGLU, P. Backtracking search optimization algorithm for numerical optimization problems. **Applied Mathematics and Computation**. v. 219, n. 15, p. 8121 – 8144, 2013.

COELHO, L. S. A quantum particle warm optimizer with chaotic mutation operator. **Chaos, Solitons & Fractals**, v. 37, n. 5, p. 1409 – 1418, 2008.

COELHO, L. S.; MARIANI, V. C. Particle swarm approach based on quatum mechanics and harmonic oscillator potential well for economic load dispatch with valve-points effects. **Energy conversion and Management**, v. 39, n. 11, p. 3080 – 3085, 2008.

SABAT, S. L.; COELHO, L. S.; ABRAHAM, A. MESFET DC model parameter extraction using quantum particle swarm optimization. **Microeletronics Reliability**. v. 49, n. 6, p. 660 – 666, 2009.

DAS, S.; SUGANTHAN, P.N. Differential Evolution: A survey of the State-of-the-Art. IEEE, 2010.

DAS, S.; MAITY, S.; QU, B. Y.; SUGANTHAN, P. N. Real-parameter evolutionary multimodal optimization - A survey of the state-of-the-art. **Swarm and Evolutionary Computation**, v. 1, n. 2, p. 71 – 88, 2011.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1, p. 3-18, 2011.

FARWELL, L.A.; DONCHIN, E. Talking off the top of your head: Toward a mental prosthesis utilizing Tevent-related brain potentials. **Electroencephalography Clinical Neurophysiology***,* v. 70, p. 510–523, 1988.

FOGEL, D.B. An introduction to simulated optimization. , v.5, n.1, 1994.

FORSLUND, P. **A Neural Network Based Brain-Computer Interface for Classification of Movement Related EEG**. 136 f. Dissertação (Mestrado em Física Aplicada e Engenharia Elétrica) – Instituto de Tecnologia, Universidade de Linkoping, Suécia, 2003.

FRIEDMAN, J. **Another approach to polychotomous classification.** Stanford University, Department of Statistics, 1996. Relatório técnico.

FUKUNAGA, K. **Introduction to Statistical Pattern Recognition**. 2. Editora New York: Academic, 1990.

FURDEA, A.; HALDER, S.; KRUSIENSKI, D.J.; BROSS, D.; NIJBOER, F.; BIRBAUMER, N.; KÜBLER, A. An auditory oddball (P300) spelling system for brain-computer interfaces. **Psychophysiology**, v. 46, p. 617–625, 2009.

GILJA, V.; CHESTEK, C. A.; DIESTER, I; HENDERSON, J. M.; DEISSEROTH, K.; SHENOY, K. V. Challenges and Opportunities for Next-Generation Intracortically Based Neural Prostheses. **IEEE Transactions on Biomedical Engineering**, v. 58, n. 7, p. 1891-1899, 2011.

GRAY, F. J. **Anatomy for the medical clinician**. 1. ed. Victoria: Shannon Books Pty Ltd, 2002.

HOCHBERG, L. R., BACHER, D., JAROSIEWICZ, B., MASSE, N. Y., SIMERAL, J. D., VOGEL, J. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. **Nature**, v. 485, p. 372–375, 2012.

HUANG, V. L. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. **IEEE Congress on Evolutionary Computation**, p. 17 – 24, Vancouver, Canada, 2006.

KARABOGA, D, An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

KARIM, A.A.; HINTERBERGER, T.; RICHTER, J.; MELLINGER, J.; NEUMANN, N.; FLOR, H.; KÜBLER, A.; BIRBAUMER, N. Neural internet: Web surfing with brain potentials for the completely paralyzed. **Neurorehabilitation Neural Repair**, v. *20*, p. 508–515, 2006.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. **Proceedings of IEEE International Conference on Neural Networks**, Piscataway, NJ, p. 1942-1948, 1995.

KENNEDY, P. R.; BAKAY, R. A. Restoration of neural output from a paralyzed patient by a direct brain connection. **NeuroReport**, v. 9, p. 1707–1711, 1998.

LEBEDEV, M. A.; NICOLELIS, M. A. L. Brain-machine interfaces: past, present and future. **TRENDS in Neurosciences,** v. 29, n. 9, 2006.

LEE, F.; SCHERER, R.; LEEB, R.; NEUPER, C.; BISCHOF, H.; PFURTSCHELLER, G. A Comparative Analysis of Multi-class EEG Classification for Brain-computer Interface. **10th Computer Vision Winter Workshop**. Technical University of Graz, Austria, 2005.

LEHTONEN, J. **EEG-based Brain Computer Interfaces**. 120 f. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia e Comunicações, Elétrica, Universidade de Tecnologia de Helsinki, Helsinki, 2002.

LIANG, J. J.; QIN, A. K.; SUGANTHAN, P. N.; BASKAR, S. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. **IEEE Transactions on Evolutionary Computations**. v. 10, n. 3, p. 281-295, 2006.

LOTTE, F.; CONGEDO, M.; LÉcuyer, A; LAMARCHE, F.; ARNALDI, B. A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces. **Journal of Neural Engineering**, v. 4, 2007.

MEZURA-MONTES, E.; VELAZQUEZ-REYES, J.; COELLO, C.A. A comparative study of differential evolution variants for global optimization. **Genetic Evolutionary Computation Conference**, p.485–492, 2006.

MULLER-PUTZ, G.R.; PFURTSCHELLER, G. Control of an Electrical Prosthesis With an SSVEP-Based BCI. **IEEE Transactions on Biomedical Engineering**, v. 55, p. 361–364, 2008.

NICOLAS-ALONSO, L. F.; GOMEZ-GIL, J. Brain Computer Interfaces, a Review. **Sensors**, Basel, v. 12, p. 1211-1279, 2012.

NUDO, R. J. Adaptive plasticity in motor cortex: Implications for rehabilitation after brain injury. **Journal of Rehabilitation Medicine**, v. 41, p. 7–10, 2003.

PENFIELD, W.; RASMUSSEN, T. **The Cerebral Cortex of Man**. New York: Macmillan, 1950.

PETERS, B. O.; PFURTSCHELLER, G.; FLYVBJERG, H. Automatic Differentiation of Multichannel EEG Signals. **IEEE Transactions on Biomedical Engineering**, v. 48, n. 1, 2001.

PFURTSCHELLER, G.; GUGER, C.; MÜLLER, G.; KRAUSZ, G.; NEUPER, C. Brain oscillations control hand orthosis in a tetraplegic. **Neuroscience Letters**, p. 292:211–214, 2000.

PFURTSCHELLER, G.; MÜLLER, G.R.; PFURTSCHELLER, J.; GERNER, H.J.; RUPP, R. ‘Thought’-control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia. **Neuroscience Letters** v. 351, p. 33–36, 2003.

PHILIPS, J.; DEL R. MILLAN, J.; VANACKER, G.; LEW, E.; GALAN, F.; FERREZ, P.W.; VAN BRUSSEL, H.; NUTTIN, M. Adaptive Shared Control of a Brain-Actuated Simulated Wheelchair. **In Proceedings of the IEEE 10th International Conference on Rehabilitation Robotics***,* Holanda, p. 408–414, 2007.

PRICE, K. **An introduction to differential evolution: New Ideas in Optimization**, Editora London: McGraw-Hill, 1999.

PRICE, K.; STORN, R.; LAMPINEN, J. **Differential Evolution - A Practical Approach to Global Optimization**. Berlin, Alemanha: Springer, 2005.

PURVES, D., AUGUSTINE, G.J., FITZPATRICK, D., KATZ, L.C. LAMANTIA, A.S.; MCNAMARA, J.O. **Neuroscience**. 3. ed. Sunderland, Inglaterra, 2004.

REBSAMEN, B.; CUNTAI, G.; HAIHONG, Z.; CHUANCHU, W.; CHEELEONG, T.; ANG, M.H.; BURDET, E. A brain controlled wheelchair to navigate in familiar environments. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, v. 18, p. 590–598, 2010.

ROIJENDIJK, L. **Variability and nonstationarity in brain computer interfaces**. 45f. Dissertação (Mestrado em Inteligência Artificial). Radboud University, Holanda, 2009.

SAINI, L.M.; AGGARWAL, S.K.; KUMAR, A. Parameter optimisation using genetic algorithm for support vector machine-based price-forecasting model in National electricity market. **IET Generation, Transmission and Distribution**, v. 4, n. 1, p. 36–49, 2010.

SALADIN, K. **The central nervous system**. *Anatomy and Physiology: The Unity of Form and Function*. New York: McGraw-Hill, 2001.

SAMADZADEGAN, F.; SOLEYMANI, A.; ABBASPOUR, R.A. Evaluation of Genetic Algorithms for tuning SVM parameters in multi-class problems. **11th International Symposium on Computational Intelligence and Informatics**, 2010.

SCHLÖGL, A.; KEINRATH, C.; ZIMMERMANN, D.; SCHERER, R.; LEEB, R.; PFURTSCHELLER, G. A fully automated correction method of EOG artifacts in EEG recordings. **Clinical Neurophysiology**. v. 118, n. 1, p. 98-104, 2007.

SHI, Y.; EBERHART, R. C. A modified particle swarm optimizer, **Proceedings of the IEEE Congress on Evolutionary Computation**,p.. 69-73, 1998.

SHI, Y.; EBERHART, R. C. Fuzzy adaptive particle swarm optimization. **Proceedings of IEEE Congress on Evolutionary Computation**, v. 1, p. 101-106, 2001.

SHILTON, A. **Design and Training of Support Vector Machines**. Doctor of Philosophy Thesis (Department of Electrical and Electronic Engineering) – The University of Melbourne, Australia, 2006.

SMITH, R. C. **Electroencephalograph based Brain Computer Interfaces.** 159 f. Dissertação (Mestrado em Ciência da Engenharia) – Faculdade de Engenharia Elétrica e Eletrônica, Universidade Colégio de Dublin, Dublin, Irlanda, 2004.

SOUZA, B. F.; CARVALHO, A. F.; CALVO, R.; ISHII, R. P. Multiclass SVM Model Selection Using Particle Swarm Optimization. **Proceedings of the Sixth International Conference on Hybrid Intelligent Systems**, 2006.

SUGANTHAN, P. N.; HANSEN, N.; LIANG, J. J.; DEB, K.; CHEN, Y. P.; AUGER, A.;TIWARI, S. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. **Congress on Evolutionary Computation**. Nanyang Technological University. Singapore, 2005.

TANAKA, K.; MATSUNAGA, K.; WANG, H.O. Electroencephalogram-based control of an electric wheelchair. **IEEE Transactions on Robotics**, v. *21*, p. 762–766, 2005.

VAPNIK, V.N. **Statistical Learning Theory**. New York: Wiley, 1998.

VELLISTE, M.; PEREL, S.; SPALDING, M. C.; WHITFORD, A. S.; SCHWARTZ, A. B. Cortical control of a prosthetic arm for self-feeding. **Nature**, v. 453, p. 1098–1101, 2008.

WANG, Y.; CAI, Z.; ZHANG, Q. Differential evolution with composite trial vector generation strategies and control parameters. **IEEE Transactions on Evolutionary Computation**, v. 15, n. 1, p. 55-66, 2011.

WESSBERG, J.; STAMBAUGH, C. R.; KRALIK, J. D.; BECK, P. D.; LAUBACH, M.; CHAPIN, J. K. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. **Nature**, v. 408, p. 361–365, 2000.

WESTON, J.; WATKINS, C. Multi-class support vector machines. Egham, UK: Royal University of London, Department of computer science, 1998.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67-82, 1997.

YANG, R. **Signal Processing for a Brain Computer Interface**. 97 f. Dissertação (Mestrado em Ciência da Engenharia) – Escola de Engenharia Elétrica e Eletrônica, Universidade de Adelaide, Adelaide, Austrália, 2009.

ZHANG, J.; SANDERSON, A. C. JADE: adaptive differential evolution with optional external archive. **IEEE Transaction on Evolutionary Computation**, v. 13, n. 5, p. 945-

958, 2009.