UNIVESIDADE ESTADUAL PAULISTA

Júlio de Mesquita Filho

Pós-Graduação em Ciência da Computação

Douglas Rodrigues

Seleção de Características Utilizando Algoritmos Evolucionistas e suas Aplicações em Reconhecimento de Padrões

UNESP

2014

Rodrigues, Douglas.
Seleção de características utilizando algoritmos evolucionistas e suas aplicações em reconhecimento de padrões / Douglas Rodrigues. -- São José do Rio Preto, 2014
44 f. : il., gráfs., tabs.

Orientador: João Paulo Papa
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Processamento de imagens - Técnicas digitais. 3. Reconhecimento de padrões. 4. Aprendizado do computador. 5. Floresta de caminhos ótimos. 6. Algoritmos evolutivos. I. Papa, João Paulo. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 518.72:76

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Douglas Rodrigues

Prof. Dr. João Paulo Papa (Orientador)

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação - Área de Concentração em Computação Aplicada como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UNESP

2014

Departamento de Computação
Universidade Estadual Paulista

Douglas Rodrigues

Abril de 2014

Seleção de Características Utilizando Algoritmos Evolucionistas e suas Aplicações em Reconhecimento de Padrões

Banca Examinadora:

- Prof. Dr. João Paulo Papa (Orientador)
- Profa. Dra. Roberta Spolon (DCo/FC/UNESP)
- Prof. Dr. Alexandre Luís Magalhães Levada (DC/UFSCar)

Agradecimentos

Agradeço, primeiramente, Deus pelo dom da vida. Aos meus pais, Ivone e Osmar, por todo o incentivo e amor dedicado. Ao meu orientador, Prof. Dr. João Paulo Papa, pela oportunidade, amizade, conhecimento e confiança.

Agradeço, também, os amigos que fiz durante essa jornada, pelo conhecimento compartilhado e pela disposição em ajudar: Luís, Clayton, Mizobe, Rafael, Adriana, Luís (Japa), Silas e muitos outros. Ao Prof. Dr. João Francisco Escobedo, pela oportunidade, amizade e o incentivo.

Ao grupo RECOGNA, Universidade Estadual Paulista, professores e funcionários do Departamento de Computação. À CAPES, pela ajuda financeira para o desenvolvimento deste trabalho.

Enfim, a todos que colaboraram direta ou indiretamente com este trabalho.

“Escolha um trabalho de que gostes,
e não terás que trabalhar nem um dia na tua vida.”
Confúcio

Resumo

Técnicas para seleção de características tem sido amplamente estudadas pela comunidade científica de reconhecimento de padrões e áreas afins, dado que o problema de encontrar o subconjunto das características que maximiza a taxa de acerto de uma técnica de classificação de padrões pode ser modelado como um problema de otimização. Metodologias baseadas em inteligência evolucionista, tais como aquelas que simulam dinâmicas sociais e de interação entre morcegos, algumas espécies de aves e outros insetos, tem sido recentemente aplicadas nesse contexto. Assim sendo, o presente trabalho visou o estudo e desenvolvimento de técnicas de seleção de características utilizando abordagens de otimização evolucionistas, sendo elas: BBA - *Binary Bat Algorithm*, BCSS - *Binary Charged System Search*, BCS - *Binary Cuckoo Search*, BKH - *Binary Krill Herd* e BSSO - *Binary Social-Spider Optimization*. Experimentos realizados em seis bases de dados utilizando as técnicas propostas em conjunto com outras cinco técnicas (BGA - *Binary Genetic Algorithm*, BPSO - *Binary Particle Swarm Optimization*, BFA - *Binary Firefly Algorithm*, BGSA - *Binary Gravitational Search Algorithm*, BHS - *Binary Harmony Search*) mostraram a eficácia das técnicas evolucionistas propostas quando utilizadas em conjunto com o classificador OPF. O BSSO - *Binary Social-Spider Optimization* apresentou a melhor acurácia em 3 bases, chegando a aumentar a taxa de acerto do classificador OPF em até 19%, bem como, selecionou o menor número de características em cinco das seis bases. Em relação ao tempo de execução, o BKH - *Binary Krill Herd* obteve o segundo melhor tempo em cinco bases, ficando atrás somente do BHS - *Binary Harmony Search*.

Abstract

Techniques for feature selection have been widely studied by the pattern recognition scientific community and related fields, as the problem of finding the subset of features that maximizes the classifier rate can be modeled as a optimization problem. Methodologies based on evolutionary intelligence, such as those that simulate social dynamics and interaction between bats, some species of birds and other insects, have recently been applied in this context. Therefore, this work aimed to the study and development of feature selection techniques using evolutionary optimization approaches: BBA - *Binary Bat Algorithm*, BCSS - *Binary Charged System Search*, BCS - *Binary Cuckoo Search*, BKH - *Binary Krill Herd* e BSSO - *Binary Social-Spider Optimization*. Experiments conducted in six databases using the proposed techniques together with five other techniques (BGA - *Binary Genetic Algorithm*, BPSO - *Binary Particle Swarm Optimization*, BFA - *Binary Firefly Algorithm*, BGSA - *Binary Gravitational Search Algorithm*, BHS - *Binary Harmony Search*) have shown the effectiveness of proposed evolutionary techniques when used with the OPF classifier. The BSSO - *Binary Social-Spider Optimization* showed the best accuracy on 3 datasets coming to increase the OPF classification rate in up to 19%. Also, SSO has selected the smallest number features in five of the six datasets. Regarding the runtime, BKH - *Binary Krill Herd* was the second fastest technique in five datasets, being only slower then BHS - *Binary Harmony Search* technique.

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

Técnicas de classificação de padrões tem sido amplamente aplicadas nas mais diversas áreas do conhecimento, desde reconhecimento automático de plantações em imagens obtidas por sensoriamento remoto, passando por auxílio ao diagnóstico médico em sistemas especialistas até identificação de intrusões em redes de computadores [1].

A ideia consiste, basicamente, em extrair características relevantes do domínio do problema em questão e criar conjuntos de treinamento e teste para o aprendizado do comportamento dos dados e posterior avaliação do classificador de padrões. De acordo com a quantidade de conhecimento *a priori* do conjunto de dados de treinamento, podemos dividir as técnicas em três tipos: (i) supervisionadas, nas quais o rótulo de todas as amostras do conjunto de treinamento é conhecido, (ii) semi-supervisionadas, onde são conhecidos os rótulos de apenas um subconjunto das amostras de treinamento e (iii) não supervisionadas, também conhecidas por técnicas de agrupamento, nas quais nenhuma informação sobre as amostras é disponibilizada [1].

Visando cada vez mais a especialidade das áreas de pesquisa, uma boa parte dos pesquisadores de reconhecimento de padrões tem se dedicado a estudar técnicas de representação e descrição de objetos (dados em geral) com o intuito de se obter melhor separabilidade do espaço de características e, consequentemente, melhores taxas de reconhecimento. Nesse contexto, podemos dividir a etapa de representação e descrição de objetos nas seguintes fases: (i) extração de características, (ii) ponderação e seleção de características e (iii) fusão de características. Enquanto a primeira etapa é responsável pela escolha e aplicação de algoritmos para extração de informações sobre o domínio do problema, a última objetiva combinar as características que, muitas vezes, são obtidas

por diferentes metodologias, com o intuito de melhorar a taxa de classificação do sistema. Um bom exemplo seria a extração de características de cor, forma e textura em imagens objetivando a combinação das mesmas.

A etapa de seleção de características possui o objetivo de encontrar o subconjunto de características que são realmente relevantes para o problema. Essa etapa é bastante interessante, dado que podemos identificar características que não são utilizadas pelo classificador de padrões em seu aprendizado e que, muitas vezes, são bastante onerosas de serem extraídas. Adicionalmente, temos que a seleção de características pode ser modelada como sendo um problema de otimização, pois a ideia é escolher o subconjunto das características que maximiza a taxa de reconhecimento do classificador, por exemplo. Uma outra possibilidade seria escolher as características que maximizam a separabilidade das amostras o que, de fato, acaba refletindo na taxa de acerto do classificador.

Dentre as técnicas de otimização, uma considerável atenção por parte dos pesquisadores tem sido dada às metodologias baseadas em inteligência evolucionista, ou seja, abordagens que propõem resolver diversos problemas, dentre eles otimização, utilizando conceitos baseados em dinâmicas sociais e comportamento de diversos seres vivos. Dentre as mais conhecidas, podemos citar Algoritmo Genético (*Genetic Algorithm* - GA) [2], Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) [3], Busca Harmônica (*Harmony Search* - HS) [4] e o Algoritmo de Busca Gravitacional (*Gravitational Search Algorithm* - GSA) [5], dentre outras.

Huang et al. [6] propuseram uma abordagem para seleção de características baseada em GA, e Firpi e Goodman [7] aplicaram o *Binary PSO* (BPSO) proposto por Kennedy e Eberhart [8] no mesmo contexto. Falcon et al. [9] propuseram uma versão binária do *Firefly Algorithm* (FA) proposto por Yang [10] no contexto de identificação de falhas de sistemas distribuídos. Posteriormente, Palanisamy e Kanmani [11] aplicaram o *Binary Artificial Bee Colony* (BABC) proposto por Karaboga e Basturk [12] para solucionar problemas de seleção de características. Rashedi et al. [13] propuseram o *Binary Gravitational Search Algorithm* (BGSA), o qual é uma versão do algoritmo GSA [5] para espaços de busca binários. Em seguida, Ramos et al. [14] propuseram aplicar o BGSA no contexto de caracterização de perdas comerciais em sistemas elétricos de energia. Temos,

também, para o contexto de seleção de características, o *Binary Bat Algorithm* (BBA), *Binary Charged System Search* (BCSS) e o *Binary Cuckoo Search* (BCS) [15, 16, 17].

Com a finalidade de guiar o processo de convergência das técnicas evolucionistas a cada iteração, ou seja, direcionar os agentes para a solução ótima, faz-se necessário o uso de uma função de aptidão (ajuste). Neste trabalho a função a ser otimizada e que guiará o processo de busca será a taxa de acerto do classificador Floresta de Caminhos Ótimos (*Optimum-Path Forest* - OPF) [18, 19]. No entanto, vale ressaltar que qualquer classificador pode ser utilizado neste processo. Desta forma, o presente trabalho objetiva o estudo e a aplicação de novas técnicas de inteligência evolucionista no contexto de seleção de características, tais como *Binary Bat Algorithm* [15], *Binary Charged System Search* [16], *Binary Cuckoo Search* [17], *Binary Krill Herd* e *Binary Social-Spider Optimization*. As Seções 2 e 3 apresentam, respectivamente, o classificador OPF e uma breve descrição dos algoritmos evolucionistas. Na Seção 4 é apresentada a metodologia utilizada e na Seção 5 os experimentos realizados e os resultados obtidos. Finalmente, na Seção 6 são feitas conclusões do trabalho realizado e atividades futuras.

2 Classificador de Padrões Baseado em Floresta de Caminhos Ótimos - OPF

Esta seção tem por objetivo apresentar o classificador baseado em floresta de caminhos ótimos com aprendizado supervisionado. Tal classificador modela o problema de reconhecimento de padrões como um problema de floresta de caminhos ótimos em um grafo definido no espaço de atributos, onde os nós são as amostras, as quais são representadas pelos seus respectivos vetores de atributos, e os arcos são definidos de acordo com uma relação de adjacência pré-estabelecida.

Nesta versão, os arcos são ponderados, e diversas funções de custo podem ser empregadas com o intuito de particionar o grafo em árvores de caminhos ótimos, as quais são enraizadas pelos seus respectivos protótipos (sementes) na fase de treinamento. O rótulo de uma amostra a ser classificada é o mesmo do protótipo mais fortemente conexo a ela.

2.1 Classificação supervisionada

O algoritmo OPF com grafo completo foi, primeiramente, apresentado por Papa et al. [18, 19] e tem sido amplamente utilizado em diversas aplicações. A técnica utilizada neste trabalho modela as amostras como sendo os nós de um grafo completo, onde os elementos mais representativos de cada classe do conjunto de treinamento, isto é, os protótipos, são escolhidos como sendo os elementos pertencentes às regiões de fronteira entre as classes.

Os protótipos participam de um processo de competição disputando as outras amostras oferecendo-lhes caminhos de menor custo e seus respectivos rótulos. Ao final deste processo, obtemos um conjunto de treinamento particionado em árvores de caminhos óti-

mos, sendo que a união das mesmas nos remete a uma floresta de caminhos ótimos. Esta abordagem apresenta vários benefícios com relação a outros métodos de classificação de padrões supervisionados: (i) é livre de parâmetros, (ii) possui tratamento nativo de problemas multiclasses e (iii) não faz alusão sobre forma e/ou separabilidade das classes. As próximas seções irão discutir a fundamentação teórica e os algoritmos de treinamento e classificação do algoritmo baseado em OPF utilizando grafo completo.

2.1.1 Fundamentação Teórica do Classificador OPF

Seja $Z = Z_1 \cup Z_2 \cup Z_3$ um conjunto de dados, tais que Z_1, Z_2 e Z_3 denotam os conjuntos de treinamento, avaliação e teste, respectivamente. Seja (Z_1, A) um grafo completo cujos nós são amostras em Z_1 e qualquer par de amostras define um arco em $A = Z_1 \times Z_1$. Os arcos não precisam ser armazenados, sendo a representação do grafo de maneira implícita. Definimos também o caminho como uma sequência de amostras distintas $\pi_t = \langle s_1, s_2, \ldots, s_{k-1}, s_k \rangle$ com término t, onde $(s_i, s_{i+1}) \in A$ para $1 \leq i \leq k-1$. O caminho é dito *trivial* se $\pi_t = \langle t \rangle$. Nós associamos a cada caminho π_t um custo $f(\pi_t)$ dado pela função de conectividade f. Um caminho π_t é considerado ótimo se $f(\pi_t) \leq f(\tau_t)$ para qualquer outro caminho τ_t. Denotamos, também, $\pi_s \cdot \langle s, t \rangle$ a concatenação de um caminho π_s e arco (s, t).

Dado com um conjunto de amostras *protótipos* $S \in Z_1$, a fase de treinamento do OPF consiste em calcular uma floresta de caminhos ótimos sobre o conjunto de treinamento. Essa floresta é, essencialmente, a coleção de árvores de caminhos ótimos com raízes em cada protótipo. A partição (árvore) da grafo completo da floresta de caminhos ótimos é computado nos $\Re^n$ pelo algoritmo da transformada imagem-floresta (IFT) [20].

Considere, agora, $R(s) \in S$ como a raiz de $s \in Z_1$. Dizemos que s é mais fortemente conectada com $R(s)$ do que qualquer outra raiz $u \in S$ em Z_1. Isto significa que, uma vez s é conquistada por alguma amostra, a qual pode ser $R(s)$ ou alguma outra amostra $v \in Z_1$ tal que $R(v) = R(s)$, s pertence à mesma árvore de caminhos ótimos de $R(s)$.

Como dito anteriomente, as amostras em S são escolhidas como sendo as mais próximas com rótulos diferentes em Z_1. Para encontrar tais protótipos, Papa et al. [18]

propuseram computar uma Árvore de Espalhamento Mínima (*Minimum Spanning Tree* - MST) em Z_1 e marcar as amostras conexas com rótulos diferentes como sendo os protótipos. O algoritmo 1 implementa a fase de treinamento do OPF.

Algoritmo 1 – Treinamento por Floresta de Caminhos Ótimos

Entrada:	Um conjunto de treinamento Z_1 λ-rotulado e o par (v, d) composto pelo vetor de características e as distâncias computadas.
Saída:	Floresta de Caminhos Ótimos P_1, mapa de custo C_1, mapa de rótulos L_1, e o conjunto ordenado Z_1'.
Auxiliares:	Fila de prioridade Q, conjunto S de protótipos, e variável de custo *cst*.

1. *$Z_1' \leftarrow \emptyset$ e compute via MST o conjunto de prototipos $S \subset Z_1$.*
2. *Para cada $s \in Z_1 \backslash S$, faça $C_1(s) \leftarrow +\infty$.*
3. *Para cada $s \in S$, faça*
4. *$C_1(s) \leftarrow 0$, $P_1(s) \leftarrow nil$, $L_1(s) \leftarrow \lambda(s)$, insira s em Q.*
5. *Enquanto $Q \neq \emptyset$, faça*
6. *Remova de Q uma amostra s tal que $C_1(s)$ e minimo.*
7. *Insira s em Z_1'.*
8. *Para cada $t \in Z_1$ tal que $C_1(t) > C_1(s)$, faça*
9. *Compute $cst \leftarrow \max\{C_1(s), d(s,t)\}$.*
10. *Se $cst < C_1(t)$, então*
11. *Se $C_1(t) \neq +\infty$, então remova t de Q.*
12. *$P_1(t) \leftarrow s$, $L_1(t) \leftarrow L_1(s)$, $C_1(t) \leftarrow cst$.*
13. *Insira t em Q.*
14. *Retorne o classificador $[P_1, C_1, L_1, Z_1']$.*

Após a fase de treinamento, o processo de classificação encontra o caminho ótimo entre uma amostra de teste $t \in Z_3$ (um procedimento similar é aplicado a amostras de Z_2) e um nó de treinamento, e t recebe o rótulo da amostra em Z_1 que o conquistou, isto é, $L(t) \leftarrow L(P(t))$, onde $P(t)$ denota o predecessor de t no caminho ótimo até $R(t)$. Em outras palavras, $P(t)$ denota a amostra que conquistou t. O Algoritmo 2 apresenta este procedimento.

Algoritmo 2 – Classificação por Floresta de Caminhos Ótimos

Entrada:	Classificador $[P_1, C_1, L_1, Z_1']$, conjunto de teste Z_3, e o par (v, d) composto pelo vetor de características e as distâncias computadas.
Saída:	Rótulo L_2 e predecessor P_2 mapas definidos por Z_3, e valor de acurácia Acc.
Auxiliares:	Variáveis de custo tmp e $mincost$.

1. *Para cada* $t \in Z_3$, *faça*
2. $\quad i \leftarrow 1$, $mincost \leftarrow max\{C_1(k_i), d(k_i, t)\}$.
3. $\quad L_2(t) \leftarrow L_1(k_i)$ *e* $P_2(t) \leftarrow k_i$.
4. $\quad$ *Enquanto* $i < |Z_1'|$ *e* $mincost > C_1(k_{i+1})$, *faça*
5. $\quad\quad$ *Compute* $tmp \leftarrow \max\{C_1(k_{i+1}, d(k_{i+1}, t)\}$.
6. $\quad\quad$ *Se* $tmp < mincost$, *então*
7. $\quad\quad\quad$ $mincost \leftarrow tmp$.
8. $\quad\quad\quad$ $L_2(t) \leftarrow L(k_{i+1})$ *e* $P_2(t) \leftarrow k_{i+1}$.
9. $\quad\quad$ $i \leftarrow i + 1$.
10. *Calcule a acuracia Acc de acordo com [18].*
11. *Retorne* $[L_2, P_2, Acc]$.

O algoritmo baseado em OPF pode ser utilizado com qualquer função de custo suave que pode agrupar amostras com propriedades similares [20]. Na versão OPF com grafo completo a função de custo abordada foi a f_{max}:O algoritmo baseado em OPF associa um caminho ótimo $P^*(s)$ de S a toda amostra $s \in Z_1$, formando uma floresta de caminhos ótimos P (uma função sem ciclos, a qual associa a todo $s \in Z_1$ seu predecessor $P(s)$ em $P^*(s)$, ou uma marca nil quando $s \in S$, como mostrado na Figura 2.1d). Seja $R(s) \in S$ a raiz de $P^*(s)$ a qual pode ser alcançada por $P(s)$.

$$
\begin{aligned}
f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\
f_{max}(\pi \cdot \langle s, t \rangle) &= \max\{f_{max}(\pi), d(s, t)\},
\end{aligned} \tag{2.1}
$$

sendo que $f_{max}(\pi)$ computa a distância máxima entre amostras adjacentes em π, quando π não é um caminho trivial. A Figura 2.1 ilustra todos os passos executados pelos algoritmos de treinamento, classificação e testes do classificador de padrões OPF. O algoritmo baseado em OPF associa um caminho ótimo $P^*(s)$ de S a toda amostra $s \in Z_1$, formando uma floresta de caminhos ótimos P (uma função sem ciclos, a qual associa a todo $s \in Z_1$ seu predecessor $P(s)$ em $P^*(s)$, ou uma marca *nil* quando $s \in S$, como mostrado na Figura 2.1d). Seja $R(s) \in S$ a raiz de $P^*(s)$ a qual pode ser alcançada por $P(s)$.

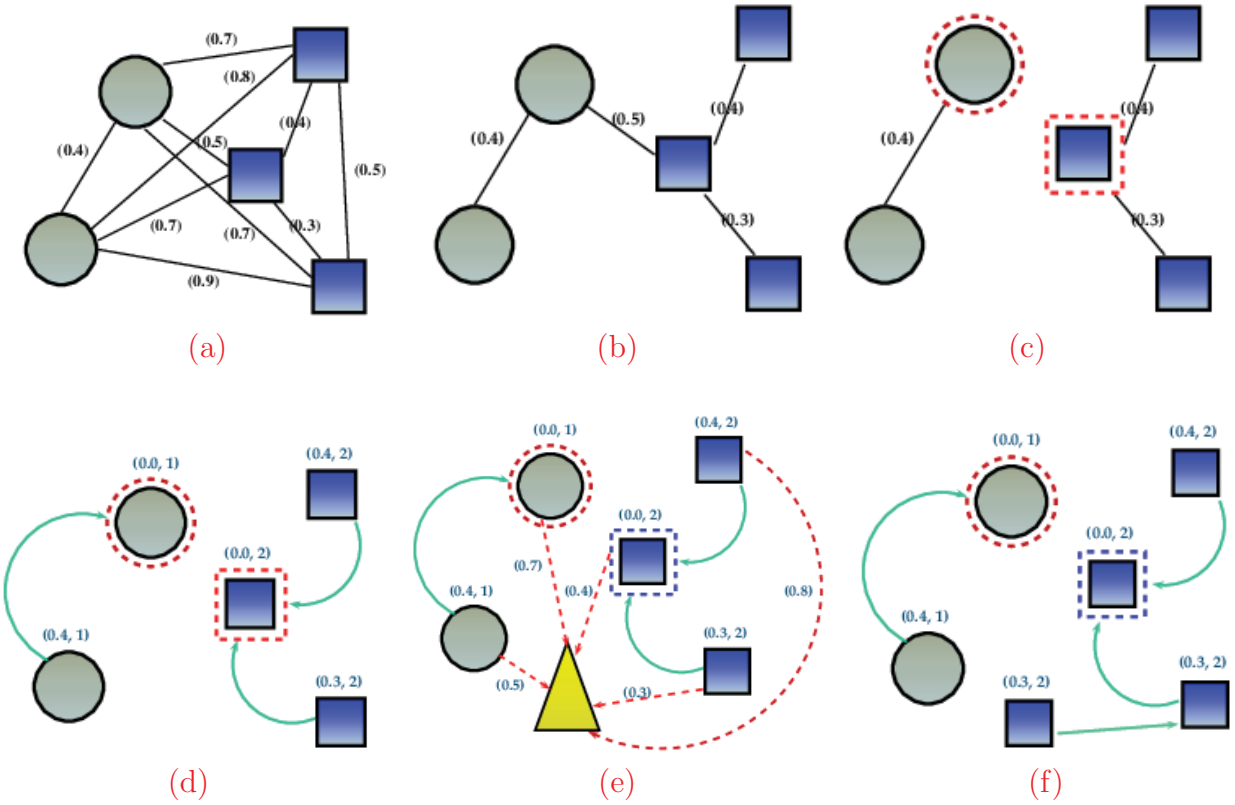


Figura 2.1: (a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) MST do grafo completo. (c) Protótipos escolhidos como sendo os elementos adjacentes de classes diferentes na MST (nós circulados). (d) Floresta de caminhos ótimos resultante para a função de valor de caminho f_{max} e dois protótipos. Os identificadores (x, y) acima dos nós são, respectivamente, o custo e o rótulo dos mesmos. A seta indica o nó predecessor no caminho ótimo. (e) Uma amostra de teste (triangulo) é conectada aos demais nós de ambas as classes (linhas pontilhadas) com os nós do conjunto de treinamento. (f) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2 e o custo de classificação 0.3 são associados a amostra de teste. Note que, mesmo a mostra de teste estando mais próxima de um nó da classe 1 (Círculo), ela foi classificada como sendo da classe 2 (Quadrado).

3 Algoritmos Evolucionistas

Exploration e *exploitation* são dois processos utilizados nos algoritmos evolucionistas para guiar a busca para a solução ótima [21, 22]. Um algoritmo ideal reduz o tamanho das regiões inexploradas, resultando em uma cobertura uniforme no espaço de busca.

1. *Exploration*: Processo que consiste em visitar novas regiões no espaço de busca.
2. *Exploitation*: Processo que consiste em visitar regiões no espaço de busca cuja vizinhança já foi previamente visitada.

Ambos processos devem ser cuidadosamente controlados afim de prover uma cobertura efetiva de todo o espaçco de busca. O algoritmo inicia com o processo de *exploration* permitindo que as regiões do espaço de busca sejam exploradas, e então o processo de *exploitation* é aplicado para refinar tais regiões. No entando, é aconselhado que mesmo na fase de *exploitation* do algoritmo evolucionista haja um baixo nível de *exploration*.

3.1 Genetic Algorithm - GA

O *Genetic Algorithm* (GA) foi proposto por Koza [2]. Baseado nos princípios da seleção natural e sobrevivência do mais apto, teoria fundamentada por Darwin em 1858, o GA é representado por uma população, onde cada indivíduo (cromossomo) é considerado uma possível solução do problema. O GA pode ser decomposto nas fases de inicialização, avaliação, seleção, cruzamento, mutação e finalização.

1. Inicialização - cria a população com as possíveis soluções;

2. Avaliação - avalia-se a aptidão de cada indivíduo para saber o quão bem eles respondem ao problema proposto;

3. Seleção - os indivíduos são selecionados para reprodução de acordo com uma probabilidade baseada na aptidão;

4. Cruzamento - as características dos indivíduos selecionados são recombinadas, gerando um novo indivíduo;

5. Mutação - os indivíduos resultantes do processo de reprodução sofrem alteração em suas características, aumentando a variedade da população;

6. Finalização - as condições de encerramento são verificadas, retornando a etapa de avaliação no caso de não serem atingidas.

3.2 Particle Swarm Optimization - PSO

Recentemente, várias aplicações tem utilizado técnicas evolucionárias como métodos heurísticos para encontrar soluções ótimas ou pseudo-ótimas. Uma atenção particular tem sido dada à técnica denominada Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO), devido à sua simplicidade e eficácia. Basicamente, PSO é um algoritmo modelado em inteligência coletiva que encontra uma solução em um espaço de busca com base na dinâmica do comportamento social [8]. Cada possível solução do problema é modelada como uma partícula do enxame que imita seu vizinho baseando-se em uma função objetivo. Outras definições consideram PSO como um algoritmo de pesquisa baseado em processos estocásticos e populacionais, onde a aprendizagem do comportamento social permite a cada solução possível (partícula) voar dentro desse espaço (enxame) a procura de outras partículas que possuam melhores características e, assim, maximizando a função objetivo. Cada partícula tem uma memória para armazenar sua melhor solução local (máximos locais) e a melhor solução global (máximos globais). Levando-se em conta essas informações, cada partícula tem a capacidade de imitar as outras que proporcionam a ela melhores posições no enxame. Este processo simula a interação social entre um bando de aves a procura de comida, por exemplo. Esse mecanismo sócio-recognitivo pode

ser resumido em três grandes princípios [8]: (i) avaliação, (ii) comparação e (iii) imitação. Cada partícula pode avaliar outras dentro de sua vizinhança através de alguma função objetivo, pode compará-la com seu próprio valor e, finalmente, pode decidir se é uma boa escolha imitá-la. O enxame é modelado como sendo um espaço multidimensional R^m, em que cada particula $p_i = (x_i, v_i) \in R^m$ tem duas características principais: (i) sua posição x_i e (ii) velocidade v_i. A melhor solução (posição no enxame) local $\hat{x_i}$ e global $\hat{s}$ são também conhecidas.

Após a definição do tamanho do enxame, ou seja, o número de partículas, cada uma delas é inicializada com valores aleatórios de velocidade e posição. Cada indivíduo é então avaliado com relação a alguma função objetivo e seu máximo local é atualizado. No final, o valor máximo global é atualizado com a partícula que alcançou a melhor posição no enxame. Este processo é repetido até que algum critério de convergência seja atingido. A posição atualizada e equações da velocidade da partícula p_i da forma mais simples que governam o PSO são, respectivamente, dadas por:

$$v_i = wv_i + c_1 r_1(\hat{x_i} - x_i) + c_2 r_2(\hat{s} - x_i) \tag{3.1}$$

e

$$x_i = x_i + v_i \tag{3.2}$$

onde w é a força de inércia que controla o poder de interação entre as partículas, e r_1 , $r_2 \in [0, 1]$ são variáveis aleatórias que dão a idéia de estocasticidade ao método PSO. As constantes c_1 e c_2 são também utilizadas para guiar as partículas e são definidas como parâmetros de entrada para o algoritmo.

3.3 Firefly Algorithm - FA

O algoritmo dos vaga-lumes foi proposto por Yang [10], e é baseado na bioluminescência que é utilizada para atrair parceiros no período de acasalamento ou atrair suas presas. O brilho de um vaga-lume é determinado por uma função objetivo e é percebido

pela intensidade de luz I dependendo da distância d de sua fonte, como segue:

$$I = I_0 e^{-\iota d}, \tag{3.3}$$

onde I_0 é a intensidade de luz original e ι é o coeficiente de absorção.

Como a atratividade de um vaga-lume é proporcional a intensidade de luz de um vaga-lume adjacente, nós podemos definir a atratividade β de um vaga-lume por

$$\beta = \beta_0 e^{-\iota d^2}, \tag{3.4}$$

onde β_0 é a atratividade em $d = 0$.

Um vaga-lume i é atraído a outro vaga-lume k com o melhor valor de aptidão, e o movimento é realizado de acordo com:

$$x_i^j(t+1) = x_i^j(t) + \beta_0 e^{-\iota d_{i,k}^2}(x_k^j - x_i^j) + \phi\left(\sigma_i - \frac{1}{2}\right), \tag{3.5}$$

onde o segundo termo afirma a atração entre ambos os vaga-lumes, $d_{i,k}^2$ é a distância entre os vaga-lumes i e k, ϕ é um fator de aleatoriedade e $\sigma_i \sim U(0,1)$.

3.4 Gravitational Search Algorithm - GSA

Rashedi et al. [5] propôs um algoritmo de otimização baseado na força gravitacional, que é uma das interações fundamentais da natureza. Essa abordagem, chamada de algoritmo de busca gravitacional, modela cada possível solução como uma partícula no universo, que interage com outras partículas de acordo com a lei gravitacional de Newton [23].

Seja p_i uma partícula no universo, e $x_i \in \mathbb{R}^n$ e $v_i \in \mathbb{R}^n$ sua posição e velocidade, respectivamente. Pode-se definir, no tempo específico t, a força da partícula k agindo na partícula i na j-ésima dimensão, como segue:

$$F_{ik}^{j}(t) = G(t)\frac{M_i(t)M_k(t)}{R_{ik}(t)+\tau}(x_k^j(t) - x_i^j(t)), \tag{3.6}$$

onde $R_{ik}(t)$ é a distância Euclidiana entre as partículas i e k, M_i é a massa da partícula i e τ é uma pequena constante para evitar divisão por zero. G é o potencial gravitacional, que é dado por

$$G(t) = G(t_0)\frac{t_0}{t}^{\zeta}, \ \zeta < 1, \tag{3.7}$$

em que ζ é um parâmetro de controle [24], $G(t)$ é o valor do potencial gravitacional no tempo t, e $G(t_0)$ é o valor do potencial gravitacional no tempo da "criação do universo"que está sendo considerado [24].

Para que o algoritmo de busca gravitacional tenha um comportamento estocástico, Rashedi et al. [5] assumiu que a força total que age na partícula i na dimensão j como uma soma de peso aleatória das forças exercidas por outros agentes:

$$F_i^j(t) = \sum_{k=1, j\neq i}^{m} \sigma_j F_{ik}^j(t), \tag{3.8}$$

em que $\sigma_i \sim U(0,1)$ e m denotam o número de partículas (tamanho do universo).

A aceleração da partícula i no tempo t e dimensão j é dado por

$$a_i^j(t) = \frac{F_i^j(t)}{M_i(t)}, \tag{3.9}$$

em que a massa M_i é calculada como:

$$M_i(t) = \frac{q_i(t)}{\sum_{k=1}^{m} q_k(t)}, \tag{3.10}$$

com

$$q_i(t) = \frac{f_i(t) - w(t)}{b(t) - w(t)}. \tag{3.11}$$

Os termos $w(t)$ e $b(t)$ denotam, respectivamente, as partículas com o pior e o melhor valor de aptidão. O termo $f_i(t)$ representa o valor de aptidão da partícula i.

Finalmente, para evitar soluções ótimas locais, só as melhores b massas, i.e., as com os maiores valores de aptidão. Sendo $\mathcal{B}$ o conjunto dessas massas. O valor de b é definido para b_0 no começo do algoritmo e diminui com o tempo. Assim, Equation 3.8 é escrita como:

$$F_i^j(t) = \sum_{b \in \mathcal{B}, b \neq i} \sigma_b F_{ib}^j(t). \tag{3.12}$$

A velocidade e posição é atualizada seguindo:

$$v_i^j(t+1) = \sigma_i v_i^j(t) + a_i^j(t) \tag{3.13}$$

e

$$x_i^j(t+1) = x_i^j(t) + v_i^j(t+1), \tag{3.14}$$

onde $\sigma_i \sim U(0,1)$.

3.5 Harmony Search - HS

O *Harmony Search* [4] é inspirado no processo musical pela busca por um estado perfeito de harmonia, como é feito durante a improvisação do jazz. Cada músico de jazz toca um possível acorde musical, que juntos, formarão uma harmonia. Caso a harmonia gerada seja boa, esta experiência é guardada em uma memória do músico, para ser usada futuramente, aumentando as chances de melhorar a harmonia gerada em uma próxima rodada.

Analogamente, no processo de otimização cada possível solução chamada harmonia é inicialmente gerada de forma aleatória dentro de um intervalo determinado, e combinadas, geram uma solução de acordo com a função objetivo. Caso esta seja uma boa solução, estas variáveis são guardadas em uma memória, para que sejam usadas na geração de novas soluções, aumentando as possibilidades de melhorar os resultados.

3.6 Bat Algorithm - BA

Morcegos são animais fascinantes e sua capacidade avançada de ecolocalização tem atraído a atenção de pesquisadores de diversas áreas. A ecolocalização funciona como um tipo de sonar: morcegos, principalmente micro-morcegos, emitem um pulso alto e curto de som, esperam atingir algum objeto, e depois de uma fração de tempo, o eco retorna a seus ouvidos [25]. Assim, morcegos conseguem saber quão distantes se encontram de um objeto [26]. Este mecanismo de orientação permite os morcegos saberem distinguir a diferença entre um obstáculo e uma presa, permitindo-os caçar mesmo na escuridão [27].

Baseado no comportamento dos morcegos, Yang [28] desenvolveu uma nova e interessante técnica de otimização chamada algoritmo dos morcegos. Essa técnica foi desenvolvida analisando como um bando de morcegos rastreiam suas presas/comida usando a capacidade de ecolocalização. Para modelar esse algoritmo, Yang [28] idealizou algumas regras, como segue:

1. Todos os morcegos usam ecolocalização para saber a distância, e eles também sabem a diferença entre comida/presa e barreiras de alguma maneira;

2. Um morcego b_i voa aleatoriamente com velocidade v_i na posição x_i com uma frequência fixa $f_{\min}$, comprimento de onda λ e sonoridade A_0 para procurar por presas. Eles podem ajustar automaticamente o comprimento de onda (ou frequência) dos pulsos emitidos e ajustar a taxa de emissão dos pulsos $r \in [0, 1]$, dependendo da proximidade do alvo;

3. Embora a sonoridade posso variar de diferentes maneiras, Yang [28] assumiu que a sonoridade varia de um valor grande (positivo) A_0 para um valor constante mínimo A_{min}.

Primeiramente, a posição inicial x_i, velocidade v_i e frequência f_i são inicializadas para cada morcego b_i. Para cada passo t, sendo T o número máximo de iterações, o movimento dos morcegos é dado pela atualização de suas velocidade e posição utilizando Equação 3.16 e 3.17, como segue:

$$f_i = f_{\min} + (f_{\min} - f_{\max})\beta, \tag{3.15}$$

$$v_i^j(t) = v_i^j(t-1) + [\hat{x}^j - x_i^j(t-1)]f_i, \tag{3.16}$$

$$x_i^j(t) = x_i^j(t-1) + v_i^j(t), \tag{3.17}$$

onde β denota um número gerado aleatóriamente no intervalo $[0, 1]$. Lembrando que $x_i^j(t)$ denota o valor da variável de decisão j para o morcego i no tempo t. O resultado de f_i (Equação 3.15) é usado para controlar a velocidade e a extenção do movimento dos morcegos. A variável $\hat{x}^j$ representa a atual melhor posição global para a variável de decisão j, que é alcançada comparando todas as soluções providas pelos m morcegos.

Com a finalidade de melhorar a variabilidade das possíveis soluções, Yang [28] propôs uma caminhada aleatória. Primariamente, uma solução é selecionada entre as atuais melhores soluções, e então a caminhada aleatória é aplicada a fim de gerar uma nova solução.

$$x_{new} = x_{old} + \epsilon\overline{A}(t), \tag{3.18}$$

em que $\overline{A}(t)$ é a sonoridade média de todos os morcegos no tempo t, e $\epsilon \in [-1, 1]$ representa direção e força da caminhada aleatória. Para cada iteração do algoritmo, a sonoridade A_i e a taxa de emissão do pulso r_i são atualizadas, como segue:

$$A_i(t+1) = \alpha A_i(t) \tag{3.19}$$

e

$$r_i(t+1) = r_i(0)[1 - \exp(-\gamma t)], \tag{3.20}$$

onde α e γ são parâmetros do algoritmo. No primeiro passo do algoritmo, a taxa de emissão $r_i(0)$ e a sonoridade $A_i(0)$ são escolhidas aleatóriamente. Geralmente, $A_i(0) \in [1, 2]$ e $r_i(0) \in [0, 1]$ [28]. No entanto, a sonoridade e a taxa de emissão serão atualizadas somente se novas soluções são encontradas, significando que os morcegos estão se movendo para a solução ótima.

3.7 Charged System Search - CSS

A lei de Coulomb é uma lei que rege a física, sendo também utilizada para descrever as interações entre partículas eletricamente carregadas. Seja uma esfera sólida com raio r e volume de densidade uniforme. A força de atração F_{ij} entre duas esferas i e j com carga total q_i e q_j é definida por:

$$F_{ij} = \frac{k_e q_i q_j}{d_{ij}^2}, \tag{3.21}$$

onde k_e é uma constante chamada de "constante de Coulomb" e d_{ij} é a distância entre as cargas.

Baseado na definição anterior, Kaveh e Talatahari [29] propuseram um novo algoritmo meta-heurístico chamado de *Charged System Search (CSS)*. Neste algoritmo, cada partícula carregada (*Charged Particle* - CP) no sistema é afetada pelos campos elétricos das outras, gerando uma força resultante sobre cada CP, que é determinada pelas leis eletrostáticas. A interação de movimento de uma CP é determinada utilizando as leis mecânicas Newtonianas. No entanto, Kaveh e Talatahari [29] resumiram CSS nas seguintes definições:

- *Definição 1*: A magnitude da carga q_i, com $i = 1, 2, ..., n$, é definida considerando a qualidade das soluções, i.e. valor da função objetivo $fit(i)$:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst}, \tag{3.22}$$

  onde $fitbest$ e $fitworst$ denotam, respectivamente, o melhor e o pior valores de aptidão de todas as partículas. A distância d_{ij} entre duas CPs é dada pela seguinte equação:

$$d_{ij} = \frac{\|\vec{x}_i - \vec{x}_j\|}{\|\frac{\vec{x}_i - \vec{x}_j}{2} - \vec{x}_{best}\| + \epsilon}, \tag{3.23}$$

  em que $\vec{x}_i$, $\vec{x}_j$ e $\vec{x}_{best}$ denotam as posições da i-ésima, j-ésima e da atual melhor CP respectivamente, e ϵ é um número positivo pequeno para evitar singularidades.

- *Definição 2*: A posição inicial $x_{ij}(0)$ e velocidade $v_{ij}(0)$ para cada j-ésima variável da i-ésima CP, com $j = 1, 2, \ldots, m$, são dadas por:

$$x_{ij}(0) = x_{i,min} + \theta(x_{i,max} - x_{i,min}) \tag{3.24}$$

e

$$v_{ij}(0) = 0, \tag{3.25}$$

onde $x_{i,max}$ e $x_{i,min}$ representam os limites superior e inferior respectivamente, e $\theta \sim U(0, 1)$.

- *Definição 3*: Para um problema de maximização, a probabilidade de cada CP mover-se na direção de outras CPs é dada por:

$$p_{ij} = \begin{cases} 1 & \text{se } \frac{fit(j)-fitworst}{fit(i)-fit(j)} > \theta e fit(i) > fit(j), \\ 0 & \text{caso contrário.} \end{cases} \tag{3.26}$$

- *Definição 4*: O valor da força resultante agindo na CP j é definido como:

$$r = 0.1 * \max(x_{i,max} - x_{i,min}) \tag{3.27}$$

$$F_j = q_j \sum_{j,i \neq j} \left(\frac{q_i}{r^3} \cdot d_{ij} \cdot c_1 + \frac{q_i}{d_{ij}^2} \cdot c_2 \right) p_{ij}(\vec{x_i} - \vec{x_j}), \tag{3.28}$$

onde $c_1 = 1$ e $c_2 = 0$ se $d_{ij} < r$, caso contrário $c_1 = 0$ e $c_2 = 1$.

- *Definição 5*: A nova posição e velocidade de cada CP são dadas por:

$$\vec{x}_j(t) = \theta_{j1} \cdot k_a \cdot F_j + \theta_{j2} \cdot k_v \cdot \vec{v}_j(t-1) + \vec{x}_j(t-1) \tag{3.29}$$

e

$$\vec{v}_j(t) = \vec{x}_j(t) - \vec{x}_j(t-1), \tag{3.30}$$

onde $k_a = 0.5(1 + \frac{t}{T})$ e $k_v = 0.5(1 - \frac{t}{T})$ são os coeficientes de aceleração e velocidade, respectivamente, sendo t a iteração atual e T o número máximo de iterações.

- *Definição 6*: O número das melhores soluções encontradas até o momento é armazenado na Memória de Carga (*Charged Memory* - CM). As piores soluções são excluídas da CM, e as melhores novas soluções são incluídas na CM.

3.8 Cuckoo Search - CS

O comportamento parasita, de algumas espécies de pássaros cucos, é extramamente intrigante. Esses pássaros podem hospedar seus ovos em ninhos de outros pássaros. Eles podem imitar características, tais como: cor e manchas. No caso dessa estratégia ser mal-sucedida, o hospedeiro descarta ovo do cuco do ninho ou abandona-o. Baseado nesse contexto, Yang e Deb [30] desenvolveram um novo algoritmo de otimização evolucionista chamado *Cuckoo Search* (CS), e ele pode ser resumido utilizando as seguintes regras:

1. Cada pássaro cuco escolhe um ninho aleatório para depositar seus ovos.
2. O número de ninhos disponíveis é fixo, e os ninhos com os ovos de maior qualidade irão para as gerações seguintes.
3. No caso do pássaro, dono do ninho, descobrir o ovo do cuco, ele descartará o ovo ou abandonará o ninho e construirá um novo. Há um número fixo de ninhos, e a probabilidade que o ovo do cuco tem a ser descoberto pelo pássaro é $p_a \in [0, 1]$.

CS realiza uma combinação balanceada entre a caminhada aleatória local e a caminhada aleatória exploratória global, controlada pelo parâmetro $p_a \in [0, 1]$. A caminhada aleatória local pode ser escrita como:

$$x_i^j(t) = x_i^j(t-1) + \alpha \cdot s \oplus H(p_a - \epsilon) \oplus (x_{k'}^j(t-1) - x_{k''}^j(t-1)), \tag{3.31}$$

onde $x_{k'}^j$ e $x_{k''}^j$ são duas soluções diferentes selecionadas pela permutação aleatória, e x_i^j representa j^{th} ovo no ninho i , $i = 1, 2, \ldots, m$ e $j = 1, 2, \ldots, d$. $H(\cdot)$ é uma função degrau, ϵ é um número aleatório de uma distribuíção uniforme, $\oplus$ denota a multiplicação entre os valores das matrizes e s é o tamanho do passo.

A caminhada aleatória global é realizada utilizando Voos de Lévy, como segue:

$$x_i^j(t) = x_i^j(t-1) + \alpha \cdot L(s, \lambda), \tag{3.32}$$

onde

$$L(s,\lambda) = \frac{\lambda \cdot \Gamma(\lambda) \cdot \sin(\lambda)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, \quad s \gg s_0 > 0 \tag{3.33}$$

3.9 Krill Herd - KH

Krill Herd (KH) é um método baseado em inteligência de bando de camarões e foi proposto por Gandomi e Alavi [31], o qual foi inspirado na simulação de bandos de camarões em responder processos biológicos e ambientais. A posição x_i de um camarão no espaço de busca é governada por três ações principais: (i) movimentação induzida por outros camarões, (ii) atividade de forrageamento, (iii) difusão aleatória, e é atualizada da seguinte forma:

$$x_i(t + \Delta t) = x_i(t) + \Delta t(N_i + F_i + D_i), \tag{3.34}$$

onde,

$$\Delta t = C_t \sum_{j=1}^{d} (UB_j - LB_j), \tag{3.35}$$

e N_i é o movimento induzido por outros camarões, F_i é o movimento de forrageamento, D_i é a difusão física do i-ésimo camarão, d é o número de variáveis a serem otimizadas, UB_j e LB_j são os limites superior e inferior da jth variável, respectivamente, e C_t é um número constante entre $[0, 2]$.

Os camarões tentam manter a densidade e movem-se para seu efeito mútuo. A direção do movimento induzido considerando um individuo i, α_i, é estimado da densidade local do bando (efeito local), a densidade alvo do bando (efeito alvo), e a densidade de repulsa do bando (efeito repulsivo). Para um camarão i, este movimento pode ser definido como:

$$N_i^{t+1} = N^{max}\alpha_i + \omega_n N_i^t, \tag{3.36}$$

onde

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target}, \tag{3.37}$$

e N^{max} é a velocidade máxima de indução, ω_n é o peso inercial do movimento induzido em $[0, 1]$, e N_i^{old} é o movimento induzido da iteração anterior.

O movimento de forrageamento é formulado em termos de localização de alimento e a experiência prévia sobre a localização do alimento, podendo ser expressa pelo i-ésimo camarão da seguinte forma:

$$F_i^{t+1} = V_f \beta_i + \omega_f F_i^t, \tag{3.38}$$

onde

$$\beta_i = \beta_i^{food} + \beta_i^{best}, \tag{3.39}$$

e V_f é a velocidade de forrageamento, ω_f é o peso inercial do movimento de forrageamento em $[0, 1]$, F_i^{old} é o último movimento de forrageamento, β_i^{food} é a atração pelo alimento, e β_i^{best} é o efeito do melhor valor de aptidão do i-ésimo camarão até o momento.

A difusão física é tomada como um processo aleatório e pode ser expressa em termos de difusão máxima e um vetor aleatório dado por:

$$D_i = D^{max}(1 - \frac{I}{I_{max}})\delta, \tag{3.40}$$

onde D^{max} é a velocidade de difusão máxima, I é a iteração e δ é o vetor aleatório em $[-1, 1]$.

3.10 Social-Spider Optimization - SSO

Social-Spider Optimization é baseado no comportamento cooperativo de algumas espécies de aranhas e foi proposto por Cuevas [32]. O algoritmo leva em consideração dois gêneros de aranhas: machos e fêmeas. Dependendo do gênero, cada agente é conduzido por um conjunto de operadores diferentes simulando o comportamento cooperativo den-

tro da colônia. O espaço de busca é assumido como sendo a teia e a posição das aranhas representam a solução ótima.

Uma característica interessante das aranhas é o enviesamento da população para as aranhas fêmeas. O número de aranhas macho dificilmente alcança 30% dos membros total na colônia. O número de fêmeas N_f é selecionado aleatóriamente com um intervalo de 65-90% da população total N, sendo calculada como:

$$N_f = [(0.9 - rand * 0.25) * N], \tag{3.41}$$

onde $rand$ é um número aleatório no intervalo $[0, 1]$. O número de aranhas macho N_m é dado por:

$$N_m = N - N_f. \tag{3.42}$$

Cada aranha recebe um peso de acordo com o valor de aptidão da solução:

$$w_i = \frac{fitness_i - worst}{best - worst}, \tag{3.43}$$

onde $fitness_i$ é o valor de aptidão obtido pela avaliação da posição da i-ésima aranha, $i = 1, 2, \ldots, N$, e $worst$ e $best$ são o melhor e o pior valores de aptidão de toda a população, repectivamente.

A teia é usada como um mecanismo para transmitir informação entre os membros da colônia. A informação é codificada na forma de pequenas vibrações e depende do peso e da distância da aranha que gerou:

$$V_{i,j} = w_j e^{-d_{i,j}^2}, \tag{3.44}$$

onde $d_{i,j}$ é a distância Euclidiana entre a aranha i e j. Pode-se considerar três relações especiais:

- As vibrações $V_{i,c}$ que são percebidas pela aranha i como resultado da informação transmitida pela aranha c, a qual é a aranha mais próxima a aranha i e é possui

peso maior, i.e., $w_c > w_i$;

- As vibrações $V_{i,b}$ que são percebidas pela aranha i como resultado da informação transmitida pela aranha b que possui o maior peso de toda a população;
- As vibrações $V_{i,f}$ que são percebidas pela aranha i como resultado da informação transmitida pela aranha fêmea f mais próxima.

As aranhas realizam uma interação cooperativa sobre outros membros da colônia dependendo do gênero. Com o intuito de simular esse comportamento cooperativo da aranha fêmea, um operador é definido na Equação 3.45. O movimento de atração ou repulsa é desenvolvido sobre outras aranhas de acordo com suas vibrações:

$$f_i^{(t+1)} = \begin{cases} f_i^{(t)} + \alpha V_{i,c}(s_c - f_i) + \beta V_{i,b}(s_b - f_i^{(t)}) + \\ +\gamma(\sigma - \frac{1}{2}) \qquad \text{if } r_m < PF; \\ f_i^{(t)} - \alpha V_{i,c}(s_c - f_i^{(t)}) - \beta V_{i,b}(s_b - f_i^{(t)}) + \\ +\gamma(\sigma - \frac{1}{2}) \qquad \text{if } r_m \geq PF, \end{cases} \tag{3.45}$$

onde r_m, α, β, γ e σ são número aleatórios entre $[0, 1]$, $f_i^{(t+1)}$ é a posição da aranha fêmea i no tempo $t + 1$, s_c, PF que denota um parâmetro ad-hoc, e s_b representa o membro mais próximo a i que possui um peso maior e a melhor aranha de toda a população, respectivamente.

A população de aranhas macho é dividida em duas classes: dominantes e não-dominantes. A classe dominante possui um valor de aptidão melhor que a não-dominante, e eles são atraídos para a fêmea mais próxima na teia. Já a classe não-dominante tende a concentrar-se no centro C da população de aranhas macho como estratégia para aproveitar os recursos disperdiçados pelos machos dominantes:

$$m_i^{(t+1)} = \begin{cases} m_i^{(t)} + \alpha V_{i,f}(s_f - m_i^{(t)}) + \gamma(\sigma - \frac{1}{2}) \\ \text{if } w_{N_f+i} > w_{N_f+C}; \\ m_i^{(t)} + \alpha \left(\dfrac{\sum_{h=1}^{N_m} m_h w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} \right) \\ \text{if } w_{N_f+i} \geq w_{N_f+C}, \end{cases} \tag{3.46}$$

onde s_f representa a fêmea mais próxima ao macho i, e $m_i^{(t+1)}$ é a posição do macho i no tempo $t+1$.

O acasalamento é realizado pelos machos dominantes e as fêmeas na colônia. Considerando r (calculado pela Equação 3.47) como sendo o raio, quando um macho dominante localiza uma fêmea dentro de r, acontece o acasalamento e uma nova aranha é criada:

$$r = \frac{\sum_{j=1}^{n} p_j^{high} - p_j^{low}}{2n}, \tag{3.47}$$

onde n é a dimensão do problema (número de variáveis a serem otimizadas), p_j^{high} e p_j^{low} são os limites superior e inferior, respectivamente. Uma vez que a nova aranha é criada, ela é comparada com a pior aranha na colônia. Se a nova aranha é melhor, a pior aranha é substituída pela nova.

4 Metodologia

As instâncias de dados são descritas como um par $(\vec{x}, y)$, em que $\vec{x} \in \mathbb{R}^n$ e y representam o vetor de características e seu rótulo, respectivamente. Seja $\mathcal{Z}(\mathcal{X}, \mathcal{Y})$ uma base de dados de nosso problema de classificação em que $\mathcal{X}$ representa o conjunto de vetores de características e $\mathcal{Y}$ o conjunto de saídas relacionadas a cada instância. Um classificador é então definido como uma função $f : \mathcal{X} \rightarrow \mathcal{Y}$ que prediz $y \in Y$ para uma dada amostra $\vec{x} \in X$ baseada no modelo aprendido do conjunto de dados rotulados (aprendizado supervisionado). A fim de proporcionar uma compreensão melhor do problema, técnicas de seleção de características visam encontrar um subespaço mínimo que melhor descrevem a distribuição de $\mathcal{X}$. Mais precisamente, nosso objetivo é selecionar um valor de $m \ll n$ e projetar cada instância de $x \in \mathbb{R}^n$ para um novo $x' \in \mathbb{R}^m$. Além disso, algoritmos de classificação podem sofrer do Fenômeno de Hughes [33] em espaços multidimensionais e, portanto, exigem muito mais carga computacional para soluções numéricas em problemas de programação dinâmica [34].

Dessa forma, apresenta-se a metodologia proposta para avaliar o desempenho das técnicas de seleção de características discutidas nas seções anteriores. Primeiramente, uma base de dados foi aleatoriamente particionada em N conjuntos, i.e., $\mathcal{Z} = \mathcal{F}_1 \cup \ldots \mathcal{F}_i \cup \ldots \mathcal{F}_N$. Ressalta-se que cada conjunto deve ser suficientemente grande para conter amostras representativas do problema. Além disso, uma instância do classificador OPF foi treinada sobre um desses conjuntos $\mathcal{F}_i$ e um dos conjuntos $\mathcal{F}_j$ é então classificado a fim de calcular a função de aptidão que irá guiar o algoritmo de otimização para selecionar o conjunto de características mais representativo, para $i \neq j$. Para cada membro da população no algoritmo é associado com uma cadeia de bits que indica a presença ou ausência de uma característica. Assim, para cada membro, foi construído um classificador a partir

do conjunto de treinamento apenas com as características selecionadas, e foi calculada a função de aptidão mediante a classificação de $\mathcal{F}_j$. Enquanto o procedimento converge, i.e., todas as gerações de uma população foram calculadas, o agente (morcego, vaga-lume, massa, harmonia, partícula) com o maior valor de aptidão possui a solução com o melhor conjunto de características. Além disso, contruímos um modelo de classificação utilizando o conjunto de treinamento e as características selecionadas e avaliamos a qualidade da solução calculando a eficácia sobre os conjuntos restantes, $\mathcal{F}_N \backslash \mathcal{F}_i \cup \mathcal{F}_j$. A Figura 4.1 ilustra a metodologia apresentada e o Algoritmo 3 detalha o procedimento adotado.

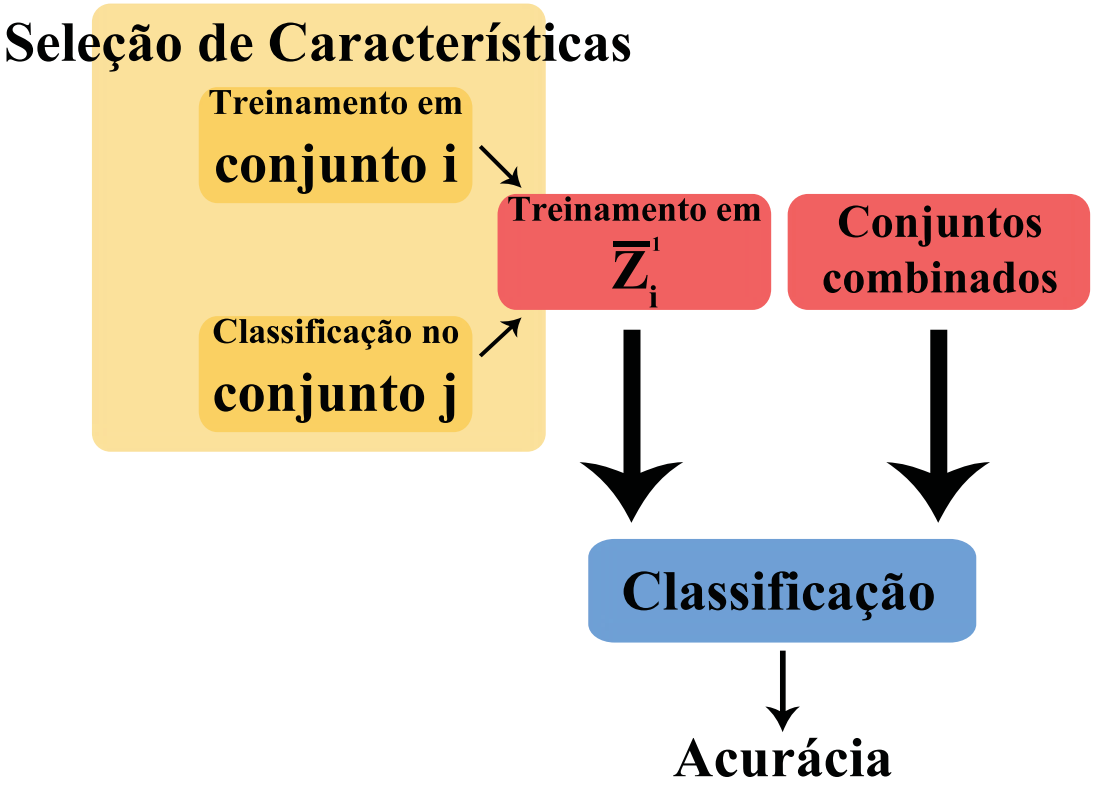


Figura 4.1: Metodologia utilizada.

Algoritmo 3 – Seleção de Características

Entrada:	Uma base de dados $\mathcal{Z}$, número de conjuntos $\mathcal{N}$, número de agentes $\mathcal{A}$, número de iterações $\mathcal{I}$.
Saída:	Performance de cada método definido pela função λ.
Auxiliares:	Um vetor $\mathcal{V}$ de características selecionadas, e os conjuntos finais de treino e teste, $\overline{\mathcal{Z}}^1, \overline{\mathcal{Z}}^2$.

1. *Para cada conjunto* $\mathcal{F}_i \in \mathcal{Z}$
2. *Para cada conjunto* $\mathcal{F}_j \in \mathcal{Z}$
3. *Para cada técnica* $\mathcal{T}$
4. $\mathcal{V} \leftarrow$ *encontra o conjunto de características utilizando* $\mathcal{T}$, $\mathcal{F}_j$, $\mathcal{F}_i$, *e os parâmetros* $\mathcal{A}, \mathcal{I}$
5. $\overline{\mathcal{Z}}^1 \leftarrow \mathcal{F}_j \backslash \mathcal{V}$
6. *Cria um classificador de* $\overline{\mathcal{Z}}^1$
7. *Para cada conjunto* $\mathcal{F}' \in \mathcal{Z} \backslash \mathcal{F}$
8. $\overline{\mathcal{Z}}^2 \leftarrow \mathcal{F}' \backslash \mathcal{V}$
9. *Classifica* $\overline{\mathcal{Z}}^2$
10. *Computa performance sobre* $\overline{\mathcal{Z}}^2$
11. *Retorna a função* λ

Nos algoritmos evolucionistas apresentados na Seção 3, os agentes movem-se pelo espaço de busca atualizando suas posições com valores contínuos. No entando, na seleção de características, o espaço de busca é modelado como um hipercubo boleano de n-dimensões e os agentes movem-se através dos cantos desse cubo, como mostra a Figura 4.2. Tendo em vista que o problema é selecionar ou não uma característica, a posição dos agentes é, então, representada por um vetor binário.

Na literatura existem diversas funções sigmoidais que podem ser utilizadas para converter os valores contínuos dos agentes em valores binários. No presente trabalho foi adotada a função logística, como proposto [15, 16, 17], porém outras funções podem ser adotadas. Ressalta-se que os agentes das técnicas GA e HS não necessitam serem convertidos, pois os valores já são binários.

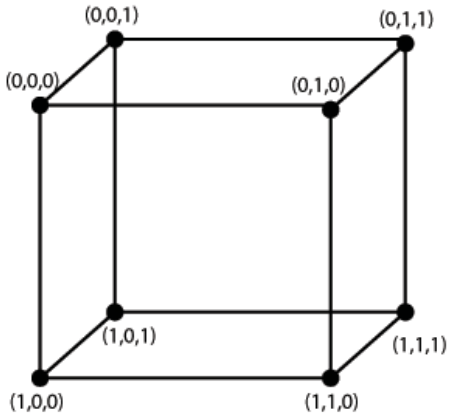


Figura 4.2: Ilustração de um 3-cubo, o qual simboliza um espaço de busca para um problema com 3 características.

$$S(v_i^j) = \frac{1}{1 + e^{-v_i^j}}. \tag{4.1}$$

5 Resultados Experimentais

Nesta seção, são apresentadas as bases de dados utilizadas para comparar as técnicas evolucionistas, bem como suas principais características. Também, são apresentados os parâmetros utilizados por cada uma das técnicas evolucionistas e, por fim, os resultados que foram obtidos. Foram avaliadas a eficiência e eficácia dos algoritmos evolucionistas apresentados na Seção 3 em encontrar o conjunto ótimo de características. Os algoritmos foram implementados na linguagem C seguindo as instruções oriundas de suas referências. Levando em consideração que em algoritmos não-determinísticos, como é o caso dos algoritmos evolucionistas, a solução obtida pelos algoritmos podem variar consideravelmente entre suas execuções, os experimentos foram realizados em 25 rodadas (valor empírico) utilizando a metodologia apresentada na Seção 4, onde foram computadas a média e o desvio padrão. Os experimentos foram executados em um computador com as seguintes configurações: Pentium Intel Core $i7^{®}$ 1.73Ghz, 6 GB de memória RAM e o sistema operacional Linux Ubuntu Desktop LTS 13.04.

5.1 Bases de dados

Em relação às bases de dados, foram utilizadas as seguintes:

- **Sonar**: 208 amostras, 2 classes, e 60 características [35].
- **Ionosphere**: 351 amostras, 2 classes, e 34 características [35].
- **Vehicle**: 846 amostras, 4 classes, e 18 características [35].
- **German Numer**: 1000 amostras, 2 classes, e 24 características [35].

- **Splice**: 1000 amostras, 2 classes, e 60 características [35].
- **Australian**: 690 amostras, 2 classes, e 14 características [35].

5.2 Parâmetros dos algoritmos evolucionistas

A Tabela 5.1 apresenta os parâmetros utilizados em cada um dos algoritmos evolucionistas. Ressalta-se ainda que foi assumido os valores de 30 agentes para o tamanho da população e 100 iterações. Os parâmetros foram escolhidos empiricamente.

Tabela 5.1: Parâmetros das técnicas de otimização meta-heurísticas: os valores W_n, W_f para o KH e o peso de inércia w do PSO foram ajustados dinâmicamente decrescendo de 0.9 para 0.4.

Técnica	**Parâmetros**
GA	mutation= 0.1
PSO	$c_1 = 2.0$, $c_2 = 2.0$
FA	$\gamma = 0.7$, $\beta_0 = 1.0$, $\alpha = 0.01$
GSA	$G_0 = 100$
HS	hmcr= 0.9
BA	$\alpha = 0.9$, $\gamma = 0.9$
CSS	–
CS	$\alpha = 1$, prob= 0.25
KH	Nmax= 0.01, Vf= 0.02, Dmax= 0.002, Ct= 0.5
SSO	PF= 0.7

5.3 Experimentos realizados

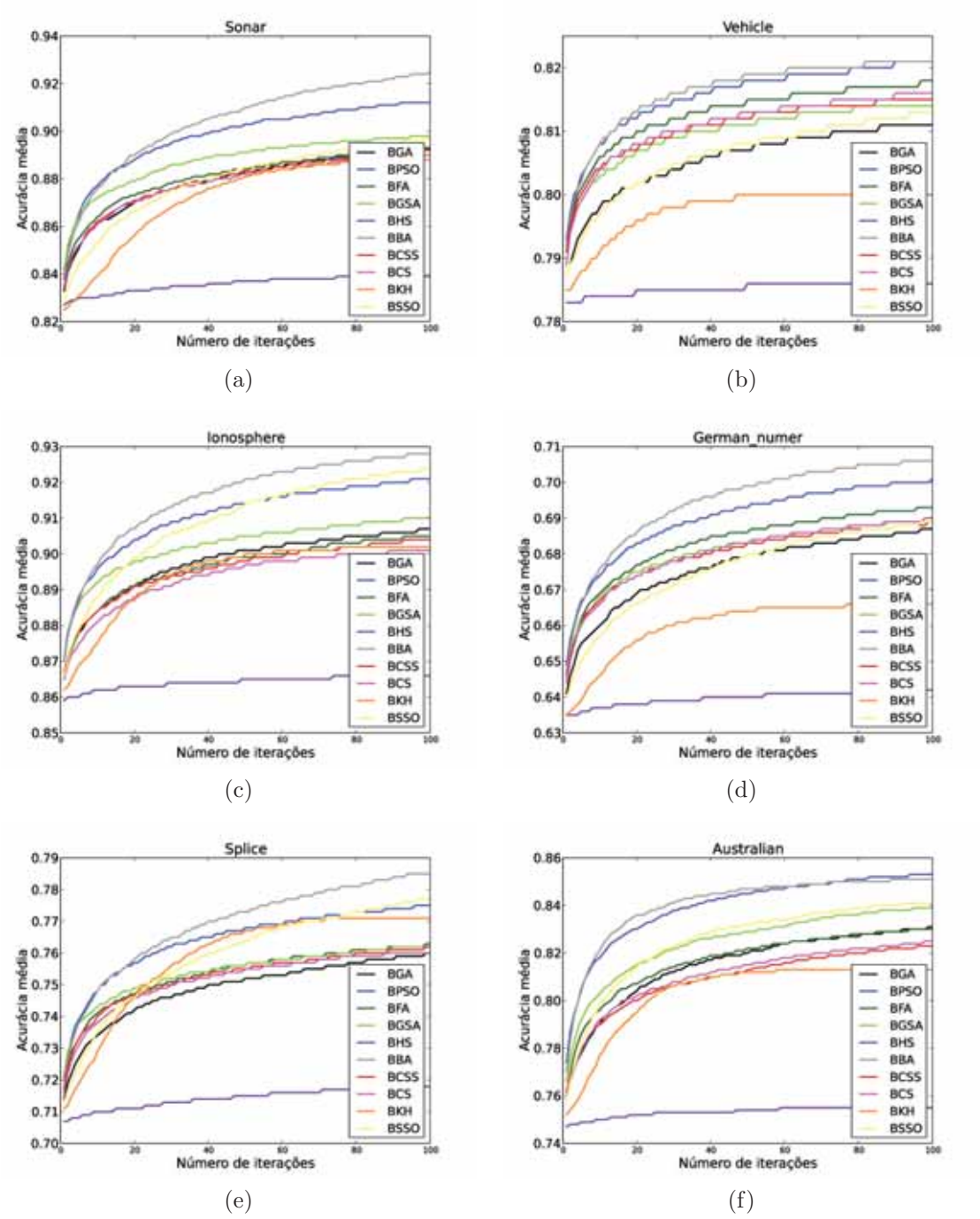
Os resultados experimentais são discutidos, primeiramente, com relação à acurácia média obtida na fase de avaliação (Figura 5.1). Em seguida, são apresentados a acurácia média e o desvio padrão obtidos acessando o conjunto de teste (Figura 5.2), bem como a média das características selecionadas (Figura 5.3) e o tempo médio de execução (ms) (Figura 5.4). Também, foi realizado o teste estatístico não-paramétrico de Wilcoxon.

Na Figura 5.1 nota-se que as técnicas convergem em direção à solução ótima, similarmente. O BHS, como pode ser observado, possui uma convergência lenta, o que pode

ser explicado devido à *exploitation* que ocorre apenas em uma harmonia a cada iteração. Também ocorre uma superestimativa na classificação com relação ao conjunto de avaliação (Figura 5.1) e o conjunto de teste (Figura 5.2). As técnicas BBA e BPSO atingiram os maiores valores na base *Sonar* (Figuras 5.1a e 5.2a), com a média de $92,45\%$ e $91,24\%$ respectivamente, enquanto que, no conjunto de teste, os valores atingem a média de $72,48\%$ e $74,13\%$. Na base *Vehicle*, o BBA e o BPSO atingiram os valores de $82,11\%$ e $82,10\%$ no conjunto de avaliação, respectivamente, e $74,02\%$ e $76,69\%$ no conjunto de teste. Já na base *Ionosphere*, os maiores valores de acurácia no conjunto de avaliação, foram do BBA, BSSO e BPSO com $92,81\%$, $92,36\%$ e $92,13\%$, respectivamente. No conjunto de teste, os valores do BBA, BSSO e BPSO chegaram a $78,68\%$, $81,19\%$ e $80,33\%$, respectivamente. Na base *German Numer* tem-se, no conjunto de avaliação, o BBA e BPSO com $70,62\%$ e $70,07\%$, respectivamente e no conjunto de teste $56,08\%$ e $58,97\%$. Na base *Splice*, os maiores valores de acurácia no conjunto de avaliação foram atingidos pelas técnicas BBA, BSSO e BPSO com os valores $78,55\%$, $77,74\%$ e $77,50\%$ e no conjunto de teste $64,28\%$, $70,45\%$ e $68,42\%$, respectivamente. E por fim, na base *Australian*, o BPSO e o BBA atingiram as maiores acurácias no conjunto de avaliação com os valores $85,33\%$ e $85,11\%$, respectivamente, porém no conjunto de teste os valores foram $81,15\%$ e $63,99\%$, respectivamente.

Na Figura 5.2 é possível notar que o classificador OPF em conjunto com as técnicas evolucionistas, ou seja, eliminando características irrelevantes e que degradam a taxa de acerto do classificador, obtiveram resultados superiores em comparação quando o classificador OPF foi utilizado nas bases de dados com todas as características. Os maiores valores foram atingidos pelo BCS com $74,11\% \pm 1,09\%$ na base *Sonar*. Já o BFA, BCSS e o BCS tiveram uma taxa de $76,77\% \pm 0,37\%$, $76,79\% \pm 0,32\%$ e $76,83\% \pm 0,33\%$ na base *Vehicle*, respectivamente. Na base *Ionosphere*, *German Numer* e *Splice*, a técnica BSSO chegou aos maiores valores de acurácia $81,19\% \pm 0,82\%$, $60,21\% \pm 1,20\%$ e $70,45\% \pm 0,74\%$, respectivamente. E na base *Australian*, o BPSO e o BSSO obtiveram as melhores acurácias médias com os valores $81,15\% \pm 1,13\%$ e $81,04\% \pm 1,65\%$, respectivamente.

Na Figura 5.3 são apresentadas as características que foram selecionadas por cada uma

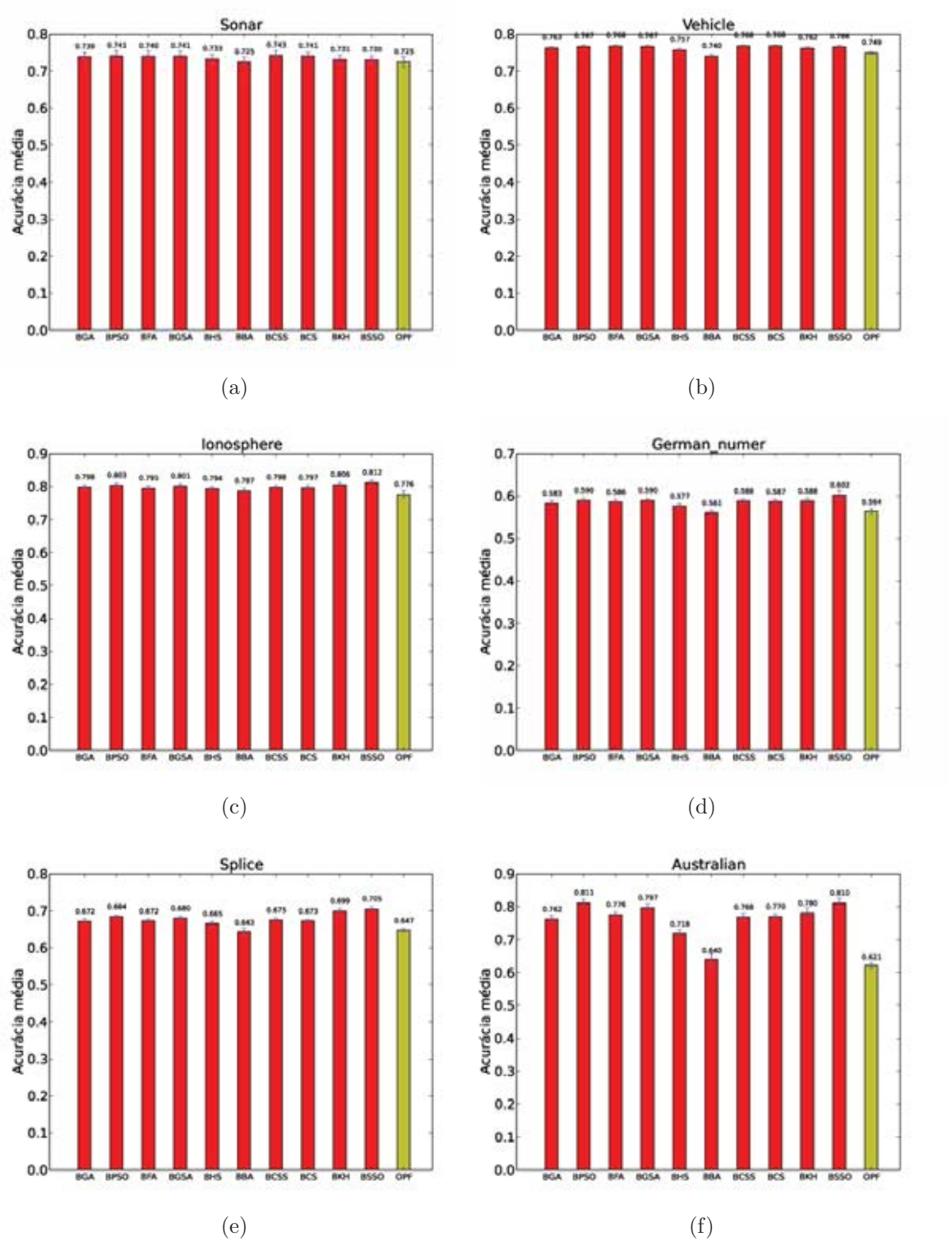


(a) (b)

(c) (d)

(e) (f)

Figura 5.1: Acurácia média no conjunto de avaliação utilizando o classificador OPF.



(a) (b)

(c) (d)

(e) (f)

Figura 5.2: Acurácia média no conjunto de teste utilizando o classificador OPF.

das técnicas evolucionistas. É importante salientar que o objetivo é maximizar a taxa de acerto do classificador e não minimizar o número de características. Porém, características irrelevantes, ou seja, aquelas que degradam a taxa de acerto do classificador, são eliminadas, restando apenas as características redundantes. Assim, temos que o BSSO foi o que utilizou o menor número de características nas bases *Sonar*, *Ionosphere*, *German Numer*, *Splice* e *Australian*, sendo elas 24, 9, 10, 20 e 3, respectivamente. Já na base *Vehicle*, é observado que as técnicas selecionaram um número semelhante de características.

Na Figura 5.4, a técnica que executou em menor tempo foi o BHS em todas as bases. E como foi explicado, o BHS atualiza a cada iteração apenas uma harmonia, sendo assim, necessário um número maior de iterações. Por fim, tem-se o BKH, que obteve o segundo melhor tempo médio de execução nas bases *Vehicle*, *Ionosphere*, *German Numer*, *Splice* e *Australian*.

Foi realizado, também, o teste estatístico de Wilcoxon [36], onde foi avaliada se a diferença entre as técnicas utilizadas neste trabalho são relevantes. As Tabelas 5.2, 5.3, 5.4, 5.5, 5.6 e 5.7 apresentam o valor-p obtido comparando-se as técnicas evolucionistas em cada uma das bases. O valor nível de significância foi estabelecido em $\alpha = 0.05$. Também, foram definidas as hipóteses do teste estatístico, onde a hipótese H_0, aceita que as técnicas são iguais e rejeitando H_0, ou seja, aceitando a hipótese H_1, aceita-se que as técnicas são estatiscamente diferentes. O teste foi realizado para cada uma das seis bases de dados e os valores em destaque (negrito) mostram que a hipótese nula H_0 foi rejeitada.

Tabela 5.2: Teste de Wilcoxon na base *Sonar*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	0.2758	0.4926	0.1919	**0.0027**	**0.0001**	**0.0303**	0.2642	**0.0049**	**0.0032**
BPSO	0.2758	–	0.7983	0.8612	**0.0011**	**0.0000**	0.3261	0.8612	**0.0042**	**0.0016**
BFA	0.4926	0.7983	–	0.9250	**0.0016**	**0.0000**	0.2642	0.6377	**0.0069**	**0.0016**
BGSA	0.1919	0.8612	0.9250	–	**0.0008**	**0.0000**	0.3819	0.8191	**0.0063**	**0.0006**
BHS	**0.0027**	**0.0011**	**0.0016**	**0.0008**	–	**0.0042**	**0.0005**	**0.0003**	0.4758	0.4273
BBA	**0.0001**	**0.0000**	**0.0000**	**0.0000**	**0.0042**	–	**0.0000**	**0.0000**	**0.0186**	**0.0422**
BCS	0.2642	0.8612	0.6377	0.8191	**0.0003**	**0.0000**	–	0.5998	**0.0017**	**0.0004**
BCSS	**0.0303**	0.3261	0.2642	0.3819	**0.0005**	**0.0000**	0.5998	–	**0.0021**	**0.0001**
BKH	**0.0049**	**0.0042**	**0.0069**	**0.0063**	0.4758	**0.0186**	**0.0021**	**0.0017**	–	0.6186
BSSO	**0.0032**	**0.0016**	**0.0016**	**0.0006**	0.4273	**0.0422**	**0.0001**	**0.0004**	0.6186	–

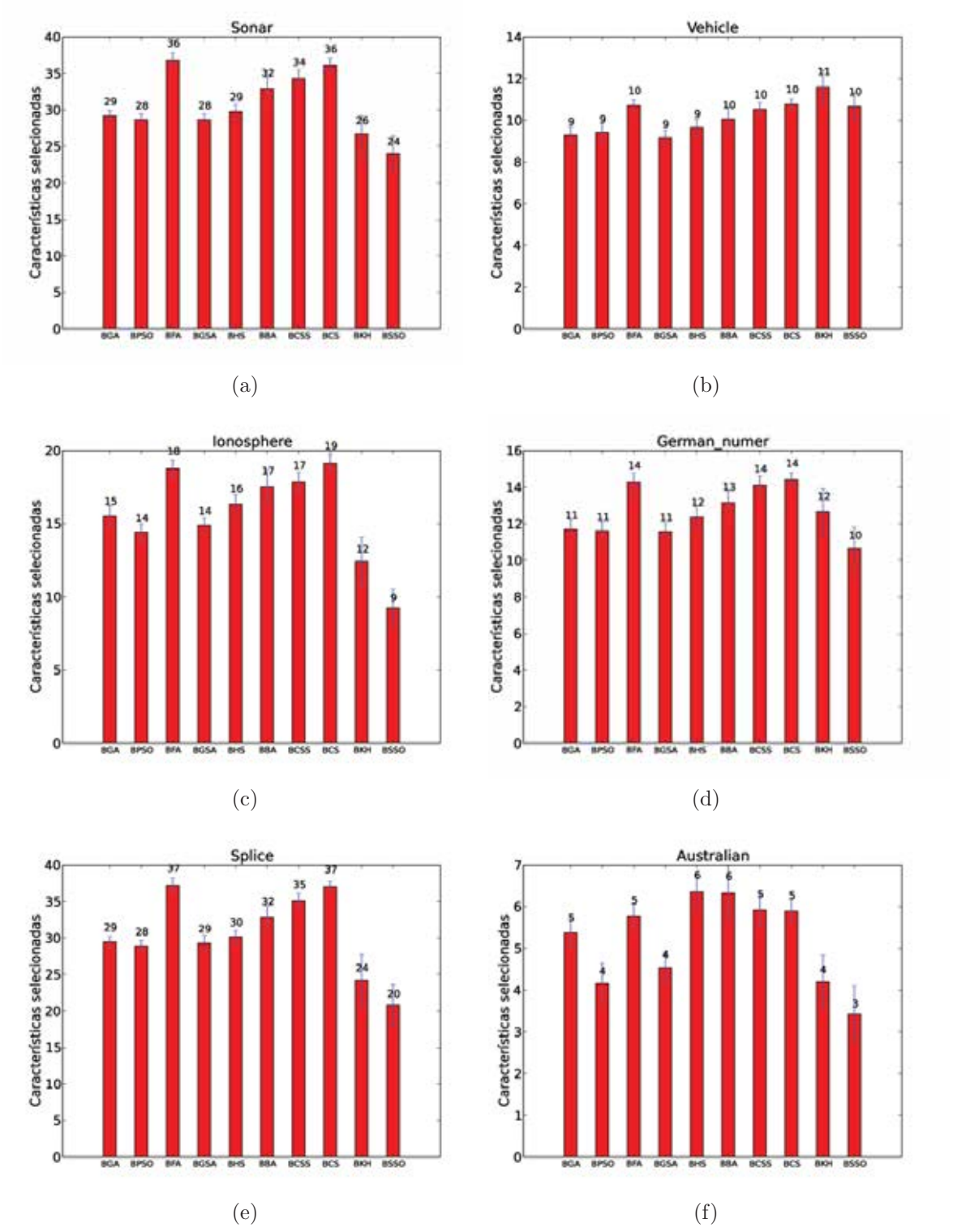


Figura 5.3: Número médio de características selecionadas utilizando o classificador OPF.

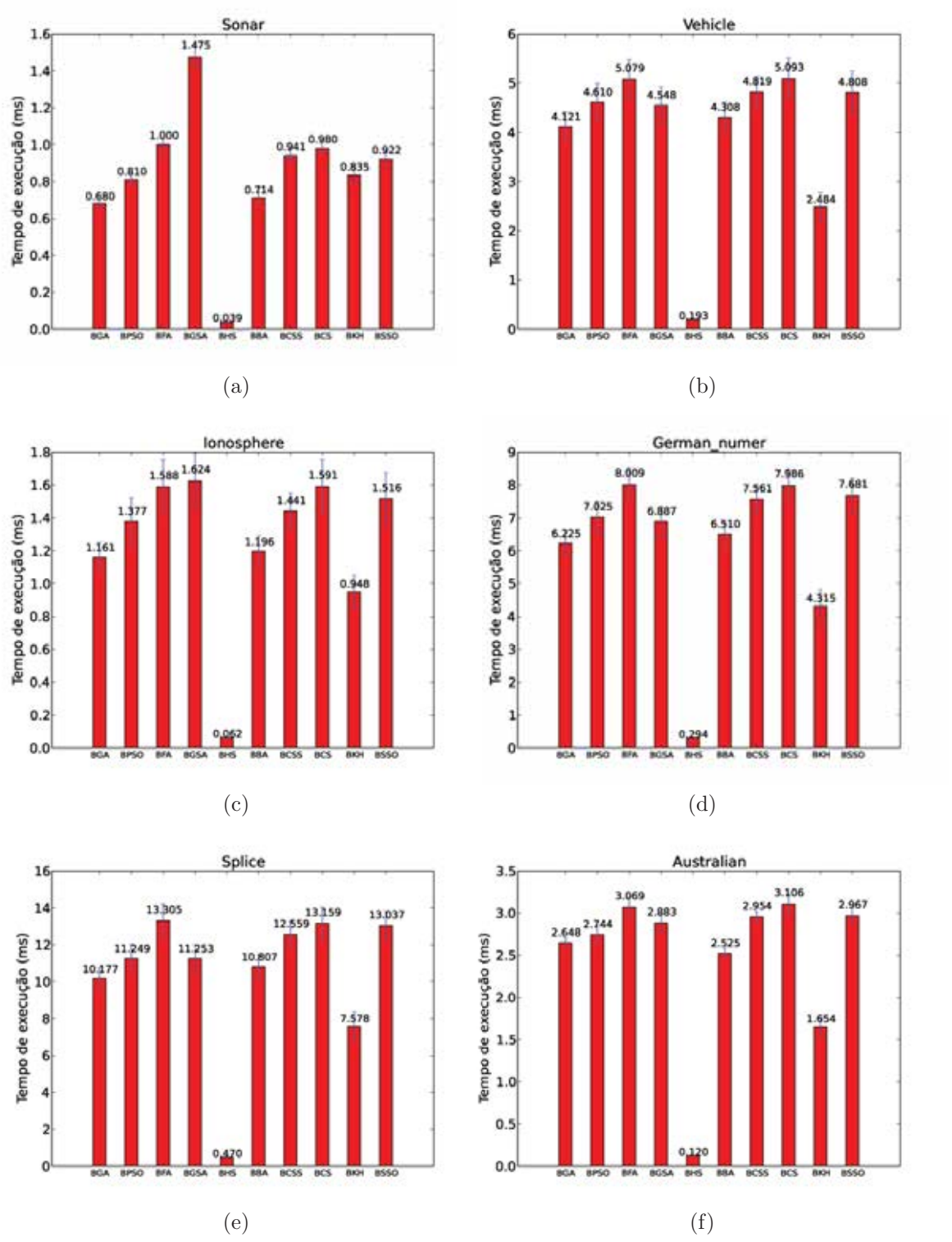


Figura 5.4: Tempo de execução (ms) médio utilizando o classificador OPF.

Tabela 5.3: Teste de Wilcoxon na base *Vehicle*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	**0.0004**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.6186	**0.0054**
BPSO	**0.0004**	–	0.2418	0.6186	**0.0000**	**0.0000**	0.2209	0.1285	**0.0000**	0.1578
BFA	**0.0000**	0.2418	–	0.2418	**0.0000**	**0.0000**	0.7570	0.5812	**0.0000**	0.0544
BGSA	**0.0000**	0.6186	0.2418	–	0.0000	0.0000	0.0827	0.0173	0.0002	0.2109
BHS	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0001**	**0.0000**
BBA	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BCS	**0.0000**	0.1285	0.5812	**0.0173**	**0.0000**	**0.0000**	–	0.3819	**0.0000**	**0.0214**
BCSS	**0.0000**	0.2209	0.7570	0.0827	0.0000	0.0000	0.3819	–	0.0000	0.0283
BKH	0.6186	0.0000	0.0000	0.0002	0.0001	0.0000	0.0000	0.0000	–	0.0021
BSSO	**0.0054**	0.1578	0.0544	0.2109	**0.0000**	**0.0000**	**0.0283**	**0.0214**	**0.0021**	–

Tabela 5.4: Teste de Wilcoxon na base *Ionosphere*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	**0.0138**	0.1094	0.1658	**0.0303**	**0.0000**	0.6766	0.4273	**0.0012**	**0.0000**
BPSO	**0.0138**	–	**0.0000**	0.2758	**0.0000**	**0.0000**	**0.0027**	**0.0011**	0.2879	**0.0004**
BFA	0.1094	**0.0000**	–	**0.0063**	0.3819	**0.0003**	0.1036	0.4926	**0.0001**	**0.0000**
BGSA	0.1658	0.2758	**0.0063**	–	**0.0012**	**0.0000**	**0.0160**	**0.0054**	**0.0303**	**0.0000**
BHS	**0.0303**	**0.0000**	0.3819	**0.0012**	–	**0.0000**	**0.0149**	0.1425	**0.0000**	**0.0000**
BBA	**0.0000**	**0.0000**	**0.0003**	**0.0000**	**0.0000**	–	**0.0000**	**0.0002**	**0.0000**	**0.0000**
BCS	0.4273	**0.0011**	0.4926	**0.0054**	0.1425	**0.0002**	–	0.4432	**0.0004**	**0.0000**
BCSS	0.6766	**0.0027**	0.1036	**0.0160**	**0.0149**	**0.0000**	0.4432	–	**0.0000**	**0.0000**
BKH	**0.0012**	0.2879	**0.0001**	**0.0303**	**0.0000**	**0.0000**	**0.0000**	**0.0004**	–	**0.0069**
BSSO	**0.0000**	**0.0004**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0069**	–

Tabela 5.5: Teste de Wilcoxon na base *German Numer*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	**0.0005**	0.0875	**0.0016**	**0.0058**	**0.0000**	**0.0069**	**0.0030**	**0.0119**	**0.0000**
BPSO	**0.0005**	–	**0.0149**	0.9036	**0.0000**	**0.0000**	0.2642	0.0875	0.5098	**0.0007**
BFA	0.0875	**0.0149**	–	**0.0054**	**0.0000**	**0.0000**	0.0578	0.5272	0.2642	**0.0001**
BGSA	**0.0016**	0.9036	**0.0054**	–	**0.0000**	**0.0000**	0.3819	**0.0160**	0.8191	**0.0006**
BHS	**0.0058**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BBA	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BCS	**0.0030**	0.0875	0.5272	**0.0160**	**0.0000**	**0.0000**	–	0.2418	0.5449	**0.0001**
BCSS	**0.0069**	0.2642	0.0578	0.3819	**0.0000**	**0.0000**	0.2418	–	0.8612	**0.0001**
BKH	**0.0119**	0.5098	0.2642	0.8191	**0.0000**	**0.0000**	0.8612	0.5449	–	**0.0000**
BSSO	**0.0000**	**0.0007**	**0.0001**	**0.0006**	**0.0000**	**0.0000**	**0.0001**	**0.0001**	**0.0000**	–

Tabela 5.6: Teste de Wilcoxon na base *Splice*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	**0.0000**	0.6571	**0.0004**	**0.0032**	**0.0000**	0.3130	0.4273	**0.0000**	**0.0000**
BPSO	**0.0000**	–	**0.0000**	**0.0138**	**0.0000**	**0.0000**	**0.0001**	**0.0000**	**0.0000**	**0.0000**
BFA	0.6571	**0.0000**	–	**0.0003**	**0.0009**	**0.0000**	0.2418	0.9036	**0.0000**	**0.0000**
BGSA	**0.0004**	**0.0138**	**0.0003**	–	**0.0000**	**0.0000**	**0.0049**	**0.0000**	**0.0000**	**0.0000**
BHS	**0.0032**	**0.0000**	**0.0009**	**0.0000**	–	**0.0000**	**0.0001**	**0.0002**	**0.0000**	**0.0000**
BBA	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BCS	0.4273	**0.0000**	0.9036	**0.0000**	**0.0002**	**0.0000**	–	0.0875	**0.0000**	**0.0000**
BCSS	0.3130	**0.0001**	0.2418	**0.0049**	**0.0001**	**0.0000**	0.0875	–	**0.0000**	**0.0000**
BKH	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0138**
BSSO	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0138**	–

Tabela 5.7: Teste de Wilcoxon na base *Australian*.

	BGA	**BPSO**	**BFA**	**BGSA**	**BHS**	**BBA**	**BCS**	**BCSS**	**BKH**	**BSSO**
BGA	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0615	**0.0264**	**0.0004**	**0.0000**
BPSO	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.6766
BFA	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0009**	**0.0214**	0.1658	**0.0000**
BGSA	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0002**	**0.0001**
BHS	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BBA	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BCS	**0.0264**	**0.0000**	**0.0214**	**0.0000**	**0.0000**	**0.0000**	–	0.8401	**0.0038**	**0.0000**
BCSS	0.0615	**0.0000**	**0.0009**	**0.0000**	**0.0000**	**0.0000**	0.8401	–	**0.0087**	**0.0000**
BKH	**0.0004**	**0.0000**	0.1658	**0.0002**	**0.0000**	**0.0000**	**0.0087**	**0.0038**	–	**0.0000**
BSSO	**0.0000**	0.6766	**0.0000**	**0.0001**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	–

6 Conclusões e Trabalhos Futuros

A seleção de características tem sido utizada para melhorar a eficácia no reconhecimento de padrões. Dado que o problema de encontrar o subconjunto das características que maximiza a taxa de acerto de uma técnica de classificação de padrões pode ser modelado como um problema de otimização. Alguns algoritmos de otimização evolucionista, baseados em dinâmicas sociais e interação de aves, insetos e outros indíviduos foram utilizados para este propósito. Neste trabalho, utilizou-se cinco técnicas já propostas na literatura, sendo elas: BGA - *Binary Genetic Algorithm*, BPSO - *Binary Particle Swarm Optimization*, BFA - *Binary Firefly Algorithm*, BGSA - *Binary Gravitational Search Algorithm*, BHS - *Binary Harmony Search* e também foram propostas outras cinco técnicas para o contexto de seleção de características: BBA - *Binary Bat Algorithm*, BCSS - *Binary Charged System Search*, BCS - *Binary Cuckoo Search*, BKH - *Binary Krill Herd* e BSSO - *Binary Social-Spider Optimization*. Tais técnicas mostraram-se eficazes quando utilizadas em conjunto com o classificador OPF. O BSSO obteve a melhor acurácia em 3 bases, sendo elas *Ionosphere*, *German Numer* e *Splice*, chegando a aumentar a taxa de acerto do classificador OPF em 19% na base *Australian*. Também, selecionou o menor número de características nas bases *Sonar*, *Ionosphere*, *German Numer*, *Splice* e *Australian*. Em relação ao tempo de execução, o BKH obteve o segundo melhor tempo nas bases *Vehicle* com $2,484ms$, *Ionosphere* com $0,948ms$, *German Numer* com $4,315ms$, *Splice* com $7,578ms$ e *Australian* com $1,654ms$ e a segunda melhor acurácia média nas bases *Ionosphere* e *Splice*. Para futuros trabalhos, uma proposta seria utilizar o *Harmony Search*, devido sua rapidez, para otimizar os parâmetros das técnicas evolucionistas, tendo em vista que devem ser ajustados para cada problema específico, ou seja, para cada uma das bases de dados. Outra ideia seria um estudo detalhado das funções sigmoidais que

fazem a conversão dos valores contínuos dos agentes de cada uma das técnicas evolucionistas em valores binários, e verificar a maneira como essas funções afetam a convergência das técnicas. Também, existe a ideia de estender o problema de seleção de características, que neste trabalho foi tratado como um problema de otimização mono-objetivo para otimização multiobjetivos, onde as funções objetivos seriam minimizar a taxa de erro do classificador para cada classe e minimizar o número de características, por exemplo.

Referências

[1] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[2] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, Cambridge, MA, 1992.

[3] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. M. Kaufman, 2001.

[4] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[5] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. GSA: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248, 2009.

[6] J. Huang, Y. Cai, and X. Xu. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters*, 28(13):1825–1844, 2007.

[7] H. A. Firpi and E. Goodman. Swarmed feature selection. In *Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop*, pages 112–118, Washington, DC, USA, 2004. IEEE Computer Society.

[8] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4108, 1997.

[9] R. Falcon, M. Almeida, and A. Nayak. Fault identification with binary adaptive fireflies in parallel and distributed systems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1359–1366, 2011.

[10] X.-S. Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal Bio-Inspired Computing*, 2(2):78–84, 2010.

[11] S. Palanisamy and S. Kanmani. Artificial bee colony approach for optimizing feature selection. *IJCSI International Journal of Computer Science Issues*, 9(3):432–438, 2012.

[12] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[13] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. BGSA: binary gravitational search algorithm. *Natural Computing*, 9:727–745, 2010.

[14] C. C. O. Ramos, A. N. de Souza, A. X. Falcão, and J. P. Papa. New insights on non-technical losses characterization through evolutionary-based feature selection. *IEEE Transactions on Power Delivery*, 27(1):140–146, 2012.

[15] D. Rodrigues, L. A. M. Pereira, R. Y. M. Nakamura, K. A. P. Costa, X. S. Yang, A. N. Souza, and J. P. Papa. A wrapper approach for feature selection based on bat algorithm and optimum-path forest. *Expert Systems with Applications*, 41(5):2250–2258, 2014.

[16] D. Rodrigues, L. A. M. Pereira, J. P. Papa, C. C. O. Ramos, A. N. Souza, and L. P. Papa. Optimizing feature selection through binary charged system search. In *Proceedings of 15th International Conference on Computer Analysis of Images and Patterns*, pages 377–384, 2013.

[17] D. Rodrigues, L. A. M. Pereira, T. N. S. Almeida, J. P. Papa, A. N. Souza, C. O. Ramos, and X.-S. Yang. BCS: A binary cuckoo search algorithm for feature selection. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 465–468, Beijing, 2013.

[18] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[19] J. P. Papa, A. X. Falcão, V. H. C. Albuquerque, and J. M. R. S. Tavares. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520, 2012.

[20] A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[21] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, 45(3):35:1–35:33, July 2013.

[22] A. E. Eiben and C. A. Schippers. On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35:35–50, 1998.

[23] D. Halliday, R. Resnick, and J. Walker. *Extended , Fundamentals of Physics, 6th Edition*. Wiley, 2000.

[24] R. Mansouri, F. Nasseri, and M. Khorrami. Effective time variation of g in a model universe with variable space dimension. *Physics Letters*, 259:194–200, 1999.

[25] D. R. Griffin, F. A. Webster, and C. R. Michael. The echolocation of flying insects by bats. *Animal Behaviour*, 8(4):141–154, 1960.

[26] W. Metzner. Echolocation behaviour in bats. *Science Progress Edinburgh*, 75(298):453–465, 1991.

[27] H.-U. Schnitzler and E. K. V. Kalko. Echolocation by insect-eating bats. *BioScience*, 51(7):557–569, July 2001.

[28] X.-S. Yang. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274, 2011.

[29] A. Kaveh and S. Talatahari. A novel heuristic optimization method: charged system search. *Acta Mechanica*, 213(3):267–289, 2010.

[30] X.-S. Yang and S. Deb. Cuckoo search via lévy flights. In *Proceedings of the 2009 World Congress on Nature and Biologically Inspired Computing*, pages 210–214, 2009.

[31] A. H. Gandomi and A. H. Alavi. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831–4845, 2012.

[32] E. Cuevas, M. Cienfuegos, D. Zaldívar, and M. Pérez-Cisnero. A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications*, 40(16):6374–6384, 2013.

[33] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[34] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 2010.

[35] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[36] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 23 / 03 / 2014

Douglas Rodrigues

Assinatura