Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Reinaldo Gomes Morais

Proposta, Implementação e Análise de um Algoritmo de Otimização Global Inspirado no Comportamento dos Pássaros de Hitchcock

Rio de Janeiro

2017

Reinaldo Gomes Morais

Proposta, Implementação e Análise de um Algoritmo de Otimização Global Inspirado no Comportamento dos Pássaros de Hitchcock

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadora: Prof.ª Dr.ª Nadia Nedjah
Orientadora: Prof.ª Dr.ª Luiza de Macedo Mourelle

Rio de Janeiro
2017

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

M827 Morais, Reinaldo Gomes
Proposta, Implementação e Análise de um Algoritmo de Otimização Global Inspirado no Comportamento dos Pássaros de Hitchcock/Reinaldo Gomes Morais.- 2017.
176f.

Orientadoras: Nadia Nedjah e Luiza de Macedo Mourelle.
Dissertação (Mestrado) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica. 2. Otimização - Enxame de partículas - Dissertações. 3. Inteligência artificial - Dissertações. I. Nedjah, Nadia. II. Mourelle, Luiza de Macedo. III. Universidade do Estado do Rio de Janeiro. IV. Título.

CDU 004.42

Assinatura

Data

Reinaldo Gomes Morais

Proposta, Implementação e Análise de um Algoritmo de Otimização Global Inspirado no Comportamento dos Pássaros de Hitchcock

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em: 11 de dezembro de 2017

Banca Examinadora:

Prof.ª Dr.ª Nadia Nedjah (Orientadora)
Faculdade de Engenharia, UERJ

Prof.ª Dr.ª Luiza de Macedo Mourelle (Orientadora)
Faculdade de Engenharia, UERJ

Prof. Dr. Fernando Buarque de Lima Neto
Escola Politécnica de Pernambuco, UPE

Prof.ª Dr.ª Nayat Sánchez Pi
Instituto de Matemática e Estatísticas, UERJ

Rio de Janeiro
2017

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Wilma Gomes Morais e Reinaldo Costa Morais, por contribuírem de forma fundamental e valiosa durante todo o meu caminho para chegar até aqui, me dando força e nunca me deixando desanimar. Agradeço também ao meu irmão mais novo, Renan Gomes Morais, por ficar sempre entusiasmado com minhas ideias e compreender minha ausência na maioria dos momentos de lazer, e a Tininha pelo carinho e atenção.

À Deus por ter colocado em minha vida uma pessoa muito especial, a quem devo grande parte do conhecimento necessário sobre o mundo e por chegar onde estou, meu avô, José Fernandes Maciel (em memória). Seu incentivo e confiança nunca serão esquecidos.

À minha companheira de todos os momentos, sem a qual sei que o caminho teria sido muito mais difícil, minha noiva, Natalie Henriques Martins, pelo carinho, incentivo e paciência.

À minha segunda família, cunhada Luise, meus sogros Simone e Márcio, Dona Helena, Tetê, Suzy, Márcia, Guigui pela torcida, apoio e por estarem presentes em mais esta etapa da minha vida.

Ao meu amigo Thallys, pela troca de conhecimento, cumplicidade e por rir das minhas piadas sem graça.

Imensamente às professoras Nadia Nedjah e Luiza de Macedo Mourelle pela disponibilidade, confiança e orientação, não somente por todos ensinamentos, mas por terem me feito aprender.

À UERJ e seu corpo docente do programa de Mestrado em Engenharia Eletrônica, pela oportunidade e pelo aprendizado adquirido ao longo do curso.

Aos meus colegas de mestrado Alexandre, Luneque, Ramon, Joelmir, Pedro e Tarso, pela troca de conhecimentos, pelo companheirismo e pelos momentos de descontração.

À todos que contribuíram de forma direta ou indireta na realização deste trabalho .

“Importante não é ver o que ninguém nunca viu, mas sim, pensar o que ninguém nunca pensou sobre algo que todo mundo vê.”

Arthur Schopenhauer

RESUMO

MORAIS, Reinaldo Gomes. *Proposta, Implementação e Análise de um Algoritmo de Otimização Global Inspirado no Comportamento dos Pássaros de Hitchcock*. 2017. 176f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

Neste estudo, é introduzido um novo algoritmo de otimização chamado Algoritmo Inspirado nos Pássaros de Hitchcock (AIPH), inspirado no comportamento agressivo de pássaros retratado por Alfred Hitchcock, no clássico de suspense "Os pássaros" de 1963. Ao juntar os elementos sobre o fenômeno dos pássaros ao longo do filme, é possível enumerar características do comportamento dos pássaros que Hitchcock retratou no filme. O AIPH é um algoritmo estocástico de inteligência de enxame que capta a essência do comportamento fictício dos pássaros exposto por Hitchcock e modela um mecanismo de otimização. O algoritmo baseou-se no padrão de ataque dos pássaros no filme, que possui as fases de espreita, ataque e reorganização, definidos pela inicialização, estratégias de movimento no espaço de busca e estratégia de saída de mínimos locais, respectivamente. A técnica possui como diferenciais o uso de parâmetros adaptativos, uma inicialização aleatória discretizada e o uso da distribuição Beta. O desempenho do algoritmo foi verificado em 8 funções de custo usadas para *benchmark*, com cenários de 10, 60 e 600 dimensões. Para a avaliação dos resultados do AIPH, foram escolhidas 14 técnicas de otimização por inteligência de enxame da literatura, considerando diferentes tipos de inspiração, como de princípios da matemática, relações humanas, física e biologia. Observou-se a média dos resultados obtidos com 30 repetições de cada algoritmo para as análises de desempenho. O AIPH produziu resultados satisfatórios e equilibrados em diferentes dimensões das funções de custo escolhidas, conseguindo se destacar em cenários de alta dimensionalidade.

Palavras-chave: Otimização; Metaheurística; Inteligência de Enxame; Algoritmo Adaptativo.

ABSTRACT

MORAIS, Reinaldo Gomes. *Proposal, Implementation and Analysis of a Global Optimization Algorithm Inspired by Hitchcock Bird Behavior.* 2017. 176f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

In this study, a new optimization algorithm called the Hitchcock Bird Inspired Algorithm (AIPH) is introduced, inspired by the aggressive bird behavior portrayed by Alfred Hitchcock in the 1963 thriller "The birds". When gathering elements about the phenomenon of birds throughout the film, it is possible to enumerate characteristics of the behavior of the birds that Hitchcock portrayed in the film. AIPH is a stochastic swarm intelligence algorithm that captures the essence of the fictional behavior of birds exposed by Hitchcock and model an optimization mechanism. The algorithm was based on the attack pattern of birds in the film, which has the stages of stalking, attack and reorganization, defined by the initialization, movement strategies in the search space and strategy of local minimum escape, respectively. The technique has as differential the use of adaptive parameters, a discretized random initialization and the use of the Beta distribution. The performance of the algorithm is verified for 8 cost functions for *benchmark*, with scenarios of 10, 60 and 600 dimensions. In order to evaluate the results of the AIPH, 14 swarm intelligence based optimization techniques in the literature were chosen, considering different types of inspiration, such as principles of mathematics, human relations, physics and biology. The obtained average results with 30 repetitions of each algorithm observed for performance analysis. The AIPH produced satisfactory and balanced results in different dimensions of the chosen cost functions, being able to stand out in high dimensionality scenarios.

Keywords: Optimization; Metaheuristic; Swarm Intelligence; Adaptative Algorithm.

LISTA DE FIGURAS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

AAA	Artificial Algae Algorithm
ABC	Artificial Bee Colony
AIPH	Algoritmo Inspirado nos Pássaros de Hitchcock
BA	Bat Algorithm
BFO	Bacterial Foraging Optimization
CS	Cuckoo Search
CSS	Charged System Search
DA	Dragonfly Algorithm
EHO	Elephant Herding Optimization
FA	Firefly Algorithm
FSS	Fish School Search
GSA	Gravitational Search Algorithm
GWO	Grey Wolf Optimizer
HS	Harmony Search
KH	Krill Herd
MBO	Monarch Butterfly Optimization
MVMO	Mean-Variance Mapping Optimization
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SBO	Stellar Mass Black Hole Optimization
SCA	Sine Cosine Algorithm
SLC	Soccer League Competition
SSA	Social Spider Algorithm
TLBO	Teaching Learning Based Optimization
VS	Vortex Search
WOA	Whale Optimization Algorithm

SUMÁRIO

SUMÁRIO

SUMÁRIO

INTRODUÇÃO

OTIMIZAÇÃO é um processo empregado para resolver problemas em muitas áreas tais como de engenharia, administração, logística, transporte, economia e biologia, entre outros.

Problemas de otimização são aqueles que visam encontrar o melhor resultado, seja este mínimo ou máximo, para uma função matemática, também conhecida como função de custo do problema em questão. No cotidiano, estamos cercados por problemas de otimização. Dentre estes problemas do cotidiano, pode-se citar a título de exemplo, a escolha do melhor conjunto de produtos no supermercado com menor custo final, o tempo de irrigação do jardim com menor desperdício de água sem comprometer o crescimento das plantas, a melhor temperatura da água do chuveiro ou ar condicionado, o tempo de dedicação para o término de uma atividade no trabalho, dentre vários outros problemas.

As técnicas de otimização global têm como objetivo encontrar a melhor solução (máximo ou mínimo) em um conjunto de soluções factíveis para um problema de uma ou mais variáveis num determinado domínio (OLIVIERI, 2004). Os algoritmos usados para a solução de um problema de otimização são baseados em um processo de melhoria iterativa e podem ser determinísticos ou estocásticos.

Os métodos de otimização baseados nos algoritmos determinísticos, que formam a maioria dos métodos clássicos, geram única solução requerendo, na maioria das vezes, o uso de pelo menos a primeira derivada da função de custo em relação às variáveis de projeto. Os métodos determinísticos requerem teoremas que lhes garantem a convergência para uma solução ótima que não é necessariamente a solução ótima global. Além disso, não apresentam um bom desempenho em otimizar funções multimodais, que possuem vários ótimos locais. Ademais, a função de custo deve ser contínua e derivável no espaço de busca (BASTOS, 2004).

Já os métodos de otimização baseados nos algoritmos estocásticos ou probabilísticos, não usam nenhum conhecimento prévio do problema, apenas a função de custo

que servirá para a avaliação das soluções criadas. Iniciam com uma geração de números aleatórios e realizam buscas simultâneas no espaço de possíveis soluções através de uma população de indivíduos. Por não utilizarem a derivada da função de custo, são considerados métodos de ordem zero. Contudo, esse métodos estocásticos não garantem uma solução ótima, permitindo encontrar uma solução "aceitável", geralmente chamada de pseudo-ótima, de maneira simples e rápida (BASTOS, 2004).

Dentre as vantagens dos algoritmos probabilísticos temos a otimização de um grande número de variáveis, desde que a avaliação da função objetivo não tenha um custo computacional demasiadamente alto. A função objetivo não precisa ser contínua ou diferenciável e ainda apresentam bom desempenho em funções multimodais (BASTOS, 2004).

Problemas intratáveis ou difíceis são comuns na natureza e nas áreas do conhecimento e possuem complexidade de tempo não polinomial, conhecidos como NP-completo, e não possuem solução através de algoritmos determinísticos. Ademais, os problemas a serem otimizados são complexos, não lineares, multimodais, o que torna o uso de algoritmos estocásticos mais adequados.

A área que estuda as metaheurísticas é considerada um subcampo primário da área de otimização probabilística. Metaheurísticas são estruturas algorítmicas gerais adaptáveis a diversos problemas de otimização e são geralmente aplicadas a problemas de alta complexidade (REZENDE, 2014). Algumas áreas bem conhecidas em otimização são: computação evolucionária e inteligência de enxame.

A Computação Evolutiva ou Evolucionária apresenta um conjunto de técnicas inspiradas em princípios da evolução biológica. A técnica de busca e otimização de soluções que se destaca dentro da computação evolutiva é o Algoritmo Genético , fundamentada no princípio Darwiniano de seleção natural e reprodução genética (HOLLAND, 1973).

Já a Inteligência de Enxame, representa modelos inteligentes inspirados em sistemas sociais do mundo real, baseando-se na interação e organização entre agentes simples para a realização de tarefas. Alguns exemplos de técnicas de IE são: *Ant Colony Optimization* (ACO) (DORIGO; MANIEZZO; COLORNI, 1996) e *Particle Swarm Optimization* (PSO) (KENNEDY; EBERHART, 1995).

A Inteligência de Enxame caracteriza-se por inspirar-se no comportamento coletivo estruturado de qualquer conjunto de agentes capazes de interagir, reproduzindo um

aspecto de movimento coletivo no espaço. No caso do PSO, a interação entre aves e o comportamento coordenado na busca por alimentos ao longo do vôo são modeladas como um mecanismo de otimização (KENNEDY; EBERHART, 1995).

Entretanto, um problema comum a muitos destes algoritmos metaheurísticos é a escolha dos valores a serem usados para configurar seus parâmetros, sendo o seu desempenho fortemente dependente do ajuste destes parâmetros. O PSO, por exemplo, possui 3 parâmetros a serem ajustados, que são duas constantes de aceleração das partículas e o coeficiente de inércia. Além disso, muitas técnicas tem um desempenho pior a medida que a dimensão do problema cresce.

O ideal para uma técnica metaheurística seria não requerer nenhum parâmetro ou poucos parâmetros a serem ajustados, ou ainda, com um ajuste dinâmico destes parâmetros durante o processo de busca. Além disso, conseguir apresentar bons resultados em problemas complexos e de alta dimensionalidade.

Este trabalho tem como objetivo criar uma nova técnica cujos parâmetros sejam adaptáveis ao avanço durante o processo da busca da solução ótima e apresente um bom desempenho em problemas de otimização com alta dimensão. Para alcançar este objetivo, usou-se como inspiração o filme Os Pássaros de Alfred Hitchcock.

O cinema tem uma forte capacidade de inspirar e pode influenciar diversas áreas do conhecimento, tanto por meio de cenários, figurinos, sentimentos, emoções e situações, quanto pela apresentação de realidades distópicas.

A concepção da técnica proposta neste trabalho deu-se a partir da observação do comportamento ficcional de pássaros apresentado durante um filme clássico dirigido por Alfred Hitchcock: Os Pássaros (1963). O filme se passa na cidade litorânea Bodega Bay, onde sem motivo aparente, várias espécies de pássaros começam a atacar os habitantes da cidade. O cenário de caos criado pelos pássaros atacando seres humanos em um pequeno vilarejo litorâneo tornou-se atemporal e foi um filme que rendeu diversas interpretações, sejam científicas, psicanalíticas ou filosóficas.

No filme, ao contrário do comportamento de enxame do PSO, por exemplo, os pássaros não possuem um voo coordenado no momento do ataque, onde apenas se organizam, definem um alvo onde se encontram muitos humanos e voam agressivamente objetivando o maior dano possível ao alvo.

Inspirado pela observação do comportamento fictício de ataque dos pássaros do filme, o Algoritmo Inspirado nos Pássaros de Hitchcock (AIPH) é uma técnica de inteligência de enxame adaptativa proposta para otimização global de problemas contínuos e complexos de otimização. Sua vantagem em relação à diversas técnicas no estado da arte, está na consistência que apresenta em seus resultados, mesmo com o aumento da dimensão do problema, e não precisar de ajustes em parâmetros de funcionamento do algoritmo para apresentar um bom desempenho.

Este trabalho propõe inicialmente apresentar a sensibilidade à definição do tamanho da revoada e a quantidade iterações necessárias para resultados satisfatórios em diferentes tipos de problemas. Para verificação do desempenho da técnica usou-se 8 funções objetivo de teste. A comparação com as técnicas da literatura é feito com quatorze técnicas metaheurísticas da literatura, visando apresentar os resultados do AIPH em cenários diferentes de dimensionalidade.

Os resultados mostram que o AIPH apresenta desempenho geral superior em problemas de alta dimensão, ao encontrar resultados satisfatórios com um máximo de até 1000 iterações, com a vantagem adicional de não precisar escolher parâmetros de funcionamento do algoritmo, utilizando parâmetros adaptativos.

O restante desta dissertação está organizada em cinco capítulos, cujos conteúdos são resumidamente descritos a seguir.

Inicialmente, o Capítulo 1 apresenta uma abordagem introdutória com os principais conceitos a cerca de otimização, problemas de otimização, métodos heurísticos e metaheurísticos.

No Capítulo 2, apresenta-se um levantamento bibliográfico de trabalhos envolvendo o desenvolvimento de técnicas de otimização metaheurísticas, apresentando de maneira sucinta uma introdução teórica sobre vinte e três técnicas da literatura. Essas técnicas são classificadas considerando o tipo de inspiração da técnica: princípios da matemática, relações humanas, propriedades da física ou bio-inspiradas. São explicadas sucintamente as inspirações de cada técnica, representações do processo de busca e descrição dos respectivos algoritmos. Neste capítulo são vistas técnicas que se destacaram no estado da arte, como exemplo, técnicas clássicas como o *Particle Swarm Optimization* (PSO), *Artificial Bee Colony* (ABC) e *Cuckoo Search* (CS). Outras técnicas mais recentes também

são vistas, como o *Elephant Herding Optimization* (EHO), *Sine Cosine Algorithm* (SCA) e *Whale Optimization Algorithm* (WOA).

O Capítulo 3 descreve detalhadamente o desenvolvimento da nova técnica de otimização, apresentando a inspiração, os critérios e a fundamentação teórica relacionada com cada recurso implementado na construção do algoritmo.

O Capítulo 4 apresenta a análise dos resultados obtidos com a aplicação da técnica AIPH. Primeiramente, é analisado a dispersão das soluções obtidas com as repetições do AIPH e o desempenho da técnica em diferentes condições de tamanho de revoada e máximo de iterações. Em seguida, são realizadas comparações entre as técnicas com diferentes dimensões, utilizando 8 funções de teste clássicas como *benchmark*. A análise permite a asserção do desempenho das técnicas citadas no Capítulo 2 e a técnica proposta neste trabalho.

Por fim, o Capítulo 5 completa esta dissertação, apresentando as principais conclusões obtidas no desenvolvimento do presente trabalho. São apresentadas também possíveis melhorias para o desenvolvimento de variações da técnica AIPH como trabalhos futuros.

Capítulo 1

OTIMIZAÇÃO, MÉTODOS HEURÍSTICOS E METAHEURÍSTICOS

ESTE capítulo oferece uma introdução aos métodos heurísticos e metaheurísticos, que são comumente empregados na área de otimização. Inicialmente, serão apresentados os diferentes tipos de problemas de otimização: combinatória, uniobjetivo e multiobjetivo. Em seguida, serão descritos os conceitos básicos de heurística e metaheurística, com suas respectivas particularidades. Por fim, será feita uma abordagem introdutória as metaheurísticas evolutivas e de inteligência de enxame.

1.1 Problemas de Otimização

Os problemas de otimização consistem em achar o melhor resultado, dentro de uma categoria de possíveis soluções a partir das variáveis do projeto, para maximizar ou minimizar uma função, geralmente chamada de função objetivo ou função custo. De acordo com o espaço de busca das variáveis, esses problemas podem ser divididos em duas categorias: aqueles cujas variáveis assumem valores pertencentes aos números reais e aqueles cujas variáveis assumem pertencentes aos números inteiros, classificados, respectivamente, como problemas de otimização contínua e otimização combinatória.

Em problemas de otimização combinatória procuram-se as melhores alternativas em um conjunto de possibilidades que é finito, entretanto, muito grande. O processo de obtenção da "melhor" solução implica em procurar a melhor combinação dos valores inteiros de certas variáveis. Um exemplo comum de problema de otimização combinatória é o caixeiro viajante, que é um problema que tenta determinar a menor rota para percorrer

uma série de cidades, visitando-as uma única vez cada uma delas, e por fim retornar à cidade de origem.

Já os problemas de otimização contínua são representados por um espaço de busca contínuo, com as variáveis podendo assumir qualquer valor dentro de um intervalo de variação. Existem muitos exemplos que se encaixam nesta categoria, encontrados em áreas como engenharia, administração, logística, economia, biologia ou de outras ciências.

Os problemas de otimização podem ainda ser classificados segundo a função objetivo, que é a representação matemática do critério de eficiência adotado no problema de otimização. Esse critério pode ser representado por uma única função objetivo ou por várias funções objetivo.

Nos problema de otimização com uma única função objetivo, os métodos de otimização tentam encontrar uma única solução satisfatória que represente o mínimo (ou máximo) de uma função objetivo.

Problemas definidos como "multiobjetivo" são aqueles que procuram obter a "melhor resposta" em mais de um aspecto, atendendo à vários objetivos simultaneamente. Em problemas deste tipo, é comum que as funções objetivo sejam conflitantes, ou seja, buscar a melhor resposta do problema em relação a um dado objetivo pode representar uma resposta pior em relação a outros objetivos do problema. Este trabalho terá foco problemas de otimização contínua e em funções com único objetivo.

1.2 Algoritmos de Otimização

Um algoritmo é um processo sistemático uma sequência finita de instruções bem definidas para a resolução de um problema (KNUTH, 1997). Os algoritmos usados para resolução de um problema de otimização podem ser classificados como determinísticos ou estocásticos.

1.2.1 Algoritmos Determinísticos

Os algoritmos determinísticos são aqueles em que dada uma determinada entrada, sempre será produzida a mesma saída. A solução é dependente do ponto de partida fornecido e pode-se provar a convergência destes métodos para uma solução ótima, porém sem evidências de que ela é a global. Os algoritmos clássicos de otimização, que requerem conhecimento das derivadas da função objetivo, são exemplos de algoritmos determinísticos.

Os melhores resultados destes métodos são para funções deriváveis e convexas (IZMAILOV A. F., 2014).

Os métodos determinísticos possuem uma grande vantagem que é o baixo número de avaliações da função objetivo. Alguns métodos clássicos para resolução são: Método do Gradiente, Método de Newton, Métodos Quase-Newton, entre outros (FRIEDLANDER, 1994).

1.2.2 Algoritmos Estocásticos

Os algoritmos conhecidos como estocásticos ou probabilísticos têm como principal característica a busca pelo ótimo através de regras de probabilidade trabalhando de maneira "aleatória". Esses métodos utilizam apenas os valores da função objetivo, podendo não requerer informações sobre suas derivadas, o que pode ser uma alternativa como um método para estes casos. Estas técnicas ganharam popularidade com a evolução dos computadores, já que requerem um grande número de avaliações da função objetivo e das restrições (SUCUPIRA, 2004). Isto é necessário para que se dê chance ao método de explorar o espaço de busca onde está contida a solução ótima. Alguns exemplos de algoritmos estocásticos são: Algoritmos Genéticos (HOLLAND, 1973), Enxame de Partículas (KENNEDY; EBERHART, 1995), dentre outros.

1.3 Métodos de Solução de Problemas de Otimização

Os métodos exatos visam encontrar a melhor solução para um problema de otimização, ou seja, o ótimo global, entretanto, devido ao número elevado de soluções existentes em dimensões elevadas, torna-se inconcebível uma enumeração completa de todas soluções. Como alternativas aos métodos exatos, os algoritmos heurísticos e os metaheurísticos são métodos que fornecem uma solução satisfatória e próxima de ótima, por meio de cálculos com um custo computacional menor. Nesta seção, serão descritas as heurísticas que adicionam uma espécie de guia aos métodos de busca e as metaheurísticas, que implementam definições para melhorar o desempenho das heurísticas.

1.3.1 Métodos Heurísticos

A capacidade heurística é uma característica humana, onde processos cognitivos empregados em decisões não racionais, sendo definidas como estratégias simplificadoras que ignoram parte da informação com o objetivo de tornar a escolha mais fácil e rápida (KAHNEMAN, 2012).

Um algoritmo é considerado um método heurístico quando não há conhecimentos matemáticos completos sobre seu comportamento, ou seja, quando, sem oferecer garantias, o algoritmo objetiva resolver problemas complexos com custos aceitáveis de tempo, para encontrar soluções razoáveis. A importância dos métodos heurísticos advém do bom desempenho médio, detectado de forma predominantemente experimental, desses métodos, quando aplicados ao tratamento de problemas, normalmente NP-difíceis, para os quais não são conhecidos métodos eficientes que forneçam garantias (LANDMANN R., 2011).

1.3.2 Métodos Metaheurísticos

A área que estuda as metaheurísticas é considerada um subcampo primário da área de otimização estocástica, classe geral de algoritmos e técnicas que empregam algum grau de aleatoriedade para encontrar soluções tão ótimas quanto possível para problemas reconhecidamente difíceis (SUCUPIRA, 2004).

O prefixo “meta” vem do grego e significa “mudança”, “além” e “após. Neste contexto, é possível denominar a metaheurística como um nível superior da heurística. A ideia principal desse conceito é a criação de heurísticas generalizadas que possam resolver diversos tipos de problemas sem a necessidade de grandes alterações na estrutura do algoritmo (FRANçA, 2005).

Os métodos e algoritmos de metaheurística são de uso e aplicação geral, normalmente não se “prendem” a ótimos locais e buscam o ótimo global. Além disso, diferentemente dos métodos heurísticos, vários parâmetros devem ser ajustados dependendo do problema particular que se está resolvendo.

Metaheurísticas, que são heurísticas genéricas mais sofisticadas, onde uma heurística mais simples é gerenciada por um procedimento que visa explorar inteligentemente a instância do problema e o seu espaço de soluções, onde, normalmente, não se “prendem” a ótimos locais e buscam o ótimo global.

Uma metaheurística é considerada eficiente quando possui um balanço dinâmico entre diversificação e intensificação. Considerando uma distinção entre os termos ingleses *exploration* e *exploitation*, tem-se que o primeiro traduz-se por diversificação, exploração diversificada, busca em largura ou simplesmente exploração, já o segundo, como exploração focada, busca em profundidade ou intensificação. A metaheurística tem como desafio encontrar o equilíbrio ideal entre diversificação e intensificação. A estratégia de busca de uma metaheurística depende da filosofia empregada por ela, e seu objetivo é escapar dos mínimos locais a fim de proceder a exploração do espaço de busca por soluções ainda melhores (BECCENERI, 2008).

Embora todos os procedimentos descritos por metaheurísticas possam ser vistos como estratégias de busca, algumas características desses procedimentos permitem classificar a maioria das metaheurísticas em tipos bastante distintos.

A classificação de um algoritmo metaheurístico está associado a origem da inspiração aplicada e ao conceito teórico utilizado na criação do mecanismo de busca, podendo ter inspiração na natureza ou não.

As metaheurísticas não inspiradas na natureza podem ser simplesmente um conjunto de regras de busca focadas na utilização de algum recurso matemático ou computacional especial, sem incorporar nenhum comportamento observado na natureza.

Uma metaheurística é dita inspirada na natureza quando suas regras de busca tentam simular mecanismos evolutivos ou aspectos do comportamento coletivo de sistemas auto-organizados, distribuídos, autônomos, flexíveis e dinâmicos de seres vivos ou da natureza. Desta forma, as metaheurísticas inspiradas na natureza podem ser classificadas em evolutivas ou de inteligência de enxame.

1.3.2.1 Metaheurísticas Evolutivas

As metaheurísticas evolutivas lidam com uma população - essencialmente, um conjunto - de soluções (possivelmente inviáveis) que evolui através da interação entre seus elementos, procurando preservar as características desejáveis (de forma a melhorar, ao longo das gerações, a qualidade média das soluções), sem comprometer a diversidade dentro da população. Um exemplo de metaheurística evolutiva são os algoritmos genéticos (KNUTH, 1997).

Nos algoritmos genéticos, cada elemento da população é chamado cromossomo, sendo a codificação de uma solução na forma de uma sequência de símbolos, chamados

genes. Um algoritmo genético também deve definir uma função que mede a aptidão de cada indivíduo, onde indivíduos que representam soluções de baixo custo tendem a ser mais aptos, enquanto os indivíduos que representam soluções inviáveis têm baixa aptidão. A população é alterada através de dois operadores principais: a mutação, que modifica um dos genes de um indivíduo e tem baixa probabilidade de ocorrer, e a recombinação, que constrói um novo indivíduo utilizando apenas a informação genética de dois outros indivíduos selecionados aleatoriamente. Indivíduos com baixa aptidão têm menor probabilidade de serem selecionados para recombinação (KNUTH, 1997).

1.3.2.2 Inteligência de Enxame

A Inteligência de Enxames, também referenciada como Inteligência de Colônias ou Inteligência Coletiva, é um conjunto de técnicas baseadas no comportamento coletivo na busca de soluções. Estes sistemas são formados por uma população de agentes computacionais simples que possuem a capacidade de perceber e modificar o seu ambiente de maneira local. Esta capacidade torna possível a comunicação entre os agentes, que captam as mudanças no ambiente geradas pelo comportamento de seus congêneres. Embora não exista uma estrutura centralizada de controle que estabelece como os agentes devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre os agentes geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema (SUCUPIRA, 2004).

A inteligência de enxame consiste em uma população de indivíduos capazes de interagir entre si e com o meio ambiente, esta capacidade de interação deve atender as propriedades (MILLONAS, 1994):

- Proximidade - os agentes devem ser capazes de interagir;
- Qualidade - os agentes devem ser capazes de avaliar seus comportamentos;
- Diversidade - permite ao sistema reagir a situações inesperadas;
- Estabilidade - nem todas as variações ambientais devem afetar o comportamento de um agente;
- Adaptabilidade - capacidade de adequação a variações ambientais.

Em um algoritmo de Enxame de Partícula, precursor dos algoritmos de Inteligência de Enxame, tem-se um conjunto de partículas que percorrem o espaço de busca apresentando comportamentos aleatórios em relação à individualidade e à sociabilidade. A individualidade de uma partícula está relacionada à ênfase dada, em seus movimentos, à melhor solução já encontrada por ela mesma, enquanto sua sociabilidade reflete o grau de importância dado por ela à melhor solução já encontrada por seus vizinhos. Além disso, a estrutura de vizinhanças é construída de forma que os progressos obtidos em cada região tenham influência, potencialmente, em todas as partículas (SUCUPIRA, 2004).

Os algoritmos de inteligência de enxame podem incorporar os mais diversos tipos de inspiração, tais como os bioinspirados, os baseados em relações humanas ou em propriedades da física ou até mesmo em seres ficcionais.

Em geral, as metaheurísticas de inteligência de enxame possuem o mesmo procedimento, consistem em maximizar ou minimizar uma função escolhendo valores de parâmetros de entrada em um conjunto permitido e computando o valor da função. São criados métodos de otimização para um dado domínio definido, encontrando os melhores valores disponíveis de alguma função objetivo. Uma metaheurística como um processo de geração iterativo que orienta uma heurística subordinada combinando conceitos inteligentemente diferentes para explorar localmente e globalmente o espaço de pesquisa, para encontrar soluções eficientemente próximas da ótima (PREMALATHA; BALAMURUGAN, 2015). Entradas e estrutura básica de uma técnica de otimização são apresentadas no Algoritmo 1.

Algoritmo 1 Representação de Metaheurísticas

Entrada: Define o critérios de parada:
- Máximo de iterações (G)
- Máximo de avaliações ($MaxAval$)
- Tolerância máxima de erro mínimo esperada (tol)
- Máximo de iterações sem melhoria(Gm)

Define a quantidade de agentes de busca (k).
Define a quantidade de dimensões D do vetor de posição dos agentes de busca x_i.
Define os limites do espaço de busca $[min_d, max_d]$, $\forall\ d \in |\ D = \{1, 2, \cdots, d\}$.
Define a função objetivo $f(x)$, $\forall\ x \in \mathbb{R} \mid x = \{x_1, x_2, ..., x_k\}$.
Inicializa o conjunto de agentes de busca x_i, $\forall\ i \in \mathbb{Z} \mid i = \{1, .., k\}$.
Avalia a aptidão de cada agente, estabelece a melhor posição como x_{best} e a melhor solução como $best$.

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Etapas do processo de busca da metaheurística.
  fim
  Retorna a melhor posição e sua respectiva solução
fim

No Algoritmo 1, cada agente de busca x_i é um vetor com um total de d dimensões sendo $d \in D$, onde a especificação de uma dimensão / variável no agente de busca será

representado por x_i^j, com $j = (1, ..., d)$ e $i = (1, ..., k)$. O conceito geral apresentado nesta seção será estendido para os subsequentes algoritmos de descrição do procedimento de busca, seguindo a mesma estrutura e consideradas as mesmas entradas básicas.

1.4 Considerações Finais

Neste capítulo foram apresentados os conceitos básicos para uma noção geral sobre otimização. Apresentaram-se os diferentes tipos de problema de otimização, dando ênfase a problemas de otimização contínua. Foram apresentadas também as diferenças entre os tipos de problemas de otimização, métodos heurísticos e metaheurísticos. Foi dada ênfase a problemas de otimização contínua. Também foram explicadas de forma geral, as diferenças entre algoritmos determinísticos e estocásticos, assim como os conceitos de heurística e metaheurística. Por último, foram abordadas as metaheurísticas evolutivas e de inteligência de enxame, apresentando a estrutura básica de um algoritmo de inteligência de enxame.

Capítulo 2

TRABALHOS RELACIONADOS

NO paradigma da otimização por inteligência de enxame o comportamento coletivo dos indivíduos em uma população gera soluções coerentes empregadas em modelos abstratos da inteligência e aprendizado coletivo.

A proposta original de um algoritmo heurístico baseado em um ambiente social foi inicialmente introduzida em 1995, por Kennedy e Eberhart, sendo um método que tem como objetivo buscar a solução ótima, em um espaço de busca, através da troca de informações entre indivíduos de uma população determinando qual trajetória cada um deles deverá tomar no espaço de busca.

Desde então, o desafio de encontrar uma solução ótima para uma grande variedade de problemas vem motivando o desenvolvimento de várias técnicas de otimização baseados em princípios da física, biologia e em interações sociais. Recentemente, muitos algoritmos de inteligência de enxame eficientes foram propostos, alguns com inspiração em propriedades da física como o Busca de Vórtices (DOAN; ÖLMEZ, 2015), bem como algoritmos bioinspirados em Algas Artificiais (UYMAZ; G.; YEL, 2015), Baleias (MIRJALILI; LEWIS, 2016) e Rebanho de Elefantes (WANG; DEB; COELHO, 2015), além de algoritmos baseados em modelos matemáticos como o Algoritmo Seno-Cosseno (MIRJALILI, 2016b).

Dentre as mais recentes técnicas, pode-se destacar também o Mapeamento de Variância Média (Erlich et al., 2010) e suas variações, devido ao seu bom desempenho nas últimas competições de otimização de funções de objetivo único no *Congress on Evolutionary Computation* (CEC), em 2014 e 2015 .

Neste capítulo, são apresentadas algumas das principais técnicas que utilizam inteligência de enxames para otimização de problemas numéricos de objetivo único, ofere-

cendo uma abordagem introdutória sobre suas inspirações e passos do processo de busca. A descrição divide as técnicas por tipo de inspiração dos algoritmos de otimização.

2.1 Otimização Inspirada em Princípios da Matemática

O campo de otimização por inteligência de enxame possui uma extensa lista de técnicas inspiradas na natureza e sociedades. Entretanto, um tipo de técnica baseado em população não-natural, que possuem como fundamento modelos matemáticos e métricas estatísticas para estratégia de busca pelo ótimo global, vem mostrando resultados satisfatórios em trabalhos acadêmicos. Nesta seção será apresentada uma das técnicas em destaque neste segmento: o Algoritmo Seno-Cosseno.

2.1.1 Algoritmo Seno-Cosseno

Desenvolvido por Mirjalili (2016), o Algoritmo Seno-Cosseno (*Sine Cosine Algorithm* - SCA) cria múltiplas soluções candidatas iniciais e é capaz de evoluir para direção da solução de resultados, usando modelos matemáticos e funções de equilíbrio. Várias variáveis aleatórias e adaptativas também são integradas em cada um deles, de modo a enfatizar a ampliação e a exploração do espaço de pesquisa em diferentes níveis de otimização (MIRJALILI, 2016b). O processo de busca da técnica pode ser visto no Algoritmo 2.

Algoritmo 2 Pseudocódigo do Algoritmo Seno-Cosseno

Entrada: Parâmetros comuns (Algoritmo 1).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Avalia cada um dos x_i a partir da função objetivo.
    Atualiza o x_{best} e *best* até o momento.
    Atualiza r1, r2, r3 e r4.
    Atualiza a posição de cada x_i usando a Equação 2.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

Existem quatro parâmetros principais em SCA: r_1, r_2, r_3 e r_4. O parâmetro r_1 dita a região da próxima posição, que poderia ser ou entre o espaço entre a solução e a orientação ou fora dela (MIRJALILI, 2016b). Este parâmetro é adaptativo e segue a Equação 1:

$$r_1 = a - a * \frac{t}{MaxGer}, \tag{1}$$

onde t é a iteração corrente, $MaxGer$ é o número máximo de iterações, e a é uma constante.

O parâmetro r_2 é um valor aleatório no intervalo $[0, 2\pi]$ que define até que ponto o movimento deve ser próximo ou afastado do destino. O parâmetro r_3 é um valor aleatório no intervalo $[0, 2]$ que traz um peso aleatório para o destino, de forma a enfatizar esteticamente ($r_3 > 1$) ou desacentuar ($r_3 < 1$) o efeito de destino na definição da distância. Finalmente, o parâmetro (r_4) é um número aleatório no intervalo [0,1] que alterna igualmente entre os componentes seno e cosseno na Equação 2 de atualização de posição (MIRJALILI, 2016b).

$$\tilde{x}_i = \begin{cases} \tilde{x}_i = x_i + r_1 * sin(r_2) * |r_3 * P_i x_i| & \text{se} \quad r_4 < 0.5, \\ \tilde{x}_i = x_i + r_1 * cos(r_2) * |r_3 * P_i x_i| & \text{se} \quad r_4 \geq 0.5. \end{cases} \tag{2}$$

2.2 Otimização Inspirada em Relações Humanas

Nesta seção serão definidos resumidamente os conceitos usados em técnicas inspiradas em relações humanas. Para exemplificar este tipo de inspiração serão mostrados algumas das técnicas relacionadas, com suas respectivas explanações e passos de busca do ótimo global usados por cada técnica.

2.2.1 Ensino Baseado na Aprendizagem

Desenvolvida por Rao et al. em 2010, é uma técnica que se baseia no efeito da influência de um professor sobre o resultado de alunos de uma classe, onde a produção é considerada em termos de resultados ou notas. O professor é geralmente considerado como uma pessoa altamente instruída que compartilha seu conhecimento com os alunos. A qualidade de um professor afeta o resultado dos alunos. Por consequência, um bom professor treina os alunos de tal forma que eles podem ter melhores resultados em termos de suas marcas ou graus (RAO; SAVSANI; VAKHARIA, 2011). Segue abaixo o Algoritmo 3 da técnica de Ensino Baseado na Aprendizagem (*Teaching Learning Based Optimization* - TLBO).

O processo de busca do TLBO é dividido em duas partes. A primeira parte consiste na Fase do Professor e a segunda parte consiste na Fase do Aluno. A Fase do Professor significa aprender com o professor (x_{best}), apresentada no Algoritmo 3 entre as linhas 5 à 8. A Fase do Aluno representa as interações entre alunos e é apresentada no Algoritmo 3 entre as linhas 9 à 17 (RAO; SAVSANI; VAKHARIA, 2011).

Na Fase do Professor a modificação dos alunos é feita usando como referência o melhor aluno até o momento. A nova posição do aluno é calculada a partir da Equação

Algoritmo 3 Pseudocódigo do Ensino Baseado na Aprendizagem

Entrada: Parâmetros comuns (Algoritmo 1).

```
início
    Calcula a média de cada dimensão desenhada (Mean).
    enquanto (critério de parada não é satisfeito) faça
        para i = 1 até k faça
            Modifica a solução baseada na melhor solução.
            se a nova solução é melhor que a existente então
                Atualiza a posição de x_i usando a Equação 3.
            fim
            Seleciona quaisquer duas soluções aleatoriamente x_u e x_v.
            se (x_u melhor que x_v) então
                Atualiza a posição de x_i usando a Equação 4.
            senão
                Atualiza a posição de x_i usando a Equação 5.
            fim
            se (a nova solução é melhor que a existente) então
                x_i = x̃_i
            fim
        fim
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

3:

$$\tilde{x}_i = x_i + r_i(x_{best} - T_F * (Mean)), \tag{3}$$

onde T_F é um fator de ensino que decide o valor da média a ser alterada, e r_i é um número aleatório no intervalo [0, 1]. O valor de T_F pode ser 1 ou 2, que é novamente um passo heurístico e decidido aleatoriamente com igual probabilidade como $T_F = [1 + rand(0, 1) * (2 - 1)]$, arredondado para um número inteiro (RAO; SAVSANI; VAKHARIA, 2011).

Na Fase do Aluno são selecionados aleatoriamente dois alunos, x_u e x_v, que serão usados para o aprendizado. Caso o aluno x_u tenha um resultado melhor na função de aptidão, usa-se a Equação 4:

$$\tilde{x}_i = x_i + r_i(x_u - x_v), \tag{4}$$

e em caso contrário a Equação 5,

$$\tilde{x}_i = x_i + r_i(x_v - x_u), \tag{5}$$

onde r_i é um número aleatório no intervalo [0, 1] (RAO; SAVSANI; VAKHARIA, 2011).

2.2.2 Competição de Liga de Futebol

Nesta técnica, apresentada em 2013 por Moosavian e Roodsari, as ideias fundamentais são inspiradas em ligas de futebol e baseadas nas competições entre equipes e jogadores. Desta forma, as competições entre as equipes em uma liga de futebol para alcançar o sucesso, e entre os jogadores para ser um jogador *star* ou *super star* pode ser simulada

para resolver problemas de otimização. Semelhante a uma liga de futebol em que cada jogador deseja ser o melhor (*super star*), em um problema de otimização cada vetor de solução procura a posição ideal global. Portanto, cada jogador em uma liga, o jogador *star* em cada equipe e o *super star* podem ser assumidos como um vetor de solução, um ótimo local e o ótimo global, respectivamente (MOOSAVIAN; ROODSARI, 2014). No Algoritmo 4 é mostrado os passos da técnica Competição da Liga de Futebol (*Soccer League Competition* - SLC).

Algoritmo 4 Pseudocódigo da Competição de Liga de Futebol

Entrada: Parâmetros comuns (Algoritmo 1).

início
  Número de times ($nTeam$).
  Número de jogadores fixos (FP).
  Número de substitutos (S).
  Parâmetros $\chi_1, \chi_2, \mu_1, \mu_2, \tau_1, \tau_2$.
  Avalia os jogadores a partir da função objetivo (poder do jogador).
  enquanto *(critério de parada não é satisfeito)* **faça**
    Ordena os jogadores.
    Aloca os jogadores em times.
    para n = *1 até* $nTeam$ **faça**
      Calcule o poder do time.
    fim
    Realiza entre todos os possíveis pares de equipes no campeonato.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

Como premissa do Algoritmo 4 tem-se que cada equipe é composta por 11 jogadores fixos, definidos por vetores de solução principais no algoritmo SLC, e alguns substitutos, descritos por vetores de solução reservados no algoritmo SLC. Para cada jogador, calcula-se uma função objetiva que representa a potência do respectivo jogador. Em um problema de minimização, valores menores de funções objetivas (função de custo) ilustram jogadores poderosos (PP). O poder total de uma equipe é definido como o valor médio de poder de seus jogadores, incluindo fixo e substituto(MOOSAVIAN; ROODSARI, 2014). A fórmula a seguir mostra como o Poder de uma Equipe (TP) é calculado:

$$TP(n) = \left(\frac{1}{nPlayer}\right) * \sum_{i=1}^{nPlayer} PP(i,n), \tag{6}$$

onde $nPlayer$ é o número total de jogadores na i-ésima equipe. $PP(i,n)$ é o poder do i-ésimo jogador na n-ésima equipe($PP(i,n) = 1/cost(i,n)$). Em cada jogo, a equipe com mais poder tem uma maior chance de ganhar. A probabilidade de vitória para cada equipe

em um jogo é dada por:

$$Pv(k) = \frac{TP(k)}{(TP(n) + TP(k))}, \tag{7}$$

$$Pv(k) = \frac{TP(n)}{(TP(n) + TP(k))}, \tag{8}$$

com Pv representando a probabilidade de vitória. Note-se que a soma de $Pv(k)$ e $Pv(n)$ é igual a 1. Após cada partida, o vencedor e o perdedor são notados e alguns jogadores (vetores de solução), incluindo os substituídos e fixos, experimentam mudanças. Estas mudanças, que visam melhorar o desempenho de jogadores e equipes, são simuladas com os seguintes operadores:

(a) *Operador de Imitação*: os jogadores fixos (FP) da equipe vencedora, imitam o jogador star (SP) em sua própria equipe e o jogador Super Star (SSP) no campeonato para melhorar suas atividades futuras. Da mesma forma, os vetores de solução relativos aos jogadores fixos na equipe vencedora caminham para a melhor solução da própria equipe e o melhor vetor de solução da liga (MOOSAVIAN; ROODSARI, 2014). No algoritmo SLC, a imitação é realizada pelas Equações 9:

$$FP(i,n) = \mu_1 * FP(i,n) + \tau_1 * (SSP - FP(i,n)) + \tau_2 * (SP - FP(i,n)), \tag{9}$$

$$FP(i,n) = \mu_2 * FP(i,n) + \tau_1 * (SSP - FP(i,n)) + \tau_2 * (SP - FP(i,n)), \tag{10}$$

onde $\mu_1 \sim U(\theta, \beta), \mu_2 \sim U(0, \theta), \tau_1 \sim U(0,2)$ e $\tau_2 \sim U(0,2)$ são números aleatórios com distribuição uniforme. $FP(i,n)$ representa o i -ésimo jogador fixo da n -ésima equipe, e $SP(n)$ é o jogador star da i-ésima equipe. Propõe-se também que: $1 < \beta < 2$ e $0 < \theta < 1$. Primeiro, o vetor de solução de jogadores fixos (PF) na equipe vencedora experimenta um grande movimento em direção à direção vetorial resultante de SP e SSP (Equação 9). Se o vetor de solução gerado recentemente nesta nova posição fosse melhor do que o vetor de solução mais antigo, ele é substituído pelo antigo. Caso contrário, o vetor solução experimenta um movimento médio em direção ao vetor resultante (Equação 10). Se esta solução fosse melhor do que a anterior, ela é reposta com o vetor antigo. No caso em que nenhum dos movimentos discutidos deu um vetor de solução melhor, o jogador é mantido na sua posição sem alteração (MOOSAVIAN; ROODSARI, 2014).

(b) *Operador de Instigação*: os substitutos de uma equipe vencedora (S) têm de provar um desempenho igual ao valor médio do nível de desempenho dos jogadores fixos na sua equipe, a fim de serem jogadores fixos (MOOSAVIAN; ROODSARI, 2014). Este processo,

que é realizado pelo operador de instigação no algoritmo SLC, é descrito por:

$$S(i,n) = C(n) + \chi_1 * (C(n) - S(i,n)), \tag{11}$$

$$S(i,n) = C(n) + \chi_2 * (S(i,n) - C(n)), \tag{12}$$

com $\chi_1 \sim U(0.9, 1)$ e $\chi_2 \sim U(0.4, 0.6)$ são números aleatórios com distribuição uniforme, e $C(n)$ é o valor médio dos vetores de solução do jogador fixo na n-ésima equipe. $S(i,j)$ é o i-ésimo substituto da n-ésima equipe.

Em primeiro lugar, o vetor de solução do jogador substituto mais fraco na equipe vencedora experimenta um movimento para frente em direção ao centro de gravidade de jogadores fixos (Equação 11). Se o vetor de solução gerado recentemente relativo a esta nova posição fosse melhor do que o último, ele é substituído pelo seu vetor antigo. Caso contrário, o jogador mencionado experimentará um movimento para trás em direção ao centro de gravidade (Equação 12). Se esta solução fosse melhor do que a solução mais fraca, este vetor é substituído pelo antigo. Na situação em que nenhum dos movimentos discutidos forneceu um vetor de solução melhor para melhorar a solução mais fraca, um novo vetor é gerado aleatoriamente e substituído pelo antigo. De um modo geral, o operador de provocação atua sobre os substitutos mais fracos dos vencedores. Se o avanço foi evidente em seu desempenho, eles são mantidos na equipe. Ou então, eles são exportados da equipe enquanto novos jogadores aleatórios (vetores de solução) são inseridos para jogos futuros (MOOSAVIAN; ROODSARI, 2014).

(c) *Operador de Mutação*: Jogadores fixos da equipe perdedora em uma partida devem revisar sua atividade para evitar falhas em jogos futuros. Para executar esta operação, as posições de alguns jogadores são alteradas aleatoriamente. Esse mecanismo usado para criar diversificação em soluções (MOOSAVIAN; ROODSARI, 2014).

(d) *Operador de Substituição*: O treinador geralmente considera novas combinações de substitutos para jogos futuros. Da mesma forma, uma abordagem baseada em aleatório é aplicada para refletir o impacto treinador principal neste algoritmo. Para fazer isso, um par de novos vetores substituto está sendo testado. Se for obtida uma resposta adequada, este par efetivo é introduzido na equipe (MOOSAVIAN; ROODSARI, 2014). Este processo, que é realizado pelo Operador de Substituição no algoritmo SLC, é descrito por:

$$\tilde{S}(i,n) = \alpha * S(i,n) + (1-\alpha) * S(i,k), \tag{13}$$

$$\tilde{S}(i,k) = \alpha * S(i,k) + (1-\alpha) * S(i,n), \tag{14}$$

onde $\alpha \sim U(0,1)$ é um vetor de números aleatórios com distribuição uniforme. O número de novos pares examinados é proposto para ser igual ao número de substitutos da equipe (MOOSAVIAN; ROODSARI, 2014). Em uma visão geral, os 4 operadores descritos têm os seguintes efeitos no algoritmo (MOOSAVIAN; ROODSARI, 2014):

- O Operador de Imitação acelera a capacidade de pesquisa do algoritmo;

- O Operador Provocação fornece soluções de alta precisão para os complexos problemas de otimização;

- Os Operadores de Mutação e Substituição ajudam o algoritmo proposto a escapar dos mínimos e planaltos locais;

Após cada jogo, 4 operadores discutidos atuam sobre os jogadores (vetores de solução) e os poderes da equipe são atualizados de acordo com as novas soluções. Obviamente, as equipes poderosas são mais prováveis ser bem sucedidas em seus momentos futuros. Este processo continua até o final da temporada e o jogador Super Star (SSP) da liga produz a melhor solução para o problema de otimização. Após cada temporada, os jogadores são organizados tendo em conta o seu poder atualizado. Antes de começar uma nova liga, os melhores jogadores são dedicados às melhores equipes, jogadores médios são alocados para as equipes com um desempenho médio, e os jogadores mais fracos são transferidos para as equipes de baixo na tabela de campeonato (MOOSAVIAN; ROODSARI, 2014).

2.3 Otimização Inspirada em Propriedades Físicas

Nesta seção serão mostradas algumas técnicas que possuem como inspiração em fundamentos da teoria de Física para sua estratégia de busca do valor ótimo, mostrando os conceitos usados e o processo de busca do ótimo global pelas respectivas técnicas.

2.3.1 Algoritmo de Busca Gravitacional

Com sua introdução em 2009 por Rashedi et al., é um algoritmo inspirado pela lei newtoniana da gravidade e do movimento. Baseia-se na Lei da Gravidade e interações de massa. O algoritmo é composto por uma coleção de agentes de busca que interagem uns com os outros através da força de gravidade. Os agentes são considerados como objetos e seu desempenho é medido pelas suas massas. A força da gravidade causa um movimento global onde todos os objetos se movem para outros objetos com massas mais pesadas. O movimento lento de massas mais pesadas garante o passo de exploração do algoritmo e

corresponde a boas soluções (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009). O pseudocódigo do Algoritmo de Busca Gravitacional (*Gravitational Search Algorithm* - GSA) pode ser visto no Algoritmo 5.

Algoritmo 5 Pseudocódigo do Algoritmo de Busca Gravitacional

```
Entrada: Parâmetros comuns (Algoritmo 1).
         Define a constante gravitacional (G_0).
         Define a constante α.
         Define elite.
         Define distância entre os dois objetos.
         Define percentual de agentes que aplicam força aos outros.

início
    enquanto (critério de parada não é satisfeito) faça
        Avalia a aptidão dos objetos a partir da função f(x).
        Atualiza G, x_best, x_worst.
        para i = 1 até k faça
            Atualiza as massas dos objetos (M).
            Calcula a aceleração e velocidade.
        fim
        Atualiza a posição dos objetos.
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

As massas seguem à Lei da Gravidade como mostrado na Equação 15 e a Lei do Movimento na Equação 16 (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009):

$$F = G * (M_1 * M_2/R^2), \tag{15}$$

$$a = F/M. \tag{16}$$

Com base na Equação 15, F representa a magnitude da força gravitacional, G é constante gravitacional, M_1 e M_2 são a massa do primeiro e segundo objetos e R é a distância entre os dois objetos. A Equação 15 mostra que na lei de gravidade de Newton, a força gravitacional entre dois objetos é diretamente proporcional ao produto de suas massas e inversamente proporcional ao quadrado da distância entre os objetos. Enquanto que para a Equação 16, a segunda lei de Newton mostra que quando uma força, F, é aplicada a um objeto, sua aceleração, a, depende da força e sua massa, M (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009).

Em GSA, o objeto tem quatro parâmetros que são posição, massa inercial, massa gravitacional ativa e massa gravitacional passiva. A posição da massa representa a solução do problema, onde as massas gravitacional e inercial são determinadas usando uma função de aptidão. O algoritmo procede ajustando as massas gravitacionais e de inércia, enquanto que cada massa apresenta uma solução. Massas são atraídas pela massa mais pesada. Assim, a massa mais pesada apresenta uma solução ótima no espaço de busca.

Os cálculos necessários para a realização das etapas do GSA são (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009):

(a) *Constante Gravitacional*: A constante gravitacional G é calculada a cada iteração, seguindo a Equação 17:

$$G = G_0 * e^{-\alpha/MaxGer}. \tag{17}$$

São inicializados na entrada e serão reduzidos com o tempo para controlar a precisão da pesquisa. $MaxGer$ é o número total de iterações (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009) e α é uma constante.

(b) *Massas dos Objetos*: massas gravitacionais e de inércia para cada objeto são calculadas em cada iteração (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009).

$$M_{ai} = M_{pi} = M_{ii} = M_i, i = 1, 2, ..., k, \tag{18}$$

$$m_i = \frac{f(x_i) - f(x_{worst})}{f(x_{best}) - f(x_{worst})}, \tag{19}$$

$$M_i = \frac{m_i}{\sum_{n=1}^{k} m_n}, \tag{20}$$

com M_{ai} e M_{pi} são as massas gravitacionais ativas e passivas respectivamente, enquanto M_{ii} é a massa de inércia do i-ésimo objeto.

(c) *Aceleração*: a aceleração dos i -ésimo objeto na iteração t é calculada a partir da Equação 21:

$$a_{i,j} = F_{i,j}/M_{ii}, \tag{21}$$

$F_{i,j}$ é a força total que atua no i-ésimo objeto calculada pela Equação 22 como:

$$F_{i \to n} = \sum_{n=1}^{nbest} rand_n * F_{i \to n,j}, \tag{22}$$

$nbest$ é o conjunto de primeiros k objetos com o melhor valor de aptidão e maior massa. $nbest$ irá diminuir linearmente com o tempo e no final haverá apenas um objeto aplicando força aos outros. $F_{i \to n}$ é calculado como a Equação 23:

$$F_{i \to n,j} = G * (M_{pi} * M_{aj}/R_{ij} + \xi) * (x_j^d - x_i^d), \tag{23}$$

onde $F_{i \to n,j}$ é a força que atua sobre o objeto i do objeto n na j-ésima dimensão e t-ésima iteração. R_{ij} é a distância euclidiana entre dois objetos i e n na iteração t . G é a constante gravitacional calculada na mesma iteração, enquanto ξ que é uma pequena constante (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009).

(d) *Velocidade e Posição dos Objetos*: a velocidade e a posição dos objetos na próxima iteração (t + 1) são calculadas com base nas seguintes Equações 24 (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009):

$$\tilde{v}_i = rand_i * x * v_i + a_i, \tag{24}$$

$$\tilde{x}_i = x_i + \tilde{v}_i. \tag{25}$$

2.3.2 Busca de Sistema Carregado

Desenvolvido por Kaveh Talatahari, em 2010, este algoritmo de otimização inspira-se na Lei de Coulomb conhecida pela eletrostática e pelas leis do movimento da mecânica newtoniana. Contém um número de agentes que são chamados partículas carregadas (PC). Cada partícula carregada é considerada como uma esfera, que tem uma densidade de carga de volume uniforme e pode impor uma força elétrica em outras partículas carregadas de acordo com a Lei de Coulomb. Esta força é atrativa e sua magnitude para a partícula carregada localizada dentro da esfera é proporcional à distância de separação entre as partículas carregadas. Para a partícula carregada localizado fora da esfera é inversamente proporcional ao quadrado da distância de separação entre as partículas carregadas. As forças sobrepostas e as leis para o movimento determinam a nova localização das partículas carregadas. Nesta fase, cada partícula carregada move-se na direção às forças resultantes e da sua velocidade anterior (KAVEH; TALATAHARI, 2010). Os passos da técnica de Busca de Sistema Carregado (*Charged System Search* - CSS) pode ser vista no Algoritmo 6.

Algoritmo 6 Pseudocódigo da Busca de Sistema Carregado

Entrada: Parâmetros comuns (Algoritmo 1).
Define as velocidades associadas as partículas carregadas (Regras 1 e 2).
Define o tamanho da memória carregada (CMS).

início
  Inicializa as partículas.
  Ordena das soluções.
  Armazena as CMS primeiras partículas carregadas e suas soluções.
  enquanto *(critério de parada não é satisfeito)* **faça**
    Determina a probabilidade de mover cada PC em direção a outros (Regra 3).
    Calcula o vetor de força de atração para cada PC (Regra 4).
    Move cada uma das partículas carregadas para a nova posição e procura as velocidade (Regra 5).
    se *cada partícula carregada existe em um espaço de busca permitido* **então**
      Corrija sua posição usando a (Regra 7).
    fim
    Avalia e compara os valores da função objetivo para as novas partículas carregadas, ordenando de forma crescente.
    se *alguma nova partícula carregada é melhor que alguma na memória carregada* **então**
      Exclui-se a pior e acrescenta a melhor partícula carregada encontrada (Regra 6).
    fim
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

Conforme visto no Algoritmo 6, para executar o processo de busca do CSS, são usadas as seguintes regras (KAVEH; TALATAHARI, 2010): (a) *Regra 1*: Muitos dos algoritmos de evolução natural mantêm uma população de soluções que evoluem através de alterações aleatórias e seleção. Da mesma forma, o CSS considera um número de PC, representado por partículas carregadas (x_i). Cada x_i tem uma magnitude de carga (q_i) e, como resultado, cria um campo elétrico ao redor de seu espaço. A magnitude da carga é definida considerando a qualidade de sua solução, conforme Equação 26:

$$q_i = \frac{f(x_i) - f(x_{worst})}{f(x_{best}) - f(x_{worst})}, i = 1, 2, , k, \tag{26}$$

onde $f(x_{best})$ e $f(x_{worst})$ são até agora o melhor e a pior aptidão de todas as partículas; $f(x_i)$ representa o valor da função objetivo ou a adequação da PC n; k é o número total de x_is. A distância de separação r_{ij} entre duas partículas carregadas é definida como na Equação 27:

$$r_{ij} = \frac{|x_i - x_j|}{|(x_i - x_j)/2 - x_{best}| + \varepsilon}, \tag{27}$$

onde x_i e x_j são as posições das i-ésima e j-ésima das x_is, x_{best} é a posição da melhor corrente x_i, e ε é um pequeno número positivo para evitar singularidades.

(b) *Regra 2*: As posições iniciais dos x_is são determinadas aleatoriamente no espaço de pesquisa:

$$x_{i,j}^{(0)} = x_{j,min} + rand(0,1) * (x_{j,max} - x_{j,min}), i = 1, 2, ..., k, \tag{28}$$

onde $x_{j,j}^{(0)}$ determina o valor inicial da j-ésima variável para o i-ésimo x_i; $x_{j,min}$ e $x_{j,max}$ são os valores mínimo e máximo admissíveis para a j-ésima variável; $rand$ é um número aleatório no intervalo [0,1]; e D é o número de variáveis. As velocidades iniciais das partículas carregadas são zero:

$$v_{i,j}^{(0)} = 0, i = 1, 2, ..., k. \tag{29}$$

(c) *Regra 3*: Três condições poderiam ser consideradas relacionadas ao tipo de forças atrativas:

- Qualquer x_i pode afetar outro; Isto é, um x_i ruim pode afetar um bom e vice-versa ($p_{ij} = 1$).
- Um x_i pode atrair outro se sua quantidade de carga elétrica (aptidão com relação de revisão na minimização de problemas) é melhor do que outros. Em outras palavras, um

x_i bom atrai um x_i ruim:

$$p_{ij} = \begin{cases} 1, se f(x_j) > f(x_i), \\ 0, \text{caso contrário.} \end{cases} \tag{30}$$

- Todos os x_is bons podem atrair x_is ruins e somente alguns das PC ruins atraem boas PC, considerando a seguinte função de probabilidade na Equação 31:

$$p_{ij} = \begin{cases} 1, se \frac{f(x_i)-f(x_{worst})}{f(x_{best})-f(x_{worst})} > rand(0,1) \vee f(x_j) > f(x_i), \\ 0, \text{caso contrário.} \end{cases} \tag{31}$$

De acordo com as condições acima, quando um boa PC atrai um mau, a capacidade de exploração para o algoritmo é fornecida, e vice-versa se um x_i ruim atrai um x_i bom, a exploração é fornecida. Quando um x_i se move em direção a um boa PC melhora seu desempenho, e assim o princípio de auto-adaptação é garantido. Movendo um bom x_i para um mau pode causar a perda da boa solução anterior ou, pelo menos, aumentar o custo computacional para encontrar uma boa solução. Para resolver esse problema, uma memória que salva a solução melhor até agora pode ser considerada. Portanto, parece que o terceiro tipo de condições acima é a melhor regra por causa de fornecer forte capacidade de exploração e uma exploração eficiente.

(d) *Regra 4*: O valor da força elétrica resultante que atua sobre um x_i é determinado usando a Equação 32 como:

$$F_j = q_j * \sum_{j,i \neq j} (\frac{q_i}{a^3} * r_{ij} * i_1) + (\frac{q_i}{r_{ij}^2} * i_2) * p_{ij} * (x_i - x_j), \begin{cases} j = 1, 2, ..., k \\ i_1 = 1, i_2 = 0 \leftrightarrow r_{ij} < a, \\ i_1 = 0, i_2 = 1 \leftrightarrow r_{ij} \geq a, \end{cases} \tag{32}$$

onde F_j é a força resultante que atua sobre o j-ésimo x_i. Neste algoritmo, cada x_i é considerado como uma esfera carregada com raio a, que tem uma densidade de carga de volume uniforme, com a magnitude de a é ajustada à unidade. No entanto, para exemplos mais complexos, o valor apropriado para a deve ser definido considerando o tamanho do espaço de pesquisa. Pode-se utilizar a a Equação 33 como uma fórmula geral:

$$a = 0.10 * max(\{|x_{j,max} - x_{j,min}|, i = 1, 2, ..., k\}). \tag{33}$$

De acordo com esta regra, na primeira iteração onde as PC estão distantes uma da outra a magnitude da força resultante que atua sobre um x_i é inversamente proporcional ao quadrado da separação entre as partículas. Assim, o poder de exploração global nessa condição é alto por causa de mais pesquisas nas primeiras iterações, enquanto que, à medida que as iterações vão passando a capacidade de exploração local aumenta e exploração global diminui.

(e) *Regra 5*: A nova posição e velocidade de cada x_i é determinada considerando as Equações 34 e 35, como segue:

$$\tilde{x}_j = rand_{j1} * k_a * \frac{F_j}{m_j} * \Delta t^2 + rand_{j2} * k_v * V_j * \Delta t + x_j, \tag{34}$$

$$\tilde{V}_j = \frac{\tilde{x}_j - x_j}{\Delta t}, \tag{35}$$

onde k_a é o coeficiente de aceleração; k_v é o coeficiente de velocidade para controlar a influência da velocidade anterior; e $rand_{j1}$ e $rand_{j2}$ são dois números aleatórios uniformemente distribuídos na faixa de (0,1). Aqui, m_j é a massa do i-ésimo x_i que é igual a q_j. Δt é o passo do tempo e está definido para unidade. O efeito da velocidade e da força resultante que atua sobre um x_i pode ser diminuído ou aumentado com base nos valores de k_v e k_a, respectivamente. Busca excessiva nas iterações iniciais pode melhorar a capacidade de exploração. No entanto, deve ser diminuída gradualmente, como descrito anteriormente. Uma vez que k_a é o parâmetro relacionado com as forças de atração, a seleção de um grande valor para este parâmetro pode causar uma convergência rápida e vice-versa um pequeno valor pode aumentar o tempo computacional. De fato k_a é um parâmetro de controle da exploração. Portanto, a escolha de uma função incremental pode melhorar o desempenho do algoritmo. Também, a direção da velocidade permeável de um x_i não é necessariamente a mesma que a força resultante. Assim, pode-se concluir que o coeficiente de velocidade k_v controla o processo de exploração e, portanto, uma função decrescente pode ser selecionada. Assim, k_v e k_a são definidos na Equação 36 como:

$$k_v = 0.5 * (1 - iter/MaxGer), k_a = 0.5 * (1 + iter/MaxGer), \tag{36}$$

onde $iter$ é o número da iteração e $MaxGer$ é o número máximo de iterações. Com a Equação 36, k_v diminui linearmente para zero enquanto k_a aumenta para um quando o número de iterações sobe. Desta forma, hiperesferas salva-se o equilíbrio entre a exploração e a rápida taxa de convergência. Considerando os valores destes parâmetros, Equação 34 e Equação 35 podem ser reescritos como:

$$\tilde{x}_j = k_a * rand_{j1} * \sum_{j,i \neq j} \left(\frac{q_i}{a^3} * r_{ij} * i_1 \right) + \left(\frac{q_i}{r_{ij}^2} * i_2 * p_{ij} * (x_i - x_j) \right) + (k_a * rand_{j2} * V_j) + x_j, \tag{37}$$

$$\tilde{V}_j = \tilde{x}_j - x_j. \tag{38}$$

(f) *Regra 6*: Considera-se uma memória que salva os melhores vetores x_i e seus valores de

função objetivo relacionados para melhorar o desempenho do algoritmo sem aumentar o custo computacional. Para cumprir esse objetivo, a memória carregada (CM) é utilizada para salvar um número das melhores soluções até o momento. No artigo de Kaveh eTalatahari (2010), o tamanho do CM (isto é CMS) é tomado como $N/4$. Outro benefício do CM consiste em utilizar esta memória para orientar os x_is atuais. Por outras palavras, os vetores armazenados no CM podem atrair x_is correntes. Em vez disso, presume-se que o mesmo número das partículas piores atuais não podem atrair outras.

(g) *Regra 7*: Qualquer componente do vetor de solução que viole os limites variáveis pode ser regenerado a partir do CM como na Equação 39:

$$x_{i,j} = \begin{cases} \text{prob. } CMCR & \Longrightarrow \text{seleciona um novo valor para a componente vindo de CM,} \\ & \Longrightarrow \text{prob. (1 - PAR) de fazer nada,} \\ & \Longrightarrow \text{prob. PAR de escolher um valor de um vizinho,} \\ \text{prob. } (1 - CMCR) & \Longrightarrow, \text{selecionar um valor aleatoriamente,} \end{cases} \tag{39}$$

onde prob. é a abreviatura de probabilidade; $x_{i,j}$ é a j-ésima componente do CP_j; O $CMCR$ é a taxa de consideração de memória carregada, variando entre 0 e 1, define a taxa de escolha de um valor no novo vetor a partir dos valores históricos armazenados no CM e $(1 - CMCR)$ define a taxa de escolha aleatória de um valor dentre os possíveis valores. O processo de ajuste do passo é realizado somente após um valor ser escolhido de CM. O valor $(1 - PAR)$ define a taxa de não fazer nada, e PAR define a taxa de escolha de um valor de vizinho o melhor x_i.

2.3.3 Busca de Vórtices

O algoritmo de Busca de Vórtices (*Vortex Search* - VS) proposto por Dogan Ölmez em 2014, é inspirado no padrão de turbilhões criados pelo fluxo vortical de fluídos agitados. Para proporcionar um bom equilíbrio no comportamento exploratório de uma busca, o método proposto modela seu comportamento de busca como um padrão de vórtice usando um esquema adaptativo de ajuste de tamanho de passo (DOAN; ÖLMEZ, 2015). O Algoritmo 7 apresenta o passos da técnica.

O algoritmo considera padrões circulares (hiperesferas em problemas de maior dimensão) dentro do espaço de busca. Em seguida, a simulação do padrão de vórtice para resolver o problema de otimização é obtida em duas etapas: geração de soluções candidatas e processo de decremento de raio. A geração de soluções candidatas é realizada por meio de uma distribuição Gaussiana, gerada aleatoriamente em torno do centro do círculo atual

Algoritmo 7 Pseudocódigo da Busca de Vórtices

Entrada: Parâmetros comuns (Algoritmo 1).
Define o centro inicial μ_0, calculado usando Equação 44.
Define o raio inicial r_0 (ou o devio padrão, σ_0), computado usando a Equação 45.

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Gera soluções candidatas $C_t(s)$ usando a distribuição gaussiana ao redor do centro μ_t com desvio padrão (raio) r_t.
    se $C_t(s)$ *fora dos limites* **então**
      Mude os valores $C_t(s)$ dentro dos limites como na Equação 46.
    fim
    Seleciona a melhor solução (s) de $C_t(s)$ para substituir o atual centro.
    μ_t
    se *(s < best)* **então**
      $best = s$
    senão
      Matenha a melhor solução *best* até o momento.
    fim
    Atualiza o centro para a posição da melhor solução até o momento.
    Reduz o desvio padrão (r_t) para a próxima iteração.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

μ_0, dado pela Equação 40, para um espaço de pesquisa de dimensão d:

$$p(x|\mu_0, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} exp(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)). \tag{40}$$

Para que essa distribuição seja esférica, a matriz de covariância Σ deve ser definida conforme Equação 41:

$$\Sigma = \sigma^2[I]_{d\times d}, \tag{41}$$

onde $[I]_{d\times d}$ representa a matriz de identidade de tamanho $d \times d$. Depois de todas as soluções candidatas terem sido geradas, o melhor candidato será o centro para o próximo círculo, a partir do qual soluções candidatas alternativas serão geradas novamente na próxima etapa do algoritmo. Este processo é realizado em conjunto com um decremento do raio do círculo no qual as soluções candidatas podem ser definidas. O procedimento para o decremento do raio do círculo é feito com base no inverso da função gama incompleta (DOAN; ÖLMEZ, 2015). A função gama incompleta é dada pela Equação 42:

$$\gamma(y, a) = \int_0^y e^{-t} * t^{a-1} dt, \tag{42}$$

onde $a > 0$ é o parâmetro de forma, e $y \leq 0$ é uma variável aleatória. O inverso dessa função é usado para calcular o novo raio na iteração t do algoritmo VS:

$$r_t = \sigma * (1/y) * \gamma(y, a_t)^{-1}, \tag{43}$$

onde $\gamma(y, at)^{-1}$ representa a função gama incompleta inversa, e $a_t = 1 - (t/MaxGer)$, onde $MaxGer$ representa o número máximo de iterações na busca e t a iteração (DOAN;

ÖLMEZ, 2015). Supondo um problema de otimização bidimensional, um padrão de vórtice pode ser modelado por um número de círculos aninhados. Aqui, o círculo exterior (maior) do vórtice é primeiro centrado no espaço de pesquisa, onde o centro inicial μ_0 pode ser calculado usando Equação 44:

$$\mu_0 = \frac{max + min}{2}, \tag{44}$$

onde $max.limit$ e $min.limit$ são $d \times 1$ vetores que definem as restrições vinculadas do problema em D espaço dimensional. Já o desvio padrão inicial (σ_0) é calculado a partir da Equação 45:

$$\sigma_0 = \frac{max(max.limit) - min(min.limit)}{2}. \tag{45}$$

O σ_0 também pode ser considerado como o raio inicial (r_0) do círculo externo para um problema de otimização bidimensional. Como uma localidade fraca é necessária nas fases iniciais, r_0 é escolhido para ser um valor grande. Assim, uma cobertura completa do espaço de pesquisa pelo círculo exterior é fornecida no passo inicial. Na fase de seleção, a melhor posição (x_{best}) é selecionada e memorizada de $C_0(s)$ para substituir o centro do círculo atual μ_0. Antes da fase de seleção, as soluções candidatas devem ser garantidas para estar dentro dos limites da procura. Para este propósito, as soluções que excedem os limites são deslocadas para os limites, como na Equação 46:

$$x_i^j = \begin{cases} rand * (max.limit^j - min.limit^j) + min.limit^j, & x_i^j < min.limit^j, \\ x_i^j, & min.limit^j \leq x_i^j \leq max.limit^j, \\ rand * (max.limit^j - min.limit^j) + min.limit^j, & x_i^j > max.limit^j, \end{cases} \tag{46}$$

onde $i = 1, 2, ..., k$ e $j = 1, 2, ..., D$ e $rand$ é um número aleatório uniformemente distribuído. Em seguida, a melhor posição memorizada x_{best} é designada para ser o centro do segundo círculo (o interno). Na fase de geração da segunda etapa, o raio efetivo (r_1) deste novo círculo é reduzido, e então, um novo conjunto de soluções $C_1(s)$ é gerado em torno do novo centro. Note que na segunda etapa, a localidade dos vizinhos gerados aumentou com o raio diminuído. Na fase de seleção do segundo passo, o novo conjunto de soluções $C_1(s)$ é avaliado para selecionar uma posição $x_{best} \in C_1(s)$. Se a solução selecionada é melhor do que a melhor solução encontrada até agora, então esta solução é atribuída a ser a nova melhor solução e é memorizada. Em seguida, o centro do terceiro círculo é designado para ser a melhor solução memorizada encontrada até agora. Este processo realiza iterações até que a condição de término seja atendida (DOAN; ÖLMEZ, 2015).

2.4 Otimização Bioinspirada

A metaheurística é bioinspirada quando suas regras de busca tentam simular alguns aspectos do comportamento de seres vivos. Nesta seção, como exemplificação da aplicação deste tipo de inspiração, serão mostradas algumas técnicas já consagradas e outras mais atuais.

2.4.1 Enxame de Partículas

James Kennedy e Russel Eberhart, em 1995, inspirados no comportamento social dos pássaros estudados por Heppner, desenvolveram uma técnica de otimização que veio a ser conhecida como enxame de partículas. A otimização do enxame de partículas, que tem raízes na vida artificial e na psicologia social, bem como na engenharia e ciência da computação, difere dos métodos de computação evolutiva pelo fato de os membros da população, chamados partículas, serem transportados pelo hiperespaço do problema.

A Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) é uma técnica baseada em padrões da natureza, mais populares e surgiu como sendo um algoritmo promissor para a resolução dos mais diversificados problemas de otimização. Desde a sua criação, já foram desenvolvidas muitas variantes para a resolução de problemas práticos relativos à otimização. O pseudocódigo do PSO canônico pode ser visto no Algoritmo 8.

Algoritmo 8 Pseudocódigo do Enxame de Partículas Canônico

```
Entrada: Parâmetros comuns (Algoritmo 1).
         Define os pesos ω, φ1 e φ2.

início
    enquanto (critério de parada não é satisfeito) faça
        para i = 1 até k faça
            se f(x_i) > f(p_i) então
                para j = 1 até d faça
                    p_{i,j} = x_{i,j}
                fim
            fim
            Escolhe arbitrariamente g = i.
            para n = índices dos vizinhos de g faça
                se f(p_n) > f(p_g) então
                    g = n
                fim
            fim
            para j =1 até d faça
                Atualiza as velocidades e posições das partículas.
            fim
        fim
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

Quando a população é inicializada, além das variáveis que recebem valores aleatórios, são atribuídas velocidades estocásticas. A cada iteração, a velocidade de cada partícula é acelerada estocasticamente em direção à sua melhor (onde tinha o seu valor mais elevado de aptidão) e para uma melhor posição da vizinhança (a posição de maior aptidão por qualquer partícula na sua vizinhança)(KENNEDY; EBERHART, 1995). A atualização da velocidade pode ser visto na Equação 47:

$$v_{i,j} = \omega * v_{i(t-1),j} + \varphi_1(p_{i,j} - x_{i(t-1),j}) + \varphi_2(x_{best} - x_{i(t-1),j}), v_{i,j} \in (-V_{max}, +V_{max}) \quad (47)$$

onde φ_1 é o coeficientes de aceleração individual, φ_2 coeficientes de aceleração coletivo e ω é o peso de inércia; $p_{i,j}$ é a melhor posição da vizinhança; x_{best} é a melhor posição do enxame. A mudança de posição da partícula é definida pela Equação 48:

$$x_{i,j} = x_{i(t-1),j} + v_{i,j}. \quad (48)$$

2.4.2 Algoritmo de Morcegos

Algoritmo baseado no comportamento de eco localização dos morcegos, proposto em 2010 por Xin-She Yang. Usa as seguintes regras: (a) Todos os morcegos usam a eco localização para detectar a distância, e eles também sabem a diferença entre as barreiras e o alimento de alguma maneira mágica; (b) Os morcegos voam aleatoriamente com velocidade v_i na posição x_i com um f_{min} frequência fixa, variando λ comprimento de onda e sonoridade A_0 para procurar presas. Eles podem ajustar automaticamente o comprimento de onda (ou frequência) dos seus impulsos emitidos e ajustar a taxa de emissão de impulsos $r \in [0, 1]$, dependendo da proximidade do seu alvo; (c) Embora o volume pode variar de muitas maneiras, assume-se que a intensidade varia de um grande A_0 (positivo) a um valor constante mínima A_{min} (YANG, 2010b). O pseudocódigo da técnica Algoritmo de Morcegos (*Bat Algorithm* - BA) pode ser visto no Algoritmo 9.

Para a estratégia de otimização do BA, são realizados os seguintes cálculos (YANG, 2010b):

(a) *Movimento de Morcegos*: Define as regras para a atualização das frequências f_i, posições x_i e velocidades $v_i$em um espaço de pesquisa D dimensional. As novas soluções x_i e velocidades v_i no passo de tempo t são dadas por:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \quad (49)$$

$$\tilde{v_i} = v_i + (x_i - x_{best})f_i, \quad (50)$$

Algoritmo 9 Pseudocódigo do Algoritmo de Morcegos

Entrada: Parâmetros comuns (Algoritmo 1).
Define valores de α e γ.

início
Inicializa: - Frequência de pulso f_i.
- Velocidade v_i.
- Taxa de pulso r_i.
- Sonoridade A_i.
enquanto *(critério de parada não é satisfeito)* **faça**
Gera novas soluções para ajustar frequência, atualizar velocidades e soluções (Equação 49 a 52).
se $(rand(0,1) > r_i)$ **então**
Seleciona uma solução entre as melhores soluções.
Gera uma nova solução ao redor da melhor solução selecionada.
fim
Gera uma nova solução de voo aleatório.
se $(rand(0,1) < A_i) e f(x_i) < f(\tilde{x}_i)$ **então**
Aceita a nova solução.
Incrementa r_i e reduz A_i.
fim
Ordena os morcegos e procura o atual melhor
$x*$.
fim
Retorna a melhor posição e sua respectiva solução.
fim

$$\tilde{x}_i = x_i + v_i, \tag{51}$$

onde $\beta \in [0, 1]$ é um vetor aleatório desenhado a partir de uma distribuição uniforme. Aqui x_{best} é a melhor posição global atual, que está localizado depois de comparar todas as soluções entre todos os k morcegos. Para a parte de pesquisa local, uma vez que uma solução é selecionada entre as melhores soluções atuais, uma nova solução para cada morcego é gerada localmente usando caminhada aleatória.

$$\tilde{x}_i = x_i + \varepsilon A, \tag{52}$$

onde $\varepsilon \in [-1, 1]$ é um número aleatório, enquanto $A(t) =< A_i >$ é a sonoridade média de todos os morcegos nesta etapa de tempo.

(b) *Sonoridade e Emissão de Pulso*: A sonoridade A_i e a taxa de emissão de impulsos r_i devem ser atualizadas conforme prosseguem as iterações. Como a intensidade geralmente diminui uma vez que um morcego encontrou sua presa, enquanto a taxa de emissão de pulso aumenta, a intensidade pode ser escolhida como qualquer valor de conveniência. Tem-se:

$$\tilde{A}_i = \alpha A_i, \tag{53}$$

$$\tilde{r}_i = r_i(0)[1 - exp(-\gamma t)], \tag{54}$$

onde α e γ são constantes. Para qualquer $0 < \alpha < 1$ e $\gamma > 0$, temos:

$$\alpha A_i \longrightarrow 0, r_i \longrightarrow r_i(0), \tag{55}$$

com $t \longrightarrow \infty$. Simplificando, pode-se usar $\alpha = \gamma$. A escolha dos parâmetros requer alguma experimentação. Inicialmente, cada morcego deve ter diferentes valores de sonoridade e taxa de emissão de pulso, e isso pode ser conseguido por randomização. A sonoridade e as taxas de emissão serão atualizadas apenas se as novas soluções são melhoradas, o que significa que esses morcegos estão se movendo em direção à solução ideal (YANG, 2010b).

2.4.3 Busca Cuco

A Busca Cuco (*Cuckoo Search* - CS) é um dos famosos algoritmos inspirados na natureza, desenvolvido em 2009 por Xin-She,Yangand e SuashDeb. Baseado no parasitismo da ninhada de algumas espécies de cuco, CS é realçado pelo método de passeio aleatório de Levy (YANG, 2010a).

Para simplificar a descrição da Busca Cuco, pode-se resumir o processo de busca em três regras (YANG, 2010a):

- Cada cuco coloca um ovo de cada vez, e despeja seu ovo de forma aleatória ninho escolhido;
- Os melhores ninhos com ovos de alta qualidade serão transferidos para as próximas gerações;
- O número de ninhos hospedeiros disponíveis é fixado e o ovo colocado por um cuco é descoberto pelo pássaro hospedeiro com uma probabilidade $p_a \in [0, 1]$. Nesse caso, o pássaro hospedeiro pode se livrar do ovo, ou simplesmente abandonar o ninho e construir um ninho completamente novo.

Como uma aproximação adicional, esta última suposição pode ser fração p_a dos n ninhos hospedeiros são substituídos por novos ninhos (com novas soluções) (YANG, 2010a).
Com base nessas três regras, os passos básicos do CS podem ser resumidos como o pseudo-código mostrado no Algoritmo 10.

As novas gerações são calculadas usando o voo de Levy conforme a Equação 56:

$$\tilde{x}_i = x_i + \alpha \oplus Levy(\lambda), \tag{56}$$

onde $\alpha > 0$ é o tamanho do passo que deve estar relacionado com as escalas do problema de interesses. Na maioria dos casos, pode-se usar $\alpha = O(L/10)$ onde L é a característica de escala do problema de interesse. A Equação 56 é essencialmente uma equação estocástica para uma caminhada aleatória. Em geral, uma caminhada aleatória é uma cadeia de Markov onde o próximo local depende somente da localização atual (o primeiro termo

Algoritmo 10 Pseudocódigo da Busca Cuco

Entrada: Parâmetros comuns (Algoritmo 1).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Obtém aleatoriamente um cuco.
    Gerar uma solução pelo voo de Levy.
    Avalia a aptidão (F_i).
    Escolha aleatoriamente um ninho.
    se $F_i > F_j$ **então**
      Substitua a solução de j pela a nova solução i.
    fim
    Uma fração (p_a) de piores ninhos é abandonada.
    Novas soluções são geradas para o lugar dos ninhos abandonados.
    Mantém-se as melhores soluções (ou ninhos com soluções de qualidade).
    Classifique as soluções e seleciona o atual melhor.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

no acima da equação) e a probabilidade de transição (o segundo termo). O produto $\oplus$ significa multiplicações de entrada. Esta multiplicação de entrada é similar aos usados em PSO, mas aqui a caminhada aleatória através do voo de Levy é mais eficiente em explorar o espaço de busca, como seu comprimento de passo é muito mais extenso no longo prazo (YANG, 2010a). O voo de Levy fornece essencialmente uma caminhada aleatória cujo comprimento do passo aleatório é extraído de uma distribuição de Levy:

$$Levy \sim \mu = t_{-\lambda}, (1 < \lambda \leq 3) \tag{57}$$

que tem uma variância infinita com uma média infinita. Aqui os passos essencialmente do processo de caminhada aleatória com uma distribuição de comprimento de cauda pesada. No entanto, uma fracção substancial das novas soluções deve ser gerada de longe e cujas localizações devem estar suficientemente longe da melhor solução atual, isso fará com que o sistema não fique preso em um ótimo local (YANG, 2010a).

2.4.4 Colônia de Abelhas Artificiais

O algoritmo da Colônia de Abelhas Artificiais (*Artificial Bee Colony* - ABC), proposto por Karaboga em 2005 para a otimização de parâmetros reais, é um algoritmo de otimização que simula o comportamento de forrageamento de uma colônia de abelhas. O modelo mínimo de seleção inteligente de forragem uma colônia de abelhas que o algoritmo ABC simula, consiste em três tipos de abelhas: abelhas empregadas, abelhas observadoras e abelhas escoteiras (AKAY; KARABOGA, 2012).

Metade da colônia consiste de abelhas empregadas e a outra metade inclui abelhas observadoras. As abelhas empregadas são responsáveis pela exploração das fontes de

néctar exploradas antes e dando informações às abelhas em espera (abelhas observadoras) na colmeia sobre a qualidade dos locais de fonte de alimento que estão explorando. As abelhas observadoras esperam na colmeia e decidem sobre uma fonte de alimento para explorar com base na informação compartilhada pelas abelhas empregadas. Os escoteiros pesquisam aleatoriamente o ambiente a fim de encontrar uma nova fonte de alimento, dependendo de uma motivação interna ou baseada em possíveis pistas externas (AKAY; KARABOGA, 2012).

No algoritmo ABC, a posição de uma fonte de alimento representa uma possível solução para o problema de otimização e a quantidade de néctar de uma fonte de alimento corresponde à rentabilidade (aptidão) da solução associada. Cada fonte de alimento é explorada por apenas uma abelha empregada. Em outras palavras, o número de abelhas empregadas é igual ao número de fontes de alimento existentes ao redor da colmeia (número de soluções na população). A abelha empregada cuja fonte de alimento foi abandonada torna-se uma escoteira (AKAY; KARABOGA, 2012). O processo de busca de Colônia de Abelhas Artificiais pode ser visto no Algoritmo 11.

Algoritmo 11 Pseudocódigo da Colônia de Abelhas Artificiais

Entrada: Parâmetros comuns (Algoritmo 1).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para *cada uma das abelhas empregadas* **faça**
      Produzir novas soluções (posições de fonte de alimento) usando a Equação 59.
      Calcula a aptidão $f(x_i)$.
      Aplique o processo de seleção guloso entre x_i e v_i.
    fim
    Calcula a probabilidade p_i envolvida na seleção probabilística a partir da Equação 61.
    para *cada abelha do espectador* **faça**
      Selecione uma solução x_i dependendo de p_i.
      Produzir nova solução v_i.
      Calcule o valor $f(x_i)$.
      Aplique o processo de seleção guloso.
    fim
    se *houver uma solução abandonada para escoteiro* **então**
      Substituí-lo por uma nova solução que será produzida aleatoriamente pela Equação 58.
    fim
    Memoriza a melhor solução até o momento.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

Usando a inteligência emergente no forrageamento de abelhas e o algoritmo ABC, as unidades do algoritmo ABC básico podem ser explicadas da seguinte forma (AKAY; KARABOGA, 2012):

(a) Produção de locais de fonte de alimento inicial: Se o espaço de pesquisa é considerado como sendo o ambiente da colméia que contém os locais de fonte de alimento, o algoritmo

começa com a produção aleatória de fontes alimentares que correspondem às soluções no espaço de busca. As fontes iniciais de alimentos são produzidas aleatoriamente dentro do intervalo dos limites dos parâmetros.

$$x_i^j = x_i^{min} + rand(0,1) * (x_i^{max} + x_i^{min}) \tag{58}$$

onde $i = 1 \ldots SN$, $j = 1 \ldots D$, SN é o número de fontes de alimento e D é o número de parâmetros de otimização.

(b) Envio de abelhas empregadas para os locais de origem alimentar: Como mencionado anteriormente, cada abelha empregada está associada a apenas um local de fonte de alimento. Assim, o número de locais de fonte de alimento é igual ao número de abelhas empregadas. Uma abelha empregada produz uma modificação na posição da fonte de alimento (solução) na sua memória, dependendo da informação local (informação visual) e encontra uma fonte de alimento vizinha e, em seguida, avalia a sua qualidade. No ABC, encontrar uma fonte de alimento vizinha é definida pela Equação 59:

$$\upsilon_i^j = x_i^j + \phi_i^j * (x_i^j - x_k^j) \tag{59}$$

Dentro da vizinhança de cada local de fonte de alimento representado por x_i, uma fonte de alimento υ_i é determinada mudando um parâmetro de x_i. Na Equação 59, j é um inteiro aleatório na faixa $[1, D]$ e $k \in 1, 2, ..., SN$ é um índice aleatório escolhido que tem de ser diferente de i, onde SN é o número de fontes de alimento e D é o número de dimensões. Já ϕ_i^j é um número aleatório real uniformemente distribuído no intervalo $[-1, 1]$.

Como pode ser visto a partir da Equação 59, à medida que a diferença entre os parâmetros dos x_i^j e x_k^j diminui, a perturbação na posição x_i^j diminui. Assim, à medida que a procura se aproxima da solução ótima no espaço de pesquisa, o comprimento do passo é reduzido adaptativamente. Se um valor de parâmetro produzido por esta operação exceder seus limites predeterminados, o parâmetro pode ser ajustado para um valor aceitável. Neste trabalho, o valor do parâmetro que excede seu limite é definido para seus limites. Se $x_i > x_i^{max}$ então $x_i = x_i^{max}$; Se $x_i > x_i^{min}$ então $x_i = x_i^{min}$. Depois de produzir υ_i dentro dos limites, um valor de aptidão para um problema de minimização pode ser atribuído à solução υ_i pela Equação 60.

$$\text{Aptidão}_i = \begin{cases} 1/(1 + f_i), \text{ se } f(x_i) \geq 0, \\ 1 + abs(f_i), \text{ se } f(x_i) < 0, \end{cases} \tag{60}$$

onde $f(x_i)$ é o valor de custo da solução v_i. Para problemas de maximização, a função de custo pode ser usada diretamente como uma função de aptidão. Uma seleção gananciosa é aplicada entre x_i e v_i; Então o melhor é selecionado dependendo dos valores de aptidão representando a quantidade de néctar das fontes de alimento em x_i e v_i. Se a fonte v_i é superior à de x_i em termos de rentabilidade, a abelha empregada memoriza a nova posição e esquece a antiga. Caso contrário, a posição anterior é mantida na memória. Se x_i não puder ser melhorado, seu contador que contém o número de testes é incrementado em 1, caso contrário, o contador é redefinido para 0.

(c) Cálculo de valores de probabilidade envolvidos na seleção probabilística: Depois que todas as abelhas empregadas terminam suas buscas, compartilham sua informação relacionada às quantidades do néctar e às posições de suas fontes com as abelhas do espectador na área da dança. Esta é a característica de interação múltipla das abelhas artificiais do ABC. Uma abelha observadora avalia as informações de néctar tomadas de todas as abelhas empregadas e escolhe um local de fonte de alimento com uma probabilidade relacionada à sua quantidade de néctar. Essa seleção probabilística depende dos valores de adequação das soluções na população. Um esquema de seleção baseado em aptidão pode ser uma roda de roleta, baseado em ranqueamento, amostragem universal estocástica, seleção de torneio ou outro esquema de seleção.

No ABC básico, o esquema de seleção de roleta em que cada fatia é proporcional em tamanho ao valor de aptidão é empregado na Equação 61:

$$p_i = \frac{f(x_i)}{\sum_{i=1}^{SN} f(x_i)}, \tag{61}$$

onde $f(x_i)$ é a aptidão da i-ésima abelha.

Neste esquema de seleção probabilística, à medida que a quantidade de néctar das fontes de alimento (a aptidão das soluções) aumenta, o número de espectadores que visitam os mesmos aumenta também. Este é o recurso de *feedback* positivo do ABC.

(d) Seleção de locais de fonte de alimento por espectadores com base nas informações fornecidas por abelhas empregadas: No algoritmo ABC, um número real aleatório dentro do intervalo $[0, 1]$ é gerado para cada fonte. Se o valor de probabilidade (p_i na Equação 61) associado a essa fonte for maior que este número aleatório, então a abelha observadora produz uma modificação na posição deste local de fonte de alimento usando a Equação 60 como no caso da abelha empregada. Depois que a fonte é avaliada, a seleção gulosa é aplicada e a abelha do espectador ou memoriza a posição nova esquecendo a velha

ou mantém a velha. Se a solução x_i não puder ser melhorada, seus testes de retenção de contadores são incrementados em 1, caso contrário, o contador é reposto a 0. Esse processo é repetido até que todos os espectadores sejam distribuídos em locais de fonte de alimento.

2.4.5 Rebanho de Camarões

Baseado na simulação do comportamento de rebanho de camarões, desenvolvido em 2012 por Gandomi e Alavi. No processo de busca, as distâncias mínimas de cada camarão para o alimento e para a maior densidade do rebanho são consideradas como a função objetivo para o movimento dos camarões. A posição dependente do tempo dos camarões e é formulado por três fatores principais: (a) movimento induzido pela presença de outros indivíduos (b) atividade alimentar, (c) difusão aleatória (GANDOMI; ALAVI, 2012). O Algoritmo 12 apresenta os passos da técnica de Rebanho de Camarões (*Krill Herd* - KH).

Algoritmo 12 Pseudocódigo do Rebanho de Camarões

Entrada: Parâmetros comuns (Algoritmo 1).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Cálculo do movimento.
    Movimento induzido pela presença de outros indivíduos.
    Movimento de *foraging*.
    Difusão física.
    Implementa os operadores genéticos.
    Atualiza as posições dos camarões nos espaço de busca.
    Avalia cada um dos camarões de busca a partir da função objetivo.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

A predação remove indivíduos, leva à redução da densidade média de camarões e distingue o enxame de camarões da localização dos alimentos. Este processo é assumido como a fase de inicialização no algoritmo KH. No sistema natural, a adequação de cada indivíduo é suposto ser uma combinação da distância do alimento e da maior densidade do enxame de camarões (GANDOMI; ALAVI, 2012).

Sabe-se que um algoritmo de otimização deve ser capaz de pesquisar espaços de dimensionalidade arbitrária. Portanto, a Equação 62 com o modelo Lagrangeano é generalizado para um espaço de decisão n dimensional:

$$\frac{dx_i}{dt} = N_i + F_i + D_i, \tag{62}$$

onde N_i é o movimento induzido por outros indivíduos do camarões; F_i é o movimento forrageamento, e D_i é a difusão física dos i-ésimo camarão (GANDOMI; ALAVI, 2012).

Desta forma, para calcular o modelo Lagrangeano, é necessário calcular (GANDOMI; ALAVI, 2012):

(a) *Movimento Induzido por Outros Indivíduos*: Para um indivíduo do camarões, este movimento pode ser definido como na Equação 63:

$$\tilde{N}_i = N_{max}\alpha_i + \omega_n N_i, \tag{63}$$

onde,

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target}, \tag{64}$$

e N_{max} é a velocidade máxima induzida, ω_n é o peso inercial do movimento induzido no intervalo [0, 1], N_i é o último movimento induzido, α_i^{local} é o efeito local fornecido pelos vizinhos e α_i^{target} é o alvo pelo efeito de direção fornecido pelo melhor indivíduo de camarões.

O efeito dos vizinhos pode ser assumido como uma tendência atrativa / repulsiva entre os indivíduos para uma pesquisa local. O efeito dos vizinhos num indivíduo de movimento de camarões é determinado na Equação 65 :

$$\alpha_i^{local} = \sum_{j=1}^{kk} \hat{K}_{ij}\hat{x}_{ij}, \tag{65}$$

com:

$$\hat{x}_{ij} = \frac{x_j - x_i}{\|x_j - x_i\| + \epsilon}, \tag{66}$$

$$\hat{k}_{ij} = \frac{K_j - K_i}{K_{worst} - K_{best}}, \tag{67}$$

onde K_{best} e K_{worst} são os melhores e os piores valores de aptidão dos indivíduos de camarões até o momento; K_i representa a aptidão ou o valor da função objetivo do i-ésimo camarão; K_j é a adequação de j-ésimo ($j = 1, 2, ..., kk$) vizinho; x representa as posições relacionadas; e kk é o número dos vizinhos. Para evitar as singularidades, um pequeno número positivo (ϵ) é adicionado ao denominador.

O vetor alvo conhecido de cada indivíduo de camarões é a menor adequação de um camarão. O efeito individual dos camarões com a melhor aptidão no i-ésimo camarão é levado em consideração usando Equação 68. Este nível leva-o ao ótimo global e é formulado na Equação 68 como:

$$\alpha_i^{target} = C_{best}\hat{K}_{i,best}\hat{x}_{i,best}, \tag{68}$$

onde, C_best é o coeficiente efetivo do indivíduo do camarões com a melhor adequação ao i-ésimo camarão. Este coeficiente é definido uma vez que um alvo i leva a solução para o ótimo global e deve ser mais eficaz do que outros indivíduos camarões, como vizinhos. Aqui, o valor de C_best é definido na Equação 69 como:

$$C_{best} = 2(rand(0,1) + \frac{I}{MaxGer}), \tag{69}$$

onde $rand$ é um valor aleatório uniformemente entre 0 e 1 e é para melhorar a exploração, I é o número de iteração real e $MaxGer$ é o número máximo de iterações.

(b) *Movimento de Forrageamento*: O movimento de forrageamento é formulado em termos de dois parâmetros eficazes principais. O primeiro é a comida e o segundo é a experiência anterior sobre a localização da comida. Este movimento pode ser expresso para o i-ésimo indivíduo do camarões conforme a Equação 70:

$$\tilde{F}_i = V_f \beta_i + \omega_f F_i, \tag{70}$$

onde:

$$\beta_i = \beta_i^{food} + \beta_i^{best}, \tag{71}$$

e V_f é a velocidade de forragem, ω_f é o peso de inércia do movimento de forragem na faixa [0, 1], é o último movimento de forrageamento, β_i^{food} é o alimento atraente e β_i^{best} é o efeito da melhor aptidão do i-ésimo camarão até o momento.

(c) *Difusão Física*: A difusão física dos camarões é considerada um processo aleatório. Este movimento pode ser expresso em termos de uma velocidade máxima de difusão e de um vetor direcional aleatório. Pode ser formulado conforme a Equação 72:

$$D_i = D^{max}\left(1 - \frac{I}{MaxGer}\right)\delta, \tag{72}$$

onde D^{max} é a velocidade de difusão máxima, e δ é o vetor direcional aleatório com valores aleatórios entre -1 e 1. Após a realização dos cálculos acima, pode-se calcular o processo de movimento. Em geral, os movimentos definidos mudam frequentemente a posição de um indivíduo de camarões em direção à melhor aptidão, de acordo com a Equação 73:

$$x_i(t + \Delta t) = x_i(t + \Delta t) + \Delta t \frac{dx_i}{dt}, \tag{73}$$

Deve-se notar que Δt é uma das constantes mais importantes e deve ser cuidadosamente definido de acordo com o problema de otimização. Isso ocorre porque este parâmetro

funciona como um fator de escala do vetor de velocidade. Δt depende completamente do espaço de busca e parece que ele pode ser obtido simplesmente a partir da Equação 74 :

$$\Delta t = C_t \sum_{j=1}^{D} (UB_j - LB_j), \tag{74}$$

onde D é o número total de variáveis, e LB_j e UB_j são limites inferiores e superiores das j-ésima variáveis (j = 1,2, ..., D), respectivamente. Portanto, o absoluto de sua subtração mostra o espaço de pesquisa. Constata-se empiricamente que C_t é um número constante entre [0, 2]. Também é óbvio que valores baixos de C_t perm que os indivíduos do camarões busquem cuidadosamente o espaço.

Para melhorar o desempenho do algoritmo, os mecanismos de reprodução genética são incorporados ao algoritmo. Os mecanismos de reprodução genética adaptativa introduzidos são *crossover* e mutação que são inspirados nos algoritmos evolutivos (GANDOMI; ALAVI, 2012):

(a) *Crossover*: O *crossover* é controlado por uma probabilidade de *crossover* (Cr) e pode ser realizada de maneira binomial. O esquema binomial executa *crossover* em cada um dos componentes j. Ao gerar um número aleatório uniformemente distribuído entre 0 e 1, a m-ésima componente de x_i ($x_{i,m}$) é manipulada conforme a Equação 75 :

$$x_{i,m} = \begin{cases} x_{r,m}, rand_{i,m} < Cr, \\ x_{r,m}, \text{caso contrário.} \end{cases} \tag{75}$$

com Cr sendo:

$$Cr = 0.2\hat{K}_{i,best}, \tag{76}$$

onde $r \in \{1, 2, ..., i-1, i+1, ..., k\}$. Usando esta nova probabilidade de *crossover*, a probabilidade de *crossover* para o melhor global é igual a zero e aumenta com a diminuição da aptidão.

(b) *Mutação*: A mutação é controlada por uma probabilidade de mutação (Mu). O esquema de mutação adaptativa utilizado é visto na Equação 77:

$$x_{i,m} = \begin{cases} x_{best,m} + \mu(x_{p,m} - x_{q,m}), rand_{i,m} < Mu, \\ x_{i,m}, \text{caso contrário.} \end{cases} \tag{77}$$

com Mu, calculado pela Equação 78:

$$Mu = 0.05/\hat{K}_{i,best}, \tag{78}$$

onde $p, q \in \{1, 2, ..., i-1, i+1, ..., k\}$ e μ é um número entre 0 e 1. Deve-se notar em $\hat{K}_{i,best}$ é encontrado por $K_i - K_{best}$. Usando esta nova probabilidade de mutação, a probabilidade de mutação para o melhor global é igual a zero e aumenta com a diminuição da aptidão.

2.4.6 Algoritmo de Algas Artificiais

Inspirado nos comportamentos de vida das microalgas, Uymaz et al. (2015) desenvolveram um algoritmo que baseia-se no processo de evolução, adaptação e o movimento de microalgas.

A técnica é resultado da inspiração tirada de estilos de vida de microalgas, que são espécies fotossintéticas. O Algoritmo de Algas Artificiais (*Artificial Algae Algorithm* - AAA) baseia-se na produção de algas, na adaptação e na natação, que emerge com o movimento de estar perto da luz como um organismo fotossintético. Em AAA, o ótimo global da função objetiva foi definido como o ponto em que as algas podem receber luz ótima para a fotossíntese (UYMAZ; G.; YEL, 2015).

As algas artificiais, semelhante às algas reais, podem se mover para a fonte de luz para fotossintetizar com natação helicoidal, e eles podem se adaptar ao ambiente, são capazes de mudar a espécie dominante e pode reproduzir por divisão mitótica. Assim, o algoritmo foi composto de 3 partes básicas chamadas: processo evolutivo, adaptação e movimento helicoidal. Essas partes serão melhor explicadas a seguir (UYMAZ; G.; YEL, 2015):

(a) *Processo Evolutivo*: Com colônia de algas recebendo luz suficiente e nutrientes, ela cresce e se reproduz para gerar duas novas células de alga no tempo t, semelhante à divisão mitótica real. Pelo contrário, a colônia de algas que não recebe luz suficiente sobrevive por algum tempo, mas acaba morrendo. A cinética de crescimento da colônia de algas foi calculada com o modelo de Monod dado na Equação 79 onde μ é a taxa de crescimento específico, μ_{max} é a taxa de crescimento específico máximo, S é a concentração de nutrientes, que é o valor de aptidão ($f(x_i)$) no tempo t no modelo e k é a substrato semi-saturação constante da colônia de algas. μ_{max} foi assumido como 1 (uma vez que a quantidade máxima convertida em biomassa deve ser igual à quantidade de substrato consumida em unidade de tempo de acordo com o princípio da conservação da massa). k foi calculada como a taxa de crescimento condições de colônia algas no tempo t. O tamanho da i-ésima colônia de algas no tempo $t + 1$ na equação de Monod é dado na Equação 79 :

$$\tilde{G}_i = \mu_i * G_i, i = 1, 2, ..., N, \tag{79}$$

onde G_i é o tamanho da i-ésima colônia de algas no tempo t, N é o número de colônias de algas no sistema. Colônias de algas fornecendo boas soluções, com custo-eficiente e a

solução mais adequada, crescem mais como a quantidade de nutriente que eles obtêm é alta. Para cada célula de alga da menor colônia de algas morrendo no processo evolutivo, a célula de alga da maior colônia de algas é replicada, conforme visto a partir das Equações 80 a 82.

$$biggest^t = max(G_i), i = 1, 2, ..., N, \tag{80}$$

$$smallest^t = min(G_i), i = 1, 2, ..., N, \tag{81}$$

$$smallest^t_m = biggest^t_m, m = 1, 2, ..., D, \tag{82}$$

onde D é a dimensão do problema, maior é a maior colônia de algas e menor é a menor. Em AAA, as colônias de algas são classificadas de acordo com os seus tamanhos no tempo t. Em qualquer dimensão selecionada aleatoriamente, a célula de alga da menor colônia de algas morre e a célula de alga da maior colônia se reproduz.

(b) *Adaptação*: A colônia de algas, que não pode crescer suficientemente em um ambiente, tentar adaptar-se ao meio ambiente e, como resultado, as espécies dominantes mudança. A adaptação é o processo no qual uma colônia de algas insuficientemente cultivada tenta se assemelhar à maior colônia de algas no ambiente. Este processo acaba com o nível de fome no algoritmo. O valor inicial de fome é zero para cada alga artificial. O valor de inanição aumenta com o tempo t, quando a célula de alga recebe luz insuficiente. A alga artificial com maior valor de inanição (Equação 83) adaptou-se (Equação 84).

$$starving = max(A_i), i = 1, 2, ..., N, \tag{83}$$

$$\tilde{starving} = starving + (biggest^t - starving) * A_p, \tag{84}$$

onde A_i é o valor de fome da colônia de algas i-ésima no tempo t, $starving$ é a colônia de algas com o maior valor de inanição no tempo t. O parâmetro de adaptação (A_p) determina se o processo de adaptação seria aplicado no tempo t ou não. A_p é constante no intervalo [0, 1].

(c) *Movimento Helicoidal*: Células de algas e colônias geralmente nadar e tentar ficar perto da superfície da água por causa da luz adequada para a sobrevivência está disponível lá. Eles nadam helicoidalmente no líquido com seus flagelos, que proporcionam movimento para frente que é restrito pela gravidade e pelo arrasto viscoso. Movimentos de células de algas diferem. À medida que a superfície de fricção da célula de alga em crescimento cresce, a frequência de movimentos helicoidais aumenta aumentando sua capacidade de

busca local. Cada célula de alga pode se mover proporcionalmente à sua energia. A energia de uma célula de alga no tempo t é diretamente proporcional à quantidade de absorção de nutrientes nesse tempo. Portanto, quanto mais uma célula de alga estiver mais próxima da superfície, mais energia ela tem, e encontra mais chance de se mover dentro do líquido. Pelo contrário, se a superfície de atrito é menor, sua distância de movimento no líquido é maior. Portanto, sua capacidade de busca global é maior. No entanto, eles podem se mover menos na proporção de sua energia.

O movimento de uma célula de alga é helicoidal como é na vida real. Em AAA, a gravidade que restringe o movimento é exibida como 0 e arrasto viscoso é exibido como força de corte, que é proporcional ao tamanho da célula de alga. Está na forma esférica e o tamanho dele é seu volume no modelo. Portanto, a superfície de atrito torna-se a área de superfície do hemisfério (Equações 85 e 86).

$$\tau(x_i) = 2 * \pi * r^2, \tag{85}$$

$$\tau(x_i) = 2 * \pi * \left(\sqrt[3]{\frac{3 * G_i}{4 * \pi}} \right)^2, \tag{86}$$

onde (x_i) é a superfície de atrito. Três dimensões para o movimento helicoidal da célula de alga são determinadas aleatoriamente. Um destes fornece o movimento linear na Equação 87 e outras duas dimensões proporcionam movimento angular nas Equações 88 e 89. A Equação 87 é usado para problemas unidimensionais e células de algas se movem em uma única direção. Em problemas bidimensionais, o movimento de algas é sinusoidal, conforme as Equações 87 e 89. No caso de três ou mais dimensões, o movimento das algas é helicoidal e as Equações 87 a 89. A superfície de fricção e distância para a fonte de luz determinam o tamanho do passo do movimento:

$$\tilde{x}_{i,m} = x_{i,m} + (x_{j,m} - x_{i,m}) * (\Delta - \tau(x_i)) * p, \tag{87}$$

$$\tilde{x}_{i,k} = x_{i,k} + (x_{j,k} - x_{i,k}) * (\Delta - \tau(x_i)) * cos(\alpha), \tag{88}$$

$$\tilde{x}_{i,l} = x_{i,l} + (x_{j,l} - x_{i,l}) * (\Delta - \tau(x_i)) * sen(\beta), \tag{89}$$

onde $x_{i,m}$, $x_{i,k}$e $x_{i,l}$ são coordenadas x, y e z da i-ésima célula de alga no instante t; α e $\beta \in [0, 2]$; $p \in [-1, 1]$; Δ é a força de cisalhamento; $\tau(x_i)$ é a de superfície de fricção da alga. No Algoritmo 13, pode-se verificar o processo de busca empregado pela técnica.

Algoritmo 13 Pseudocódigo do Algoritmo de Algas Artificiais

Entrada: Parâmetros comuns (Algoritmo 1).
Avalia o tamanho (G) das n colônias de algas.
Define Força de Cisalhamento (Δ), Perda de energia (e) e Parâmetro de adaptação (Ap).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Avalia a energia (E) e a superfície de atrito (θ) das n colônias de algas.
    para i = *1 até* k **faça**
      $fome = verdadeiro$.
      enquanto $(E(x_i) > 0)$ **faça**
        Escolha j entre todas as soluções via método de seleção de torneio.
        Escolha aleatoriamente três dimensões e realiza o movimento helicoidal.
        Avalia as novas soluções.
        $E(x_i) = E(x_i) - (e/2)$ energia perdida causada pelo movimento.
        se *a nova solução é melhor* **então**
          Atualiza a colônia de algas i.
          $fome = falso$.
        fim
        Senao $E(x_i) = E(x_i) - (e/2)$ energia perdida causada pelo movimento.
      fim
      se $fome = verdadeiro$ **então**
        Incrementa a fome $A(x_i)$.
      fim
    fim
    Avalia o tamanho (G) da população.
    Escolhe uma dimensão para reprodução, r.
    $menor^r = maior^r$.
    se $rand(0,1) < A_p$ **então**
      $starving = starving + (biggest^t - smallest^t) * A_p$
    fim
    Avalia cada alga de busca de acordo com sua nova posição e encontre a atual melhor solução.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

2.4.7 Rebanho de Elefantes

O método é inspirado pelo comportamento de rebanho do grupo de elefante. Na natureza, os elefantes pertencem a diferentes clãs que vivem juntos sob a liderança de uma matriarca, e os elefantes machos vão deixando seu grupo familiar quando crescem. Estes dois comportamentos podem ser modelados em dois operadores seguintes: operador de atualização do clã e de separação. Na técnica de Rebanho de Elefantes (*Elephant Herding Optimization* - EHO), os elefantes em cada clã são atualizados pela sua posição atual e matriarca através de operador de atualização do clã. Isso é seguido pela aplicação do operador de separação que podem aumentar a diversidade da população na fase de pesquisa posterior (WANG; DEB; COELHO, 2015).No Algoritmo 14 tem-se os passos da técnica.

A fim de tornar o comportamento de rebanho de elefantes para resolver problemas de otimização global, a regras idealizadas foram simplificadas da seguinte forma (WANG; DEB; COELHO, 2015):

- A população de elefantes é composta por alguns clãs, e cada clã tem número fixo de elefantes;

Algoritmo 14 Pseudocódigo do Rebanho de Elefantes

Entrada: Parâmetros comuns (Algoritmo 1).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Ordena todos os elefantes de acordo com sua aptidão.
    Implementa o operador de atualização do clan usando o Algoritmo 15.
    Implementa o operador de separação mostrado no Algoritmo 16.
    Avalia a população considerando as posições atualizadas.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

- Um número fixo de elefantes machos deixará seu grupo familiar e viverão solitariamente longe do grupo principal de elefantes em cada geração;
- Os elefantes em cada clã vivem juntos sob a liderança de uma matriarca.

Os dois operadores que regem o processo de busca baseado no comportamento de rebanho de elefantes são (WANG; DEB; COELHO, 2015):

(a) Operador de Atualização de Clan: Como mencionado antes, todos os elefantes vivem juntos sob a liderança de uma matriarca em cada clã. Portanto, para cada elefante no clã ci, sua próxima posição é influenciada pelo matriarca ci. Para o elefante i no clã ci, ele pode ser atualizado pela Equação 90:

$$\tilde{x}_{ci,i} = x_{ci,i} + \alpha * (x_{ci,best} - x_{ci,i}) * r, \tag{90}$$

onde $\tilde{x}_{ci,i}$ e $x_{ci,i}$ são recém-atualizados e posição antiga para elefante i no clã ci, respectivamente. A variável $\alpha \in [0, 1]$ é um fator de escala que determina a influência do matriarca ci em $x_{ci,i}$. $x_{ci,best}$ representa matriarca ci, que é o elefante mais apto individual no clã ci. A variável $r \in [0, 1]$, utilizando-se uma distribuição uniforme. O elefante mais apto em cada clã ($x_{ci,best}$) não pode ser atualizado pela Equação 90 . Para o mais apto, a atualização deve ser feita a partir da Equação 91:

$$\tilde{x}_{ci,i} = \beta * x_{ci,center}, \tag{91}$$

onde [0,1] é um fator que determina a influência do $x_{ci,center}$ em $\tilde{x}_{ci,i}$. Pode-se ver, o novo indivíduo $\tilde{x}_{ci,i}$ na Equação 91 é gerado pela informação obtida por todos os elefantes no clã ci. $x_{ci,center}$ é o centro do clã ci, e para o j-ésima dimensão pode ser calculada com a Equação 92:

$$x^d_{ci,center} = \frac{1}{n_{ci}} * \sum_{i=1}^{n_{ci}} * x^d_{ci,i}, \tag{92}$$

onde $1 \leq d \leq D$ indica a j-ésima dimensão, e D é sua dimensão total. n_{ci} é o número de elefantes no clã ci. $x^d_{ci,i}$ é o j-ésimo componente do elefante indivíduo $x_{ci,i}$. O centro do

clã *ci*, $x_{ci,center}$, pode ser calculado através dos cálculos D pela Equação 92 . O Algoritmo 15 apresenta os passos do operador de atualização de clan.

Algoritmo 15 Operador de Atualização do Clan

```
início
    para ci = 1 até nClan (para todos clans na população de elefantes) faça
        para i = 1 até n_ci (para todos elefantes no clan ci) faça
            Atualiza x_{ci,i} e gera x̃_{ci,i} com a Equação 90.
            se x_{ci,i} = x_{best_ci} então
                Atualiza x_{ci,i} e gera x̃_{ci,i} com a Equação 91.
            fim
        fim
    fim
fim
```

(b) Operador de separação: No grupo dos elefantes, os elefantes masculinos deixarão seu grupo da família e viverão sozinhos quando alcançam a puberdade. Este processo de separação pode ser modelado por operador de separação ao resolver problemas de otimização. A fim de melhorar ainda mais a capacidade de busca do método EHO, supondo que os indivíduos de elefante com a pior aptidão irão implementar o operador de separação em cada geração como mostrado na 93 .

$$x_{ci,worst} = x_{min} + (x_{max} - x_{min} + 1) * rand(0, 1), \tag{93}$$

onde x_{max} e x_{min} são respectivamente limite superior e inferior da posição de elefante individual. $x_{ci,worst}$, *ci* é o pior elefante individual no clã *ci*. $rand \in [0, 1]$ seguindo uma distribuição uniforme na faixa [0, 1]. O Algoritmo 16 apresenta os passos do operador de separação.

Algoritmo 16 Operador de Separação

```
início
    para ci = 1 até nClan (para todos clans na população de elefantes) faça
        Substitua o pior elefante no clan ci com a Equação 93.
    fim
fim
```

2.4.8 Algoritmo de Baleias

A técnica criada por Mirjalili e Lewis (2016) foca no método de caça especial das baleias jubarte. Esse comportamento de forrageamento é chamado método de alimentação rede de bolhas. As baleias jubarte preferem caçar camarões ou de peixes pequenos perto da superfície. Observou-se que este forrageamento é feito criando bolhas distintivas ao longo de um círculo ou trajetória em forma de ”9”. Antes de 2011, este comportamento só foi

investigado com base na observação da superfície. No entanto, Goldbogen et al. (2013) investigaram esse comportamento utilizando sensores de *tags*. Eles capturaram 300 eventos de alimentação de bolha-rede derivados de *tag* de 9 baleias jubarte. Eles encontraram duas manobras associadas à bolha e chamaram-nas de espirais ascendentes e "duplo-*loops*". Devido seu comportamento único de caça, a manobra espiral de alimentação de rede de bolhas é matematicamente modelada para realizar a otimização nesta técnica (MIRJALILI; LEWIS, 2016). Os passos do Algoritmo de Baleias (*Whale Optimization Algorithm* - WOA) pode ser visto no Algoritmo 17.

O algoritmo WOA considerada 3 elementos principais para seu processo de busca: circundar a presa, manobra de alimentação espiral de rede de bolhas e a procura de presas. Os modelos matemáticos que reproduzem estes elementos são mostrados abaixo (MIRJALILI; LEWIS, 2016):

(a) *Circundar Presa*: As baleias jubarte podem reconhecer a localização das presas e circundá-las. Uma vez que a posição do desenho ótimo no espaço de pesquisa não é conhecida a priori, o algoritmo WOA assume que a solução melhor candidata atual é a presa alvo ou está próxima do ótimo. Depois que a melhor baleia é definida, as outras baleias, portanto, tentam atualizar suas posições para a melhor baleia. Esse comportamento é representado pela Equação 94:

$$\tilde{x_i} = x_{best} - A * D, \tag{94}$$

com,

$$D = |C * x_{best} - x_i|, \tag{95}$$

$$A = 2 * a - a, \tag{96}$$

$$C = 2 * r, \tag{97}$$

onde t indica a iteração atual, A e C são vetores de coeficientes, x_{best} é o vetor de posição da melhor solução obtida até agora, x_i é o vetor de posição. Vale a pena mencionar aqui que x_{best} deve ser atualizado em cada iteração se houver uma solução melhor. A variável a é linearmente diminuída de 2 para 0 ao longo das iterações (tanto nas fases de exploração global como de exploração local) e r é um vetor aleatório em [0,1].

(b) *Manobra de Alimentação Espiral de Rede de Bolhas*: Com o objetivo de modelar matematicamente o comportamento da rede de bolhas das baleias-jubarte, duas abordagens são projetadas da seguinte forma:

(b1) Mecanismo de Encolhimento Envolvente: Esse comportamento é conseguido diminuindo o valor de a na Equação 96 . Observe que o intervalo de flutuação de A também é diminuído por a. Em outras palavras A é um valor aleatório no intervalo $[-a, a]$ onde a é diminuída de 2 para 0 ao longo das iterações. Definir valores aleatórios para A em [-1,1], a nova posição de uma baleia pode ser definida em qualquer lugar entre a posição original da baleia e a posição da melhor baleia atual.

(b2) Posição de Atualização Espiral: As baleias jubarte nadam em torno da rapina dentro de um círculo encolhendo e ao longo de um dado trajeto em forma espiral simultaneamente. Para modelar esse comportamento simultâneo, assume-se que existe uma probabilidade de 50% para escolher entre o mecanismo de encolhimento encurvado ou o modelo espiral para atualizar a posição das baleias, conforme Equação 98 :

$$\tilde{x}_i = \begin{cases} x_{best} - A * D, \text{se } p < 0.5, \\ \tilde{D} * e^{bl} * cos(2\pi * l) + x_{best}, \text{se } p \geq 0.5, \end{cases} \tag{98}$$

com,

$$\tilde{D} = |x_{best} - x_i|, \tag{99}$$

que indica a distância da baleia à presa (x_{best}), b é uma constante para definir a forma da espiral logarítmica, l é um número aleatório em [-1,1]. A variável p é um número aleatório no intervalo [0,1].

(c) *Procura de Presas*: A mesma abordagem com base na variação do intervalo A um vetor pode ser utilizado para procurar presas. Na verdade, as baleias jubarte pesquisam aleatoriamente de acordo com a posição de cada um. Portanto, usa-se A com valores aleatórios maiores que 1 ou menores que -1 para forçar a baleia a se afastar de uma baleia de referência. Em contraste com a fase de exploração local, atualiza-se a posição de uma baleia de acordo com uma baleia escolhida aleatoriamente, em vez da melhor baleia encontrada até agora. Este mecanismo e $|A| > 1$ enfatizam a exploração e permite que o algoritmo WOA realize uma busca global. O modelo matemático é visto na Equação 100:

$$\tilde{x}_i = x_{rand} - A * D, \tag{100}$$

com,

$$D = |C * x_{rand} - x_i|, \tag{101}$$

onde x_{rand} é uma baleia aleatória escolhida da população atual.

Algoritmo 17 Pseudocódigo do Algoritmo de Baleias

Entrada: Parâmetros comuns (Algoritmo 1).

```
início
    enquanto (critério de parada não é satisfeito) faça
        para i = 1 até k faça
            Atualiza a, A, C, l e p.
            se (p < 0.5) então
                se (|A| < 1) então
                    Atualiza a posição atual da baleia de busca pela Equação 94.
                fim
                senão
                    Seleciona uma baleia.
                    Atualiza a posição atual da baleia pela Equação 100.
                fim
            fim
            senão
                Atualiza a posição atual da baleia pela Equação 98.
            fim
        fim
        Trata qualquer violação de limitações.
        Calcula a aptidão de cada baleia.
        Atualiza x_best se for uma posição melhor.
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

2.5 Características das Técnicas da Literatura

Como o objetivo deste trabalho é criar uma técnica que não possua parâmetros ajustados manualmente e possua uma boa performance em alta dimensionalidade, nesta seção será apresentado um resumo destas características, a partir do levantamento realizado dos artigos de apresentação das formas canônicas de cada técnica. A Tabela 1, apresenta a quantidade de parâmetros (P), a dimensão máxima testada (d) e a referência da técnica.

Tabela 1: Resumo das características das técnicas

Inspiração	Sigla	P	d	Referência
Princípios da Matemática	SCA	–	20	(MIRJALILI, 2016b)
Propriedades da Física	CSS	4	8	(KAVEH; TALATAHARI, 2010)
Propriedades da Física	GSA	5	49	(ELDOS, 2013)
Propriedades da Física	VS	–	30	(DOAN; ÖLMEZ, 2015)
Relações Humanas	TLBO	–	13	(RAO; SAVSANI; VAKHARIA, 2011)
Relações Humanas	SLC	5	200	(MOOSAVIAN; ROODSARI, 2014)
Bioinspirada	PSO	3	13	(EBERHART; KENNEDY, 1995)
Bioinspirada	BA	2	2	(YANG, 2010b)
Bioinspirada	CS	2	8	(YANG, 2010a)
Bioinspirada	ABC	2	100	(AKAY; KARABOGA, 2012)
Bioinspirada	KH	3	20	(GANDOMI; ALAVI, 2012)
Bioinspirada	AAA	3	50	(UYMAZ; G.; YEL, 2015)
Bioinspirada	EHO	5	15	(WANG; DEB; COELHO, 2015)
Bioinspirada	WOA	–	30	(MIRJALILI; LEWIS, 2016)

Na coluna que apresenta a quantidade de parâmetros, as técnicas com "−" não possuem parâmetros ajustados manualmente. A Tabela 1 mostra que a maioria das técnicas levantadas possui 2 ou mais parâmetros que devem ser ajustados para o bom funcionamento da técnica. As técnicas GSA, SLC e EHO são as possuem a maior quantidade de parâmetros ajustáveis manualmente. Em seus artigos de apresentação, apenas o ABC e o SLC realizaram testes em problemas de 100 ou mais dimensões.

2.6 Considerações Finais

Este capítulo apresentou algumas das diversas técnicas metaheurísticas existentes. A partir dos estudos realizados foi possível observar os passos, a complexidade e particularidades de cada uma das técnicas. Observou-se que, dentre as quatorze técnicas levantadas, dez possuíam pelo menos 2 parâmetros para ajuste manual. Em termos de comparação, cada uma das técnicas em seus artigos de estreia usaram muitas vezes funções de custo diferentes para *benchmark*, bem como cenários de teste diversos, sendo muito difícil fazer comparações entre elas. No Capítulo 4, será mostrado o desempenho da técnica AIPH e das técnicas citadas neste capítulo, a partir do uso de um mesmo conjunto de funções de teste.

Capítulo 3

ALGORITMO INSPIRADO NOS PÁSSAROS DE HITCHCOCK

NO Algoritmo Inspirado nos Pássaros de Hitchcock (AIPH) é simulado o comportamento violento da revoada de pássaros apresentado no filme e seu movimento para encontrar os pontos com maior potencial de dano aos humanos. A estratégia de ataque dos pássaros do filme Os Pássaros é regida por três processos, que são: a espreita, ataque e reorganização. Nas próximas seções, será descrita a inspiração para cada um desses processos e os recursos matemáticos usados.

3.1 Inspiração

Alfred Joseph Hitchcock (1899-1980) foi um cineasta britânico, considerado o "mestre do suspense" e um dos pioneiros no suspense psicológico. Os filmes de Hitchcock possuem como principal característica o uso do suspense a partir de tramas que exploram psicologicamente os temores humanos, com elaborados movimentos de câmera, edição e trilha sonora que realçam o clima de suspense e terror.

Em 28 de março de 1963, Alfred Hitchcock estreou seu novo filme de suspense, Os Pássaros (*The Birds*), em Nova York. Com base em uma história de Daphne du Maurier, o roteiro tem como premissa um ataque inesperado à raça humana pelos pássaros.

Alfred Hitchcock criou uma provocação elaborada em Os Pássaros, como se para provar que o suspense e as emoções podem ser induzidos tanto pela expectativa de horror como pelo próprio horror. Com uma premissa simples, a suposição de um ataque repentino de pássaros, a direção de Hitchcock apresenta a revolta dos pássaros e seu comportamento de forma convincente.

Os filmes de Hitchcock caracterizam-se por apresentarem duas parcelas: no início, há uma trama inócua, mas em seguida ocorre algo inesperado. O filme começa com um flerte entre Mitch Brenner (Rod Taylor) e Melanie Daniels (Tippi Hedren) em uma loja de animais de estimação de São Francisco, depois culmina na cidade costeira da Bodega Bay, onde a sensação de segurança dos personagens é lentamente corroída pelo curioso comportamento dos pássaros na área.

Em um primeiro momento, em diversas partes da cidade, os pássaros aparecem espreitando os habitantes e fazendo ataques pontuais. Em seguida, aves de todas as descrições, principalmente, corvos e gaivotas, começam a atacar em bandos os humanos da cidade. Com o início da onda de ataque, o comportamento dos pássaros mostra um padrão delineado sutilmente. Estes atacam os humanos com uma intensidade crescente, trazendo o maior caos possível, param o ataque quando o efeito já não é mais percebido e por fim, se agrupam para organizar o próximo ataque (HITCHCOCK, 1963).

O filme termina sem uma explicação sobre por que os pássaros começaram o ataque e sem um final convencional na época, a tela escura com o *The End*, mas finaliza com um cenário quase apocalíptico do resultado do poder destrutivo que um ataque de uma revoada de pássaros trouxe àquela cidade (HITCHCOCK, 1963).

3.2 Comportamento dos Pássaros

Hitchcock acreditava no impacto da imagem. Para ele, olhares, gestos e movimentos podem dizer muito mais do que diálogos explicativos. O diretor inovava ao usar recursos técnicos que elevassem o potencial de narrar visualmente uma história. Por conta disso, em Os Pássaros, elementos são lançados ao longo do filme dando pistas para o expectador do fenômeno que está acontecendo e trazendo-o para àquele cenário. Ao juntar estes elementos, é possível enumerar algumas das características do comportamento dos pássaros que Hitchcock retratou no filme (HITCHCOCK, 1963):

- *A reunião de várias espécies de pássaros para os ataques, oriundos de vários pontos da cidade*: Os pássaros não estão espalhados de forma inteiramente aleatória. No entanto, há pássaros distribuídos em vários pontos do ambiente. A Figura 1 ilustra os momentos do filme que ocorreram investidas de diferentes pássaros.

(a) Andorinhas

(b) Corvos

(c) Gaivotas

Figura 1: Cenas com tipos de pássaros

- *Não há organização nos movimentos para ataque*: Os pássaros não apresentam o movimento de *flocking* ao atacar, cada um, individualmente, tem apenas o objetivo de atingir um alvo. A Figura 2 apresenta o momento em que inicia-se o ataque ao centro da cidade, quando o ataque melhor ilustra esta não organização no movimento coletivo dos pássaros.

Figura 2: Ataque das gaivotas

- *Um sentimento de raiva e vingança contra os humanos*: Sendo um dos motivos especulados pelos personagens do filme para o ataque dos pássaros, esses tipos de sentimentos sugerem a inteligência e raciocínio dos pássaros, trazendo os conceitos de flexibilidade e prospecção nos seus movimentos. A agressividade define a intensidade

do ataque, o discernimento na localização do alvo e quantos pássaros morrem no processo. A Figura 3 mostra o momento em que iniciam-se os diálogos especulando o motivo para o comportamento dos pássaros, com uma das integrantes da conversa sendo uma ornitóloga.

Figura 3: Diálogos sobre motivações dos pássaros

- *Existe um padrão, eles atacam, param e se agrupam*: Este padrão é citado pelo personagem Mitch Brenner próximo do fim do filme. Os pássaros atacam o alvo em ondas. Enquanto há resultado nas investidas, o ataque prossegue. A Figura 4 apresenta o momento em que o personagem Mitch Brenner comenta com a personagem Melanie Daniels sobre o comportamento percebido.

Figura 4: Diálogo sobre comportamento dos pássaros

- *Alguns pássaros se mantêm próximos ao alvo antes do ataque*: Este comportamento pode ser visto na cena onde alguns corvos estão concentrados nas proximidades na área do *playground* e em seguida, ao primeiro sinal de comoção das pessoas, inúmeros corvos começam a atacar. Pode ser entendido como um elitismo este grupo que demarca este lugar promissor à um ataque. A cena do *playground* é ilustrada na Figura 5.

Figura 5: Cena dos corvos no *playground*

- *Novos pássaros aparecem a cada nova onda de ataque*: Devido a agressividade do ataque, muitos pássaros morrem no processo e precisam ser repostos por outros novos para manter a numerosidade da revoada. Estes pássaros vêm de lugares diferentes e se dirigem ao ponto que está para ser atacado. A Figura 6 ilustra os momentos onde mostram-se que o volume de pássaros é grande, mesmo após diversos ataques intensos.

(a) Após ataque ao centro da cidade

(b) Próximo do fim do filme

Figura 6: Cenas com volume de pássaros após ataques

3.3 Estrutura do AIPH

O AIPH é um tipo de inteligência de enxame que tenta captar a essência do comportamento fictício dos pássaros exposto por Hitchcock no filme e modelar um mecanismo de otimização. Seguindo os elementos encontrados no filme, a analogia comportamental do algoritmo considera cada pássaro uma partícula, a área atacada é equivalente ao espaço de busca, o dano causado pelo pássaro representa a aptidão e o maior dano, como a morte de um humano, à encontrar uma solução ótima.

O AIPH resolve um problema de otimização criando uma revoada de pássaros e movendo estes pássaros dentro do espaço de busca. Fórmulas matemáticas definem a

intensidade do seu deslocamento seguindo a sua agressividade, distância do alvo, posição dos melhores pássaros e a estagnação dos resultados do ataque. Este último central para todo funcionamento do algoritmo, representando a falta de progresso no ataque.

A estagnação dos resultados é um fator que tem grande importância no comportamento do movimento dos pássaros no AIPH. Este fator é usado tanto individualmente quanto no grupo. Determinando o crescimento da agressividade de cada pássaro a cada iteração que não se obtém um resultado melhor de aptidão. No grupo, determinando a mudança de pássaros "mortos" ao verificar se há estagnação do conjunto de pássaros, ou seja, a melhor aptidão geral não se altera.

Por apresentar a característica de buscar cada vez um dano maior aos humanos, o AIPH é um algoritmo de otimização por maximização, onde a ideia fundamental é o estabelecimento, em cada passo ou iteração, do movimento de cada um dos pássaros da revoada, no sentido de alcançar um resultado de valor maior que o anterior.

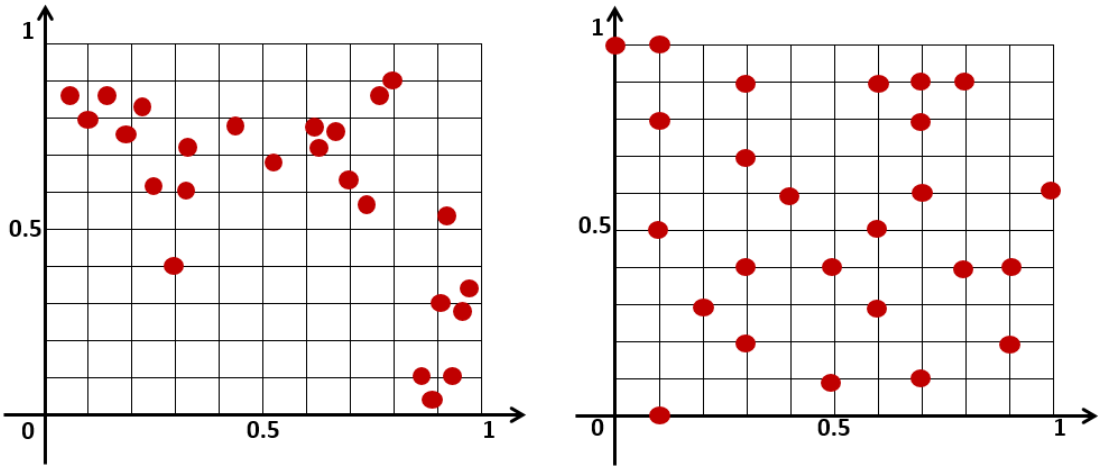
Usando os elementos vistos na Seção 3.2 pode-se resumir o processo que define a estratégia de ataque dos pássaros em três partes: a espreita, o ataque e a reorganização. A espreita representa a inicialização dos pássaros no espaço de busca visando um melhor posicionamento. O ataque representa o movimento dos pássaros no objetivo de atingir um alvo. A reorganização é etapa que vem após a estagnação, onde são excluídos e repostos os pássaros "mortos". Cada um dos três passos será detalhado nas seções seguintes.

3.4 Espreita

Durante o filme, a chegada dos pássaros caracteriza-se por sua origem vir de vários pontos da cidade. Eles ficam posicionados em lugares que favoreçam o ataque, espreitando, aguardando algum estímulo ou oportunidade de atacar. Devido a isso, a forma como eles se distribuem no espaço antes do ataque é não-aleatória e premeditada, tendendo a ficar próximos de pássaros na melhor região. Usando este conceito, a inicialização no algoritmo não pode seguir um espalhamento livremente aleatorizado no espaço e sim algo mais orientado.

No AIPH, a espreita representa a inicialização dos pássaros no espaço de busca. Para reproduzir a forma premeditada de posicionamento dos pássaros do filme, propõe-se um espalhamento aleatório discretizado dos pássaros e em seguida seu deslocamento para a área próxima do pássaro melhor posicionado.

O algoritmo começa com o espalhamento dos pássaros, que, ao invés de criar valores reais uniformemente aleatórios para as coordenadas dos pássaros, considera-se para as coordenadas uma seleção dentro de conjunto de possibilidades com mesma probabilidade de serem selecionadas aleatoriamente. Na Figura 7 pode-se acompanhar a diferença entre os processos de inicialização considerando 25 pássaros.



(a) Espalhamento aleatório simples (b) Espalhamento aleatório discretizado

Figura 7: Comparação entre tipos de inicialização

Diferentemente do espalhamento aleatório simples, conforme Figura 7, no espalhamento discretizado cada uma das possibilidades em um intervalo [0,1] representa uma aresta delineada por suas combinações, criando uma grade. Desta forma, deixa-se de considerar infinitos valores reais possíveis para cada coordenada e passa-se a considerar, em um exemplo de dimensão 2 com 11 possibilidades, uma quantidade de coordenadas igual a 121 (11^2).

A escolha de cada uma das possibilidades é um processo aleatório com reposição, isto é, um mesmo valor dentro do conjunto de possibilidades pode ser escolhido mais de uma vez. Com isso, ter uma revoada de pássaros igual a quantidade de possibilidades não garante que todas as possibilidades serão usadas no espalhamento, podendo ter mais de um pássaro em uma aresta. A probabilidade de ter mais de um pássaro em uma aresta é bem pequena, sendo $0,8\%$ $(1/121)$ em um caso de 2 dimensões e cada vez menor a medida que a quantidade de dimensões cresce.

Este processo de inicialização visa melhorar a geração de pássaros com posições puramente aleatórias, que assumem qualquer valor real dentro dos limites do espaço de busca, podendo deixar regiões do ambiente sem nenhum pássaro, comprometendo a busca do ótimo global.

Após o espalhamento "premeditado" dos pássaros eles tendem a se deslocar para próximo da melhor dentre as posições assumidas, para espreitar o local antes do ataque. Para isso, é avaliada a aptidão de cada pássaro e usado como referência o que possui o maior valor de aptidão dentre estes. O processo completo de inicialização pode ser descrito em 4 etapas:

1. geração de uma revoada inicial atribuindo-se aleatoriamente uma coordenada de localização discretizada do pássaro da revoada, onde os valores para cada coordenada só podem assumir 11 possíveis valores no intervalo [0,1] com passo 0,1, onde cada valor representa uma partição na malha do espaço de busca;

2. em seguida, transforma as coordenadas do intervalo [0,1] para o intervalo real do espaço de busca usando a Equação 102:

$$x_i^d = (max_d - min_d) * \hat{x}_i^d + min_d, \tag{102}$$

   onde, x_i^d é a d-ésima coordenada do i-ésimo pássaro, $\hat{x}_i^d$ é a d-ésima coordenada discretizada no intervalo [0,1] do i-ésimo pássaro, max_d é o máximo da d-ésima coordenada d e min_d é o mínimo da d-ésima coordenada no espaço de busca.

3. ordena-se decrescentemente os pássaros de acordo com sua aptidão e cada pássaro ganha uma classificação r_i com o valor 1 representando o melhor pássaro e k representando o último pássaro, onde k é o tamanho da revoada de pássaros;

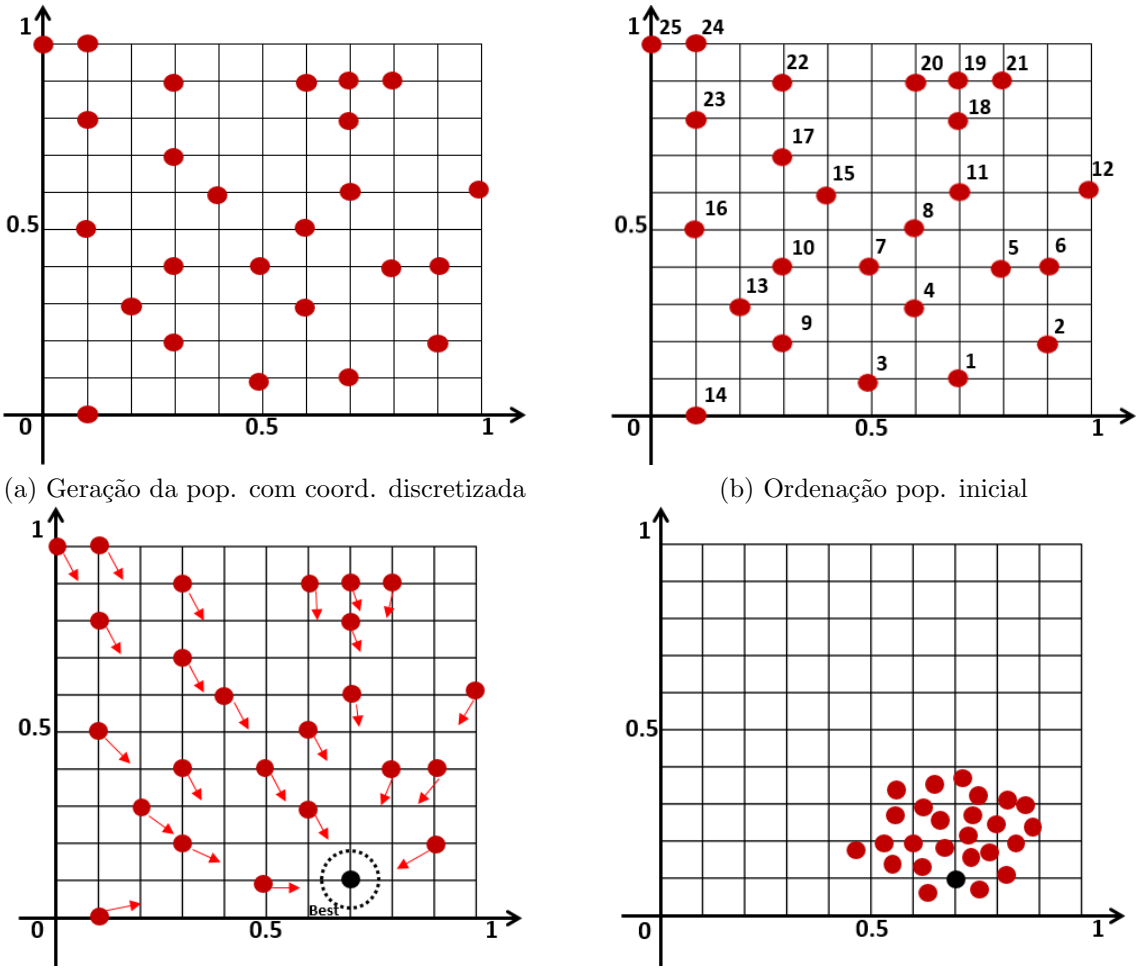
4. verifica-se qual é a melhor posição dentre as geradas e determina-se uma aproximação de cada pássaro para essa posição, de acordo com sua ordenação. O deslocamento dos pássaros segue o cálculo visto na Equação 103:

$$\tilde{x}_i = x_i - ((x_{best} - x_i)/r_i), \tag{103}$$

   onde, $\tilde{x}_i$ é a nova posição do i-ésimo pássaro, x_i é a posição anterior do i-ésimo pássaro, x_{best} é a posição do melhor pássaro e r_i é a classificação do i-ésimo pássaro na ordenação. O r_i é considerado na equação como uma forma de não mover os pássaros para muito próximo de um mínimo local do espaço de busca.

5. Após a definição da nova posição do pássaro, verifica-se a nova aptidão e é feita uma crítica sobre se o deslocamento vale a pena. Caso a posição anterior tenha uma aptidão melhor do que o novo, o pássaro manterá sua posição anterior.

Para explicitar todo o processo enumerado nas etapas acima, é ilustrado na Figura 8 o espalhamento e o deslocamento dos pássaros para a área do pássaro com a melhor aptidão.



(a) Geração da pop. com coord. discretizada

(b) Ordenação pop. inicial

(c) Movimento de aprox. dos pássaros

(d) Final do processo de inicialização

Figura 8: Processo de inicialização discretizada

A Figura 8a apresenta o espalhamento dos pássaros da etapa 1 da inicialização. Em seguida, Figura 8b mostra a ordenação após a transformação das coordenadas para os valores reais do espaço de busca, visto nas etapas 2 e 3. Na Figura 8c, observa-se a definição do melhor pássaro e o deslocamento dos demais pássaros para àquele ponto. Ao final, na Figura 8d encontra-se as novas posições assumidas pelos pássaros.

Seguindo o processo explicitado acima nesta subseção, o Algoritmo 18 mostra o pseudocódigo usado para realizar a etapa de inicialização. Como a inicialização possui etapas de avaliação e deslocamento dos pássaros, uma das iterações, ou a quantidade de avaliações, gastas neste processo são subtraídos do total definido como critério de parada,

como consequência, as fases de ataque e recolocação não considerarão estas quantidades em suas operações.

Algoritmo 18 Inicialização Aleatória Discretizada

Entrada: k: quantidade de pássaros.
$f(x)$: função objetivo.

início
  gera k pássaros com coordenadas discretas aleatórias.
  transforma coordenadas em valores dentro do espaço de busca real (Equação 102).
  avalia cada x_i a partir da $f(x_i)$.
  ordena os $f(x_i)$ do melhor resultado ao pior.
  atribua um valor de ordem r_i de acordo com a colocação de $f(x_i)$.
  encontra o melhor pássaro do grupo (x_{best}).
  para $i = 1, \cdots, k$ **faça**
    $\tilde{x}_i = x_i - ((x_{best} - x_i)/r_i)$
    se $f(\tilde{x}_i) > f(x_i)$ **então**
      $x_i = \tilde{x}_i$
    fim
  fim
  retorna x
fim

3.5 Ataque

Fazendo uma associação do filme com um comportamento real na natureza, o comportamento de ataque dos pássaros exposto por Hitchcock no filme se assemelha muito ao comportamento de *mobbing* de algumas aves. Segundo Shields (1984),o *mobbing* caracteriza-se por uma aproximação em direção a um predador potencialmente perigoso, seguida de frequentes mudanças de posição, com movimentos voltados para o predador. Exibições visuais e vocais estereotipadas comumente acompanham a ação. Frequentemente o *mobbing* inclui investidas e rasantes no predador e pode até incluir ataque direto.

Este comportamento é totalmente adaptativo às adversidades da investida do ataque ao predador, podendo variar a quantidade de pássaros que participam do *mobbing*, a forma de atuação e a extensão do ataque para espantar o predador. Outra característica do *mobbing* é a chamada a partir da vocalização dos pássaros durante o ataque, usada para convocar pássaros próximos para cooperar no ataque.

A partir destes conceitos formulou-se o comportamento de ataque no AIPH, que considera ajustes adaptativos de parâmetros: elite, nível de agressividade e fim do ata-

que. A elite representa a quantidade de pássaros que demarcam a área ataca, o nível de agressividade define a intensidade do ataque e o fim do ataque representa a extensão da continuidade do ataque. O ataque divide-se em duas estratégias, referindo-se à quando o pássaro está longe ou perto da área do ataque. Nas subseções seguintes serão apresentados os conceitos usados na formulação do ataque dos pássaros retratado no filme de Hitchcock.

3.5.1 Elitismo

Desde quando foi formulada, sendo uma medida bibliométrica, a Lei do Elitismo de Price vem sendo utilizada como objeto de estudos em várias ciências (SILVA; MAROLDI; LIMA, 2014). Em essência, considerando a Lei do Elitismo, se k representa o número total da revoada , então $\sqrt{k}$ representaria a elite desta revoada. Baseando-se neste conceito, a definição da quantidade de pássaros na elite será adaptativo, de acordo com a quantidade total de pássaros escolhida:

$$\epsilon = \lceil \sqrt{k} \rceil, \tag{104}$$

onde, k é a quantidade total de pássaros.

3.5.2 Distribuição Beta e Intensidade de Agressividade

Segundo Moretin & Bussab (2004), as variáveis para as quais os possíveis valores pertencem a um intervalo de números reais são chamadas de variáveis aleatórias contínuas. A partir de um conjunto de valores para variáveis aleatórias contínuas pode-se construir modelos teóricos escolhendo-se adequadamente a distribuição de probabilidade para descrevê-la. O conceito de distribuição de probabilidade é formalizado matematicamente pela teoria da medida, onde uma distribuição de probabilidade é uma medida muitas vezes vista como uma distribuição que descreve o comportamento de uma variável aleatória discreta ou contínua.

Existem diversas distribuições de probabilidade que são úteis para o estudo de diversas situações. Cada distribuição de probabilidade possui sua função densidade de probabilidade (FDP), que é uma função que descreve a probabilidade relativa de uma variável aleatória tomar um valor dado.

No AIPH, a intensidade de agressividade pode ser vista como uma variável aleatória contínua, pois seus valores encontram-se dentro do conjunto do números reais. Espera-se na construção do algoritmo que a intensidade da agressividade assuma valores aleatórios

menores com um probabilidade maior no início do ataque e valores maiores ao fim do ataque, desta forma, é necessária um distribuição de probabilidade que não apresente uma forma fechada, podendo mudar de acordo com o cenário. Para isto, escolheu-se a distribuição Beta.

A distribuição Beta é uma distribuição de probabilidade para variáveis contínuas, definidas no intervalo [0,1] parametrizado por dois parâmetros positivos, denotados por α e β, que aparecem como expoentes da variável aleatória e controlam o formato da distribuição (EVANS et al., 2000).

A distribuição Beta não apresenta forma fechada para sua função de densidade de probabilidade, onde sua curva de densidade varia seu formato de acordo com seus parâmetros. A expressão geral da função densidade de probabilidade Beta é dada por:

$$g(x) = \begin{cases} \frac{(x-a)^{\alpha-1} * (b-x)^{\beta-1}}{B(\alpha,\beta) * (b-a)^{\alpha+\beta+1}}, & \text{se } a \leq x \leq b \text{ e } (\alpha,\beta) > 0; \\ 0, & \text{caso contrário,} \end{cases} \tag{105}$$

onde:

$$B(\alpha,\beta) = \int_0^1 x^{\alpha} * (1-x)^{\beta-1} dx. \tag{106}$$

A distribuição Beta é apropriada para modelar proporções, devido ao seu domínio $[0,1]$ e também pela variedade de forma que a função de densidade de probabilidade pode assumir, de acordo com os valores especificados de α e β.

Aplicando a distribuição Beta na geração de valores para a intensidade de agressividade, tem-se o parâmetro α assumindo o papel de nível de agressividade, sendo iniciado em 1 e crescendo a cada investida não sucedida até alcançar um valor máximo ao passo 1 ou mantém seu valor em casos de sucesso. Os parâmetros de α e β são definidos conforme abaixo:

$$\begin{aligned} \alpha &= \theta_i; \\ \beta &= \Theta - \theta_i + 1, \end{aligned} \tag{107}$$

onde θ_i é o nível de agressividade do pássaro x_i e Θ é igual ao valor que determina o fim do ataque para a reorganização.

O Algoritmo 19 apresenta os passos usados para a atualização do nível de agressividade dos pássaros, que será usado posteriormente para a geração da intensidade da agressividade.

Já a intensidade de agressividade é o valor gerado aleatoriamente no intervalo [0,1], com probabilidade de acontecer seguindo a distribuição Beta, com parâmetros definidos de acordo com seu nível de agressividade. A tarefa realizada pelo Algoritmo 20 é basicamente

Algoritmo 19 Atualização do Nível de Agressividade

Entrada: k: quantidade de pássaros.
θ_i: nível de agressividade de x_i.
$f(x)$: função objetivo.
Θ: fim da onda de ataque calculada.

início
  para $i = 1, \cdots, k$ **faça**
    se $f(\tilde{x}_i) < f(x_i)$ **então**
      $\theta_i = \theta_i + 1$
    fim
    se $\theta_i > \Theta$ **então**
      $\theta_i = \Theta$
    fim
  fim
  retorna θ
fim

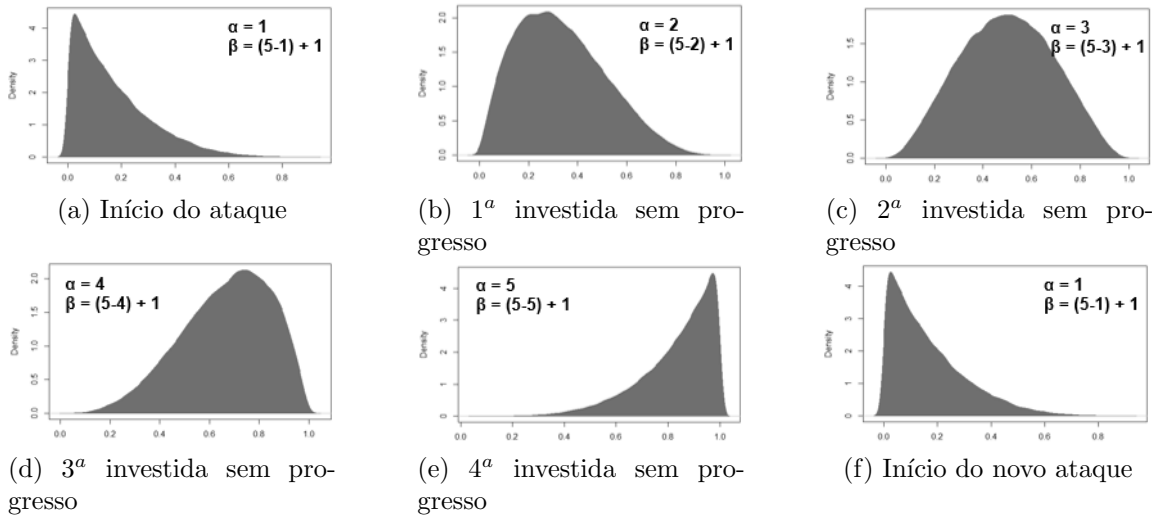
gerar para cada pássaro os valores aleatórios a partir da distribuição Beta, que representam a intensidade da agressividade, com parâmetros definidos conforme o seu respectivo nível de agressividade.

Algoritmo 20 Atualização da Intensidade de Agressividade

Entrada: k: quantidade de pássaros.
θ_i: nível de agressividade de x_i.
Θ: fim da onda de ataque.

início
  para $i = 1, \cdots, k$ **faça**
    $\rho_i = Beta(\alpha = \theta_i, \beta = \Theta - \theta_i + 1)$
  fim
  retorna ρ
fim

A Figura 9 exemplifica o processo de mudança adaptativa dos parâmetros α e β, considerando uma agressividade máxima de 5. Observa-se que, quanto maior o valor do nível de agressividade θ_i, maior a probabilidade de apresentar valores percentuais próximos de 1 para a intensidade de agressividade. Neste exemplo, pode-se ver também que, após uma sequência de investidas sem progresso, θ_i se iguala a Θ, e na próxima investida de ataque reinicia com θ_i o seu valor inicial igual a 1.



(a) Início do ataque

(b) 1^a investida sem progresso

(c) 2^a investida sem progresso

(d) 3^a investida sem progresso

(e) 4^a investida sem progresso

(f) Início do novo ataque

Figura 9: Representação de processo de mudança de valores de α e β

3.5.3 Movimento de Ataque

O movimento de ataque possui como parâmetros para sua extensão a intensidade de agressividade e a iteração corrente no processo de busca, onde a intensidade da agressividade é minimizada a medida que a quantidade de iterações vai acabando.

Além disso, considera-se uma "miopia", devida à distância à posição do pássaro com maior aptidão, funcionando como uma condicional que determina a estratégia que o pássaro irá seguir para voar na direção deste alvo. Para a escolha da estratégia de ataque que será assumida é considerada uma padronização da distância Euclidiana, conforme a Equação 108:

$$\Delta(x_i, x_{best}) = \frac{\delta(x_i, x_{best})}{max(\delta(x_i, x_{best}), \cdots, \delta(x_k, x_{best}))}, \tag{108}$$

onde $\Delta(x_i, x_{best})$ é a distância Euclidiana padronizada entre um pássaro x_i e o pássaro com melhor aptidão, $\delta(x_i, x_{best})$ é distância Euclidiana entre o pássaro x_i e o melhor pássaro x_{best} e o $max(\delta(x_i, x_{best})$ é a máxima distância entre um pássaro da revoada e o melhor pássaro x_{best}. O Algoritmo 21 apresenta os passos para a determinação do movimento de ataque e as respectivas estratégias utilizadas.

Conforme visto no Algoritmo 21, o cálculo da nova posição sempre assume como referência o x_{best}, com diferentes geradores de pesos aleatórios de acordo com a distância.

No caso do pássaro apresentar um $\Delta(x_i, x_{best})$ menor 0,5, considera-se a distribuição Beta com α e β iguais a g, que, a medida que g cresce, assume uma forma semelhante à uma distribuição Normal centrada em 0.5, privilegiando assim que o movimento para o

Algoritmo 21 Estratégias de Ataque

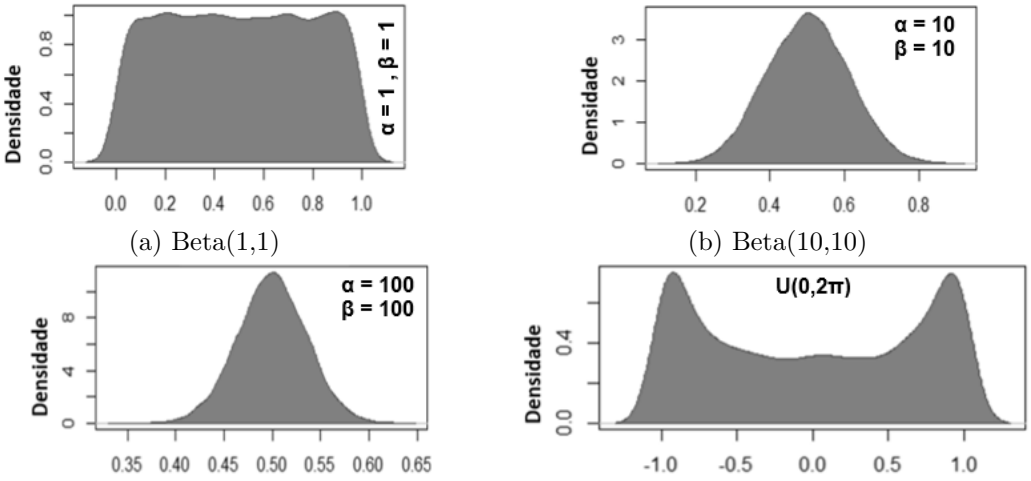
Entrada: g: iteração atual.
G: máximo de iterações.
k: quantidade de pássaros.
ρ_i: intensidade de agressividade.
Θ: fim da onda de ataque calculada.

início
  $W = \rho_i - (\rho_i * ((g-1)/G)$
  para $i = 1, \cdots, k$ **faça**
    calcula a $\Delta(x_i, x_{best})$.
    seleciona aleatoriamente um conjunto V com parte das dimensões do conjunto D com total de dimensões.
    se $(\Delta(x_i, x_{best}) < 0.5)$ **então**
      para v *em* V **faça**
        $\tilde{x}_i^v = x_{best}^v + W * (2 * Beta(\alpha = g, \beta = g) - 1) * (x_{best}^v * \rho_i - x_i^v)$
      fim
    fim
    senão
      $\tilde{x}_i = x_{best} + W * cos(\mathcal{U}(0, 2\pi)) * (x_{best} * \mathcal{U}(0, 1) - x_i)$
    fim
  fim
  retorna $\tilde{x}_i$
fim

meio do caminho do intervalo entre o pássaro i e o *best* tenha uma probabilidade maior. O cálculo do movimento é feito em parte das coordenadas e possui uma penalização da intensidade da agressividade ρ_i na comparação entre um pássaro x_i e x_{best}.

Para distâncias padronizadas maiores que 0.5, usa-se como distribuição a função cosseno fixa entre 0 e 2π, já que sua distribuição privilegia os extremos entre -1 e 1, fazendo com que o pássaro se movimente para direções mais desconexas com o x_{best}. Aqui, todas as dimensões podem sofrer uma alteração, com uma penalização aleatória uniforme no intervalo [0,1] na comparação entre um pássaro x_i e x_{best}. A forma das distribuições de Beta e Cosseno podem ser vistas na Figura 10.

As Figuras 11a e 11b demonstram as estratégias de ataque, onde as zonas mais escuras são as com maior probabilidade de serem atingidas no movimento, seguindo a intensidade da agressividade e distância para o x_{best}.



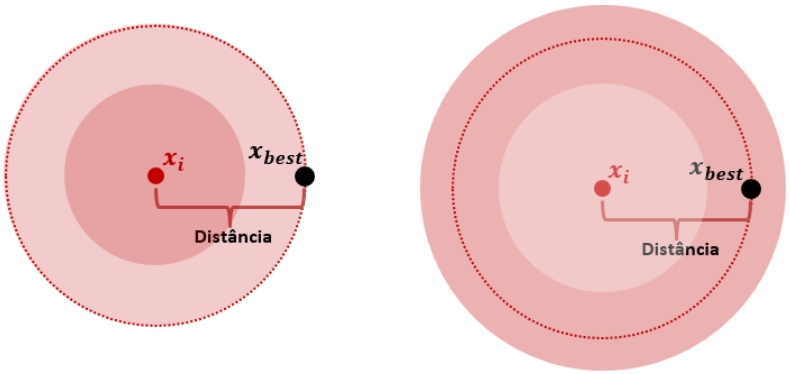
(a) Beta(1,1)

(b) Beta(10,10)

(c) Beta(100,100)

(d) Coseno($U(0, 2\pi)$)

Figura 10: Distribuições para estratégia de ataque



(a) Área de movimento quando $\Delta(x_i, x_{best}) < 0.5$

(b) Área de movimento quando $\Delta(x_i, x_{best}) \geq 0.5$

Figura 11: Movimento de ataque

3.5.4 Fim do Ataque

A quantidade limite de ataques em sequência é uma característica que permite estabelecer o momento para a chegada de novos pássaros para substituir os "mortos", que possuem os piores resultados, e a organização dos sobreviventes ao redor do centro da elite.

O valor para este limite é determinado de acordo com a complexidade do problema, tomando como referência a quantidade de dimensões do espaço de busca. O cálculo do limite para fim do ataque é visto na Equação 109:

$$\Theta = \lceil log(d) + 1 \rceil, \tag{109}$$

onde o fim do ataque (Θ) é um valor inteiro e d é a quantidade de dimensões.

3.6 Estagnação e Reorganização

O objetivo da reorganização, com a definição de pássaros como "mortos" e geração de novos para substituí-los, é a continuidade da exploração do espaço de busca, onde coordenadas de áreas poucos exploradas são privilegiadas. Esta estratégia diminui a chance do processo de busca ficar preso em um mínimo local.

No AIPH, o fim de uma sequência de investidas de ataque, conforme vista no filme, deve-se a estagnação, que é uma sequência prolongada de iterações sem apresentar melhora no resultado da busca de ótimo global. Quando isso acontece, é verificado os desempenhos dos pássaros para a definição dos que serão declarados como mortos. Excluindo-se os pássaros que estão na elite, a escolha dos pássaros declarados como mortos é feita probabilisticamente conforme sua aptidão, seguindo a probabilidade de morte de cada pássaro calculada a partir na Equação 110 .

$$P(x_i = \text{“morto”}) = \frac{f(x_i)}{\sum_{i=1}^{k} f(x_i)}, \tag{110}$$

onde k é a quantidade de pássaros da revoada e $f(x_i)$ é a aptidão do pássaro x_i.

Após o cálculo da probabilidade de morte de cada pássaro, a seleção dos pássaros que serão excluídos da revoada será feito a partir de uma amostragem com reposição, com a probabilidade de exclusão do pássaro representada pela probabilidade de morte $P(x_i = \text{“morto”})$. O Algoritmo 22 apresenta o processo de seleção de pássaros que serão excluídos da revoada.

Algoritmo 22 Seleção de Pássaros "Mortos"

Entrada: k: quantidade de pássaros.
x_i: pássaros.
$f(x_i)$: aptidão do pássaro x_i.

início
  para *i = 1, ⋯, k* **faça**
    Calcula probabilidade morte do pássaro x_i.
  fim
  cria um conjunto M com os pássaros com $f(x_i)$ menor que a média de $f(x)$.
  defina m a quantidade de pássaros em M.
  $\tilde{M} = amostra(M, \mathcal{U}(1, m), [P(x_1 = \text{“morto”}), \cdots, P(x_m = \text{“morto”})])$
  defina $\tilde{m}$ a quantidade de pássaros distintos em $\tilde{M}$.
  retorna $\tilde{M}$ e $\tilde{m}$
fim

O algoritmo de seleção de pássaros "mortos" cria um conjunto M de pássaros que apresentaram aptidão menor do que a média de aptidão de todos os pássaros. Estes pássaros serão os candidatos a exclusão da revoada. Dada a probabilidade de morte de cada pássaro, é feita uma amostra aleatória com reposição, de tamanho também aleatório, no conjunto de pássaros M, que selecionará quais dos pássaros candidatos será excluído da revoada.

Selecionados quais serão excluídos, a geração de pássaros novos traz pássaros de posições diferentes das já vistas. Os pássaros novos considerarão um espaço discretizado, analogamente à inicialização, entretanto, quanto maior a frequência que um das 11 possibilidades do intervalo [0,1] apareceu em uma coordenada, menor sua chance de aparecer em um pássaro novo. A Equação 111 apresenta o cálculo da probabilidade de cada um dos decis q_j para valor de x_m^d.

$$P(x^d = q_j) = \frac{1/n_j^d}{\sum_{j=1}^{11} 1/n_j^d}, \tag{111}$$

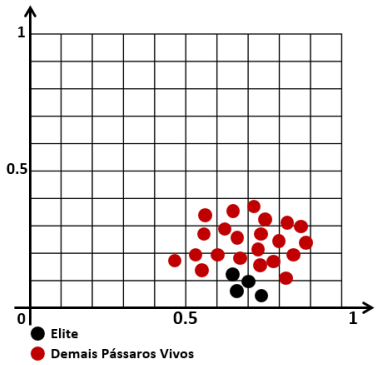
onde q_j é uma das 11 possibilidades do intervalo [0,1], x^d é a d-ésima coordenada do conjunto de k pássaros e n_j^d é a quantidade de vezes que uma das 11 possibilidades apareceu na d-ésima coordenada. O procedimento de geração de pássaros novos é observada no Algoritmo 23.

Algoritmo 23 Geração de Pássaros Novos

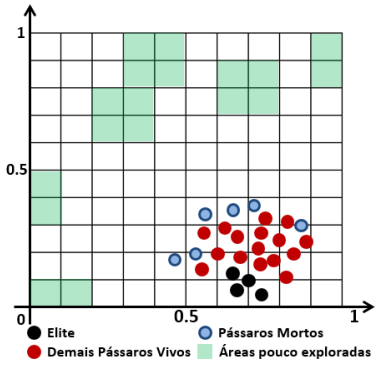
Entrada: x_i:posições dos pássaros atuais.
Q:intervalo [0,1] com passo 0,1.
D: conjunto de dimensões.
$\tilde{M}$: conjunto de pássaros "mortos".
$\tilde{m}$: quantidade de pássaros "mortos".

início
  calcula $P(x^d = q_j)$ de cada uma das 11 possibilidades para as D dimensões.
  gera $\tilde{m}$ pássaros com coordenadas discretas aleatórias.
  para $o = 1, \cdots, \tilde{m}$ **faça**
    para d *em* D **faça**
      $x_o^d = amostra(Q, 1, [P(x^d = q_1), \cdots, P(x^d = q_{11})])$
      transforma a coordenada x_o^d em um valor dentro do espaço de busca real (Equação 103).
    fim
  fim
  $V := [x_1, \cdots, x_{\tilde{m}}]$
  retorna V
fim

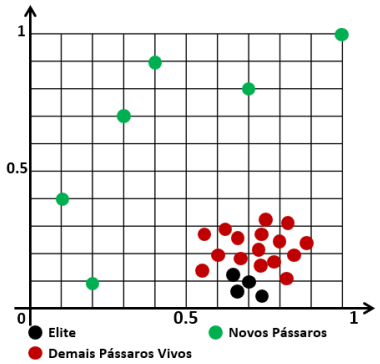
Observa-se no Algoritmo 23 que a construção da geração dos novos pássaros é bem parecida com a inicialização aleatória discreta, entretanto, a área de onde virão os pássaros será uma área que foi pouco explorada pelo AIPH até o momento. A seleção dos valores discretos em cada coordenada tem uma probabilidade inversamente proporcional ao seu aparecimento na revoada, seguindo o cálculo $P(x^d = q_j)$ com $j = 1, \cdots, 11$. Para cada coordenada é feita uma amostragem de tamanho 1, dentro de Q com sua respectivas probabilidades de cada possibilidade dentro daquela coordenada. As Figuras 2.6 ilustram ao procedimento de substituição dos pássaros "mortos".



(a) Posições no fim do ataque



(b) Seleção de pássaros "morto" e áreas pouco exploradas



(c) Geração de pássaros novos oriundos das áreas pouco exploradas

Figura 12: Processo de exclusão e reposição de pássaro discretizada

Na Figura 12a é observada a posição em que encontram-se os pássaros quando o ataque estagnou. Em seguida,na Figura 12b, são selecionados os pássaros que irão sair da revoada e as áreas poucos exploradas no ataque. Por fim, na Figura 12c são vistos os novos pássaros gerados nas arestas das áreas pouco exploradas.

3.7 Algoritmo

Com a apresentação de todos os conceitos e o processo envolvido na estrutura do AIPH, pode-se combinar as partes na esquematização do algoritmo que rege o processo de busca do AIPH. Os passos do AIPH são descrito pelo Algoritmo 24.

Algoritmo 24 Pseudocódigo do Algoritmo AIPH

Entrada: tamanho da k revoada.
parâmetros do espaço de busca (D, max_d e min_d).
critério de parada (máx. de avaliações, máx. de iterações ou tol. ao erro).

início
  $C := 0$
  $\theta_i := 1$
  inicializa a revoada de forma aleatória discretizada (Algoritmo 18).
  calcula Elite (ϵ) (Equação 104).
  calcula Fim da Onda de Ataque (Θ) (Equação 109).
  enquanto *critério de parada não é satisfeito* **faça**
    se *($C \leq \Theta$)* **então**
      para $i = 1, \cdots, k$ **faça**
        avalia cada um dos x_i a partir da função objetivo.
        atualiza o x_{best} e *best* até o momento.
        realiza o movimento de ataque (Algoritmo 21).
        atualiza θ_i e ρ_i (Algoritmo 19 e 20).
      fim
    fim
    senão
      exclua pássaros "mortos" (Algoritmo 23).
      gera pássaros novos (Algoritmo 23).
      $C := 0$.
      $\theta_i := 1$.
    fim
    se *($f(\tilde{x}_{best}) \leq f(x_{best})$)* **então**
      $C = C + 1$
    fim
  fim
  retorna x_{best} e $f(x_{best})$.
fim

No início do algoritmo definem-se então os valores referentes a revoada e ao espaço de busca, como: tamanho da revoada, quantidade de dimensões, limites máximo e mínimo do espaço de busca. Determina-se o critério de parada, que podem ser: tolerância de erro, quantidade de máxima de tempo, avaliações e/ou iterações. A revoada de pássaros

é inicializada com posição gerada aleatoriamente de forma discreta. A primeira iteração do algoritmo calcula os parâmetros de elite (ϵ) e fim da onda de ataque (Θ). Tendo já definido os valores para posição dos pássaros e suas respectivas intensidades de agressividade, aplica-se o cálculo da aptidão a cada pássaro da desta revoada. Com os pássaros da revoada avaliados, extraem-se o x_{best}, isto é, a melhor posição encontrada pela revoada. Em seguida, realiza o movimento de ataque dos pássaros. Após o ataque, o nível de agressividade de cada pássaro e, consequentemente sua respectiva intensidade de agressividade são atualizadas. Diante das novas posições, caso o critério de parada tenha sido atingido, a solução do problema encontrada é apresentada. Caso contrário, é verificado se o fim da onda de ataque foi alcançado, em caso afirmativo, realiza a reorganização dos pássaros e reiniciam-se os valores de C, senão, repete-se a etapa de movimentação de ataque e a cada iteração sem melhora soma-se acumulativamente à C o valor 1, até alcançar o valor de Θ. O laço prossegue até o critério de parada ter sido atingido.

3.8 Considerações Finais

Este capítulo apresentou as inspirações e os operadores envolvidos no processo de busca do AIPH. Destacam-se como diferenciais da técnica AIPH o uso de parâmetros adaptativos, dispensando a necessidade de ajuste manual, utilizando a distribuição de probabilidade Beta. Além disso, foi proposto um tipo de inicialização diferenciada, considerando um espaço de possibilidades discreta. No próximo capítulo de resultados, com o entendimento das particularidades do AIPH, será mostrado seu desempenho em comparação às técnicas apresentadas no Capítulo 2.

Capítulo 4

RESULTADOS

NESTE capítulo serão avaliadas a sensibilidade da técnica AIPH e seu desempenho em relação a outras técnicas da literatura. A Seção 4.1 apresenta as premissas dos experimentos e Seção 4.2 detalha as características das funções de custo usadas para comparação de desempenho. Na Seção 4.3 são expostos as soluções referentes à análise sensibilidade do AIPH à diferentes escolhas de configuração de tamanho de revoada e de máximo de iterações. Por fim, na Seção 4.4, é feita a avaliação do AIPH, mediante comparações com outras técnicas em diferentes cenários de dimensionalidade.

4.1 Metodologia de Avaliação

Com o objetivo de proporcionar uma análise abrangente sobre o AIPH, a avaliação do desempenho da técnica de otimização proposta no trabalho divide-se em: apresentação das funções de custo, verificação da dispersão dos resultados das repetições realizadas, testes de configuração de tamanho de revoada e máximo de iterações para o AIPH, comparação e avaliação do desempenho geral em relação à outras técnicas.

Nas Subseções 4.4.1 e 4.4.2, dentre os testes com diversos tamanhos de revoada, para ilustrar a influência do tamanho da revoada na busca da melhor solução, serão considerados 10, 50, 100 e 200 pássaros. Na Subseção 4.4.3, para o máximo de iterações, será visto o desempenho do AIPH com quantidades máximas de 100, 500 e 1000 iterações. Em todas as análises, serão verificados os cenários de 10, 60 e 600 dimensões, para observar a necessidade de pássaros e iterações com o aumento da dimensão.

A Seção 4.5, dedica-se a parte experimental de comparação com outras técnicas da literatura, visando conduzir uma avaliação de desempenho do AIPH. Para tal, são realizados experimentos comparativos entre a técnica proposta e as técnicas apresentadas

no Capítulo 2. Os experimentos abordam 8 funções de custo para *benchmark*, utilizadas para avaliar o desempenho de técnicas de otimização em geral. Na comparação, para uma avaliação justa, será considerada a solução encontrada pelas técnicas ao fim de 1000 iterações, com uma quantidade de 100 agentes de busca. Como o AIPH é uma técnica de maximização, converteram-se as funções de custo em problemas de maximização mudando o sinal da função objetivo.

Para a Seção 4.4 realizaram-se inúmeros testes com diversos tamanhos de revoada e de máximos de iteração, bem como na Seção 4.5, onde testaram-se outras funções de custo, contudo, para não tornar o trabalho muito extenso, foi dado destaque aos experimentos que ilustrassem mais objetivamente o comportamento e desempenho do AIPH. Ainda assim, o total de experimentos do AIPH apresentado neste trabalho foi de 168 para a Seção 4.4 e de 24 para a Seção 4.5.

Cada experimento é composto por 30 repetições da técnica, com o objetivo de obter uma quantidade suficiente de observações para o cálculo de estatísticas mais robustas. Para a comparação, será considerada como métrica de avaliação a média dos resultados das 30 repetições. Para diminuir problemas de imprecisão devido aos arredondamentos, todos os valores iguais ou abaixo de $1e-10$ serão considerados zero.

O processamento dos experimentos foi conduzido em um notebook com a seguinte configuração básica: processador Intel Core i7; frequência de *clock* de 2,71 GHz; Capacidade de memória RAM de 16 GB; Sistema Operacional: Windows 10.

4.2 Técnicas para Análise Comparativa

Os algoritmos das técnicas usadas para comparação foram executados em *Matlab*®, que é um *software* pago, de alto nível e destinado à computação técnico-científica no desenvolvimento de sistemas (MATSUMOTO, 2002). É o software que possui a grande maioria das técnicas de otimização metaheurísticas de inteligência de enxame desenvolvidas na literatura.

O uso da linguagem *R* é gratuita e de código aberto, por conta disso, possui várias bibliotecas de funções desenvolvidas de técnicas estatísticas e computacionais, o que propicia excelente ambiente para análises e com recursos gráficos de alta qualidade (MORAIS R. G.; DIAS, 2015). O AIPH foi implementado e executado em *R*, com o objetivo de contribuir com este trabalho em oferecer mais uma opção de algoritmo de otimização

para esta linguagem, que carece de uma diversidade de opções de metaheurísticas de inteligência de enxame.

Objetivando uma análise comparativa mais ampla da qualidade das soluções do AIPH, o seu desempenho é confrontado com as soluções obtidas por outras 14 técnicas da literatura. O conjunto de técnicas escolhidas para comparação terá 8 técnicas bioinspiradas, 3 de propriedades da física, 2 de relações humanas e 1 de princípios da matemática, conforme Tabela 2.

Como o AIPH apresenta-se na sua forma canônica neste trabalho, escolheram-se técnicas de otimização em suas versões iniciais, vistas em seus primeiros artigos publicados. Todas as técnicas citadas no Capítulo 2 foram usada para análise.

Em geral, tentou-se selecionar técnicas que representassem diferentes inspirações, priorizando técnicas consagradas, com PSO, ABC, CS e BA, e técnicas mais recentes, como WOA, SCA e EHO, publicadas em 2016. As técnicas SBO e FSS, até a conclusão deste trabalho, não possuiam o algoritmo na linguagem *Matlab*® ou *R*. Já o MVMO, não possuia o código em *Matlab*® para sua forma canônica e sim para uma das suas variações, o MVMO-SH. O SA foi descartado devido ao diferente paradigma usado em seu processo de busca.

Algumas técnicas que ficaram de fora mesmo com o código *Matlab*®, são do conjunto de técnicas bioinspiradas, que possui a maior variedade, desta forma, para não tornar o trabalho muito extenso, optou-se pela escolha das mais recentes ou clássicas. Outras técnicas não usadas na análise, mas com grande contribuição ao estudo da arte, podem ser vistas no Apêndice D.

Tabela 2: Técnicas Usadas para Comparação de Desempenho

Inspiração	Sigla	Nome	Referência
Princípios da Matemática	SCA	*Sine Cosine Algorithm*	(MIRJALILI, 2016b)
Relações Humanas	TLBO	*Teaching Learning Based Optimization*	(RAO; SAVSANI; VAKHARIA, 2011)
Relações Humanas	SLC	*Soccer League Competition*	(MOOSAVIAN; ROODSARI, 2014)
Propriedades da Física	CSS	*Charged System Search*	(KAVEH; TALATAHARI, 2010)
Propriedades da Física	GSA	*Gravitational Search Algorithm*	(RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009)
Propriedades da Física	VS	*Vortex Search*	(DOAN; ÖLMEZ, 2015)
Bioinspirada	PSO	*Particle Swarm Optimization*	(KENNEDY; EBERHART, 1995)
Bioinspirada	BA	*Bat Algorithm*	(YANG, 2010b)
Bioinspirada	CS	*Cuckoo Search*	(YANG, 2010a)
Bioinspirada	ABC	*Artificial Bee Colony*	(AKAY; KARABOGA, 2012)
Bioinspirada	KH	*Krill Heard*	(GANDOMI; ALAVI, 2012)
Bioinspirada	AAA	*Artificial Algae Algorithm*	(UYMAZ; G.; YEL, 2015)
Bioinspirada	EHO	*Elephant Herding Optimization*	(WANG; DEB; COELHO, 2015)
Bioinspirada	WOA	*Whale Optimization Algorithm*	(MIRJALILI; LEWIS, 2016)

4.3 Funções de Custo

Problemas de otimização podem ter múltiplos ou até mesmo um número infinito de soluções ótimas, dependendo do domínio do espaço de pesquisa. O objetivo de qualquer bom algoritmo de otimização global é encontrar soluções globalmente ótimas ou pelo menos sub-ótimas.

As funções de custo para *benchmark* podem ser caracterizadas pela quantidade de ótimos locais, diferenciabilidade, continuidade, separabilidade e escalabilidade (JAMIL; YANG, 2013).

Os conceitos de continuidade e de diferenciabilidade de uma função possuem uma relação estreita. Por definição, uma função f é diferenciável, se f é derivável em todo ponto a pertencente ao domínio de f e, dado que é diferenciável em a, então f é contínua em a. Entretanto, apenas sabendo que f é contínua em a, nada pode-se afirmar sobre a diferenciabilidade da função f em a (STEWART, 2006).

Uma função com mais de um ótimo local é chamada multimodal. Essas funções são usadas para testar a capacidade de um algoritmo para escapar de qualquer mínimo local. Se o processo de exploração de um algoritmo é mal projetado, então ele não pode pesquisar o espaço de função de forma eficaz. Isso, por sua vez, leva a um algoritmo ficar preso em um mínimo local. Funções multimodais com muitos mínimos locais estão entre a classe mais difícil de problemas para muitos algoritmos. Funções com superfícies planas representam uma dificuldade para os algoritmos, uma vez que a planicidade da função não dá ao algoritmo qualquer informação para direcionar o processo de busca para os mínimos(JAMIL; YANG, 2013).

Outro grupo de problemas de teste é formulado por funções separáveis e não separáveis. Em algumas funções, a área que contém os mínimos globais é muito pequena, quando comparada com todo o espaço de pesquisa. Se o algoritmo não pode manter as mudanças de direção nas funções com um vale curvo estreito ou não pode explorar o espaço de busca de forma eficaz, o algoritmo irá falhar para esses tipos de problemas ((JAMIL; YANG, 2013) apud Junior et al, 2004).

As funções também podem ser caracterizadas em escaláveis e não escaláveis, que refere-se a dimensionalidade da função objetivo. A dificuldade de um problema geralmente aumenta com sua dimensionalidade. Segundo Winston (1992) , quando o número de dimensão aumenta, o espaço de pesquisa também aumenta exponencialmente. Para

problemas altamente não-lineares, esta dimensionalidade pode ser uma barreira significativa para quase todos os algoritmos de otimização.

Com o objetivo de verificar a eficiência e o desempenho da implementação do AIPH, dentre as funções de custo usadas para *benchmark*, vistas em (JAMIL; YANG, 2013), foram escolhidas 8 funções objetivo para este trabalho: Ackley, Dixon-Price, Esfera, Griewank, Rastrigin, Rosenbrock, Schwefel 2.22 e Schwefel 2.26.

4.3.1 Ackley

A função Ackley é uma largamente utilizada para testar algoritmos de otimização. Na sua forma bidimensional, caracteriza-se por uma região externa quase plana e um grande orifício no centro. A dificuldade desta função para algoritmos de otimização, particularmente algoritmos de escalada, está no risco de ficarem presos em um dos seus muitos mínimos locais.

A função objetivo Ackley é escalável, contínua, não separável e é uma função de teste multimodal. Sua definição geral pode ser observada na Equação 112:

$$f_1(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e, \qquad (112)$$

onde o espaço de busca utilizado é restrito ao intervalo $-32 \leq x_i \leq 32$, $i = 1, \ldots, d$. Essa função possui mínimo global $f_1(x) = 0$ obtido no ponto de coordenadas $x_i = 0, i = 1, \ldots, d$ (JAMIL; YANG, 2013). A curva da função Ackley pode ser observada na Figura 13.

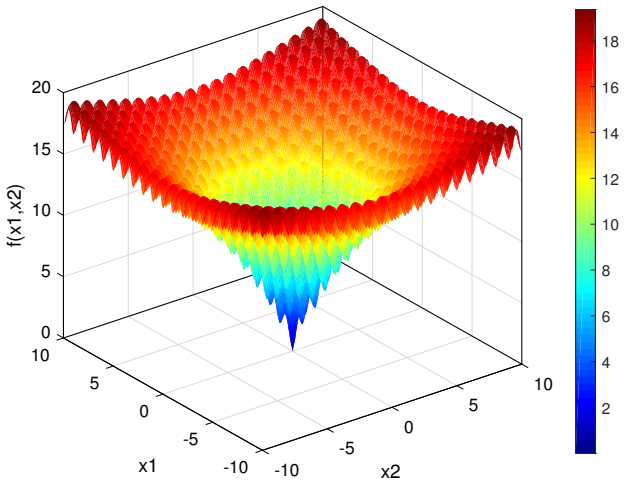


Figura 13: Função Ackley de 2 dimensões

.

4.3.2 Dixon-Price

A função Dixon-Price é contínua, diferenciável, não separável, escalável, unimodal e não convexa. A definição geral pode ser observada na Equação 113:

$$f_2(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2, \tag{113}$$

onde o espaço de busca utilizado é restrito ao intervalo $-10 \leq x_i \leq 10$, $i = 1, \ldots, d$ (JAMIL; YANG, 2013). Essa função possui mínimo global $f_2(x) = 0$ obtido no ponto de coordenadas $x_i = 2^{\frac{-2^i-2}{2^i}}$, $i = 1, \ldots, d$. A curva da função Dixon-Price pode ser observada na Figura 20.

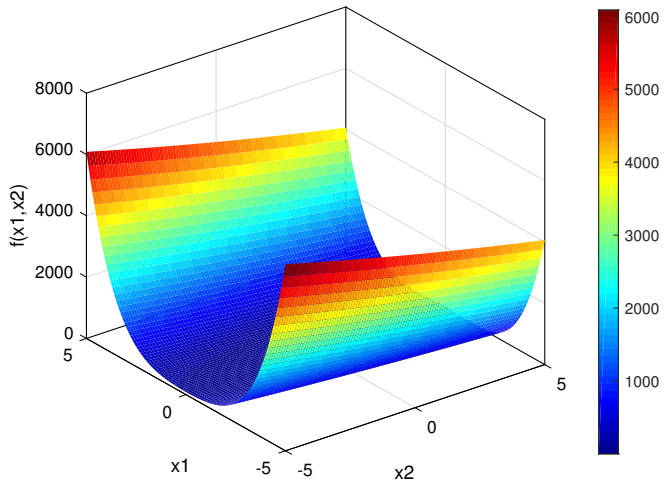


Figura 14: Função Dixon-Price de 2 dimensões

.

4.3.3 Esfera

É um função também conhecida como primeira função de De Jong é uma das mais simples *benchmark* de teste. A função Esfera é escalável, contínua, convexa e unimodal. Sua definição geral pode ser observada na Equação 114:

$$f_3(x) = \sum_{i=1}^{n} (x_i)^2, \tag{114}$$

onde o espaço de busca utilizado é restrito ao intervalo $-100 \leq x_i \leq 100$ e $i = 1, \ldots, d$. A função possui um mínimo global $f_3(x) = 0$ obtido no ponto de coordenadas $x_i = 0, i = 1, \ldots, d$ (JAMIL; YANG, 2013). A curva proporcionada pela função Esfera pode ser observada na Figura 15.

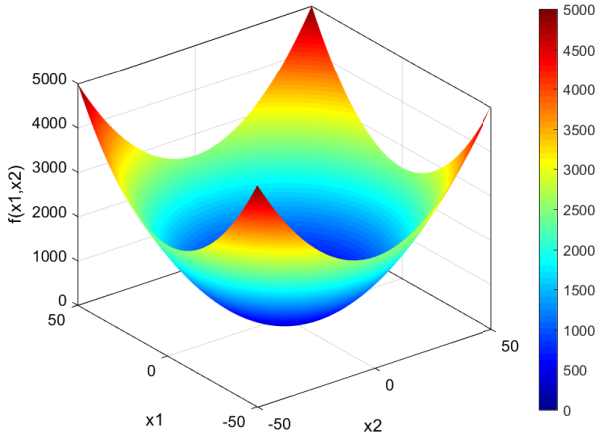


Figura 15: Função Esfera de 2 dimensões

.

4.3.4 Griewank

A função Griewank tem muitos mínimos locais, que são regularmente distribuídos. É uma função contínua, escalável, não convexa e multimodal. A definição geral pode ser observada na Equação 115:

$$f_4(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{115}$$

onde o espaço de busca utilizado é restrito ao intervalo $-600 \leq x_i \leq 600$, $i = 1, \ldots, d$. Essa função possui mínimo global $f_4(x) = 0$ obtido no ponto de coordenadas $x_i = 0$, $i = 1, \ldots, d$ (JAMIL; YANG, 2013). A curva da função Griewank pode ser observada na Figura 16.

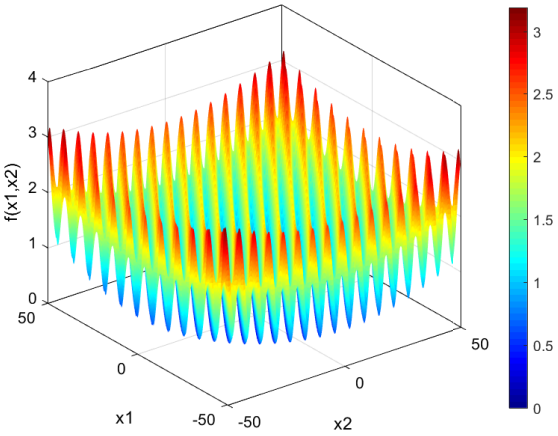


Figura 16: Função Griewank de 2 dimensões

.

4.3.5 Rastrigin

A função de Rastrigin baseia-se na função de De Jong com a adição de modulação de cosseno para produzir vários mínimos locais. É uma função bastante complexa e caracteriza-se por possuir vários mínimos locais regularmente distribuídos. É uma função contínua, escalável, não convexa e multimodal. Sua definição geral pode ser observada na Equação 116:

$$f_5(x) = 10d + \sum_{i=1}^{d}(x_i^2 - 10cos(2_i)) \tag{116}$$

onde o espaço de busca utilizado é restrito ao intervalo $-5,12 \leq x_i \leq 5,12$, $i = 1, \ldots, d$. Essa função possui um mínimo global $f_5(x) = 0$ obtido no ponto de coordenadas $x_i = 0, i = 1, \ldots, d$ (JAMIL; YANG, 2013). A curva da função Rastrigin pode ser observada na Figura 17.

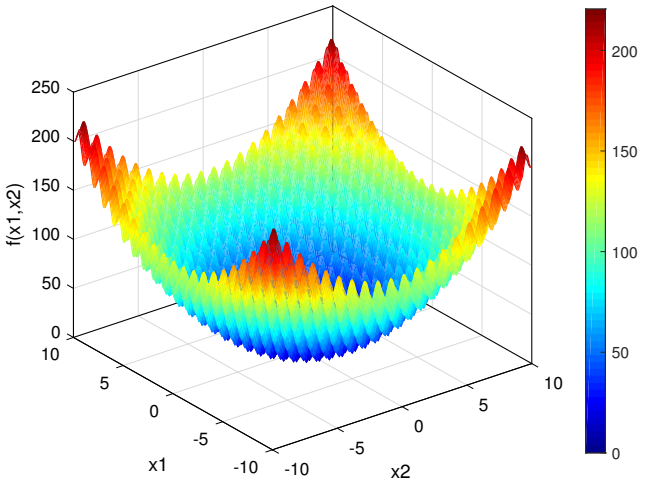


Figura 17: Função Rastrigin de 2 dimensões

.

4.3.6 Rosenbrock

A função Rosenbrock, ou função Banana, é contínua, convexa, escalável, multimodal e diferenciável. A função Rosenbrock é um clássico problema de otimização, com difícil convergência para o ótimo global. A curva da função Rosenbrock pode ser observada na Figura 18.

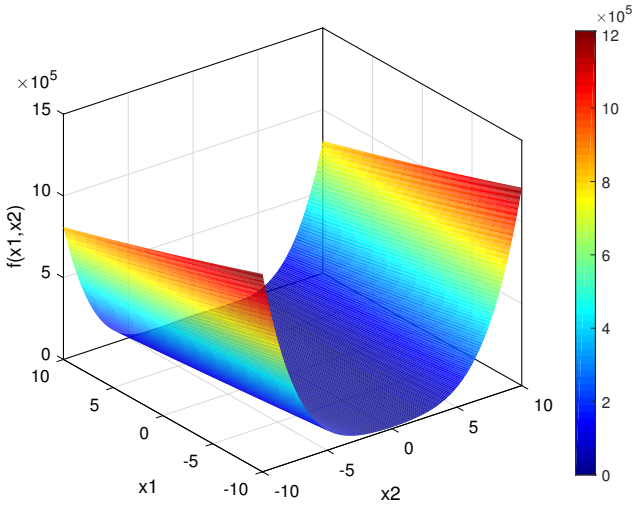


Figura 18: Função Rosenbrock de 2 dimensões

.

A função é unimodal em até 2 dimensões e em maiores dimensões passa a ser multimodal. Sua definição geral pode ser observada na Equação 117:

$$f_6(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{117}$$

onde o espaço de busca utilizado é restrito ao intervalo $-2,048 \leq x_i \leq 2,048$, $i = 1, \ldots, d$. Essa função possui mínimo global em $f_6(x) = 0$ obtido no ponto de coordenadas $x_i = 0, i = 1, \ldots, d$ (JAMIL; YANG, 2013).

4.3.7 Schwefel 2.22

A função objetivo Schwefel 2.22 é unimodal, escalável, convexa, separável e não diferenciável. Sua definição geral pode ser observada na Equação 118:

$$f_7(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i \tag{118}$$

onde o espaço de busca utilizado é restrito ao intervalo $-10 \leq x_i \leq 10$, $i = 1, \ldots, d$. Essa função possui mínimo global $f_7(x) = 0$ obtido no ponto de coordenadas $x_i = 0, i = 1, \ldots, d$ (JAMIL; YANG, 2013).

Por apresentar em sua equação o produtório dos valores x_i, dependendo da dimensionalidade, faz com que a solução obtida na inicialização do algoritmo de otimização seja muito elevada. Dada essa implicação, o algoritmo de otimização pode ter dificuldade em convergir à uma solução satisfatória em poucas iterações. A curva da função Schwefel 2.22 pode ser observada na Figura 19.

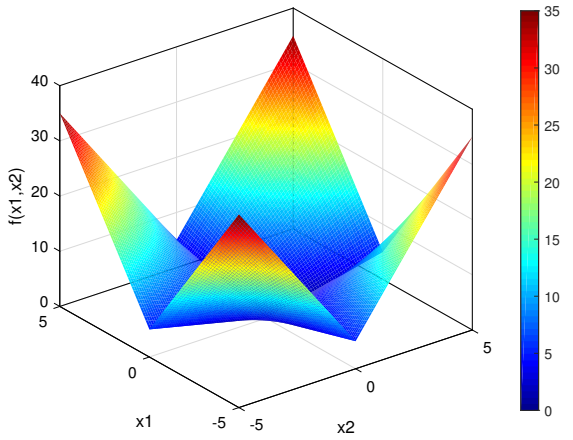


Figura 19: Função Schwefel 2.22 de 2 dimensões

.

4.3.8 Schwefel 2.26

A solução global da função Schwefel 2.26 encontra-se geometricamente distante das soluções locais. Desta forma, os algoritmos de otimização são potencialmente propensos pela busca na direção incorreta. Sua definição geral pode ser observada na Equação 119:

$$f_8(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{x_i}) \tag{119}$$

onde o espaço de busca utilizado é restrito ao intervalo $-500 \leq x_i \leq 500, i = 1, \ldots, d$. Essa função possui mínimo global $f_8(x) = 0$ obtido no ponto de coordenadas $x_i = 420,9687, i = 1, \ldots, d$ (JAMIL; YANG, 2013). A curva da função Schwefel 2.26 pode ser observada na Figura 20.

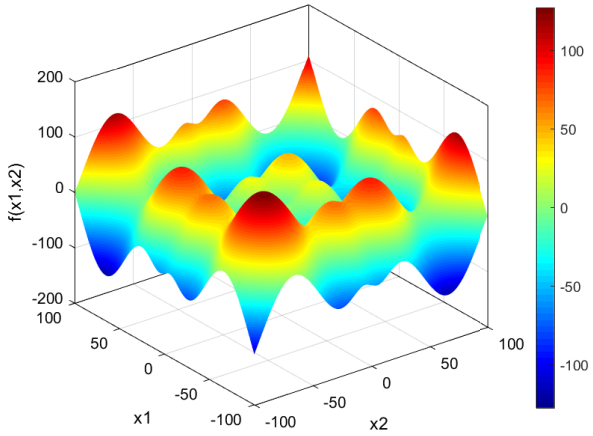


Figura 20: Função Schwefel 2.26 de 2 dimensões

.

4.4 Análise de Sensibilidade

A escolha da quantidade de agentes de busca e o máximo de iterações é uma parte importante para qualquer algoritmo de otimização, que pode afetar a eficiência do processo de busca e o desempenho global. Nesta seção será feita uma análise da dispersão dos resultados das 30 repetições e da sensibilidade do desempenho do AIPH quanto a escolha de diferentes tamanhos de revoada e de máximo de iterações. Com isso, espera-se mostrar o quanto a escolha destes parâmetros impactaria a solução encontrada pelo AIPH.

4.4.1 Dispersão dos Resultados

A realização de várias repetições de um algoritmo de otimização para um mesmo experimento é fundamental para a verificação da estabilidade de suas soluções e medir seu desempenho médio. Uma quantidade baixa de repetições pode fazer com que um bom algoritmo seja resumido a uma falha de convergência eventual.

Para a realização dos experimentos deste trabalho, escolheu-se uma quantidade de 30 repetições, que é uma quantidade razoável para calcular estatísticas robustas das soluções encontradas por cada algoritmo de otimização testado. Entretanto, ao analisar os resultados das repetições do AIPH para as 8 funções, considerando 1000 iterações e diferentes cenários de tamanho de revoada e dimensões, observou-se que a dispersão das soluções destas repetições pode ser alta. Devido a isto, mesmo que na maioria das repetições alcancem boas soluções, as poucas repetições que não convergiram de forma satisfatória, trazem soluções desproporcionalmente piores, comprometendo o resultado médio da técnica. O objetivo desta subseção é analisar o efeito dos resultados atípicos no resultado geral do AIPH por meio do coeficiente de variação (CV) e o índice Herfindahl−Hirschman (IHH).

O coeficiente de variação é uma medida de dispersão empregada para estimar a precisão de experimentos e representa o desvio-padrão expresso como porcentagem da média. Como medida de dispersão, a principal qualidade do CV é a capacidade de comparar resultados de diferentes trabalhos que envolvem a mesma variável-resposta, permitindo quantificar a precisão das pesquisas (Kalil, 1977). Uma classificação é proposta por Gomes (1990), que considera os coeficientes de variação como baixos quando inferiores a 10%, médios entre 10 e 20%, altos entre 20 e 30% e muito altos se superiores a 30%. O

cálculo do coeficiente de variação é dado pela Equação 120:

$$CV = \frac{\sigma}{\mu} \tag{120}$$

onde σ é o desvio padrão e μ é a média.

Já o índice Herfindahl-Hirschman, é uma medida de concentração muito usada em economia. O índice consiste em obter as quotas de mercado dos respectivos competidores, elevá-las ao quadrado e somar o conjunto. Se o resultado estiver acima de uma pontuação, então os economistas consideram que existe uma concentração no mercado e qual a intensidade (KWOKA, 1985).

Podendo se estender este conceito a qualquer estudo de concentração, o IHH é utilizado na teoria econômica para indicar o grau de concentração de um determinado setor, conforme Equação 121:

$$IHH = \sum_{i=1}^{N} \mathcal{P}_i^2 \times 1000, \tag{121}$$

onde, N é a quantidade de elementos do conjunto e $\mathcal{P}_i$ é a proporção de cada um desses elementos no conjunto, dado pela Equação 122:

$$\mathcal{P}_i = X_i / \sum_{i=1}^{N} X_i \tag{122}$$

onde X_i sendo o valor referente ao elemento. Para esclarecer a criação do indicador, considere um conjunto com 6 elementos (0.1, 2, 5, 10, 2.2 , 3), o cálculo de $\mathcal{P}$ se dará da seguinte forma:

$$\mathcal{P} = \left(\frac{0.1}{22,3}, \frac{2}{22,3}, \frac{5}{22,3}, \frac{10}{22,3}, \frac{2.2}{22,3}, \frac{3}{22,3} \right) \tag{123}$$

e em seguida, calcula-se o IHH como:

$$IHH = (0.00^2 + 0.09^2 + 0.22^2 + 0.45^2 + 0.10^2 + 0.13^2) \times 1000 = 287. \tag{124}$$

O IHH pode ser classificado conforme a sua pontuação, onde abaixo de 150 indica a não concentração, entre 150 a 250 indica um conjunto com concentração moderada e acima de 250 indica uma elevada concentração. Desta forma, pode-se considerar o IHH do exemplo como uma evidência de concentração elevada (MOREIRA, 2015).

A Tabela 3 apresenta os CV e IHH de cada experimento realizado. Note que, todos os casos não listados nessa tabela tem como média nula dos resultados e portanto CV é não definido, devido a um denominador nulo. Para os experimentos que aparecem

com o valor com um hífen, são casos onde o resultado médio foi menor que $1e-10$, desta forma, podem ser vistos como um CV próximo de zero. A partir dos resultados pode-se observar 50 dos 96 experimentos, com CV muito superior a 30%, que é o percentual que já classificaria o CV como muito alto. Entretanto, caso o IHH tenha um valor acima de 150, significa que o CV alto é fruto de valores atípicos.

Nas colunas referentes ao IHH, foram destacados em vermelho os casos em que o CV foi maior que o 30% e o IHH foi menor que 150. Desta forma, consegue-se observar que a dispersão dos resultados se deve também a função de custo, pois há um acúmulo de casos em vermelho nas funções Dixon-Price e Rosenbrock, que são funções com superfícies planas, característica que torna mais difícil o processo de busca.

Uma observação que pode ser feita é sobre as soluções melhorarem em algumas funções com o aumento da dimensionalidade, como é o caso da função Rastrigin, com k = 10 e d = 600. Isto se deve ao parâmetro adaptativo do AIPH, Fim do Ataque (Θ), que define quantas iterações o ataque insistirá, sendo maior proporcionalmente a quantidade de dimensões. Sendo um experimento de 600 dimensões, o fim do ataque seria após 4 iterações com a melhor solução estagnada. Como a definição do cálculo de Θ tem como objetivo acompanhar a complexidade dimensional da função de custo, contudo, o benefício da continuidade do ataque compensa a complexidade e ainda oferece um acréscimo de melhoria ao desempenho.

A partir dos resultados obtidos na Tabela 3, pode-se verificar o efeito das piores soluções encontradas pelo AIPH, muitas vezes podem ser tão desproporcionalmente diferentes dos demais valores das repetições, que fazem com que haja uma dispersão alta e efeito na média geral, impactando sua eficiência. Mesmo com aumento do tamanho da revoada, a alta dispersão persiste. Com isso, mesmo o AIPH apresentando uma concentração de tentativas com resultados satisfatórios, existe a probabilidade dele falhar em convergir para a solução, tornando indispensável a realização de várias repetições.

4.4.2 Tamanho da Revoada

A escolha de um tamanho de revoada (k) razoável é importante para que o processo de busca do algoritmo de otimização contenha indivíduos suficientemente diversificados para exploração do espaço de busca.

Tabela 3: Indicadores de Dispersão

Função	Tam. Revoada	Média			CV			IHH		
		d=10	d=60	d=600	d=10	d=60	d=600	d=10	d=60	d=600
Ackley	k = 10	4.6E-01	1.2E-01	3.3E-04	321%	548%	519%	365	1000	1000
	k = 50	1.2E-01	0	0	548%	-	-	1000	-	-
Dixon-Price	k = 10	3.1E-01	6.1E-01	4.5E+01	109%	26%	535%	71.4	35.6	957.3
	k = 50	1.8E-01	5.0E-01	6.7E-01	169%	56%	1%	125	43.4	33.3
	k = 100	6.7E-02	4.2E-01	6.3E-01	305%	77%	20%	333.3	52.5	34.7
	k = 200	4.4E-02	4.2E-01	6.4E-01	381%	77%	17%	500	52.6	34.2
Esfera	k = 10	0	0	3.4E-07	-	-	376%	-	-	-
Griewank	k = 10	5.2E-03	0	3.2E-06	548%	-	526%	1000	-	-
Rastrigin	k = 10	8.3E+00	6.0E+00	6.8E-06	141%	305%	515%	97.6	333.3	-
	k = 50	2.3E+00	9.9E+00	2.0E+01	184%	227%	548%	142.9	200	1000
	k = 100	2.0E+00	8.0E+00	0	203%	259%	-	166.7	250	-
	k = 200	9.9E-01	1.4E+01	0	305%	184%	-	333.3	142.9	-
Rosenbrock	k = 10	3.8E+00	5.0E+01	5.9E+02	95%	40%	0%	62.2	38.5	33.3
	k = 50	2.4E-01	3.6E+01	5.5E+02	548%	72%	27%	1000	50	35.7
	k = 100	2.1E-01	2.5E+01	5.1E+02	548%	109%	40%	1000	71.4	38.5
	k = 200	2.0E-08	3.2E+01	4.7E+02	337%	83%	51%	-	55.6	41.7
Schwefel 2.22	k = 10	0	0	3.1E-08	-	-	286%	-	-	-
Schwefel 2.26	k = 10	2.3E+02	4.7E+02	2.7E+00	238%	381%	237%	216.3	500	213.9
	k = 50	1.2E+02	2.4E+02	9.0E-03	305%	548%	51%	333.3	1000	41.4
	k = 100	3.9E+01	2.4E+02	2.4E+03	548%	548%	548%	1000	1000	1000
	k = 200	1.2E+02	7.6E-04	7.6E-03	305%	0%	0%	333.3	33.3	33.3

O tamanho de revoada não pode ser pequena a ponto de perder a diversidade necessária para convergir a uma boa solução, pois fornecem uma pequena cobertura do espaço de busca do problema. Contudo, se for muito grande, irá demorar em avaliar a função de custo de todo o conjunto a cada iteração e consequentemente necessitará de maior tempo de processamento.

Para a análise foram consideradas 8 funções de custo que aparecem na Seção 4.3. As possibilidades testadas para o tamanho de revoada são 10, 50, 100 e 200 pássaros, com um máximo de iterações de até 1000 iterações. A análise considerará cenários de 10, 60 e 600 dimensões para a verificação da necessidade de pássaros e iterações com o aumento da dimensão. Para a avaliação seram usadas a Tabela 3 e os gráficos com as curvas de convergência para cada função de custo em diferentes dimensões vistos no Apêndice A.

4.4.2.1 10 dimensões

A primeira avaliação realizada usa um cenário de 10 dimensões. A partir da Tabela 3, verifica-se que, na maioria das funções de custo, a solução encontrada pelos tamanhos de revoada de 100 e 200 pássaros são bem melhores que com 10 ou 50 pássaros. Para as funções Esfera e Schwefel 2.22 não houve diferença de desempenho entre os tamanhos de revoada.

No Apêndice A, a Figura 26 apresenta as soluções obtidas para as funções de custo com 10 dimensões. Observa-se que Ackley, Esfera, Griewank e Schwefel 2.22, em até 1000 iterações, a diferença entre os tamanhos de revoada não é tão visível. Na função Rastrigin, o tamanho de 10 pássaros não convergiu na mesma medida que os demais tamanhos de revoada avaliados. A diferença entre os tamanhos de revoada avaliados fica mais evidente nas funções Dixon-Price, Rosenbrock e Schwefel 2.26.

4.4.2.2 60 dimensões

Na avaliação do cenário de 60 dimensões, a partir da Tabela 3, na função Rastrigin o aumento de pássaros não trouxe uma melhora significativa na solução, bem como para a função Ackley, que só aumento de 10 para 50 pássaros fez efeito para a solução da função de custo. As funções Esfera e Schwefel 2.22 continuam sem incremento de desempenho com aumento de pássaro e a função Rastrigin apresenta o mesmo comportamento neste cenário.

A Figura 27 apresenta as soluções obtidas para as funções de custo em um cenário de 60 dimensões. Exceto para as funções Dixon-Price, para todas as outras funções percebe-se que o impacto de usar um tamanho de revoada de 10 e 50 pássaros fica muito mais explícito, quando comparado com o experimento de 10 dimensões da Figura 26. Ainda assim, apesar do descolamento observado, pode-se observar que para as funções de custo Esfera e Rosenbrock, a curva de progressão da busca pela solução global para um tamanho de revoada de 10 pássaros, alcança as demais entre a iteração 30 e 50.

Na Figura 27, observa-se que o tamanho de revoada de 10 pássaros não conseguiu se aproximar do desempenho dos outros tamanhos de revoada. Já a função Rosenbrock, o tamanho de revoada de 200 pássaros, apresentou um desempenho bem melhor que os demais tamanhos de revoada

4.4.2.3 600 dimensões

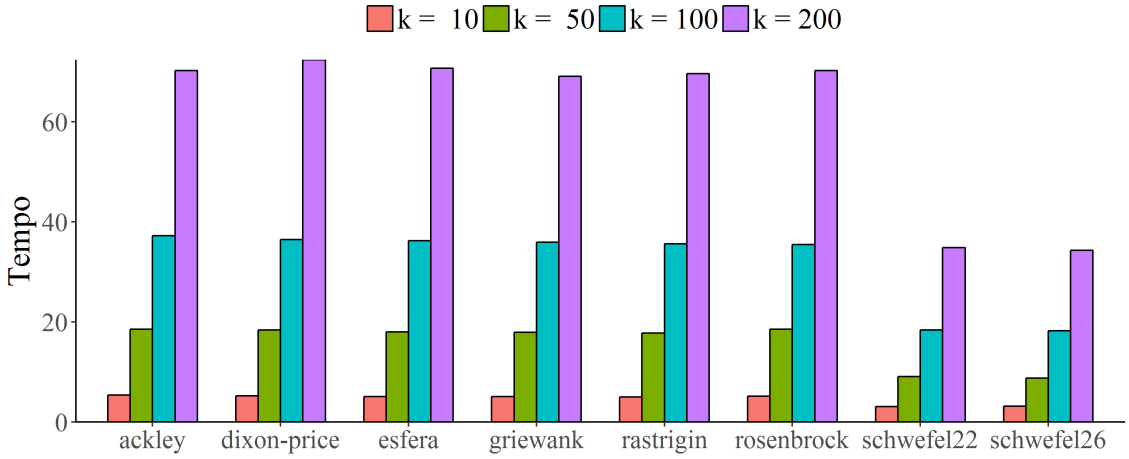
No último cenário com 600 dimensões, mostrado na Figura 28, as soluções obtidas mostram de forma nítida as disparidades entre as curvas de cada tamanho de revoada nas funções de custo. Entretanto, na Figura 28 pode-se notar que nas funções Dixon-Price, Rastrigin, Rosenbrock e Schwefel 2.22 as curvas para 100 e 200 pássaros estão muito próximas, além disso, mesmo nas funções de custo que há um descolamento no início das iterações, a partir da iteração 40 elas praticamente se sobrepõe. Este comportamento leva a crer que um tamanho de revoada maior não trará benefício a evolução do processo de busca.

4.4.2.4 Visão geral

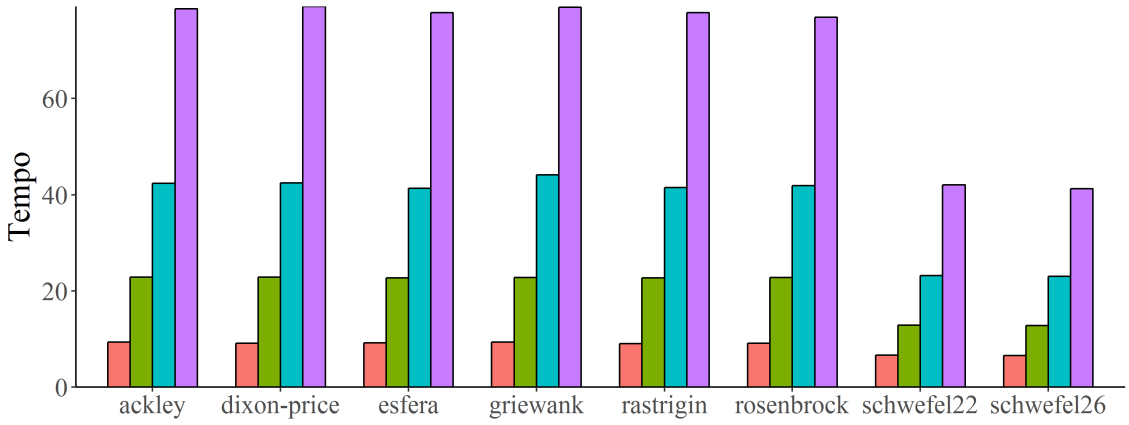
Para auxiliar na decisão, a Figura 21 nesta subseção, mostra a diferença entre os tempos de processamento de cada função de custo em segundos e os respectivos tamanhos de revoada em diferentes dimensões.

Observando a Figura 21, percebe-se que a diferença entre o tempo de processamento com o uso de 100 ou 200 é alta, porém, o incremento na melhoria da solução vindo do aumento de pássaros é bem pequena. Desta forma, aumentar o tamanho da revoada traz um custo de processamento, não compensado pela pequena melhoria na solução, dado que espera-se que o algoritmo traga uma solução satisfatória o mais rápido possível.

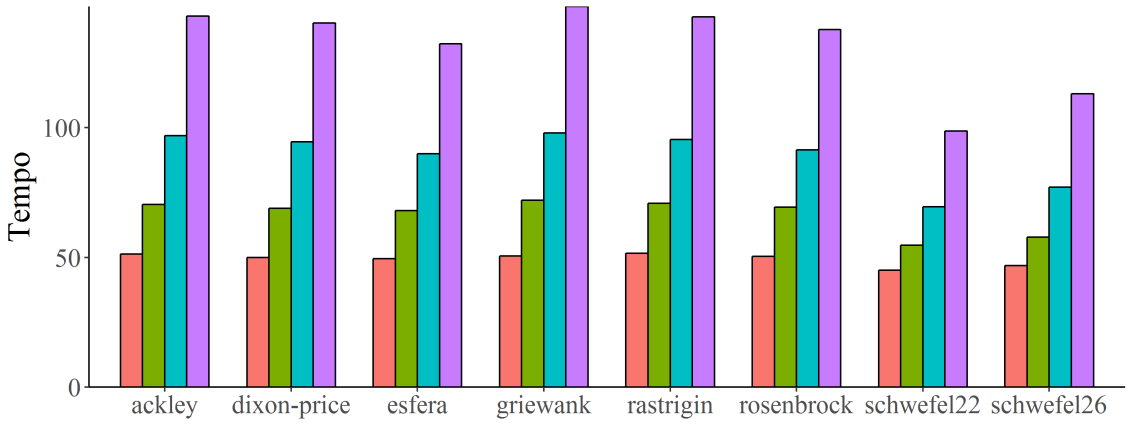
Observa-se que o tempo de processamento aumenta consideravelmente a cada incremento no tamanho da revoada. As funções de custo Schwefel 2.22 e Schwefel 2.26 tem os



(a) d=10



(b) d=60



(c) d=600

Figura 21: Tempo de processamento dos tamanhos de revoada para cada função de custo.

menores tempos de processamento, devido a suas funções matemáticas serem compostas por operações mais simples que as demais.

Dando um enfoque a diferença entre os tamanhos de revoada de 100 e 200 pássaros, pode-se concluir que optar por usar 200 pássaros não é uma boa alternativa, já que uma quantidade de 100 pássaros traz soluções bem parecidas em menos tempo.

A partir dos experimentos realizados nesta subseção, pode-se verificar que um tamanho de revoada adequado pode trazer benefícios, tanto na busca pela solução ótima global quanto no tempo de processamento. Devido à análise feita, para os próximos experimentos será usado um tamanho de revoada de 100 pássaros.

4.4.3 Máximo de Iterações

O máximo de iterações (G), quando usada como critério de parada, define o número máximo de ciclos que será usado na tentativa de alcançar o ótimo global da função de custo. Seguindo esta definição, pode-se supor que, quanto mais iterações, melhor será a solução final. Entretanto, os algoritmos de otimização muitas vezes convergem para uma solução satisfatória antes de alcançar o máximo de iterações, sendo desnecessárias as demais iterações a partir daquele ponto. Além disso, mesmo que a convergência não aconteça, não é possível esperar uma quantidade ilimitada de iterações para se alcançar um valor ótimo. Neste sentido, espera-se que o algoritmo de otimização consiga alcançar uma solução razoável em uma menor quantidade possível de iterações.

Outra questão importante vem do uso do máximo de iterações em parâmetros adaptáveis que regulam o processo de busca de uma técnica de otimização. O AIPH é uma técnica que possui um parâmetro adaptável, o parâmetro W conforme visto no Capítulo 3, bem como o SCA e o VS possuem r_1 e a_t, respectivamente, vistos no Capítulo 2. Em muitos casos, este tipo de parâmetro regula a velocidade de evolução do algoritmo na busca da solução, fazendo com que convirja mais lentamente.

Para verificar estes efeitos, a análise feita nesta subseção considerará uma comparação do AIPH com as técnicas SCA e VS em quantidades de 100, 500 e 1000 iterações para a avaliação de 8 funções de custo escaláveis com 10, 60 e 600 dimensões, onde o SCA possui apenas parâmetros adaptativos e o VS possui como parâmetro da distribuição gama que a compõe o valor 0,1. A comparação com estas outras duas técnicas tem como propósito mostrar os efeitos do máximo de iterações na evolução da busca da solução global. Para reduzir a representação a uma escala mais fácil de ser visualizada e manejada, usou-se a escala logarítmica para a apresentação das soluções.

4.4.3.1 Máximo de 100 Iterações

A Figura 29 apresenta uma visão comparativa entre as técnicas AIPH, SCA e VS para um cenário de 10 dimensões e um máximo de 100 iterações. As técnicas VS e SCA apresentam um progresso mais lento até iteração 50, enquanto que, na maioria das funções de custo, o AIPH apresenta um progresso constante ou aproxima-se da convergência em uma iteração anterior à quinquagésima iteração. Na função Schwefel 2.26 e Ackley, o progresso da busca da solução global se estende até próximo do fim de 100 iterações, com um progresso mais agressivo que o visto no SCA e VS.

Na Figura 30 tem-se um cenário de 60 dimensões e um quantidade máxima de 100 iterações. Mesmo em um cenário de 60 dimensões, o AIPH continua com progresso rápido na convergência logo nas primeiras iterações, enquanto que o SCA e o VS, continuam com um progresso lento até 50 iterações.

Observando a Figura 31, nota-se que, com o aumento da dimensionalidade, o VS apresentou resultados significativamente piores. No cenário de 600 dimensões, o AIPH apresentou um progresso na convergência mais lento do que visto nas dimensões 10 e 60, havendo progresso na convergência até a centésima iteração.

A Figura 31 usa um cenário alta dimensionalidade e um quantidade máxima de 100 iterações para a comparação entre as técnicas. Bem como no experimento apresentado na Figura 31, as curvas mostram que, na maioria das funções de custo, o AIPH apresenta um progresso da busca da solução global ao longo as iterações muito mais perceptível do que as outras duas técnicas. Neste experimento, verifica-se que o VS apresentou o pior progresso da busca da solução global ao longo das iterações e o AIPH apresentou uma constância de decrescimento visível nas funções Esfera, Griewank, Rastrigin, Rosenbrock e Schwefel 2.26.

4.4.3.2 Máximo de 500 Iterações

Observa-se na Figura 32 que ao aumentar a quantidade de iterações, as técnicas SCA e VS conseguem alcançar melhores soluções, comparando-se com o experimento de 100 iterações. Nas funções Ackley, Rosenbrock e Esfera, o AIPH continuou com o comportamento de conversão rápida e apresentou soluções que equiparam-se ao SCA e VS.

Verifica-se o comportamento do AIPH de alcance de uma convergência em menos iterações também na Figura 33, que apresenta um cenário de 60 dimensões e 500 iterações.

Entretanto, com o aumento da dimensionalidade, o AIPH apresentou vantagem em relação as duas outras técnicas na maioria das funções de custo.

No cenário de 600 dimensões, apresentado na Figura 34, observa-se que o AIPH apresentou uma constância de progresso da solução ao longo das iterações nas funções Ackley, Esfera, Griewank, Rastrigin, Rosenbrock e Schwefel 2.26. Em relação a Figura 31, verifica-se que o aumento do número de iterações trouxe uma melhora visível para o AIPH em 3 funções, que são a Ackley, Esfera e Griewank.

4.4.3.3 Máximo de 1000 Iterações

Para o experimento com 1000 iterações e 10 dimensões, visto na Figura 35, observa-se que o AIPH continuou com um progresso acentuado nas primeiras iterações, entretanto, conseguiu alcançar soluções melhores nas funções Schwefel 2.26 e Rosenbrock.

Observa-se para os experimentos de 60 dimensões, vistos nas Figuras 36, que o AIPH converge antes de 200 iterações em todas as funções de custo testadas. Já para os experimentos de 60 dimensões, vistos nas Figuras 37, o AIPH apresentou visível superioridade nas 8 funções de custo, entretanto, convergindo próximo a 500 iterações.

4.4.3.4 Visão Geral

A partir das soluções obtidas nos experimentos realizados com 10, 60 e 600 dimensões, pode-se observar que, diferentemente do SCA e do VS, o progresso da convergência não é lento nas primeiras iterações por conta do parâmetro adaptativo que considera o máximo de iterações em sua equação.

Com o objetivo de evidenciar que o uso do parâmetro adaptativo não desnivela o comportamento de convergência, as curvas apresentadas nas Figuras 22, 23 e 24 comparam as 100 primeiras gerações dos 3 máximos de iterações testados. Desta forma, os gráficos apresentam a totalidade de iterações para os experimentos com máximo de 100 iterações e as 100 primeiras iterações dos experimentos com 500 e 1000 iterações.

No cenário de 10 dimensões, visto na Figura 22, percebe-se que apenas nas funções Schwefel 2.26 e Rastrigin que a escolha de um máximo de 100 iterações faz com que a convergência seja mais rápida e eficiente até a centésima iteração. Já no cenário de 60 dimensões, visto na Figura 23, o maior distanciamento entre as curvas dos 3 diferentes máximos de iteração, acontece nas funções Ackley, Rastrigin e Schwefel 2.26.

Observando o cenário de 600 dimensões, na Figura 24, observa-se que a progressão dos resultados até a centésima iteração foi mais eficiente para um máximo de 100 iterações em 5 funções de custo: Ackley, Dixon-Price, Esfera, Rosenbrock e Schwefel 2.22.

Esta representação demonstra que, mesmo o AIPH possuindo um parâmetro adaptável, que dependente das iterações corrente e máxima, isto não compromete a intensidade com que a convergência progride ao longo das iterações, como acontece em alguns casos com o SCA e o VS.

No geral, pode-se observar que, nas dimensões 10, 60 e 600, o descolamento entre as curvas é muito sutil na maioria das funções de custo. A partir da análise realizada nesta subseção, mesmo que haja uma maior eficiência no uso de poucas iterações, há benefício em algumas funções de custo em considerar um número máximo de iterações maior. Desta forma, na seção de comparação com outras técnicas da literatura será considerado um máximo de 1000 iterações para os experimentos, para todas as técnicas terem uma quantidade razoável de iterações para alcançar uma solução satisfatória.

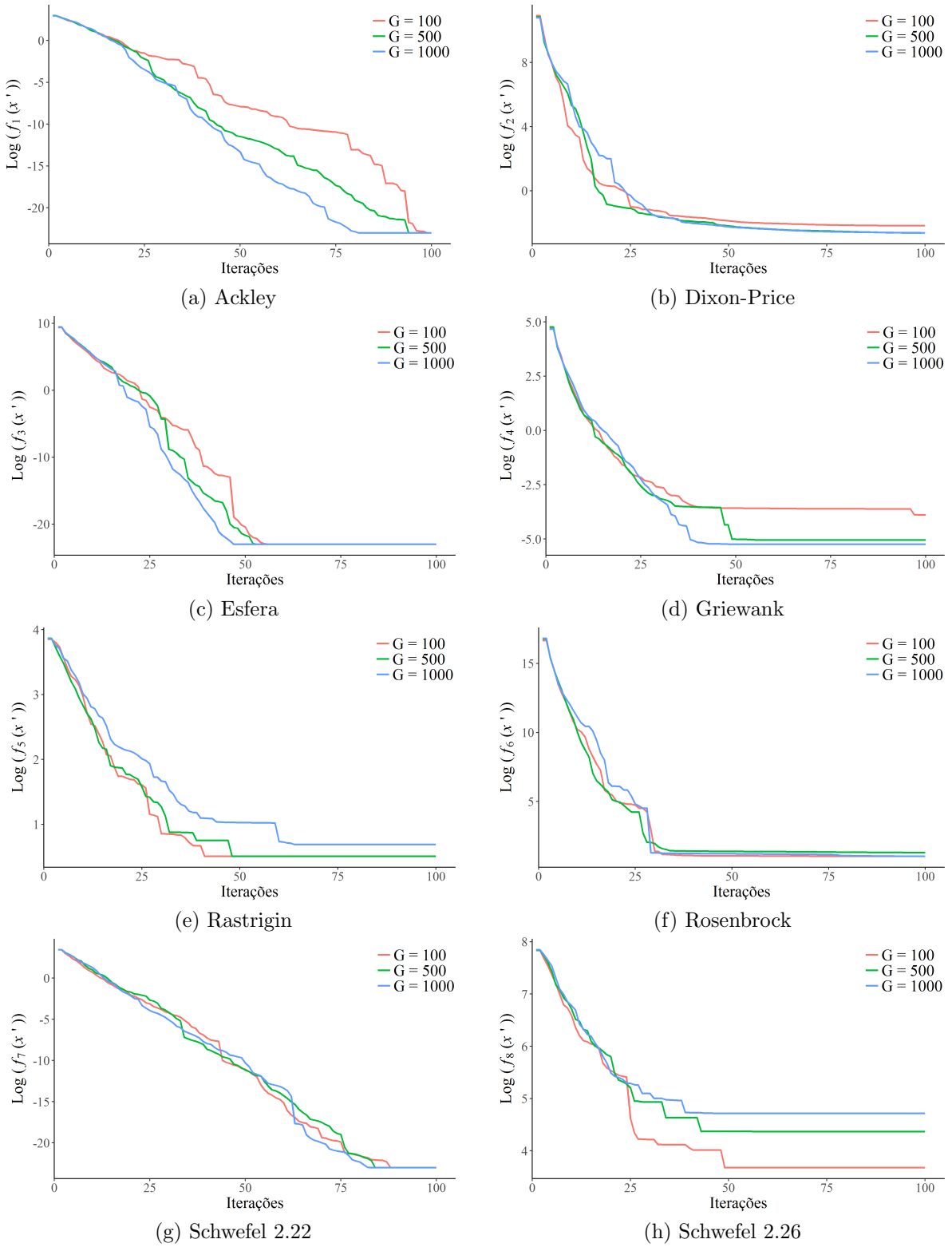
4.5 Comparação com Outras Técnicas

Para não tornar extenso o trabalho de comparação do AIPH com outras técnicas, dentre todas as técnicas disponíveis na literatura, escolheram-se algumas técnicas que possuíam implementação em *Matlab*® e experimentos usando as funções de custo de *benchmark*.

Dentre os objetivos propostos com a técnica AIPH, o principal é obter um bom desempenho em problemas de alta dimensão. Para esta avaliação, serão consideradas 8 funções separáveis com dimensão 10, 60 e 600 e 14 técnicas de otimização da literatura.

Para a comparação de desempenho, em comum as técnicas utilizadas, consideram-se como quantidade de agentes de busca, máximo de iterações e máximo de avaliações iguais a 100, 1000 e 100.000, respectivamente. No geral, os demais parâmetros são definidos conforme seus artigos de origem.

As técnicas WOA, TLBO, SCA e VS não possuem parâmetros extras para ajustar. Os coeficientes de aceleração (c_1 e c_2) do algoritmo PSO são ambos ajustados para 1.8 e o coeficiente de inércia é definido como 0.6. Os parâmetros de força de cisalhamento, de perda de energia e de adaptação da técnica AAA são definidos como 2, 0,3 e 0,5, respectivamente. Para o ABC, o coeficiente de aceleração será igual a 1 e limite de abandono igual a 360.000. Os parâmetros do BA são definidos como 0,5 para a intensidade, 0,5 para



(a) Ackley (b) Dixon-Price

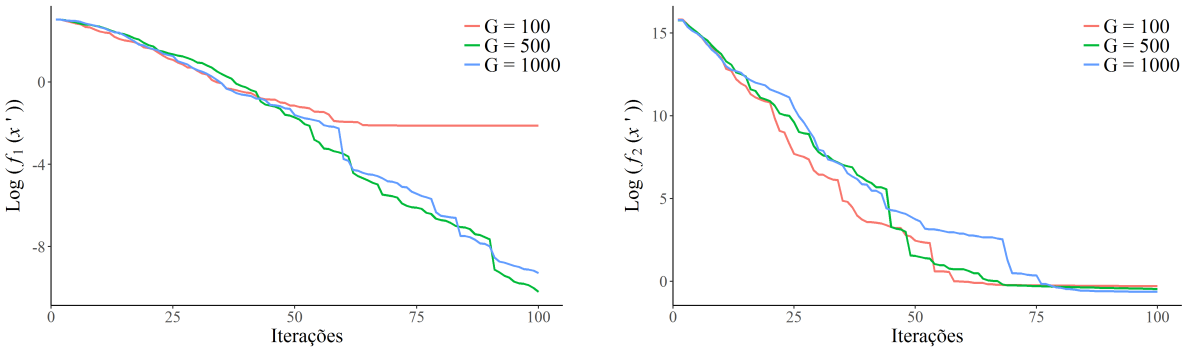
(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

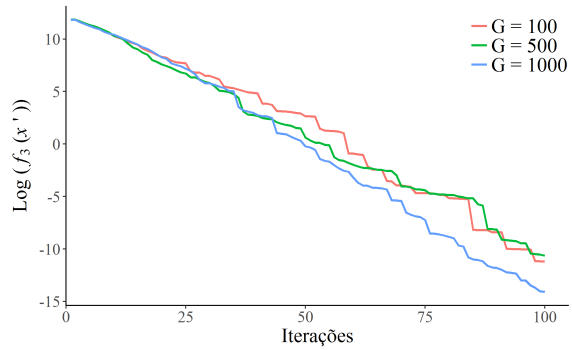
Figura 22: Efeito do máximo de iterações para cada função de custo em $d = 10$.

a taxa de pulso e as frequências entre 0 e 2. Para o CS, considera-se a taxa de descoberta de ovos igual a 0,25 e o parâmetros da distribuição gama igual a 1,5. Na técnica EHO, define-se os parâmetros com o total de clãs de 5, a quantidade de elefantes em cada clã

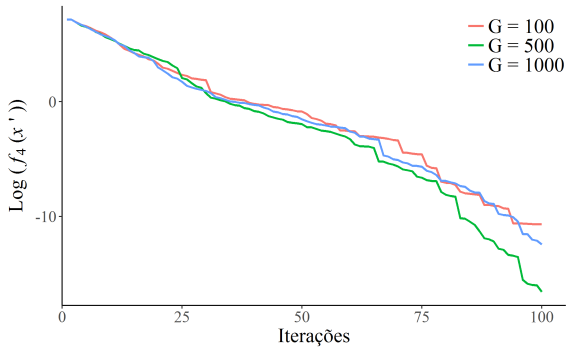


(a) Ackley

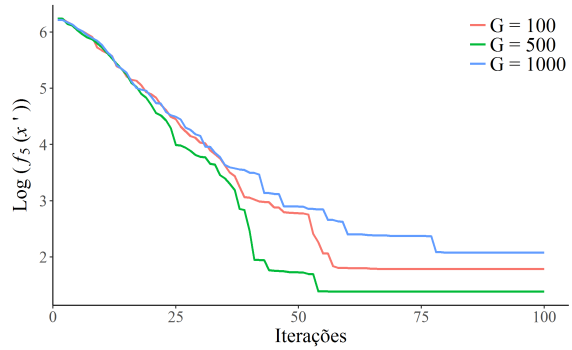
(b) Dixon-Price



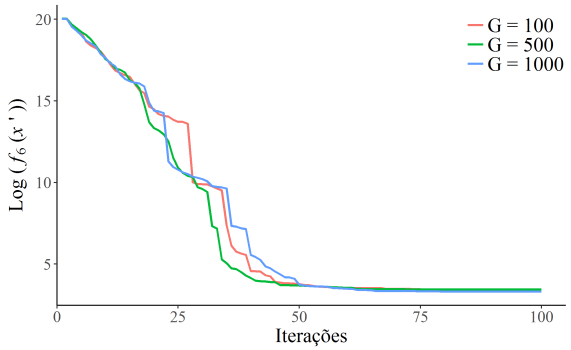
(c) Esfera



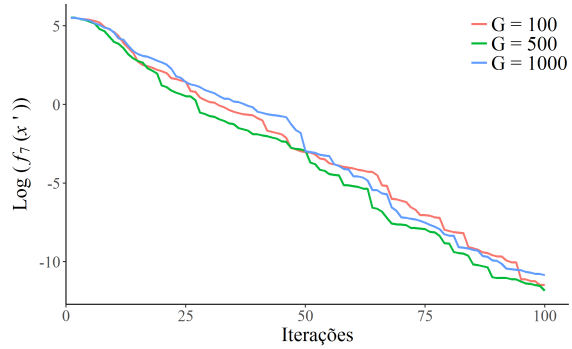
(d) Griewank



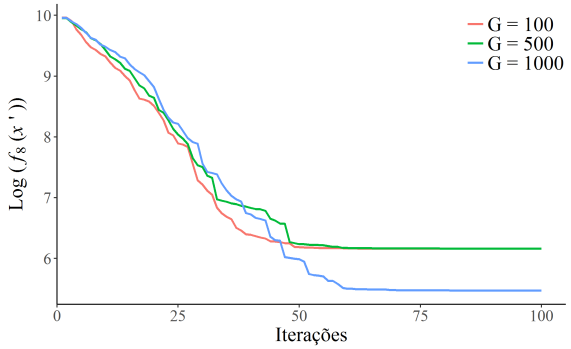
(e) Rastrigin



(f) Rosenbrock



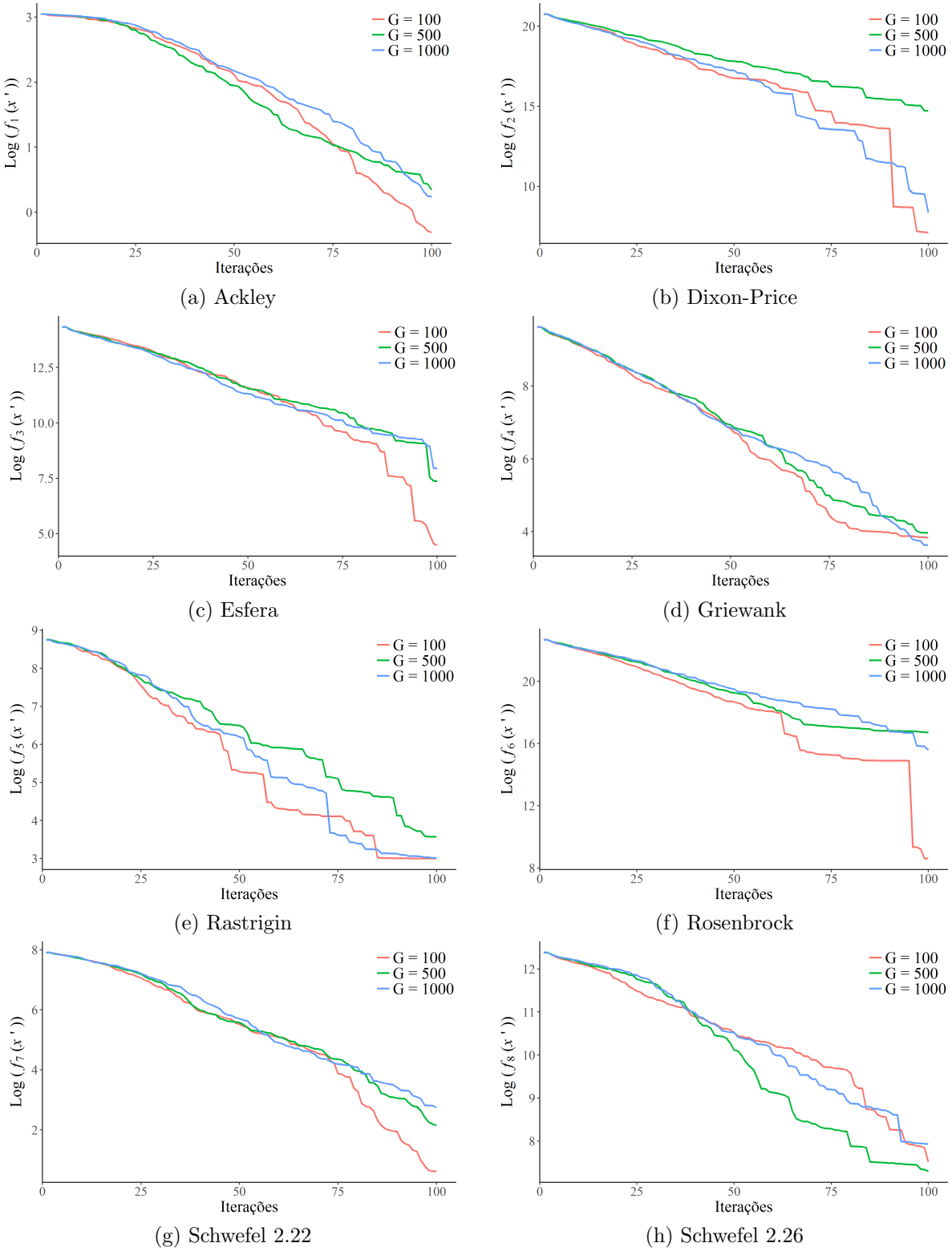
(g) Schwefel 2.22



(h) Schwefel 2.26

Figura 23: Efeito do máximo de iterações para cada função de custo em $d = 60$.

igual a 20, elitismo igual a 2, alfa igual a 0,5 e beta igual a 0,1. No KH, a velocidade de forragem v_f igual a 0,02, velocidade máxima induzida igual a 0,01 e a máxima velocidade de difusão igual a 0.005.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 24: Efeito do máximo de iterações para cada função de custo em $d = 600$.

Para o CSS, o tamanho da memória carregada é definido como 25, o raio da esfera carregada igual a 1, a memória carregada considerando taxa igual a 0,85 e a taxa de escolha de um valor próximo da melhor partícula carregada é definida como 0,15. Define-

se para a técnica GSA uma elite de tamanho e distância entre os dois objetos igual a 1, alfa igual 20, uma constante gravitacional inicial igual 100 e 2% dos agentes aplicam força aos outros. A técnica SLC, foram definidos o número de equipes igual a 4, o número de titulares e reservas igual a 3, a probabilidade de mutação igual a 0,1 e a taxa de mutação igual a 0,2.

4.5.1 Avaliação dos Resultados

Nesta subseção serão mostradas as médias das soluções obtidas com as 30 repetições de cada técnica, ao final das 1000 iterações de cada função objetivo, para a comparação do desempenho do AIPH com as técnicas mostradas no Capítulo 2. Estes resultados servirão para verificação do desempenho de cada uma das técnicas e a identificação das técnicas que se destacam na busca de uma solução ótima global para as 8 funções de custo escolhidas.

Para comparação, consideraram-se as 14 técnicas citadas no início deste capítulo. Foram consideradas 8 funções de teste que aparecem na Seção 4.2.8 e cada experimento com as técnicas reportadas foi repetida 30 vezes com sementes aleatórias diferentes. Para diminuir problemas de imprecisão devido aos arredondamentos, todos os valores iguais ou abaixo de $1e-10$ serão considerados zero. Usou-se a escala logarítmica para a apresentação dos resultados, para reduzir a representação a uma escala mais fácil de ser visualizada e manejada.

Como uma ilustração resumida da diferença de desempenho entre as técnicas, para uma maior explicitação dos resultados, as curvas de convergência apresentadas no Apêndice C possuem a evolução da solução a cada iteração em cada uma das técnicas e também serão usados na avaliação das técnicas.

A partir das soluções encontradas pelas técnicas em um experimento com 10 dimensões, visto na Tabela 4, pode-se observar que o AIPH conseguiu o valor mínimo dentre as técnicas em apenas 3 das 8 funções de custo: Ackley, Esfera e Schwefel 2.22. Neste experimento, a técnica que se destacou em relação as demais foi a SLC, conseguindo o valor mínimo dentre as técnicas em 5 das 8 funções de custo, entretanto, o AIPH conseguiu melhores soluções que o SLC em funções mais difíceis, como a Dixon-Price e Rosenbrock, onde o valor mínimo dentre as técnicas não chegou próximo de $1e-10$. O CS apresentou o pior desempenho dentre as técnicas, apresentando o valor máximo dentre as técnicas em quase todas as funções de custo.

Tabela 4: Resultados das técnicas ao final das iterações com $d = 10$

Técnica	Ackley	Dixon-Price	Esfera	Griewank	Rastrigin	Rosenbrock	Schwefel 2.22	Schwefel 2.26
AIPH	0.0E+00	6.7E-02	0.0E+00	5.2E-03	2.0E+00	2.1E-01	0.0E+00	3.9E+01
AAA	1.0E+01	3.5E+01	1.0E+02	1.6E+00	1.7E+01	2.4E+04	7.3E-02	3.6E+03
ABC	0.0E+00	1.1E-06	0.0E+00	1.6E-03	0.0E+00	1.2E-01	0.0E+00	0.0E+00
BA	1.2E+01	6.2E-01	4.8E+02	1.8E+01	2.7E+01	3.5E+02	5.4E+00	0.0E+00
CS	1.9E+01	2.9E+04	9.5E+03	7.9E+01	9.0E+01	1.1E+07	3.4E+01	3.8E+03
EHO	1.2E-06	7.4E-01	0.0E+00	0.0E+00	0.0E+00	8.7E+00	8.4E-07	1.8E+03
KH	1.2E-06	6.7E-01	2.3E-05	5.2E-02	3.3E+00	8.2E+00	7.5E-06	3.7E+03
PSO	3.2E-01	6.4E-01	0.0E+00	8.2E-02	7.3E+00	3.4E+00	0.0E+00	1.7E+03
WOA	0.0E+00	6.0E-01	0.0E+00	2.2E-02	0.0E+00	4.8E+00	0.0E+00	4.0E+02
CSS	1.8E+00	1.5E-01	0.0E+00	4.5E-02	1.5E-03	9.5E-01	6.1E-03	5.6E-04
GSA	9.1E-10	6.7E-01	0.0E+00	1.8E-02	1.2E+00	5.5E+00	2.2E-09	2.5E+03
VS	3.5E+01	2.1E-09	6.4E-01	0.0E+00	2.6E-01	3.2E+02	0.0E+00	1.8E+01
SCA	0.0E+00	6.7E-01	0.0E+00	2.5E-03	0.0E+00	6.7E+00	0.0E+00	1.8E+03
SLC	0.0E+00	6.7E-01	0.0E+00	0.0E+00	0.0E+00	4.8E+00	0.0E+00	4.2E+02
TLBO	0.0E+00	6.7E-01	0.0E+00	3.4E-04	7.7E-01	4.1E-03	0.0E+00	4.0E+02
Mínimo	0.0E+00	2.1E-09	0.0E+00	0.0E+00	0.0E+00	4.1E-03	0.0E+00	0.0E+00
Máximo	3.5E+01	2.9E+04	9.5E+03	7.9E+01	9.0E+01	1.1E+07	3.4E+01	3.8E+03

Em relação às técnicas bioinspiradas, observando as curvas da Seção C.1 do Apêndice, pode-se ver que as técnicas que se destacaram foram a ABC e EHO, com a ABC apresentando uma solução melhor nas funções Dixon-Price e Rosenbrock e a EHO na função Griewank. Nas técnicas com inspiração nas propriedades da física, a técnica VS se destacou com solução melhor que a vista no AIPH nas funções Dixon-Price e Rosenbrock, enquanto a CSS se destacou nas funções Rastrigin e Schwefel 2.26. As demais técnicas, com inspirações em relações humanas ou princípios da matemática, tem-se a SLC com uma solução melhor que o AIPH nas funções Rastrigin e Griewank, e na TLBO, na função Rosenbrock.

No experimento com 60 dimensões, visto na Tabela 5, pode-se observar que o AIPH conseguiu o valor mínimo dentre as técnicas em mais funções do que no experimento com 10 dimensões, com apenas 6 das 8 funções de custo. A técnica que voltou a se destacar foi a SLC, continuando com a marca de 5 das 8 funções de custo com o valor mínimo dentre as técnicas. Pode-se destacar ainda duas outras técnicas, WOA e TLBO, com 4 das 8 funções de custo com o valor mínimo dentre as técnicas. O CS voltou a apresentar o pior desempenho dentre as técnicas, apresentando o valor máximo dentre as técnicas em quase todas as funções de custo.

Observando as técnicas bioinspiradas a partir das curvas da Seção C.2 do Apêndice, pode-se ver que as técnicas que se destacaram foram a WOA e EHO, com ambas apresentando uma solução melhor na função Rastrigin. Nas técnicas com inspiração nas propriedades da física, a técnica VS se destacou com solução melhor que a vista no AIPH nas funções Rastrigin e Schwefel2.26, enquanto a GSA se destacou nas funções Esfera e Dixon-Price. Nas demais técnicas, com inspirações em relações humanas ou princípios da matemática, pode-se observar que o desempenho do SCA foi o pior dentre as técnicas destas inspirações.

No experimento com 600 dimensões, visto na Tabela 6, pode-se observar que o AIPH conseguiu o valor mínimo dentre as técnicas em todas as 8 funções de custo. A solução encontrada para a função Rastrigin foi melhor do que a solução encontrada para 10 e 60 dimensões. Isto acontece devido ao efeito oriundo do parâmetro adaptativo que determina o fim do ataque (Θ). As técnicas SLC, WOA e TLBO voltaram a se destacar, sendo as técnicas que mais apresentaram soluções equivalentes ao mínimo dentre as funções.

Tabela 5: Resultados das técnicas ao final das iterações com $d = 60$

Técnica	Ackley	Dixon-Price	Esfera	Griewank	Rastrigin	Rosenbrock	Schwefel 2.22	Schwefel 2.26
AIPH	0.0E+00	4.2E-01	0.0E+00	0.0E+00	8.0E+00	2.5E+01	0.0E+00	2.4E+02
AAA	1.7E+01	3.5E+05	1.1E+04	1.2E+02	2.5E+02	2.9E+07	3.0E+01	2.2E+04
ABC	1.2E-01	5.6E+00	7.7E-06	4.3E-03	1.2E+01	6.8E+01	4.1E-03	-
BA	1.4E+01	9.9E-01	1.7E+04	1.9E+02	1.2E+02	1.5E+06	2.1E+02	-
CS	2.1E+01	6.4E+06	1.2E+05	1.1E+03	8.7E+02	4.3E+08	1.0E+10	2.4E+04
EHO	5.7E-06	9.2E-01	1.0E-09	5.3E-09	4.3E-08	5.8E+01	1.9E-04	2.0E+04
KH	1.8E+00	3.5E+00	1.9E-01	3.2E-02	2.5E+01	1.4E+02	3.6E+07	2.3E+04
PSO	4.3E+00	1.2E+01	6.9E+00	5.5E-01	5.3E+01	2.1E+02	3.9E+00	1.4E+04
WOA	0.0E+00	6.7E-01	0.0E+00	1.4E-03	0.0E+00	5.6E+01	0.0E+00	9.9E+02
CSS	1.2E+01	3.4E+03	8.3E+02	1.2E+01	5.3E+01	6.9E+04	9.8E+00	2.9E+02
GSA	2.0E-09	6.8E-01	0.0E+00	1.2E+01	2.0E+01	5.6E+01	2.7E-08	2.1E+04
VS	4.5E+02	2.4E+01	3.6E+00	8.0E-05	1.8E-02	6.3E+03	1.8E+00	1.9E+02
SCA	1.6E+01	7.5E+03	8.2E+01	1.8E+00	7.5E+01	5.9E+05	8.6E-03	1.9E+04
SLC	0.0E+00	6.7E-01	0.0E+00	0.0E+00	0.0E+00	5.8E+01	0.0E+00	2.5E+03
TLBO	0.0E+00	6.7E-01	0.0E+00	0.0E+00	4.3E+00	4.7E+01	0.0E+00	1.0E+04
Mínimo	0.0E+00	4.2E-01	0.0E+00	0.0E+00	0.0E+00	2.5E+01	0.0E+00	1.9E+02
Máximo	4.5E+02	6.4E+06	1.2E+05	1.1E+03	8.7E+02	4.3E+08	1.0E+10	2.4E+04

Verifica-se que, dentre as 8 funções de custo, o TLBO apresentou uma estagnação até G = 500. O CS e o VS apresentaram os piores desempenhos dentre as técnicas.

Observando as técnicas bioinspiradas a partir das curvas da Seção C.3 do Apêndice, pode-se ver que a técnica que se equipara ao AIPH é o WOA, em 5 funções de custo. Nas técnicas com inspiração nas propriedades da física, a técnica VS, apesar de 4 funções com o valor máximo dentre as funções, destacou-se com solução melhor que a vista nas demais técnicas desta inspiração nas funções Rosenbrock, Schwefel 2.22 e Schwefel 2.26. Nas demais técnicas, com inspirações em relações humanas ou princípios da matemática, pode-se observar que o desempenho do TLBO e SLC equiparam-se ao AIPH em quase todas as funções de custo.

4.5.2 Desempenho Geral das Técnicas

Uma comparação do desempenho geral das técnicas deve considerar uma métrica que combine as soluções encontradas nas diferentes funções de custo em um único indicador e que não sofra efeito da escala destas soluções.

Para uma comparação geral das técnicas será usado um método que considere a ordenação das soluções de cada técnica para cada função de custo. Desta forma, propõe-se a criação de um *score* para cada técnica, que combina os resultados obtidos nas 8 funções de custo testadas, conforme Equações 125 e 126.

$$\Lambda_d(t) = \sum_{f=1}^{8} \lambda_d^f(t), \tag{125}$$

onde $\Lambda_d(t)$ é o score geral da técnica t, na dimensão d, para a função f, com f representa uma das 8 funções de custo e possui valores de 1 a 8 e d podendo assumir os valores de 10, 60 ou 600. Já o indicador λ_d^f é o score da técnica na dimensão d para uma função f e é calculado a partir da Equação 126:

$$\lambda_d^f(t) = \frac{\mathcal{O}_d^f(t)}{max(\mathcal{O}_d^f)} \times 100. \tag{126}$$

onde o indicador $\mathcal{O}_d^f(t)$ é a ordem da solução encontrada pela técnica t, na dimensão d, para a função f e $max(\mathcal{O}_d^f)$ é a ordem máxima das técnicas na função f com d dimensões. Consiste em uma ordenação ascendente que organiza as soluções encontradas por ordem do primeiro ao último por cada técnica em cada função de custo testada, podendo assumir valores de 1 a 15. Desta forma, o primeiro colocado foi a técnica com a solução mais

Tabela 6: Resultados das técnicas ao final das iterações com $d = 600$

Técnica	Ackley	Dixon-Price	Esfera	Griewank	Rastrigin	Rosenbrock	Schwefel 2.22	Schwefel 2.26
AIPH	0.0E+00	6.3E-01	0.0E+00	0.0E+00	0.0E+00	5.1E+02	0.0E+00	2.4E+03
AAA	1.9E+01	1.5E+08	4.9E+05	4.3E+03	5.2E+03	1.2E+09	-	2.3E+05
ABC	1.9E+01	2.1E+08	4.7E+05	4.3E+03	4.4E+03	1.5E+09	-	-
BA	1.5E+01	3.8E+06	2.3E+05	2.2E+03	9.9E+02	1.1E+08	3.8E+79	-
CS	2.1E+01	8.3E+08	1.3E+06	1.2E+04	9.6E+03	5.6E+09	1.0E+10	2.5E+05
EHO	9.3E-05	1.0E+00	2.5E-06	5.2E-07	2.5E-05	6.0E+02	8.1E-03	2.3E+05
KH	4.6E+00	3.5E+04	5.7E+03	5.2E+01	1.8E+03	2.0E+05	-	2.5E+05
PSO	1.1E+01	6.4E+05	2.7E+04	2.6E+02	2.8E+03	4.3E+06	4.1E+02	1.7E+05
WOA	0.0E+00	6.7E-01	0.0E+00	3.2E-03	0.0E+00	5.9E+02	0.0E+00	3.6E+03
CSS	1.4E+01	2.3E+07	1.2E+05	1.0E+03	1.1E+03	3.2E+08	-	1.4E+04
GSA	5.2E+00	1.0E+05	1.4E+04	5.7E+03	1.3E+03	4.5E+05	-	2.4E+05
VS	1.5E+08	1.4E+34	1.7E+07	1.3E+05	1.1E+03	1.1E+05	1.9E+01	4.5E+03
SCA	2.0E+01	2.3E+08	1.8E+05	1.6E+03	1.4E+03	1.8E+09	8.2E+01	2.3E+05
SLC	0.0E+00	6.7E-01	0.0E+00	0.0E+00	0.0E+00	6.0E+02	0.0E+00	1.2E+05
TLBO	6.0E-01	6.7E-01	0.0E+00	0.0E+00	0.0E+00	5.9E+02	0.0E+00	1.8E+05
Mínimo	0.0E+00	6.3E-01	0.0E+00	0.0E+00	0.0E+00	5.1E+02	0.0E+00	2.4E+03
Máximo	1.5E+08	1.4E+34	1.7E+07	1.3E+05	9.6E+03	5.6E+09	3.8E+79	2.5E+05

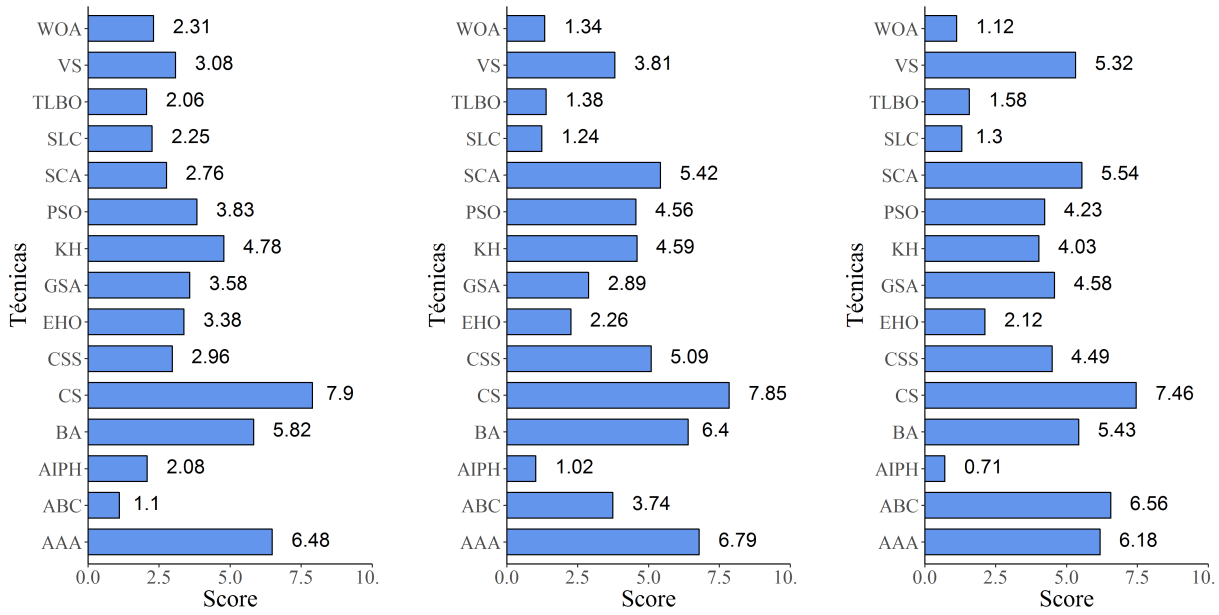
próxima de zero. Feito isso, usa-se esta ordenação como uma pontuação de cada técnica e normaliza-se esta pontuação dividindo-a pela maior pontuação encontrada. Esta normalização é feita pois, em algumas funções de custo, há empate nas soluções e consequente ocupação da mesma posição por mais de uma técnica, fazendo com que a última posição não necessariamente seja a de número 15.

A interpretação do indicador $\Lambda_d(t)$ considera que todas as funções de custo tem o mesmo peso para desempenho geral e que, quanto menor for o valor do *score da técnica*, melhor foi o desempenho da técnica. Para os casos citados na Subseção 4.5.1, onde as técnicas não tiveram capacidade de conversão nas funções Schwefel 2.22 e Schwefel 2.26, será dado a última posição da ordenação, de número 15.

Observando-se na Figura 25 o desempenho geral do uso do AIPH e as demais técnicas para busca de uma solução ótima global. No cenário de 10 dimensões, as técnicas ABC e TLBO mostraram-se melhores que o AIPH, com o ABC com um grande distanciamento em relação ao segundo colocado. Entretanto, com o aumento de dimensionalidade, o AIPH começa a se destacar.

Com o cenário de 60 dimensões o AIPH apresenta o melhor desempenho e, dentre as técnicas testadas, a SLC, WOA e TLBO apresentaram scores com valores próximos ao AIPH. Já no cenário de 600 dimensões, o AIPH continua com desempenho superior às demais técnicas, neste caso, com o score do AIPH com um grande distanciamento em relação ao segundo colocado. Percebe-se que os scores da SLC, WOA e TLBO são os mais próximos o score do AIPH.

Os resultados obtidos a partir da comparação com outras técnicas da literatura ratificam a eficiência do AIPH em problemas de alta dimensionalidade. Já o desempenho do AIPH em baixas dimensionalidades foi satisfatório, contudo, menos competitivo em relação as demais técnicas analisadas.



(a) d=10 (b) d=60 (c) d=600

Figura 25: Scores das técnicas.

4.6 Considerações Finais

Neste capítulo foram realizadas as comparações com 14 técnicas de otimização apresentadas no Capítulo 2 usando como *benchmark* 8 funções de custo. Observou-se que o AIPH apresentou soluções satisfatórias e se destacou em relação a muitas das técnicas analisadas e equiparando-se a técnicas que apresentam bom desempenho, como SLC, TLBO e WOA.

O estudo possibilitou verificar a dispersão encontrada nas soluções encontrados nas 30 repetições realizadas, apresentando o impacto que resultados atípicos tempo no cálculo do desempenho médio do AIPH .Além disso, pode-se notar que há sensibilidade da técnica AIPH à escolha de um tamanho de revoada baixo, entretanto, em muitos casos equipara-se ao desempenho de tamanho de revoada maiores, necessitando apenas de mais iterações para convergir.

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

NESTE capítulo são apresentadas as principais conclusões acerca do conceito, implementação e análise da nova técnica de otimização global proposta neste trabalho, bem como seu desempenho em comparação com outras técnicas encontradas na literatura. Adicionalmente, são apresentadas algumas possíveis aprimoramentos, assim como algumas possibilidades de trabalhos futuros.

5.1 Conclusões

Esta dissertação apresentou a proposta de uma nova técnica metaheurística de otimização global inspirado no filme Os Pássaros dirigido por Alfred Hitchcock, chamado AIPH, para problemas de otimização contínua, onde as variáveis podendo assumir qualquer número real dentro de um intervalo de variação.

O AIPH se diferencia das demais técnicas por possuir uma inspiração em algo não natural, oriundo de um universo ficcional distópico. A partir de elementos tirados do filme de Hitchcock, foi possível combinar conceitos tirados da estatística e biblioteconomia para o desenvolvimento de uma técnica cujo os parâmetros tivessem uma ajuste dinâmico durante o processo de busca. Outro diferencial é a proposta de uma técnica que tenha um desempenho superior em problemas de otimização de alta dimensionalidade.

O trabalho foi dividido em quatro partes. Inicialmente foram apresentados alguns conceitos importantes a cerca de problemas e algoritmos de otimização. Em seguida, foram levantados trabalhos relacionados ao tema proposto, estudados para a avaliação de outros algoritmos de otimização global que serviram para comparação com AIPH, para fundamentar e evidenciar a relevância da nova técnica proposta. Em seguida, apresenta-

se a concepção e detalhamento sobre o funcionamento da técnica idealizada. A quarta parte, apresentou as funções de custo que serão testadas, avalia os efeitos da escolha de diferentes tamanhos de revoada e de máximo de iterações no desempenho do AIPH, além de uma comparação com outras técnicas da literatura.

No Capítulo 4 foi possível observar que o AIPH apresenta uma dispersão alta nas soluções encontradas nas 30 repetições, provocada pelas piores soluções, mesmo que haja uma concentração de repetições com soluções satisfatórias. Verificou-se que essa dispersão é ainda maior quando o cenário é de alta dimensionalidade, como o caso de 600 dimensões. Devido à probabilidade do AIPH "falhar" no processo de convergência para o ótimo global, evidenciou-se que é indispensável a realização de várias repetições do algoritmo.

A avaliação do tamanho de revoada mostrou que o AIPH não apresenta bons resultados quando usa uma quantidade baixa de pássaros, como o caso de um tamanho de revoada de 10 pássaros. A qualidade da solução aumenta com o um número maior de pássaros e diminui a necessidade de um máximo de iterações maior. Observou-se que um tamanho de revoada razoável para eficiência do AIPH pode ser de 100 pássaros e foi a quantidade usada para os demais testes do Capítulo 4.

A análise do máximo de iterações abordou uma quantidade suficiente de iterações para um bom desempenho do algoritmo, bem como o efeito dessa quantidade no parâmetro adaptativo W do AIPH, comparando com o SCA e o VS, que possuem igual parâmetro. A partir das curvas de convergência vistas no Apêndice B, pode-se observar que há uma necessidade visível de uma quantidade maior de iterações em cenário de maior dimensionalidade e em funções complexas, como Rosenbrock e Schwefel 2.26 de 600 dimensões, entretanto, o aumento de iterações não traz uma melhoria perceptível da solução na maioria dos casos, pois o AIPH possui um comportamento de convergência mais rápido que as técnicas SCA e VS. Na seção de comparação com outras técnicas da literatura, decidiu-se usar um máximo de 1000 iterações para todas as técnicas terem uma quantidade razoável de iterações para alcançar uma solução satisfatória.

Com o objetivo de mostrar o desempenho da técnica AIPH, é comparado com outros quatorze algoritmos metaheurísticos, através de oito funções de custo em 10, 60 e 600 dimensões. Os resultados mostram que a técnica AIPH é capaz de encontrar soluções satisfatórias e eficientes, ao obter resultados superiores ou equivalentes aos vistos em técnicas já consolidadas na literatura. Apesar de apresentar um desempenho pior no cenário

de 10 dimensões, o AIPH foi capaz de oferecer a característica de escalabilidade, principal objetivo proposto, uma vez que seu desempenho não foi prejudicado pelo aumento da dimensionalidade das funções de custo testadas.

Desta forma, a partir dos resultados encontrados, pode-se dizer que a técnica alcançou o objetivo relativo à eficiência, uma vez que apresentou desempenho geral melhor na maioria dos experimentos, mesmo em funções mais complexas como Schwefel 2.22 e Schwefel 2.26 de 600 dimensões.

5.2 Trabalhos Futuros

Os trabalhos futuros consistem na continuidade do processo de pesquisa, aprimoramento e aplicações à problemas diversos. Inicialmente, será criado e publicada uma biblioteca para a linguagem R e, em seguida, uma implementação para *Matlab*®. Também serão desenvolvidas versões do AIPH para problemas de otimização combinatória e para problemas multiobjetivo.

Para continuidade das análises de desempenho do AIPH em relação à outras técnicas, espera-se realizar a comparação com as técnicas bioinspiradas citadas no Capítulo 2 e que não foram consideradas neste trabalho, bem como suas formas alternativas. As técnicas novas que forem criadas posteriormente à publicação do AIPH, também poderão ser usados para análises comparativas. As funções de custo pertencentes ao *Congress on Evolutionary Computation* (CEC), que oferece funções de custo de grande complexidade para competição entre técnicas de otimização, serão usadas para verificação do desempenho do AIPH em futuras análises.

Como propostas de aprimoramento, pode-se realizar a investigação de soluções para mitigar a probabilidade de "falha" no processo de convergência, bem como a influência e possível substituição da equação definida para determinação do fim do ataque. Testar outras equações para o definir o movimento de ataque dos pássaros, outras formas de inicialização das posições dos pássaros e o uso de operadores evolutivos, como a mutação, integrados a este algoritmo para melhorar seu desempenho. Poderá ser testado também outras equações para a determinação do fim do ataque e quantidade de pássaros na elite. A investigação de soluções para mitigar a probabilidade de "falha" no processo de convergência também serão realizados.

Além disso, o AIPH pode ser hibridado com outros algoritmos no campo da otimização estocástica para melhorar seu desempenho. Por fim, a investigação da aplicação e contribuição do AIPH à problemas das áreas de engenharia, logística, administração, economia, entre outras ciências.

REFERÊNCIAS

AKAY, B.; KARABOGA, D. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, v. 192, p. 120 – 142, 2012. Swarm Intelligence and Its Applications.

BASTOS, E. A. *Otimizaçao de Seções Retangulares de Concreto Armado Submetidas à Flexo-Compressao Oblíqua Utilizando Algoritmos Genéticos.* Dissertao (Mestrado) — Universidade Federal do Rio de Janeiro, 2004.

BECCENERI, J. C. *Meta-heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais.* São José dos Campos, SP.: Computação e Matemática Aplicada às Ciências e Tecnologias Espaciais. INPE., 2008.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 26, n. 1, p. 29–41, 1996.

DOAN, B.; ÖLMEZ, T. A new metaheuristic for numerical function optimization: Vortex search algorithm. *Information Sciences*, v. 293, p. 125 – 145, 2015.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on.* Nagoya, Japan: IEEE, 1995. p. 39–43.

ELDOS, R. A. Q. T. On the performance of the gravitational search algorithm. *International Journal of Advanced Computer Science and Applications*, v. 4, n. 8, p. 74–78, 2013.

ERLICH, I.; RUEDA, J. L. Solving the cec2016 real-parameter single objective optimization problems through mvmo-phm. *2016 IEEE Congress on Evolutionary Computation*, 2016.

EVANS, M. et al. *Statistical Distributions: 3rd ed.* New York, USA: Wiley, 2000.

FILHO, C. J. A. B. et al. A novel search algorithm based on fish school behavior. In: *2008 IEEE International Conference on Systems, Man and Cybernetics*. Singapore: IEEE, 2008. p. 2646–2651.

FRANçA, F. O. *Algoritmos Bio-Inspirados aplicados à Otimização Dinâmica.* Dissertao (Mestrado) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas - FEEC/Unicamp, 2005.

FRIEDLANDER, A. Elementos de programação não-linear. Editora da UNICAMP, 1994.

GANDOMI, A. H.; ALAVI, A. H. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, v. 17, n. 12, p. 4831 – 4845, 2012.

GONZALEZ-LONGATT, F. et al. Mean variance mapping optimization for the identification of gaussian mixture model: Test case. In: *6th IEEE International Conference Intelligent Systems*. Sofia, Bulgaria: IEEE, 2012. v. 2, p. 158–163.

HAESER, G.; GOMES-RUGGIERO, M. Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global. In: *TEMA Tend. Mat. Apl. Comput.* Campinas, SP: SBMAC, 2008. v. 9, n. 3, p. 395–404.

HITCHCOCK, A. *The Birds*. United States: Universal Studios, 1963.

HOLLAND, J. H. Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing*, v. 2, p. 88–105, 06 1973.

IZMAILOV A. F., S. M. V. Newton-type methods for optimization and variational problems. *Springer series in operations research and financial engineering.*, Springer, 2014.

JAMIL, M.; YANG, X. A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2013.

KAHNEMAN, D. *Rápido e Devagar: duas formas de pensar*. Rio de Janeiro, Brasil: Objetiva, 2012.

KAVEH, A.; TALATAHARI, S. A novel heuristic optimization method: charged system search. *Acta Mechanica*, v. 213, n. 3, p. 267–289, 2010.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings IEEE International Conference on Neural Networks*. Perth, WA, Australia: IEEE, 1995. v. 4, p. 1942–1948.

KNUTH, D. E. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1997.

KWOKA, J. The herfindahl index in theory and practice. *Antitrust Bull*, 1985.

LACERDA, M. G. P. Uma nova metaheurística de segregação de cardumes para a otimização multi-solução de problemas multimodais. 2012.

LANDMANN R., E. R. H. Uma abordagem heurística para a programação da produção na indústria de fundição com utilização da lógica fuzzy. *Gestão e Produção*, scielo, v. 18, 2011.

MATSUMOTO, E. Y. *MATLAB 6.5 - Fundamentos de Programação*. São Paulo, Brasil: Editora Érica, 2002.

MILLONAS, M. M. Swarms,phase transitions and collective intelligence. *Artificial Life III*, 1994.

MIRJALILI, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, v. 27, n. 4, p. 1053–1073, 2016.

MIRJALILI, S. Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, v. 96, p. 120 – 133, 2016.

MIRJALILI, S.; LEWIS, A. The whale optimization algorithm. *Advances in Engineering Software*, v. 95, p. 51 – 67, 2016.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. *Advances in Engineering Software*, v. 69, p. 46 – 61, 2014.

MOOSAVIAN, N.; ROODSARI, B. Soccer league competition algorithm, a new method for solving systems of nonlinear equations. In: *International Journal of Intelligence Science*. Mashhad, Iran: SCIRP, 2014. v. 4, n. 1, p. 7–16.

MORAIS R. G.; DIAS, D. M. Previsão de séries temporais do mercado de ações utilizando programação genética com inspiração quântica. 2015.

MOREIRA, E. T. Diversificação econômica: Análise da estrutura setorial das microrregiões do sul do brasil, 2002/2010. *VII Seminário Internacional sobre Desenvolvimento Regional*, 2015.

OLIVIERI, B. *Otimização de Projeto de Pontes Protendidas Pré-Moldadas pelo Método dos Algoritmos Genéticos*. Dissertao (Mestrado) — Universidade Federal do Rio de Janeiro/COPPE, 2004.

PREMALATHA, K.; BALAMURUGAN, R. A nature inspired swarm based stellar-mass black hole for engineering optimization. In: *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. Coimbatore, India: IEEE, 2015. p. 1–8.

RAO, R.; SAVSANI, V.; VAKHARIA, D. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, v. 43, n. 3, p. 303 – 315, 2011.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Gsa: A gravitational search algorithm. *Information Sciences*, v. 179, n. 13, p. 2232 – 2248, 2009.

REZENDE, M. O uso de métodos heurísticos na alocação de recursos defensivos em cenários táticos. 2014.

SILVA, D.; MAROLDI, A.; LIMA, L. Outliers na lei do elitismo. *Revista da Faculdade de Biblioteconomia e Comunicação da UFRGS*, 2014.

STEWART, J. *Cálculo*. São Paulo, Brasil: Editora Pioneira Thomson Learning, 2006.

SUCUPIRA, I. R. Métodos heurísticos genéricos: Metaheurística e hiper-heurísticas. 2004.

TEODORO, F. R.; PARPINELLI, R. S.; LOPES, H. S. Otimização por colônia de bactérias aplicada a problemas de engenharia. In: *Mecánica Computacional Vol XXIX*. Buenos Aires, Argentina: Asociación Argentina de Mecánica Computacional, 2010. p. 9615–9627.

UYMAZ, S. A.; G., T.; YEL, E. Artificial algae algorithm for nonlinear global optimization. *Applied Soft Computing*, v. 31, p. 153 – 171, 2015.

WANG, G. G.; DEB, S.; COELHO, L. d. S. Elephant herding optimization. In: *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*. Bali, Indonesia: IEEE, 2015. p. 1–5.

YANG, X.-S. Firefly algorithms for multimodal optimization. In: *Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009*. Berlin, Heidelberg: Springer, 2009. v. 5, p. 26–28.

YANG, X.-S. Harmony search as a metaheuristic algorithm. In: *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Berlin, Heidelberg: Springer, 2009.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms: Second Edition*. United Kingdom: Luniver Press, 2010.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Berlin, Heidelberg: Springer, 2010. p. 65–74.

YU, J. J.; LI, O. V. A social spider algorithm for global optimization. *Applied Soft Computing*, v. 30, p. 614 – 627, 2015.

APÊNDICE A – Resultados para Teste de Tamanho de Revoada

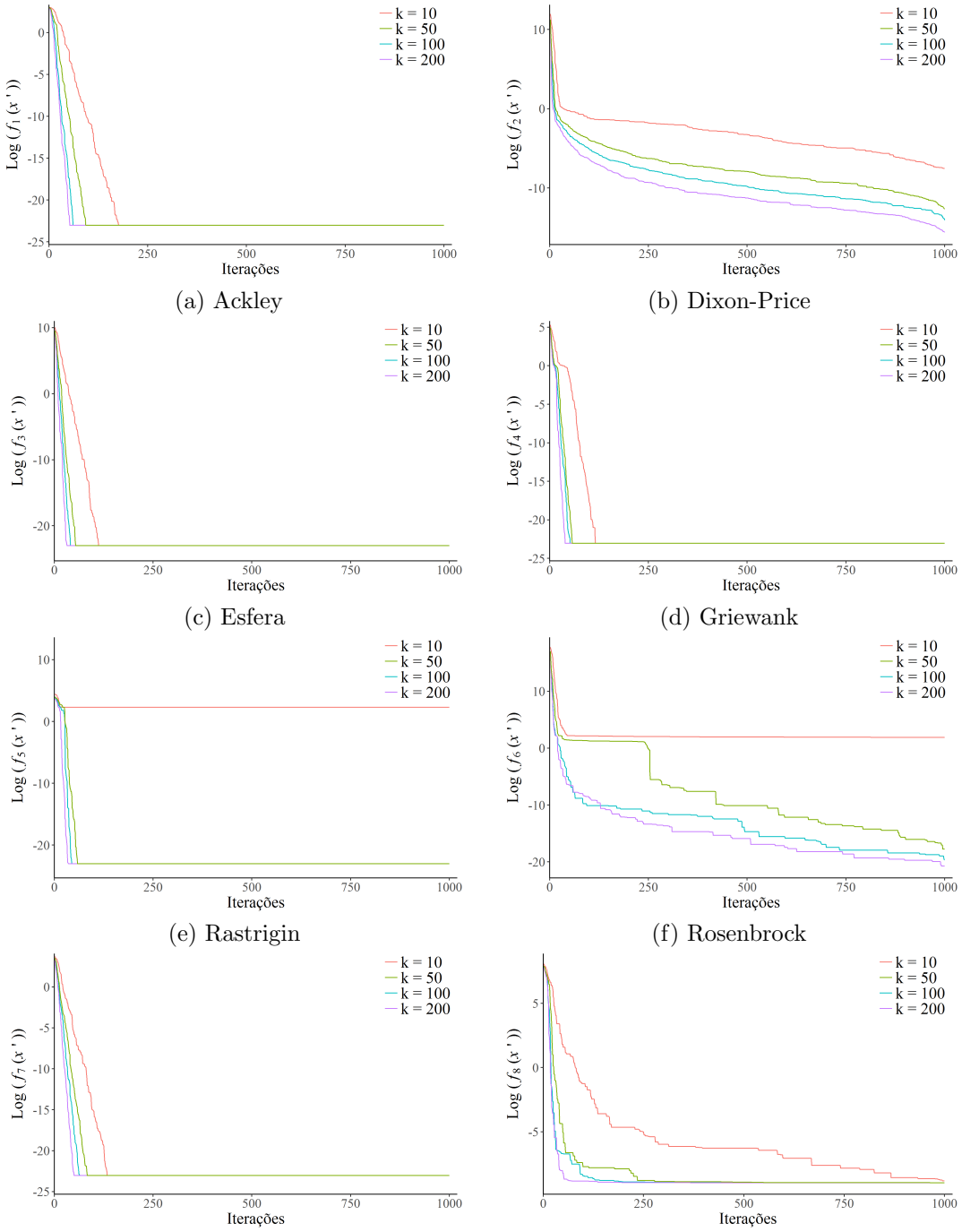
Este apêndice apresenta os resultados dos experimentos feitos para a análise do efeito do tamanho da revoada para a evolução da solução no decorrer das iterações. Estes resultados são relacionados com a análise feita na Subseção 4.4.2 do Capítulo 4.

Cada experimento é composto por 30 execuções da técnica, com o objetivo de obter uma quantidade suficiente de observações para o cálculo de estatísticas mais robustas, onde as curvas representam as médias dos resultados das repetições para cada um dos tamanhos de revoada testados. Foram consideradas 8 funções de custo que aparecem na Seção 4.3. As possibilidades testadas para tamanho de revoada são 10, 50, 10 e 200 pássaros, com um máximo de 1000 iterações.

O apêndice dividi-se em 3 seções, com cada uma representando um cenário de dimensionalidade. As Figuras apresentadas descrevem o melhor resultado em média encontrado a cada iteração pelo AIPH por cada tamanho de revoada, nas respectivas funções objetivo.

A.1 Experimento com 10 Dimensões

Os resultados apresentados nesta seção correspondem às figuras relacionadas com a verificação da configuração de tamanho de revoada considerando um experimento com 10 dimensões feita para análise na Seção 4.4.2 do Capítulo 4.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

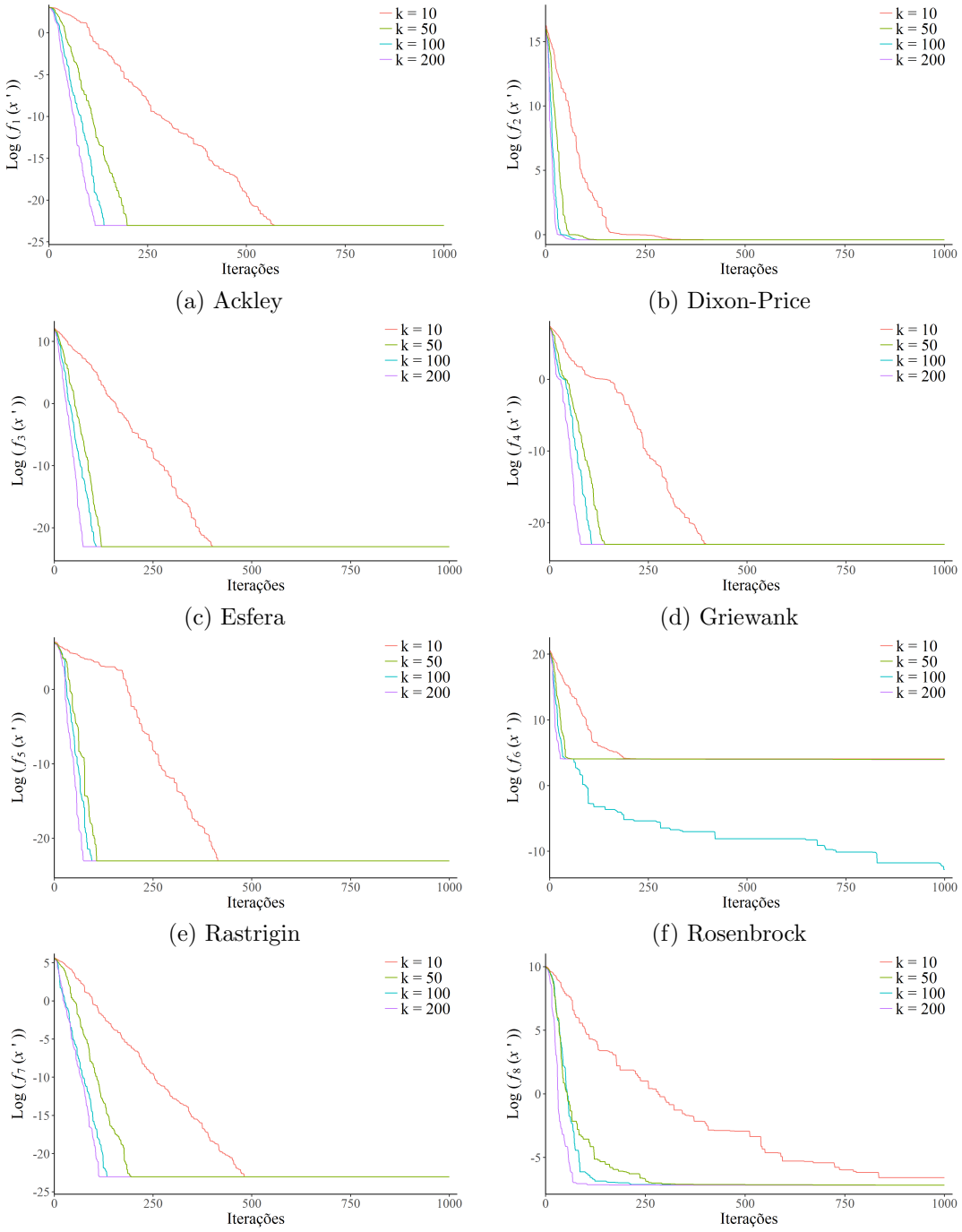
(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 26: Efeito do tamanho da revoada em $d = 10$.

A.1 Experimento com 60 Dimensões

Os resultados apresentados nesta seção correspondem às figuras relacionadas com a verificação da configuração de tamanho de revoada considerando um experimento com 60 dimensões feita para análise na Seção 4.4.2 do Capítulo 4.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

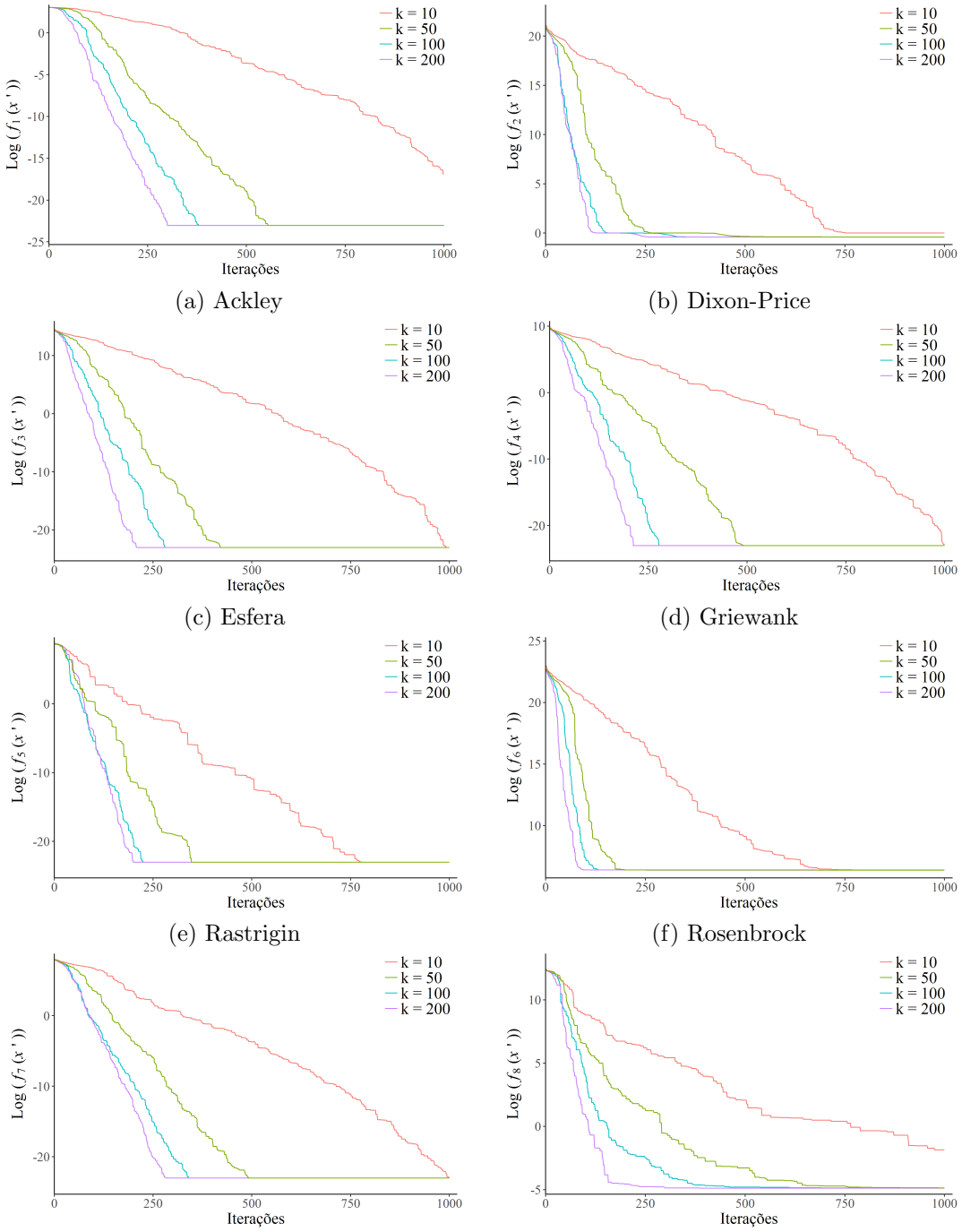
(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 27: Efeito do tamanho da revoada em $d = 60$.

A.1 Experimento com 600 Dimensões

Os resultados apresentados nesta seção correspondem às figuras relacionadas com a verificação da configuração de tamanho de revoada considerando um experimento com 600 dimensões feita para análise na Seção 4.4.2 do Capítulo 4.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 28: Efeito do tamanho da revoada em $d = 600$.

APÊNDICE B – Resultados para Teste de Máximo de Iteração

Neste apêndice serão apresentados os resultados dos experimentos feitos para a análise do efeito da quantidade de iterações para o alcance de uma solução satisfatória. Estes resultados são relacionados com a análise feita na Subseção 4.4.3 do Capítulo 4.

Cada experimento é composto por 30 execuções da técnica, com o objetivo de obter uma quantidade suficiente de observações para o cálculo de estatísticas mais robustas, onde as curvas representam as médias dos resultados das repetições para cada uma das técnicas testadas. Foram consideradas 8 funções de teste que aparecem na Seção 4.3. As possibilidades testadas para quantidade máxima de iterações são 100, 500 e 1000, utilizando-se duas outras técnicas, SCA e VS, para comparação do comportamento da evolução de resultados ao longo das iterações. A escolha destas técnicas deve-se ao fato de possuírem parâmetros dinâmicos dependentes da escolha da quantidade máxima de iterações, bem como o AIPH.

O apêndice dividi-se em 3 seções, com cada uma representando uma quantidade máxima de iterações em cenários 10, 60 e 600 dimensões. As Figuras apresentadas descrevem o melhor resultado em média encontrado a cada iteração pelo AIPH, SCA e VS, nas respectivas funções objetivo.

B.1 Experimento com 100 Iterações

Os resultados apresentados nesta seção correspondem às figuras relacionadas com a verificação da configuração de quantidade máxima de iterações considerando um experimento com 100 iterações em 10, 60 e 600 dimensões, respectivamente, feita para análise na Seção 4.4.3 do Capítulo 4.

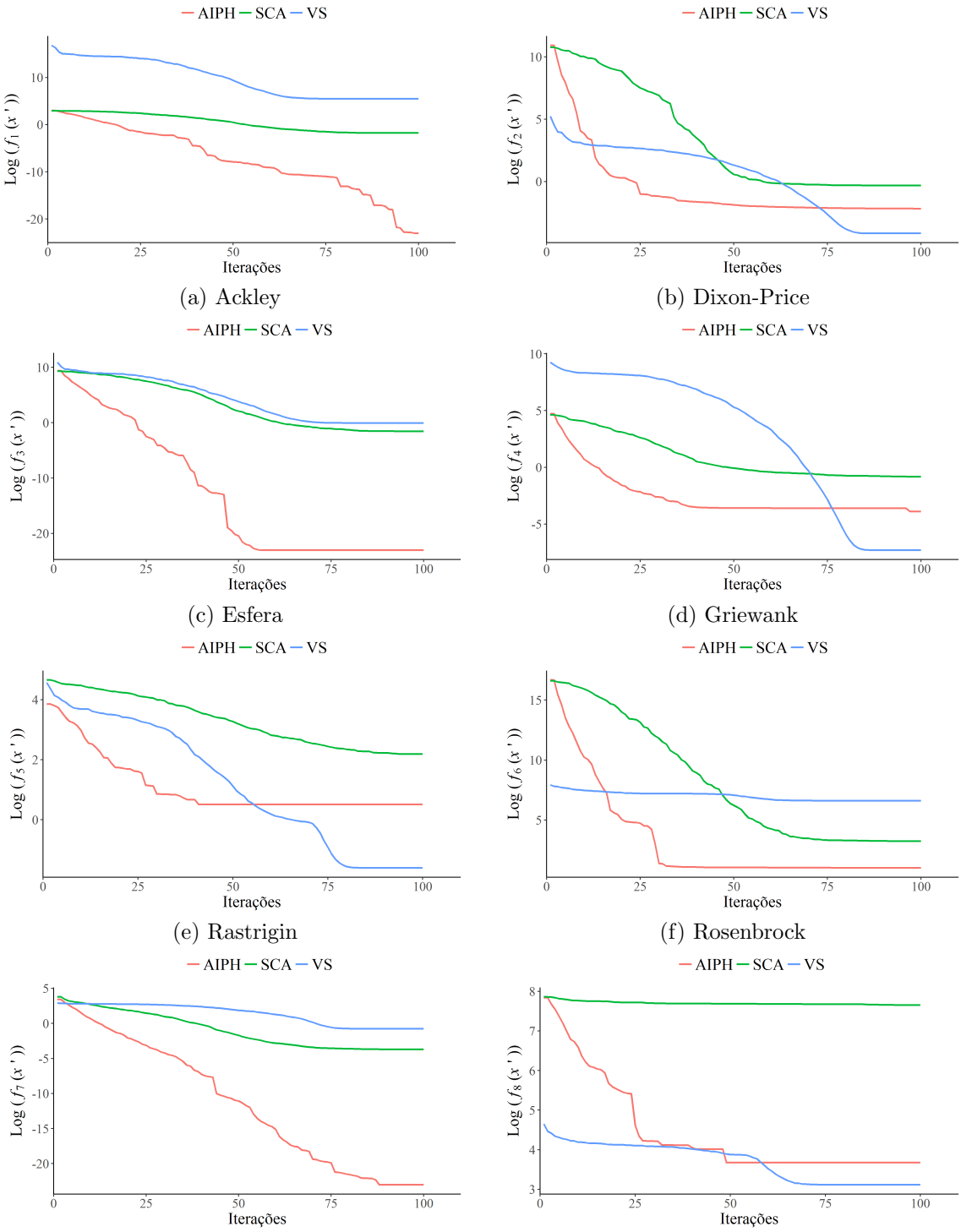


Figura 29: Curvas de convergência para $G = 100$ e $d = 10$.

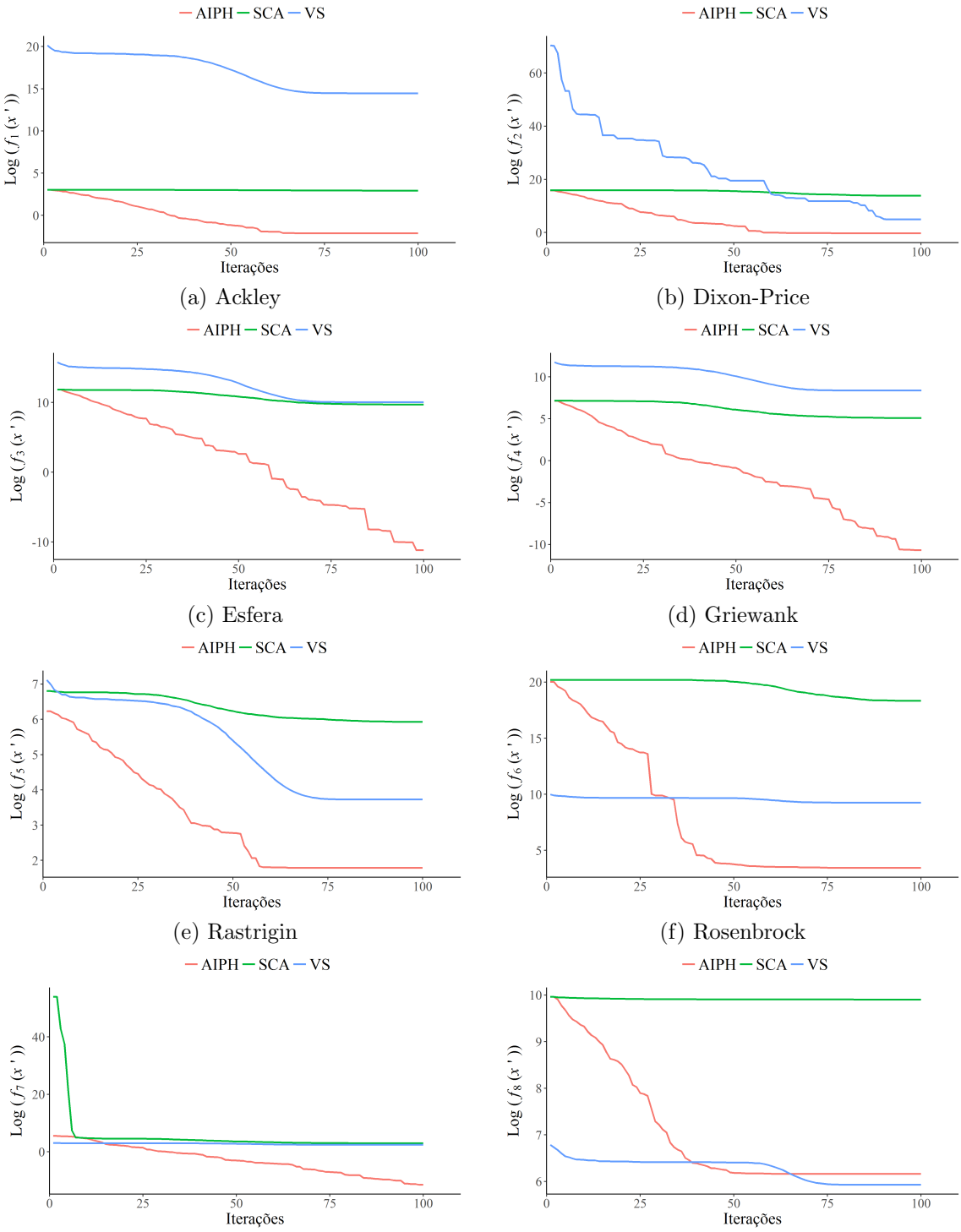


Figura 30: Curvas de convergência para $G = 100$ e $d = 60$.

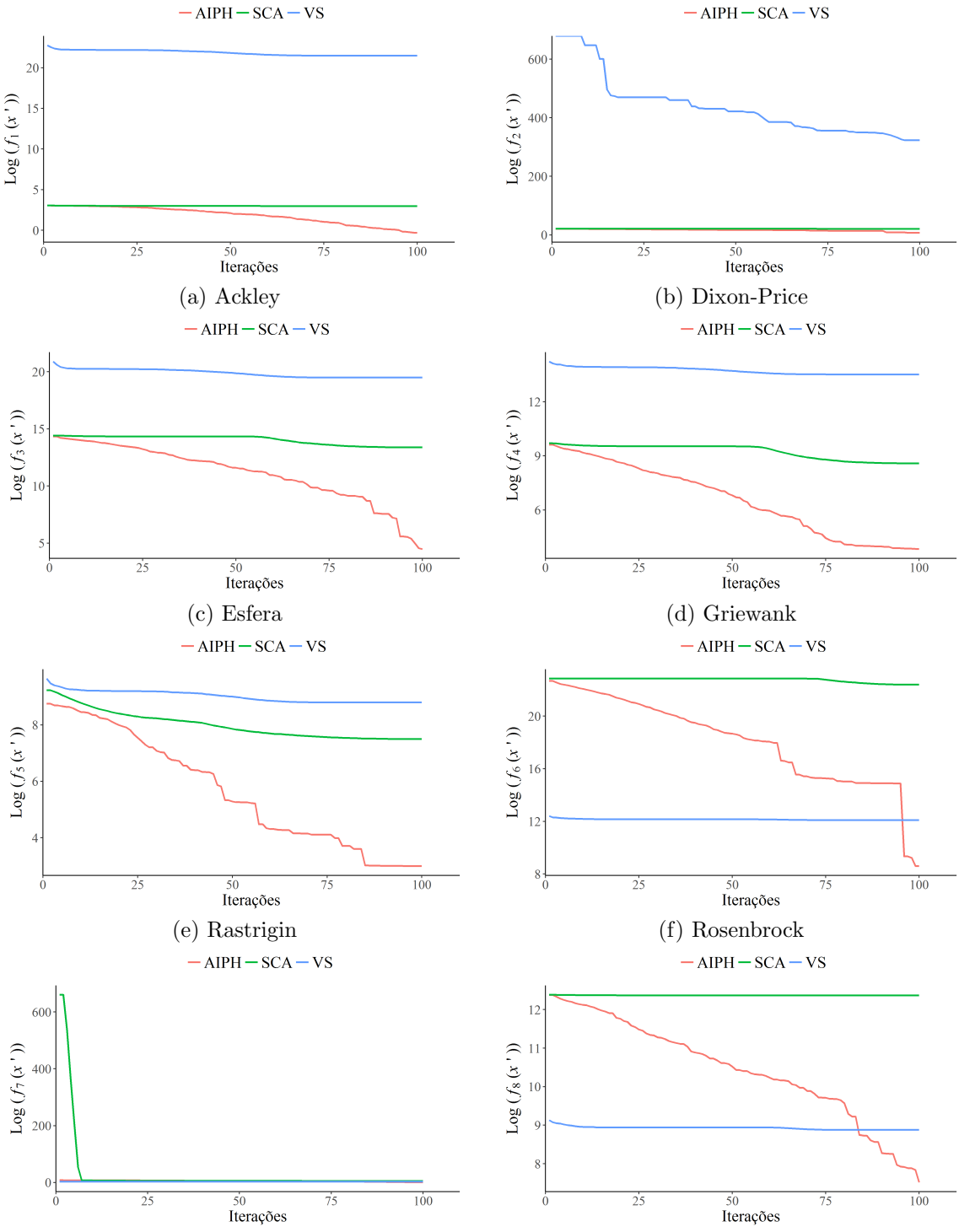
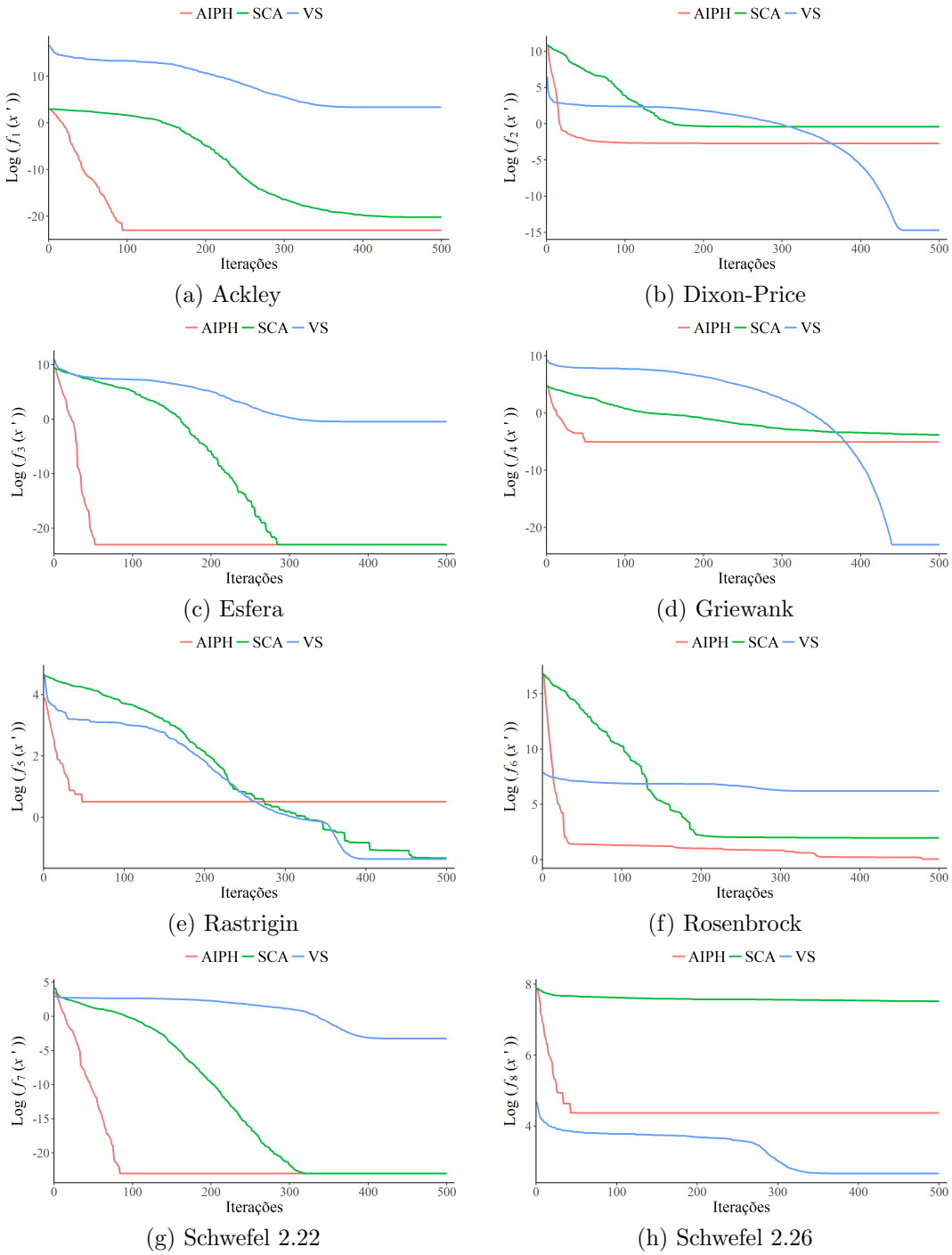


Figura 31: Curvas de convergência para $G = 100$ e $d = 600$.

B.2 Experimento com 500 Iterações

Nesta seção tem-se às figuras relacionadas com a verificação da configuração de quantidade máxima de iterações considerando um experimento com 500 iterações em 10, 60 e 600 dimensões, respectivamente, feita para análise na Seção 4.4.3 do Capítulo 4.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 32: Curvas de convergência para $G = 500$ e $d = 10$.

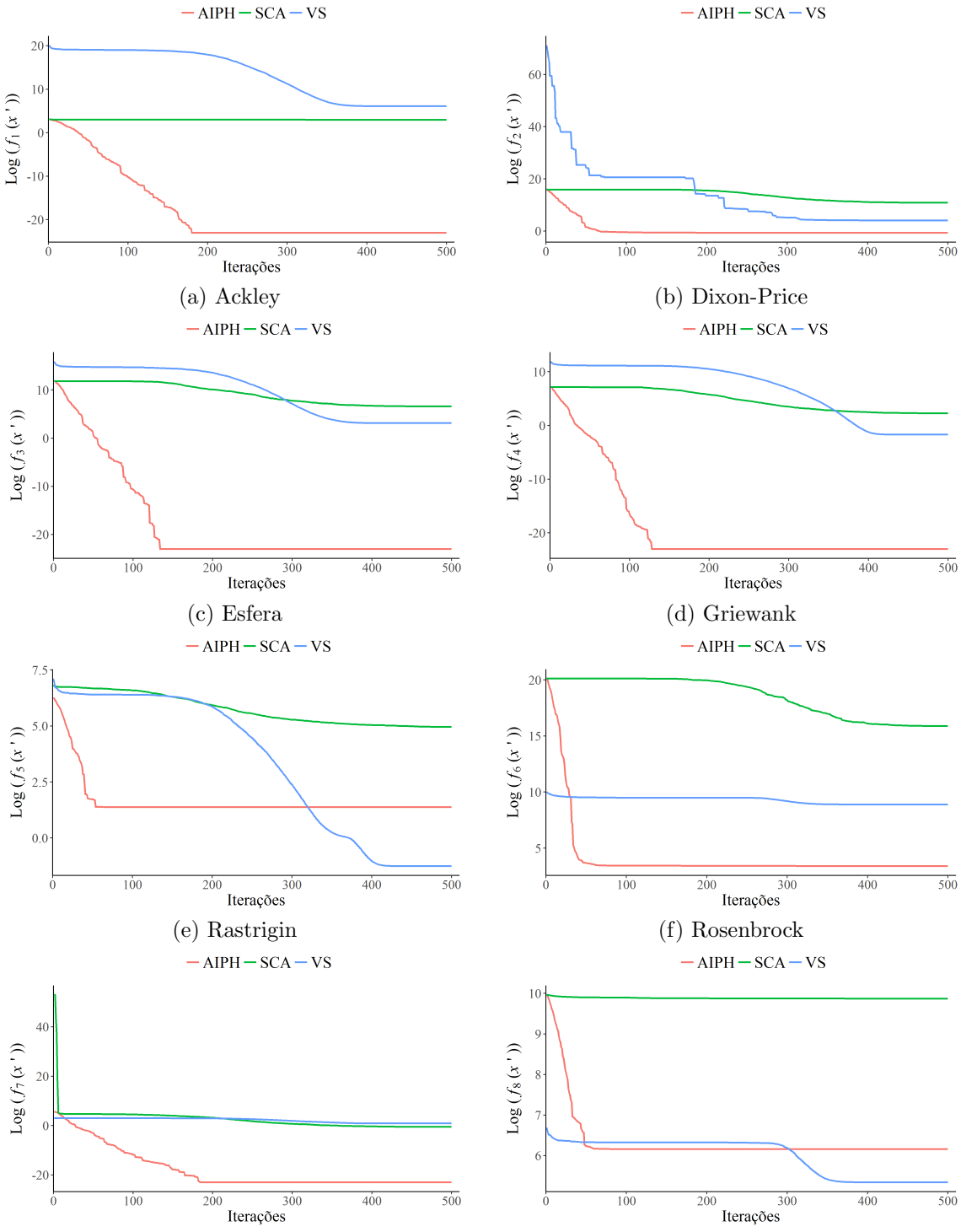


Figura 33: Curvas de convergência para $G = 500$ e $d = 60$.

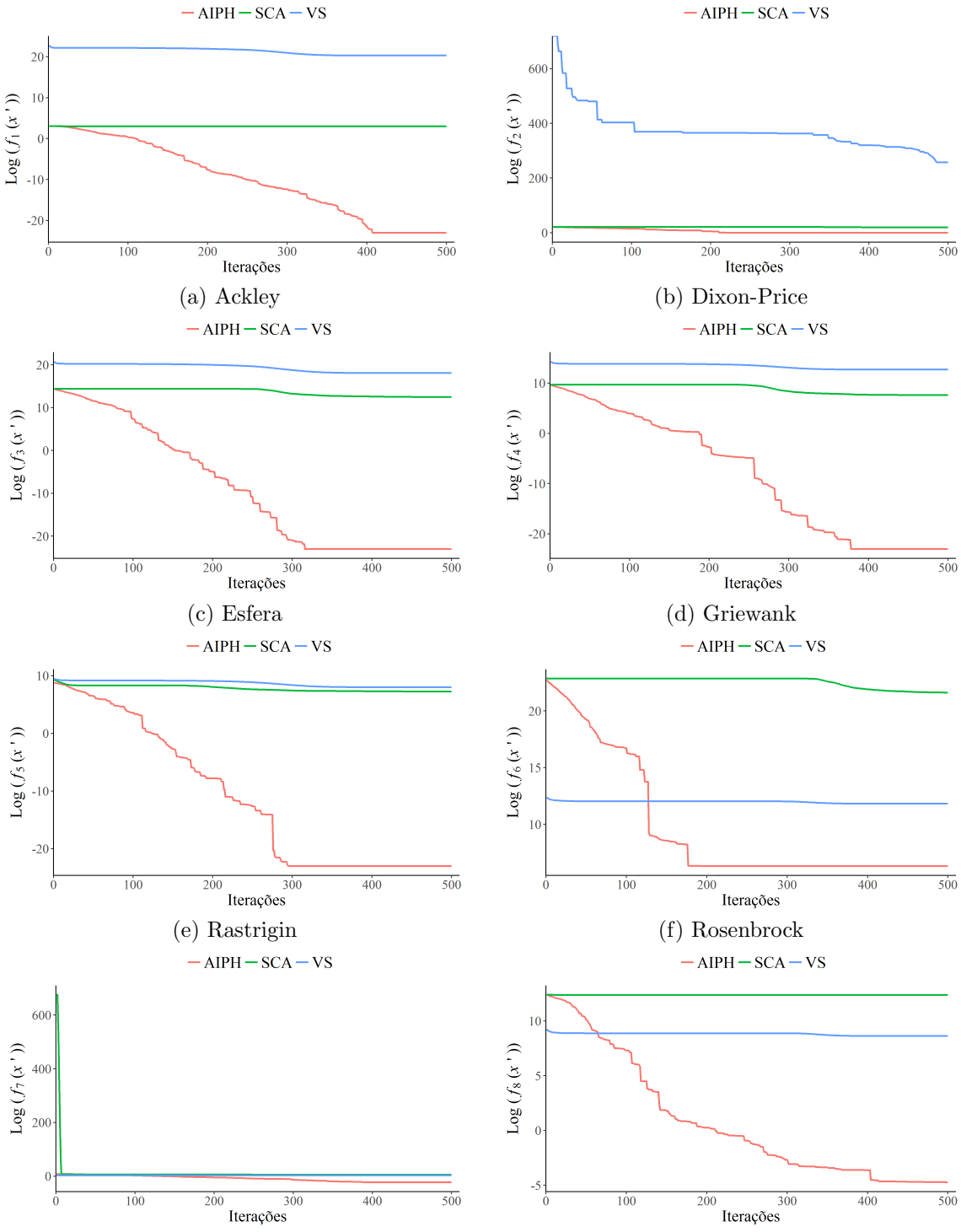
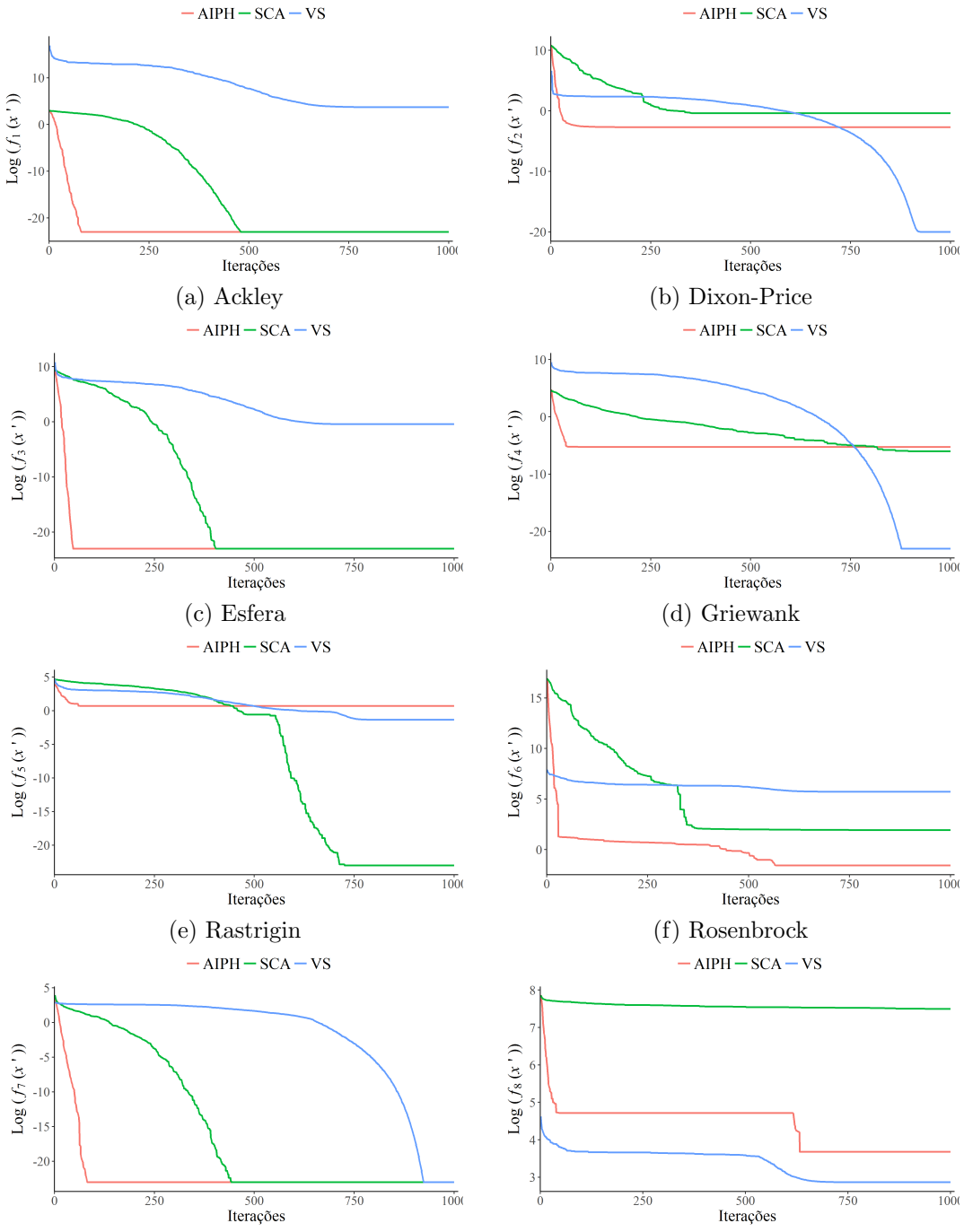


Figura 34: Curvas de convergência para $G = 500$ e $d = 600$.

B.3 Experimento com 1000 Iterações

As figuras desta seção referem-se à verificação da configuração de quantidade máxima de iterações considerando um experimento com 1000 iterações em 10, 60 e 600 dimensões, respectivamente, feita para análise na Seção 4.4.3 do Capítulo 4.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 35: Curvas de convergência para $G = 1000$ e $d = 10$.

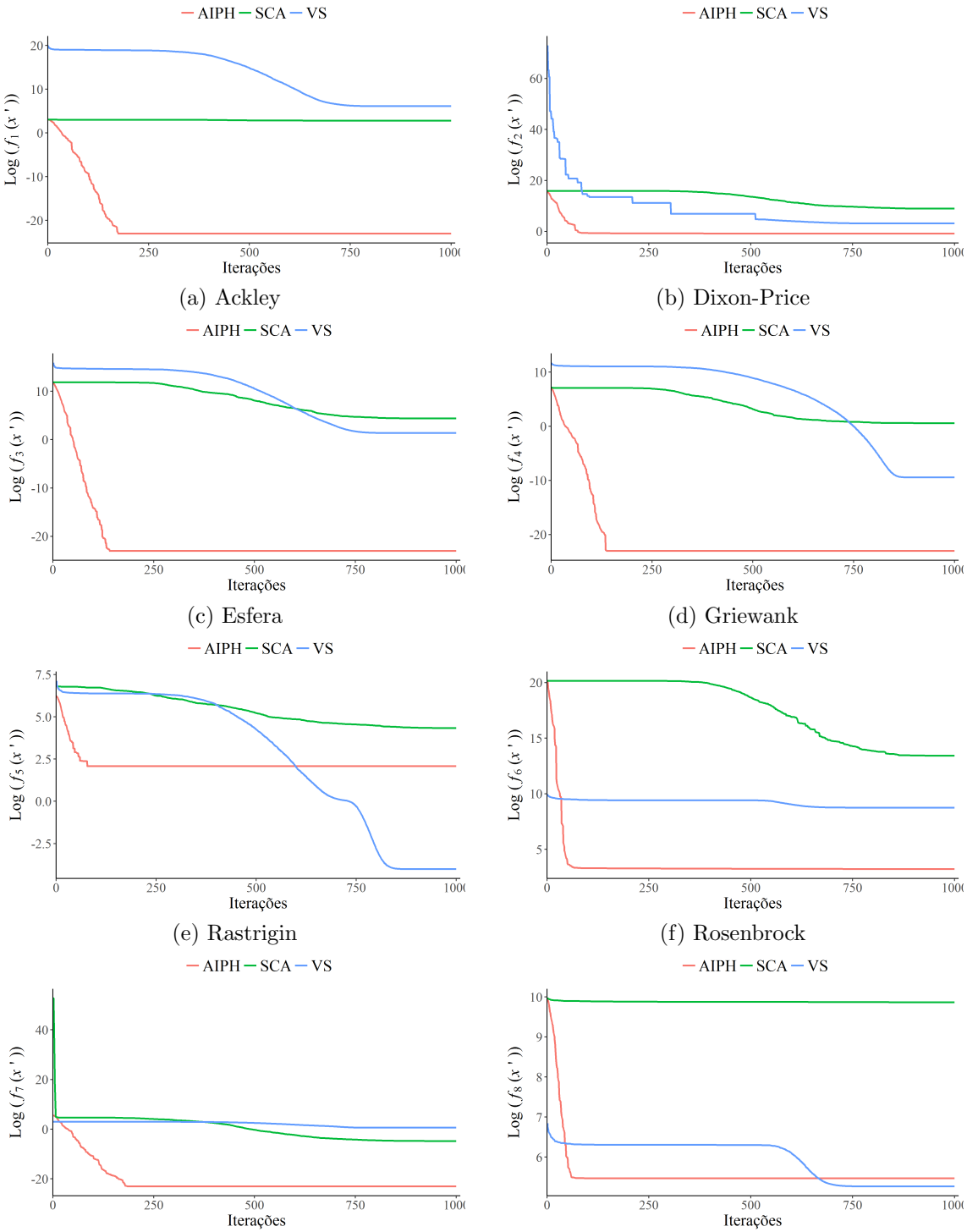


Figura 36: Curvas de convergência para $G = 1000$ e $d = 60$.

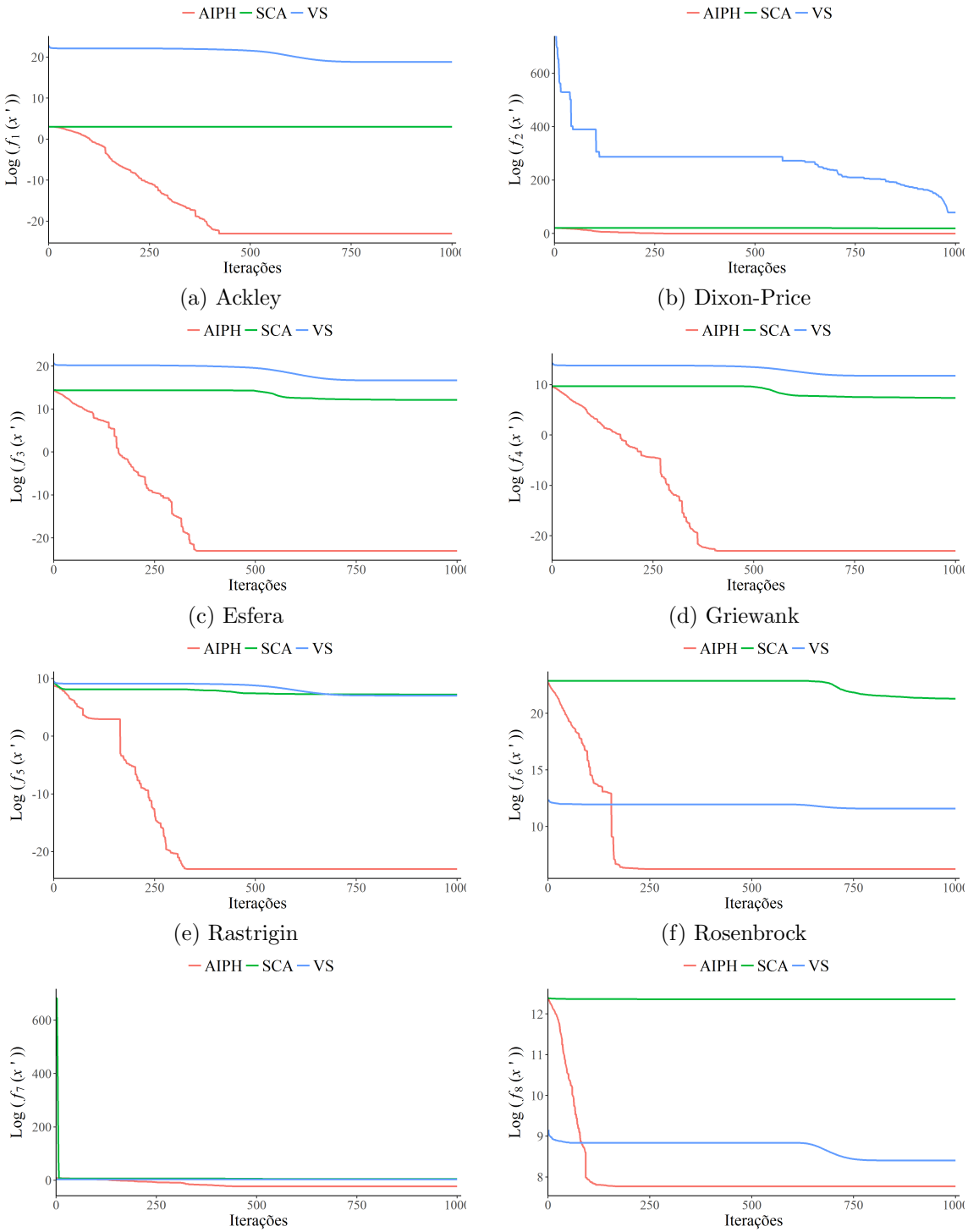


Figura 37: Curvas de convergência para $G = 1000$ e $d = 600$.

APÊNDICE C – Resultados da Comparação com Outras Técnicas

Neste apêndice serão apresentadas as curvas de convergência dos experimentos realizados para a comparação do desempenho do AIPH com algumas das técnicas mostradas no Capítulo 2. Estes resultados são relacionados com a análise feita na Subseção 4.5 do Capítulo 4.

Cada experimento é composto por 30 execuções da técnica, com o objetivo de obter uma quantidade suficiente de observações para o cálculo de estatísticas mais robustas, onde as curvas representam as médias dos resultados das repetições de cada uma das técnicas citadas no início da Seção 4.5. Foram consideradas 8 funções de teste que aparecem na Seção 4.3. Para comparação, consideraram-se 8 técnicas bioinspiradas, 3 de propriedades da física, 2 de relações humanas e 1 de princípios da matemática, que são: AAA, ABC, BA, CS, EHO, PSO, KH, WOA, CSS, GSA, VS, SLC, TLBO e SCA.

O apêndice dividi-se em 3 seções, com cada uma representando um tipo de inspiração em cenários de 10, 60 e 600 dimensões, com máximo de 1000 iterações. As Figuras apresentadas descrevem os resultados em média encontrado a cada iteração pelo AIPH e as demais técnicas, nas respectivas funções objetivo.

C.1 Experimento com 10 Dimensões

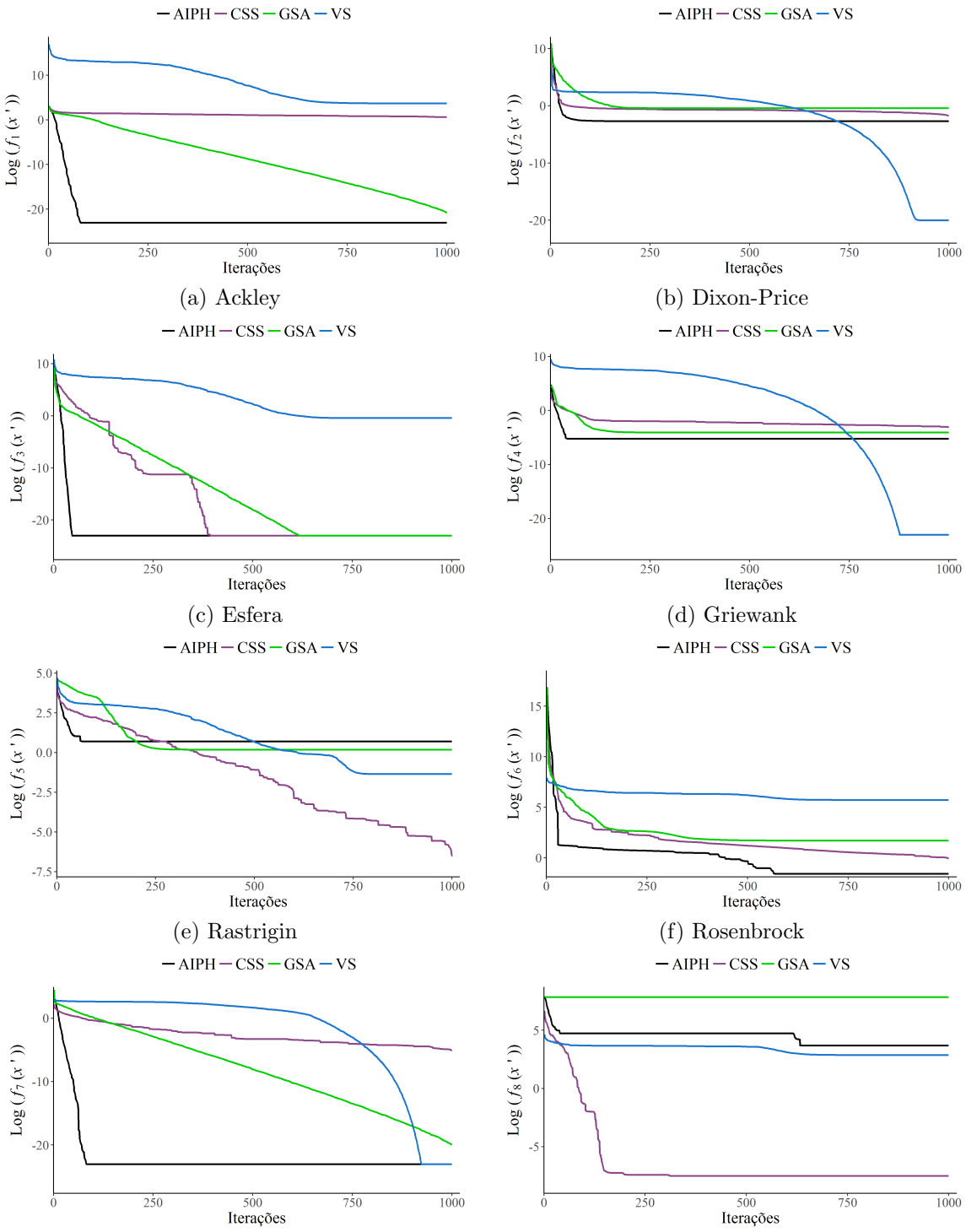
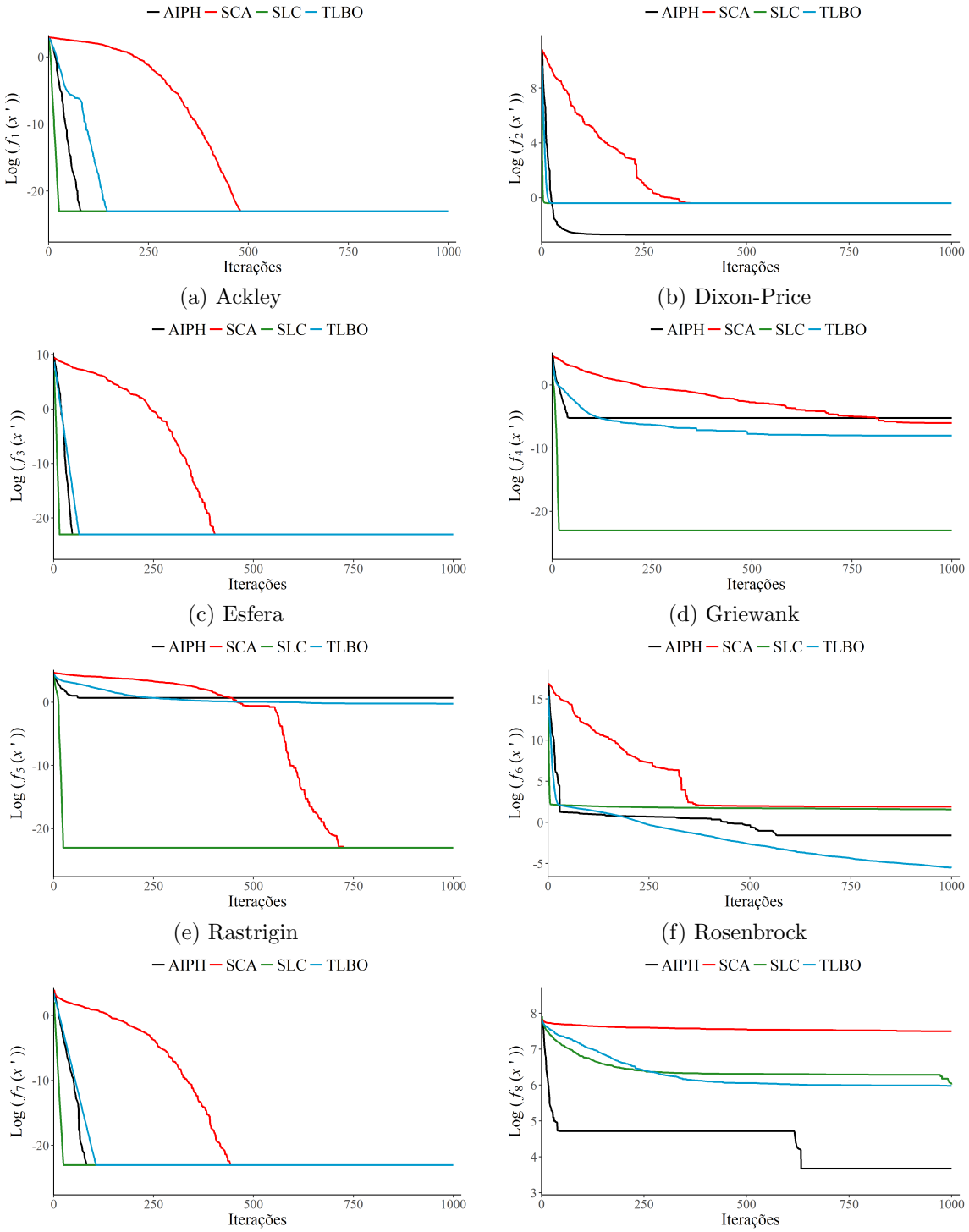


Figura 38: Comparação com técnicas inspiradas em propriedades físicas com cenário de 10 dimensões.



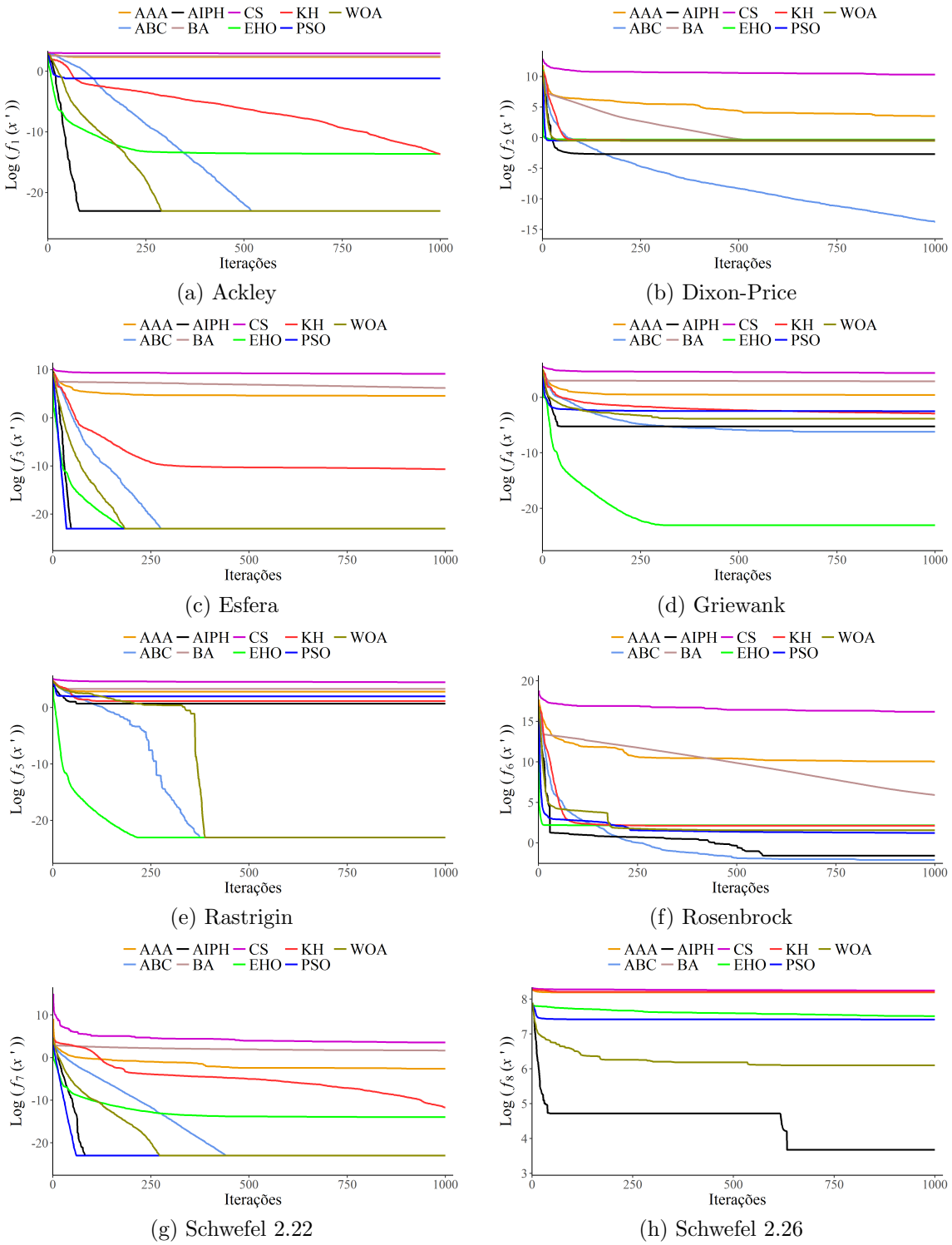
(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 39: Comparação com técnicas inspirada em princípios da matemática ou relações humanas com cenário de 10 dimensões.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 40: Comparação com técnicas bioinspiradas com cenário de 10 dimensões

C.2 Experimento com 60 Dimensões

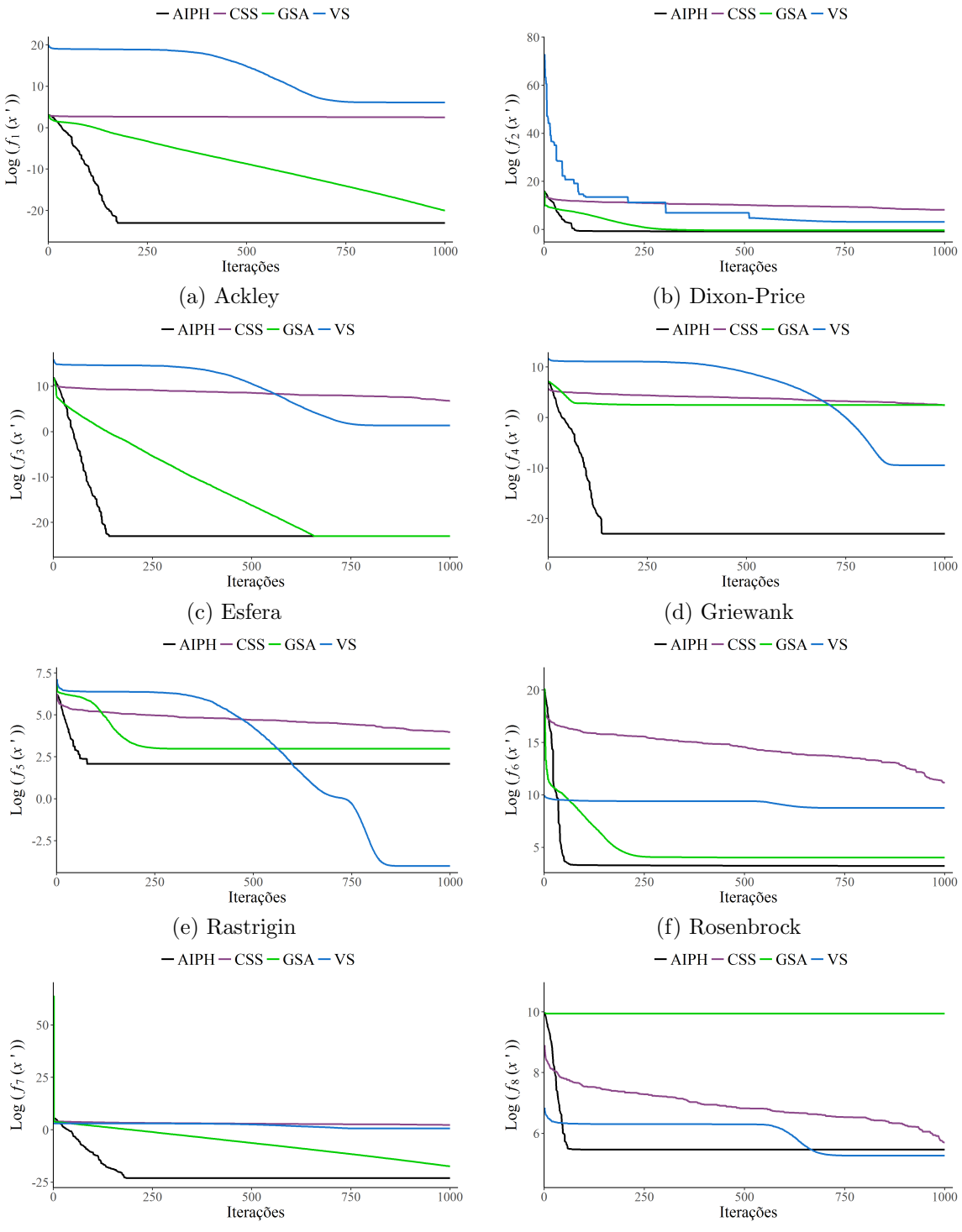
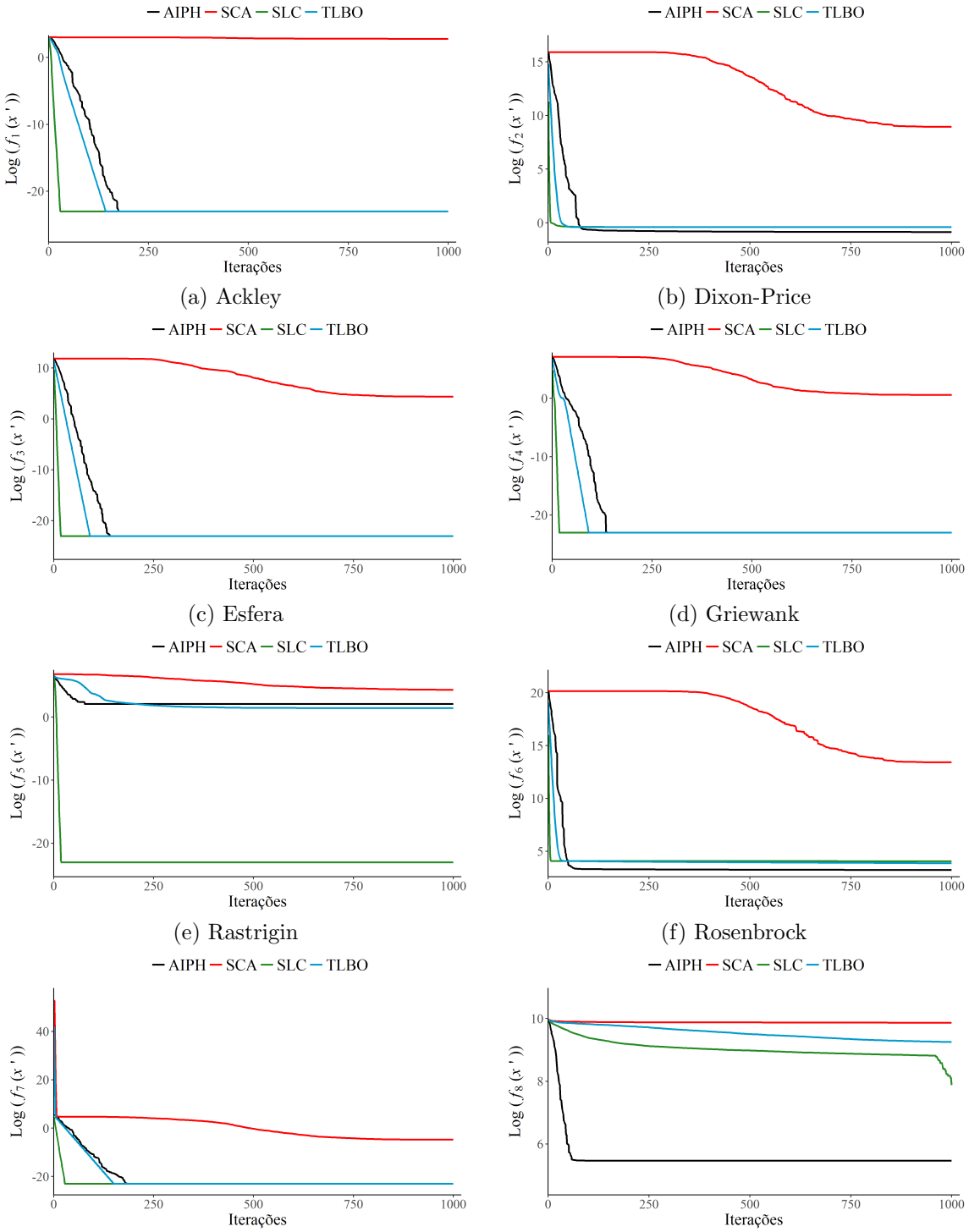


Figura 41: Comparação com técnicas inspiradas em propriedades físicas com cenário de 60 dimensões.



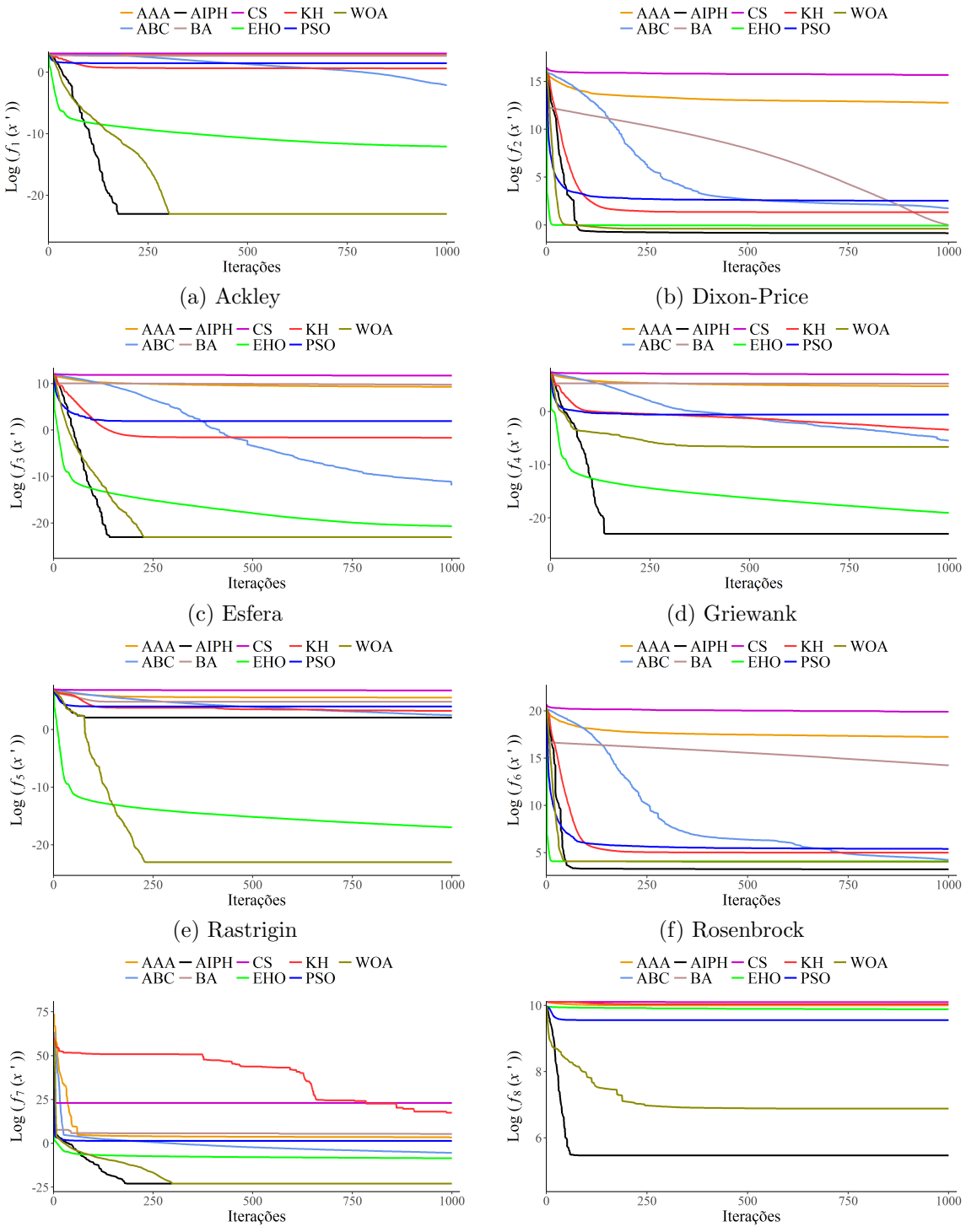
(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 42: Comparação com técnicas inspirada em princípios da matemática ou relações humanas com cenário de 60 dimensões.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 43: Comparação com técnicas bioinspiradas com cenário de 60 dimensões.

C.3 Experimento com 600 Dimensões

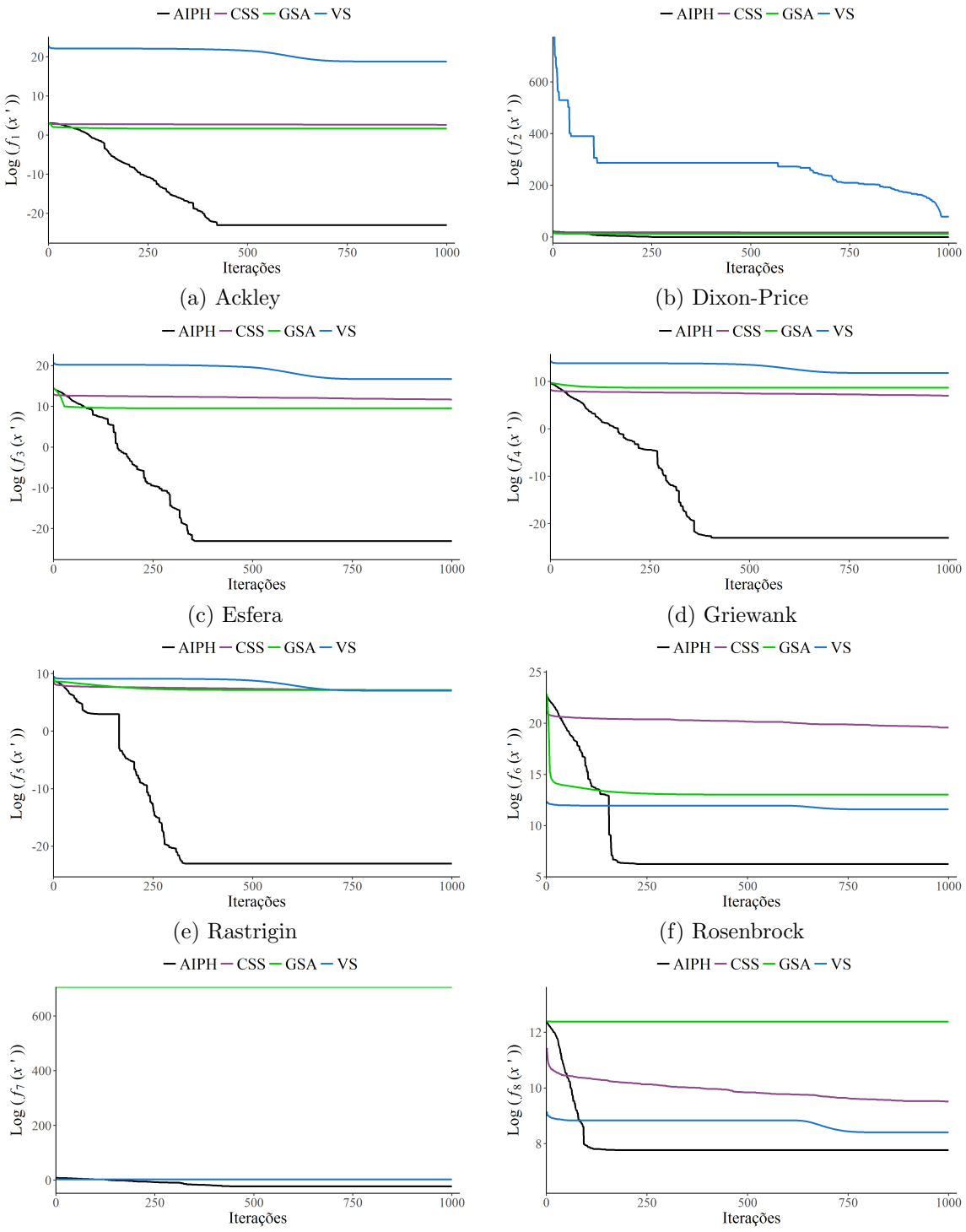


Figura 44: Comparação com técnicas inspiradas em propriedades físicas com cenário de 600 dimensões.

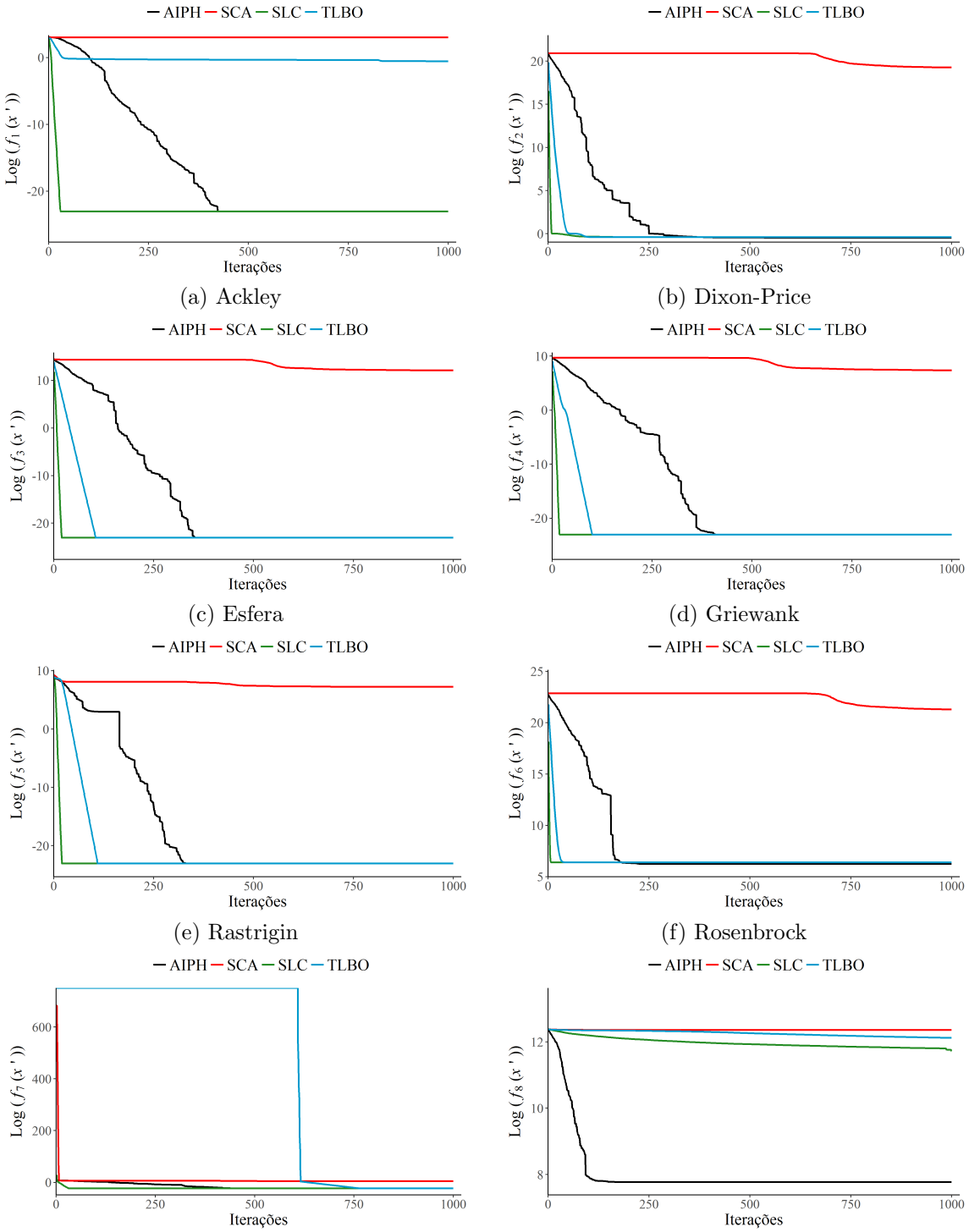
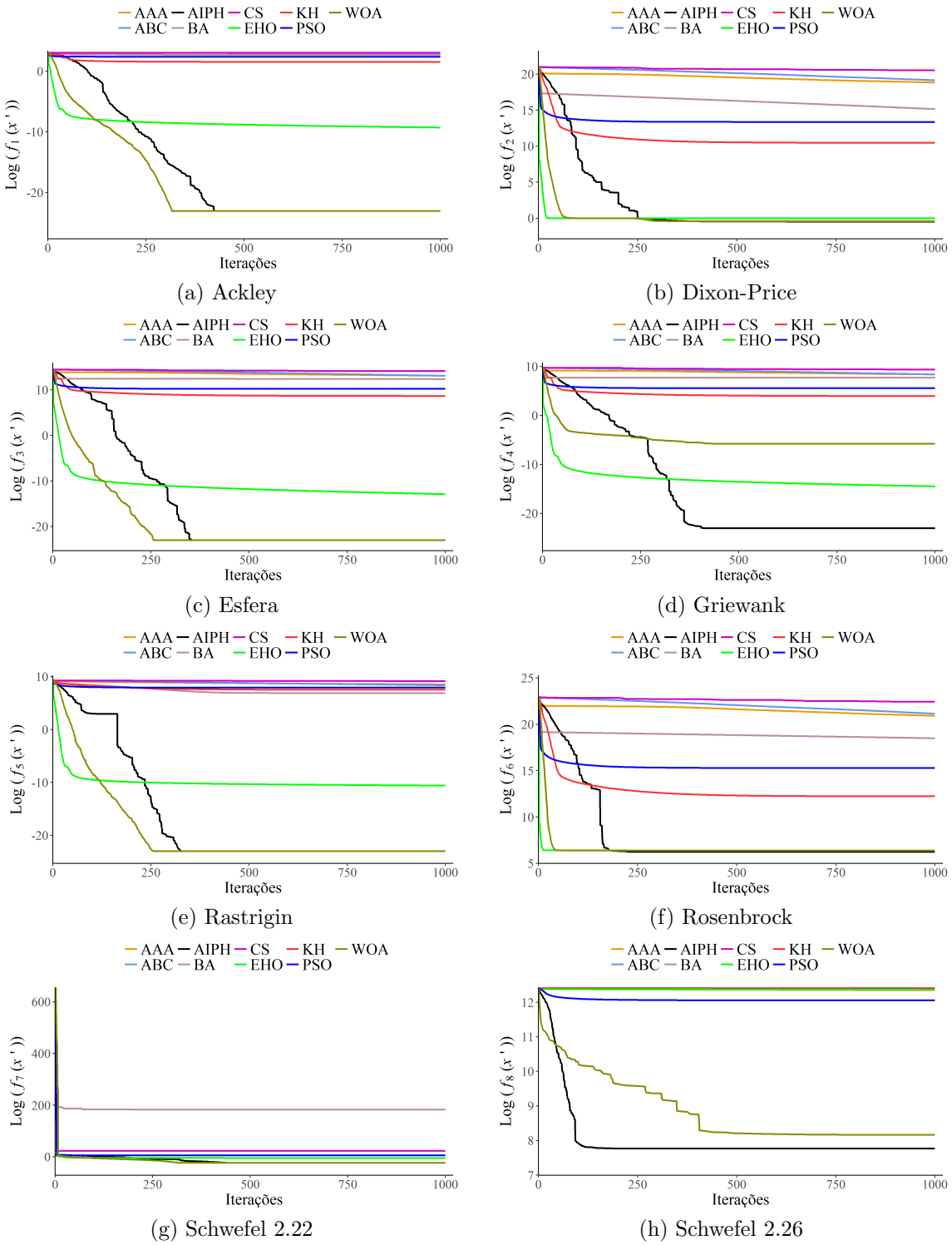


Figura 45: Comparação com técnicas inspirada em princípios da matemática ou relações humanas com cenário de 600 dimensões.



(a) Ackley (b) Dixon-Price

(c) Esfera (d) Griewank

(e) Rastrigin (f) Rosenbrock

(g) Schwefel 2.22 (h) Schwefel 2.26

Figura 46: Comparação com técnicas bioinspiradas com cenário de 600 dimensões.

APÊNDICE D – Outros Trabalhos Relacionados

Este apêndice apresenta outras técnicas que possuem grande contribuição no estado da arte, mas não foram usadas para a comparação de desempenho com o AIPH. O apêndice dividi-se em 4 seções, cada uma representado uma das inspirações citadas no Capítulo 2: princípios da matemática, propriedades da Física, bio inspiração e relações humanas.

D.1 Otimização Inspirada em Princípios da Matemática

Mapeamento de Variância Média

Com sua primeira versão sendo lançada em 2010, por Erlich et al., o Mapeamento de Variância Média (*Mean-Variance Mapping Optimization* - MVMO) está na categoria de técnica estocástica baseada em população, que possui como conceito básico certas semelhanças com outras abordagens heurísticas, porém faz uso de função especial de mapeamento aplicado para mutar os descendentes baseado na média e da variância da população N-melhor atingido até o momento. As semelhanças entre MVMO e os outros algoritmos estocásticos conhecidos está na inclusão de operadores evolucionários: seleção, mutação e *crossover* (GONZALEZ-LONGATT et al., 2012).

A propriedade elegante da MVMO é a capacidade de pesquisar em torno da melhor solução local com uma pequena chance de ser preso em um dos melhores locais. Este recurso é contribuído para a estratégia de lidar com a variância zero (GONZALEZ-LONGATT et al., 2012).

É uma técnica que vem ganhando destaque em competições de otimização com uma das suas variações, a MVMO-PHM, que a fim de se diferenciar das variantes anteriores da MVMO, possui o P, que denota a abordagem baseada na população, H a hibridização da estrutura algorítmica para incluir a estratégia de busca local, e M a adoção de uma

nova função de mapeamento (ERLICH; RUEDA, 2016). No Algoritmo 25 é apresentado o pseudocódigo da forma canônica da técnica MVMO.

Algoritmo 25 Pseudocódigo do Mapeamento de Variância Média

Entrada: Parâmetros comuns (Algoritmo 1).

início
  Normaliza as d variáveis de otimização no intervalo [0,1].
  Avalia cada um dos x_i a partir da função objetivo usando as variáveis normalizadas.
  enquanto *(critério de parada não é satisfeito)* **faça**
    Atualiza o arquivo de soluções (armazenamento as n-melhores populações).
    Gera descendência (Atribuição dos pais e mutação via função de mapeamento).
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

O arquivo de solução constitui a base de conhecimento do algoritmo para guiar a direção de busca. Assim, os melhores n indivíduos que MVMO encontrou até agora são salvos no arquivo de soluções. Além disso, também são armazenados dois parâmetros de informação relevantes, a saber, aptidão e viabilidade de cada indivíduo. As seguintes regras são estabelecidas para comparar o indivíduo gerado em cada iteração e soluções existentes arquivadas, a fim de evitar a perda de boas soluções: (a) qualquer solução viável é preferível a qualquer solução inviável, (b) entre duas soluções viáveis, a que tem melhor valor objetivo é preferida, (c) entre duas soluções inviáveis, é preferida a que tem menor violação de restrição(GONZALEZ-LONGATT et al., 2012).

Uma atualização ocorre somente se o novo indivíduo é melhor do que aqueles no arquivo. O tamanho do arquivo é fixo para todo o processo. Os indivíduos arquivados são ordenados dinamicamente para que o indivíduo classificado primeiro seja sempre o melhor. Soluções viáveis são colocadas na parte superior do arquivo. Entre essas soluções, elas são classificadas com base em seus valores objetivos originais. Soluções inviáveis são classificadas de acordo com seus valores de aptidão e, em seguida, colocadas na parte inferior do arquivo. Uma vez que o arquivo é preenchido por n soluções viáveis, qualquer solução de candidato inviável não tem chance de ser salvo no arquivo(GONZALEZ-LONGATT et al., 2012). Para cada uma das dimensões m-selecionadas, a mutação é usada para atribuir um novo valor dessa variável. Dado um número aleatório uniforme $\hat{x}_i \in [0, 1]$, o novo valor da i-ésima componente x_i é determinado por:

$$\tilde{x}_i = h_x + (1 - h_1 + h_0) * \hat{x}_i - h_0, \tag{127}$$

onde h_x, h_1 e h_0 são as saídas da função de mapeamento de transformação com base em diferentes entradas dadas por:

$$h_x = h(u_i = \hat{x}_i); h_0 = h(u_i = 0); h_1 = h(u_i = 1). \tag{128}$$

A função de mapeamento usada para gerar a descendência é parametrizada pela Equação 129:

$$h(x_i, s_{i1}, s_{i2}, u_i) = \bar{x}_i(1 - e^{u_i, s_{i1}} + (1 + \bar{x}_i)e^{(1-u_i), s_{i2}}, \tag{129}$$

onde s_{i1} e s_{i2} são fatores de forma que perm inclinações assimétricas da função de mapeamento. A inclinação é calculada na Equação 130:

$$s_i = -ln(v_i) * f_s, \tag{130}$$

onde f_s é um fator de escala, que permite o controle do processo de busca durante a iteração.

D.2 Otimização Inspirada em Relações Humanas

Busca de Harmonia

A Busca de Harmonia (*Harmony Search* - HS) é um algoritmo de otimização metaheurística baseado em música, inspirado pela observação de que o objetivo da música é procurar um estado de harmonia. Esta harmonia na música é análoga para encontrar um valor ótimo em um processo de otimização. O processo de pesquisa em otimização pode ser comparado a um processo de improviso de um músico qualificado. Quando um músico está improvisando, ele ou ela tem três opções possíveis: (a) tocar qualquer música famosa (uma série de passos em harmonia) exatamente de sua memória; (b) jogar algo semelhante a uma peça conhecida (ajustando assim o passo ligeiramente); (c) compor novas ou aleatórias notas, formalizando-as essas três opções em um processo de otimização quantitativa (YANG, 2009b). Pode-se ver os passos do processo de busca usado pela técnica de Busca de Harmonia no Algoritmo 26.

O processo de otimização usa três componentes: memória de harmonia, ajuste de afinação e randomização. A escolha da memória de harmonia é importante, pois funciona como uma escolha dos melhores indivíduos. Isto irá assegurar que as melhores harmonias serão transferidas para a nova memória de harmonia. Para usar esta memória de forma mais eficaz, é tipicamente atribuído como um parâmetro $r_{accept} \in [0, 1]$, chamado de

Algoritmo 26 Pseudocódigo da Busca de Harmonia

```
Entrada: Parâmetros comuns (Algoritmo 1).
início
    Taxa de ajuste do passo (r_pa).
    Limites do passo.
    Largura de banda.
    Taxa de aceitação de memória de harmonia (r_accept).
    enquanto (critério de parada não é satisfeito) faça
        Gera novas harmonias, aceitando as melhores harmonias.
        Ajusta o passo para obter as novas harmonias.
        se (rand(0,1) > r_accept) então
            Escolhe aleatoriamente uma harmonia existente.
        senão, se (rand(0,1) > r_pa) então
            Ajusta o passo aleatoriamente dentro dos limites.
        senão
            Gera novas harmonias via aleatorização.
        fim
        Aceita as novas harmonias se melhores
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

aceitação de memória de harmonia ou taxa de consideração. Se esta taxa é muito baixa, apenas algumas das melhores harmonias são selecionadas e pode convergir muito lentamente. Se esta taxa possui um valor próximo de 1, quase todas as harmonias são usadas na memória de harmonia, então outras harmonias não são bem exploradas, levando a soluções potencialmente erradas. Portanto, tipicamente, usa-se o r_{accept} entre 0,7 e 0,95 (YANG, 2009b).

O segundo componente é o ajuste de passo determinado por um b_{range} de largura de passo de banda e uma taxa de ajuste de passo r_{pa} . Embora na música, ajuste de afinação significa mudar as frequências, corresponde a gerar uma solução ligeiramente diferente no algoritmo Busca de Harmonia. Em teoria, o passo pode ser ajustado linearmente ou não linearmente, mas na prática, ajustamento linear é usado(YANG, 2009b). Então nós temos:

$$\tilde{x}_i = x_i + b_{range} * \xi, \tag{131}$$

onde x_i é o passo ou solução existente a partir da memória de harmonia, e $\tilde{x}_i$ é o novo passo após a ação do ajuste de passo. Isto essencialmente produz uma nova solução em torno da solução de qualidade existente variando o passo ligeiramente por uma pequena quantidade aleatória(YANG, 2009b).

Aqui ξ é um gerador de números aleatórios no intervalo de [-1,1]. O ajuste de afinação é semelhante ao operador de mutação em algoritmos evolutivos. Pode-se atribuir uma taxa de ajuste do passo (r_{pa}) para controlar o grau do ajuste. Uma taxa de ajuste de passo baixo com uma largura de banda estreita pode retardar a convergência do HS porque a limitação na exploração de apenas um pequeno subespaço de todo o espaço de

pesquisa. Por outro lado, uma taxa de ajuste de passo muito alta com uma largura de banda larga pode causar a solução dispersa em torno de algum potencial ótimo como em uma pesquisa aleatória. Assim, normalmente usa-se o r_{pa} entre 0,1 e 0,5 na maioria das aplicações(YANG, 2009b).

O terceiro componente é a randomização, que é responsável pela geração de novas harmonias via aleatorização, aumentando a diversidade das soluções. Embora o ajuste de passo tenha um papel semelhante, mas é limitado a um determinado ajuste de passo local e, portanto, corresponde a uma busca local. O uso da randomização pode impulsionar o sistema ainda mais para explorar várias soluções diversas de modo a encontrar o ótimo global(YANG, 2009b). No pseudo-código, pode-se ver que a probabilidade de randomização conforme Equação 132:

$$P_{random} = 1 - r_{accept}, \tag{132}$$

e a probabilidade real de ajustar os arremessos é:

$$P_{passo} = r_{accept} * r_{pa}. \tag{133}$$

D.3 Otimização Inspirada em Propriedades Físicas

Arrefecimento Simulado

O Arrefecimento Simulado (*Simulated Annealing* - SA) surgiu no contexto da mecânica estatística, desenvolvido por Kirkpatrick et al., em 1983, e independentemente por Cerny em 1985, utilizando o algoritmo de Metropolis de 1953 (HAESER; GOMES-RUGGIERO, 2008).

Especificamente, a técnica de arrefecimento consiste em aquecer um material a fim de lhe conferir alta energia. O material é então arrefecido lentamente, de tal modo que em cada fase a temperatura é mantida durante tempo suficiente. Esta estratégia de recozimento conduz a um estado sólido (estável) final correspondendo a um mínimo absoluto de energia nas moléculas do material. Pelo contrário, um arrefecimento rápido, descontrolado do material conduz a uma estrutura amorfa, isto é, metaestável indicando um mínimo local de temperatura (HAESER; GOMES-RUGGIERO, 2008).

O processo de otimização é realizado por níveis, simulando os níveis de temperatura no resfriamento. Em cada nível, dado um ponto $u \in S$, vários pontos na vizinhança de u são gerados e o correspondente valor de aptidão $f(x)$ é calculado. Cada ponto gerado é aceito

ou rejeitado de acordo com certa probabilidade. Esta probabilidade de aceitação decresce de acordo com o nível do processo, ou equivalentemente, de acordo com a temperatura (HAESER; GOMES-RUGGIERO, 2008). O pseudocódigo do SA pode ser visto no Algoritmo 27.

Algoritmo 27 Pseudocódigo do Arrefecimento Simulado

Entrada: Parâmetros comuns (Algoritmo 1).
Inicializa o ponto u.
Define a temperatura inicial T_0.
Define o número de transições inicial L_0.
Nível ($k = 0$).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para l = *1 até* Lk **faça**
      Gerar w da vizinhança de u.
      se $f(w) \leq f(u)$ **então**
        $u = w$
      fim
      Caso contrário, **se** $rand(0,1) < exp(\frac{f(u)-f(w)}{T_k})$ **então**
        $u = w$
      fim
      $k = k + 1$
    fim
    Calcular L_k e T_k.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

A ideia do SA é obter $i \in S$ com distribuição q. Para isso, é construída uma cadeia de Markov homogênea com T fixo até atingir-se a distribuição limite, em seguida diminuí-se T e restaura-se o equilíbrio atingindo novamente a distribuição limite para a nova temperatura. O processo é repetido até que T seja aproximadamente zero (HAESER; GOMES-RUGGIERO, 2008).

Enxame de Massa Estelar de Buracos Negros

O Enxame de Massa Estelar de Buracos Negros (*Stellar Mass Black Hole Optimization* - SBO) foi desenvolvido por Premalatha Balamurugan em 2015, baseados nas propriedades físicas absorção, emissão, coalescência e fuga de buraco negro.(PREMALATHA; BALAMURUGAN, 2015). Pode ser visto a seguir no Algoritmo 28, correspondente a técnica.

A massa estelar do buraco negro tem a seguinte característica (PREMALATHA; BALAMURUGAN, 2015):

- A massa de um buraco negro cresce quando absorve a matéria;
- Um buraco negro também cresce colidindo e fundindo-se com outros buracos negros;
- Durante a radiação, um buraco negro perde energia e sua massa é reduzida;

Algoritmo 28 Pseudocódigo do Enxame Massa Estelar de Buracos Negros

```
Entrada: Parâmetros comuns (Algoritmo 1).
         Taxa de Absorção (α).
         Taxa de emissão (β).

início
    enquanto (critério de parada não é satisfeito) faça
        para i = 1 até k faça
            Avalia a aptidão do buraco negro.
            Taxa de crescimento do buraco negro ρ_k.
            Taxa de radiação do buraco negro ϵ_k.
        fim
        para i = 1 até k faça
            Atualiza a massa do buraco negro usando a Equação 136.
        fim
        Ordena as soluções
        Substitui uma fração dos piores buracos negros com novos buracos negros.
        Colide os buracos negros próximos e gera buracos negros para as saídas deixadas na população.
        Mantenha a melhor solução.
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

- Buracos negros menores irradiam mais do que os maiores;
- A emissão torna-se cada vez maior e até terminar.

Com base nas propriedades acima, a técnica é desenvolvida incorporando diversificação e intensificação no espaço de busca. Os indivíduos têm sucesso em novas características que lhes dão uma sobrevivência, fazendo parecer uma técnica evolucionária. A partir das seguintes características são realizados os seguintes cálculos (PREMALATHA; BALAMURUGAN, 2015):

(a) *Taxa de Crescimento do Buraco Negro*: A taxa de crescimento do buraco negro é diretamente proporcional à sua massa. O buraco negro de massa mais elevada tem uma taxa de absorção mais elevada que é dada na Equação 134:

$$\rho_i = \frac{f(x_i)}{\sum_{j=1}^{k} f(x_i)}, \tag{134}$$

onde $f(x_i)$ é o valor da aptidão do buraco negro i na população $1 < i \leq k$ e k é o número de buracos negros na população.

(b) *Taxa de Radiação*: A taxa de radiação ε_i do buraco negro individual é dada na Equação 135:

$$\varepsilon_i = 1 - \rho_i, \tag{135}$$

(c) *Atualização Massa do Buraco Negro*: A atualização segue a Equação 136:

$$\tilde{b}_{i,d} = b_{i,d} + \Delta a_{i,d} - \Delta e_{i,d}, \tag{136}$$

onde $b_{i,d}$ sendo o valor do i-ésimo buraco negro na j-ésima dimensão; $\Delta a_{i,d}$ o crescimento na dimensão j no i-ésimo buraco negro, calculado pela Equação 137:

$$\Delta a_{i,d} = \begin{cases} \alpha(\rho_i * b_{i,d} + \rho_{i+1} * b_{i,d+1}), & i = 1, \\ \alpha(\rho_{i-1} * b_{i,d-1} + \rho_i * b_{i,d} + \rho_{i+1} * b_{i,d+1}), & 1 < i < k, \\ \alpha(\rho_{i-1} * b_{i,d-1} + \rho_i * b_{i,d}), & i = k, \end{cases} \tag{137}$$

com α sendo a taxa de absorção determinada entre 0 e 1; e ρ_i a taxa de crescimento do i -ésimo buraco negro. $\Delta e_{i,d}$ é a radiação na dimensão j no i-ésimo buraco negro e t a iteração, calculado pela Equação 138:

$$\Delta e_{i,d} = \beta * \varepsilon_i * b_{i,d}, \tag{138}$$

onde β é a taxa de emissão determinada, assumindo um valor entre 0 e 1, e ε_i é a taxa de radiação do i-ésimo buraco negro.

(d) *Colisão entre buracos negros*: O cenário mais possível para dois buracos negros é uma fusão. A massa total seria a massa combinada dos dois buracos negros. Se os dois buracos negros b_p e b_q estiverem muito mais próximos, então eles se fundem e formam o novo buraco negro b_n usando a Equação 139:

$$b_{n,d} = rand(0, 1) * (b_{p,d} + b_{q,d}). \tag{139}$$

D.4 Otimização Bioinspirada

Busca de Cardume de Peixes

Proposto por Bastos Filho e Lima Neto em 2009 (FILHO et al., 2008), o algoritmo Busca de Cardume de Peixes (*Fish School Search* - FSS) é inspirado no comportamento coletivo de um cardume de peixes. Em cardumes de peixes, tem-se um conjunto de indivíduos atuando coletivamente, objetivando criar uma proteção mútua em relação aos predadores e/ou a busca conjunta por alimentos. O FSS é composto principalmente de três operadores: alimentação, natação e reprodução. Usando esses operadores, computacionalmente, o algoritmo adquire uma habilidade de pesquisa abrangente, flexibilidade na capacidade de alternar entre exploração local e global, com orientação global auto-adaptável para o processo de busca (FILHO et al., 2008).

Basicamente, os comportamentos que serviram de inspiração natural para a construção passada e futura do algoritmo FSS, foram (FILHO et al., 2008):

- Alimentação: inspirada pelo instinto natural dos peixes para encontrar comida para desenvolverem-se e se reproduzirem. A procura por comida aqui refere-se a avaliação de soluções candidatas no processo de busca. Em oposição ao conceito de ingestão de alimentos, também considera-se que um indivíduo pode perder peso nadando;

- Natação: Pretende imitar o movimento compartilhado coletivo coordenado de movimento produzido por todos os peixes na escola. A natação é conduzida pela alimentação que, basicamente, orienta o processo de busca;

- Criação: baseado no mecanismo de seleção natural, onde seres com melhor aptidão tem maior probabilidade de procriar e outros indivíduos piores, são mais propensos a perecer. A geração da prole aqui permite melhorar a habilidade de exploração para soluções candidatas melhor adaptadas do processo de busca.

Inspirando-se no comportamento natural dos cardumes, o Algoritmo 29 apresenta o processo de busca do FSS.

Algoritmo 29 Pseudocódigo de Busca de Cardume de Peixes

Entrada: Parâmetros comuns (Algoritmo 1).
Define W_{scale}, $step_{ind.inicial}$, $step_{ind.final}$, $step_{vol.inicial}$ e $step_{vol.final}$.

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para i = *1 até* k **faça**
      Realiza o Movimento Individual (Equação 140).
      Executa o Processo de Alimentação (Equação 142).
      Realiza o Movimento Coletivo Instintivo (Equação 143).
      Realiza o Movimento Coletivo Volitivo (Equação 145 ou 146).
    fim
    Ordena os peixes e procura o atual melhor.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

No Operador de Movimento Individual, cada indivíduo executa um movimento aleatório e independente dentro dos limites do ambiente, porém, sempre na direção do gradiente positivo, se for o problema for de maximização, ou negativo, se for um problema de minimização. Em outras palavras, este operador em todos os ciclos do algoritmo FSS é executado apenas se a nova posição for melhor do que a anterior. O peixe avalia se a densidade de alimentos parece ser melhor do que em sua localização atual. Se este não for o caso ou se o tamanho do passo for além dos limites do ambiente, o movimento individual do peixe não ocorre (LACERDA, 2012). Tal movimento é descrito pela Equação 140:

$$\tilde{x}_i^j = x_i^j + step_{ind}, \tag{140}$$

onde x_i^j é o valor da dimensão j do vetor de posição do peixe i no tempo t, $\tilde{x}_i^j$ é o seu novo valor, r é uma variável aleatória pertencente ao intervalo [1,1] e $step_{ind}$ é o tamanho do passo individual no tempo t. Objetivando a obtenção de uma busca mais ampla no início da execução do algoritmo e um refinamento gradual da mesma com o passar das iterações, o tamanho do passo individual pode decair linearmente, como descreve a Equação 141:

$$\tilde{step}_{ind} = step_{ind} - \frac{step_{ind.inicial} - step_{ind.final}}{G}, \tag{141}$$

onde $step_{ind}$ e $\tilde{step}_{ind}$ são os tamanhos do passo do movimento individual antes e depois de sua atualização, $step_{ind.inicial}$ é o tamanho de tal passo inicial, $step_{ind.final}$ é o seu tamanho final e G é o número máximo de iterações do algoritmo. Logo após cada movimento individual, ocorre a alimentação.

O Operador de Alimentação é responsável por atualizar o peso do peixe de acordo com a quantidade de alimento ingerido através do Operador de Movimento Individual, ou seja, de acordo com o valor de melhora da aptidão após tal movimento. Seguindo o comportamento natural, os peixes do FSS direcionam-se para os alimentos espalhados no ambiente em várias concentrações. Para encontrar maiores quantidades de alimentos, os peixes no cardume podem realizar movimentos individuais. Como resultado, cada peixe pode crescer ou diminuir em peso, dependendo do seu sucesso ou falha na busca de alimentos. Propõe-se que a variação do peso do peixe seja proporcional à diferença normalizada entre a avaliação da função de aptidão da posição de peixe anterior e atual em relação à concentração de alimentos (FILHO et al., 2008). A avaliação da concentração de alimentos considera todas as dimensões do problema, como mostrado na Equação 142:

$$\tilde{W}_i = W_i + \frac{\Delta f_i}{max(|\Delta f_i|)}, \tag{142}$$

onde W_i e $\tilde{W}_i$ são os valores do peso do peixe i antes e depois da alimentação, respectivamente, Δf_i são as variações de aptidão após o Movimento Individual dos peixes. Adicionalmente, incluiram-se medidas para garantir que a convergência para áreas ricas do aquário ocorra rapidamente, que são: a variação do peso do peixe é avaliada uma vez em cada ciclo do FSS; um parâmetro adicional, denominado escala de peso (W_{scale}) foi criado para limitar o peso de um peixe, fazendo com que varie no intervalo [1,W_{scale}]; todos os peixes nascem com peso igual a $W_{scale}/2$ (FILHO et al., 2008).

O Operador de Movimento Coletivo Instintivo é o primeiro movimento coletivo a ser executado em cada iteração do algoritmo. É responsável por definir um vetor o qual será

adicionado ao vetor de posição de todos os indivíduos. Em outras palavras, neste operador, os indivíduos devem "imitar" o comportamento dos mais bem sucedidos (LACERDA, 2012). Tal operador é descrito pela Equação 143:

$$\tilde{x}_i^j = x_i^j + \frac{\sum_N^{k=1} \Delta x_k^j * \Delta f(x_k)}{\sum_N^{k=1} \Delta f(x_k)}, \tag{143}$$

onde x_i^j é o valor da dimensão j do vetor de posição do peixe i no tempo t, $\tilde{x}_i^j$ é tal valor atualizado, ou seja, no tempo $t+1$, N é o número de indivíduos na população, Δx_k^j é a variação do valor da dimensão j do vetor de posição do peixe k antes e depois do último Movimento Individual e $\Delta f(x_k)$ é variação da aptidão do mesmo peixe.

No Operador de Movimento Coletivo Volitivo, o cardume deve realizar contrações ou expansões tendo como referência o baricentro do cardume, o qual é calculado utilizando o peso de cada peixe, seguindo o cálculo da Equação 144:

$$B_j = \frac{\sum_N^{i=1} x_i^j * W_i}{\sum_N^{i=1} W_i}, \tag{144}$$

onde Em cada execução do movimento coletivo volitivo, o peso total do cardume deve ser calculado. Se houver aumento deste valor, o cardume deve contrair de forma a executar uma busca local em uma determinada região e com um aumento, significa que o processo de busca foi bem sucedido. Contudo, se o valor não aumenta, o cardume deve expandir, pois o local explorado não está oferecendo boas fontes de alimento. A expansão evita que a população fique estagnada em mínimos locais, realizando, desta forma, uma exploração mais ampla, ou seja buscando outras regiões (LACERDA, 2012). Estes processos são calculados conforme a Equação 145, representando a contração:

$$\tilde{x}_i^j = x_i^j step_{vol} * rand(0,1) * \frac{x_i^j - B_j}{d(x_i, B_j)}, \tag{145}$$

onde $step_{vol}$ é o tamanho do passo volitivo definido a partir da diferença entre $step_{vol.final}$ e $step_{vol.inicial}$, o qual, assim como ocorre com o tamanho do passo individual, pode ser decrescido linearmente ao decorrer da execução do algoritmo, $rand(0,1)$ é um número uniformemente aleatório gerado entre 0 e 1 e $d(x_i, B_j)$ é uma função que retorna a distância euclidiana entre os vetores x_i e B_j. Já a Equação 146, representa a expansão:

$$\tilde{x}_i^j = x_i^j + step_{vol} * rand(0,1) * \frac{x_i^j - B_j}{d(x_i, B_j)}. \tag{146}$$

Algoritmo de Vagalumes

É uma metaheurística proposta em por Xin-She Yang e inspirada pelo comportamento de brilho dos vagalumes (*Firefly Algorithm* - FA), apresentado em 2008. Usa três regras: 1) todo vagalume são unissex, para que os vagalumes sejam atraídos para outros vagalumes independentemente do seu sexo; 2) Atratividade é proporcional ao seu brilho, assim, para quaisquer dois vagalumes, o menos brilhante vai avançar para a mais brilhante. A atratividade é proporcional ao brilho e ambos diminuem à medida que a distância aumenta. Se não houver mais brilhante do que um determinado vagalume, se movimentará aleatoriamente; 3) O brilho de um vagalume é determinado pela paisagem da função objetivo. Para um problema de maximização, o brilho pode ser simplesmente proporcional ao valor da função objetivo (YANG, 2009a). O Algoritmo 30 apresenta o processo de busca do Algoritmo de Vagalumes.

Algoritmo 30 Pseudocódigo do Algoritmo de Vagalumes

```
Entrada: Parâmetros comuns (Algoritmo 1).
         Define o Coeficiente de absorvição de luz γ.
         Inicializa a Intensidade de luz I_i.

início
    enquanto (critério de parada não é satisfeito) faça
        para i = 1 até k faça
            para j = 1 até i faça
                se (I_i > I_i) então
                    Move o vagalume i em direção a j em d-dimensões.
                fim
                Avalia as novas soluções e atualiza a intensidade de luz.
            fim
        fim
        Ordena os vagalumes e procura o atual melhor.
    fim
    Retorna a melhor posição e sua respectiva solução.
fim
```

Como a atratividade de um vagalume é proporcional à intensidade luminosa observada pelos vagalumes adjacentes, pode-se agora definir a variação da atratividade β com a distância r por:

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{147}$$

onde β_0 é a atratividade em $r = 0$ (YANG, 2009a). O movimento de um vagalume i é atraído por outro vagalume mais atraente (mais brilhante) j é determinado por:

$$\tilde{x}_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha_t \epsilon_i, \tag{148}$$

onde o segundo termo é devido à atração. O terceiro termo é a randomização com α_t sendo o parâmetro de aleatorização e ϵ_i é um vetor de números aleatórios desenhados a

partir de uma distribuição Gaussiana ou distribuição uniforme no tempo t. Se $\beta_0 = 0$, torna-se uma simples caminhada aleatória. Por outro lado, se $\gamma = 0$, reduz-se a uma variante de otimização de enxames de partículas. Além disso, a randomização γ pode ser facilmente estendida para outras distribuições, tais como voos de Levy (YANG, 2009a). Como α_t controla essencialmente a aleatoriedade (ou, em certa medida, a diversidade de soluções), pode-se ajustar este parâmetro durante iterações para que ele possa variar com o contador de iteração t. Então, uma boa maneira de expressar α_t é usar:

$$\alpha_t = \alpha_0 \delta^t, 0 < \delta < 1, \tag{149}$$

onde α_0 é o fator de escala de aleatoriedade inicial e δ é essencialmente um fator de arrefecimento. Para a maioria das aplicações, pode-se usar $\delta = 0{,}95$ a $0{,}97$. Em relação ao α_0 inicial, as simulações mostram que o FA será mais eficiente se α_0 estiver associado com escalas de variáveis de projeto. Seja L a escala média do problema de interesse, pode-se definir $\alpha_0 = 0,01L$ inicialmente. O fator 0,01 vem do fato de que caminhadas aleatórias requerem um número de etapas para alcançar a meta enquanto equilibram a exploração local sem saltar muito longe em poucos passos(YANG, 2009a).

O parâmetro β controla a atratividade, e estudos paramétricos sugerem que $\beta_0 = 1$ pode ser usado para a maioria das aplicações. No entanto, γ deve estar também relacionada com a escala L. Em geral, pode-se definir $\gamma = 1/\sqrt{(L)}$. Se as variações de escala não são significativas, então pode-se definir $\gamma = O(1)$ (YANG, 2009a).

Otimizador de Lobos Cinzentos

Desenvolvido por Mirjalili e Lewis, em 2014, o algoritmo imita a hierarquia de liderança e mecanismo de caça de lobos cinzentos na natureza. O mecanismo de busca considera quatro tipos de lobos cinzentos como α, β, δ e ω, que são empregados para simular a hierarquia liderança. Além disso, são implementadas as 3 principais etapas de caça, busca de presas, circundando presa e atacando presas (MIRJALILI; MIRJALILI; LEWIS, 2014). Pode-se verificar o processo de busca empregado pelo Otimizador de Lobos Cinzentos (*Grey Wolf Optimizer* - GWO) no Algoritmo 31.

O algoritmo GWO é delineado baseando-se nos seguintes modelos matemáticos de hierarquia social e de caça: perseguir, rodear e atacar a presa. Os itens abaixo apresentam estes elementos da técnica(MIRJALILI; MIRJALILI; LEWIS, 2014): (a) *Hierarquia Social*: ao executar o GWO, considera-se a solução mais apta como α. Consequentemente, a segunda

Algoritmo 31 Pseudocódigo do Otimizador de Lobos Cinzentos

Entrada: Parâmetros comuns (Algoritmo 1).
Parâmetros a, A e C.
Avalia a aptidão de cada lobo cinza.
x_α: melhor lobo cinza.
x_β: segundo melhor lobo cinza.
x_θ: terceiro melhor lobo cinza.

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para *cada lobo cinza* **faça**
      Atualiza a posição do lobo cinza usando a Equação 154.
    fim
    Atualiza a, A e C.
    Calcula a aptidão de todos os lobos cinza.
    Atualiza x_α, x_β e x_θ.
    Avalia a aptidão de cada lobo cinza.
  fim
  Retorna a melhor posição e sua respectiva solução
fim

e terceira melhores soluções são denominadas β e δ, respectivamente. O resto das soluções candidatas são assumidas como sendo ω. No algoritmo GWO a otimização é guiada por α, β e δ. Os ω lobos seguem estes três lobos.

(b) *Rodeando a presa*:.Para modelar matematicamente o comportamento rodear a presa, são a Equação 150 :

$$\tilde{x}_i = x_p - A * D, \tag{150}$$

com,

$$D = |C * x_p - x_i|, \tag{151}$$

onde t indica a iteração atual, A e C são vetores de coeficientes, x_p é o vetor de posição da presa, e x_i indica o vetor de posição de um lobo cinzento. Os vetores A e C são calculados como se segue:

$$A = 2 * a * r_1 - a, \tag{152}$$

$$C = 2 * r_2, \tag{153}$$

onde componentes de a são linearmente diminuídos de 2 para 0 ao longo das iterações e r_1, r_2 são vetores aleatórios em [0, 1].

(c) *Caça*: Lobos cinzentos têm a capacidade de reconhecer a localização da presa e cercá-los. A caça é geralmente guiada pelo α. O β e o δ também podem participar na caça ocasionalmente. No entanto, em um espaço de busca abstrata não tem-se ideia sobre a localização do ótimo (presa). A fim de simular matematicamente o comportamento de caça de lobos cinzentos, supõe-se que a α (melhor candidato solução) β, e δ tem um melhor conhecimento sobre a localização potencial de presas. Portanto, guarda-se as

três primeiras melhores soluções obtidas até agora e obriga-se os outros lobos cinzentos (incluindo os ω) a atualizar suas posições de acordo com a posição dos melhores lobos cinzentos. A Equação 154 é propostas a este respeito.

$$\tilde{x}_i = \frac{x_1 + x_2 + x_3}{3}, \tag{154}$$

onde,

$$x_1 = x_\alpha - A_1 * D_\alpha, \tag{155}$$

$$x_2 = x_\beta - A_2 * D_\beta, \tag{156}$$

$$x_3 = x_\delta - A_3 * D_\delta, \tag{157}$$

onde,

$$D_\alpha = |C_1 * x_\alpha - x_i|, \tag{158}$$

$$D_\beta = |C_2 * x_\beta - x_i|, \tag{159}$$

$$D_\omega = |C_3 * x_\omega - x_i|, \tag{160}$$

com A_1, A_2 e A_3 calculados conforme a Equação 152 e C_1, C_2 e C_3 calculados conforme a Equação 153.

(d) *Atacar presas (exploração local)*: Os lobos cinzentos terminam a caça atacando a presa quando ela pára de se mover. A fim de modelar matematicamente a aproximação da presa, diminuí-se o valor de a. Observe que a faixa de flutuação de A também é diminuída por a. Por outras palavras, A é um valor aleatório no intervalo $[-2a, 2a]$ onde a é diminuído de 2 para 0 ao longo das iterações. Quando valores aleatórios de A estão em [-1, 1], a próxima posição de um lobo cinzento pode estar em qualquer posição entre sua posição atual e a posição da presa.

(e) *Pesquisa de presa (exploração global)*: a pesquisa ocorre de acordo com a posição do α, β e δ. Eles divergem uns dos outros para procurar presas e convergem para atacar presas. Para modelar matematicamente a divergência, utiliza-se A com valores aleatórios maiores que 1 ou inferiores a -1 para obrigar o lobo cinzento a divergir da presa. Isto enfatiza a exploração global e permite que o algoritmo GWO procure globalmente. Outro componente do GWO que favorece a exploração é C. O vetor C contém valores aleatórios em [0, 2]. Este componente fornece pesos aleatórios para a presa, de forma a enfatizar estocástica ($C > 1$) ou desestimular ($C < 1$) o efeito da presa na definição da distância na Equação 151.

Algoritmo de Aranha Social

Proposta por Yu Li, em 2015, a técnica usa como inspiração a Aranha Social, uma espécie de aranha cujos indivíduos formam agregações relativamente duradouras. Tal inspiração foi utilizada para resolver problemas de otimização global, baseando-se principalmente na estratégia de forrageamento dessas aranhas através das vibrações na teia para determinar a posição da presa. Esta espécie de aranhas também pode distinguir vibrações geradas pela presa com outras aranhas e recebem passivamente as vibrações geradas por outras aranhas (YU; LI, 2015).

No Algoritmo de Aranha Social (*Social Spider Algorithm* - SSA), uma aranha gerará uma vibração quando atingir uma nova posição diferente da anterior. A intensidade da vibração está correlacionada com a adequação da posição. A vibração se propagará sobre a teia e outras aranhas podem sentí-la (YU; LI, 2015). Dessa forma, as aranhas na mesma teia compartilham suas informações pessoais com outras pessoas para formar um conhecimento social coletivo. O Algoritmo 32 representa os passos da técnica SSA.

Algoritmo 32 Pseudocódigo do Algoritmo de Aranha Social

Entrada: Parâmetros comuns (Algoritmo 1).
Cria memória para as aranhas de busca.
Inicializa v_i^{tar} para cada aranha.

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para i = *1 até* k **faça**
      Avalia a aptidão das aranha.s
      Gera a vibração da posição de x_i.
    fim
    para i = *1 até* k **faça**
      Calcula a intensidade da vibração V gerada para todas as aranhas.
      Seleciona a vibração mais forte (v_i^{best}) de V.
      se *a intensidade de* v_i^{best} *é maior que* v_i^{tar} **então**
        Armazena v_i^{best} como v_i^{tar}.
      fim
      Atualiza c_i.
      Gera um número aleatório r entre $[0, 1]$.
      se $r > p_c^{c_i}$ **então**
        Atualiza a dimensão máscara m_s.
      fim
      Gera v_i^{fo}.
      Executa um passo aleatório.
      Trata qualquer violação de limitações.
    fim
  fim
  Retorna a melhor posição e sua respectiva solução
fim

A representação matemática dos elementos usados SSA considera os itens abaixo, que apresentam os elementos da técnica (YU; LI, 2015):

(a) *Vibração na Teia*: A vibração é um conceito muito importante em SSA. É uma das principais características que distinguem a SSA de outras metaheurísticas. Na SSA, usa-se

duas propriedades para definir uma vibração, ou seja, a posição da fonte e a intensidade da fonte da vibração. A posição da fonte é definida pelo espaço de busca do problema de otimização e define-se a intensidade de uma vibração no intervalo $[0, +\infty]$. Sempre que uma aranha se move para uma nova posição, ela gera uma vibração na sua posição atual. Define-se a posição da aranha a no tempo t como P_a. Utiliza-se ainda $I(P_a, P_b)$ para representar a intensidade de vibração detectada por uma aranha na posição P_b no instante t e a fonte da vibração está na posição P_a. Com estas notações pode-se usar $I(P_s, P_s)$ para representar a intensidade da vibração gerada pelas aranhas em Com estas notações pode-se usar $I(P_{x_i}, P_{x_i})$ para representar a intensidade da vibração gerada pela aranha x_i na posição da fonte. Esta intensidade de vibração na posição de fonte está correlacionada com a adequação da sua posição $f(P_{x_i})$. O valor de intensidade é calculado na Equação 161:

$$I(P_s, P_s) = log\left(\frac{1}{f(P_s) - C} + 1\right), \tag{161}$$

onde C é uma constante de confiança de valor pequeno de tal forma que todos os valores possíveis fitness são maiores do que C.

Como uma forma de energia, a vibração atenua a distância. Este fenômeno físico é contabilizado na concepção de SSA. Define-se a distância entre a aranha a e b como $D(P_a, P_b)$ e usa-se a distância de Manhattan, calculando a distância conforme a Equação 162:

$$D(P_a, P_b) = |P_a, P_b|, \tag{162}$$

o desvio padrão de todas as posições de aranha ao longo de cada dimensão é representado por σ. Com estas definições, define-se ainda a atenuação de vibração sobre a distância conforme a Equação 163:

$$I(P_a, P_b) = I(P_a, P_b) * exp\left(\frac{-D(P_a, P_b)}{\overline{\sigma} * r_a}\right), \tag{163}$$

Na Equação 163, introduz-se um parâmetro controlado pelo usuário $r_a \in (0, \infty)$. Este parâmetro controla a taxa de atenuação da intensidade de vibração sobre a distância. Quanto maior a r_a, menor a atenuação imposta à vibração. (b) *Padrão de Busca*: Cada iteração pode ser subdivididos nos seguintes sub-passos: avaliação de aptidão, geração de vibração, mudança de máscara, caminhada aleatória e restrição de manipulação.

Os valores de aptidão são avaliados uma vez para cada aranha durante cada iteração. Então estas aranhas geram vibrações em suas posições usando a Equação 161. Depois de

todas as vibrações são gerados, o algoritmo simula o processo de propagação destas vibrações usando Equação 163. Neste processo, cada aranha receberá k diferentes vibrações geradas por outras aranhas onde k é a população de aranhas. As informações recebidas destas vibrações incluem a posição de fonte da vibração e a sua intensidade atenuada. Usa-se V para representar estas k vibrações. Após a recepção de V x_i irá selecionar a vibração mais forte $v_{x_i}^{best}$ de V e comparar sua intensidade com a intensidade da vibração alvo $v_{x_i}^{tar}$ armazenados em sua memória. x_i irá armazenar $v_{x_i}^{best}$ como $v_{x_i}^{tar}$ se a intensidade de $v_{x_i}^{best}$ é maior, e c_s ou o número de iterações desde x_i mudou sua vibração alvo, é zerado, caso contrário, o v^{tar} original retido e c_s é incrementado em um. Usa-se $P_{x_i}^{i}$ e $P_{x_i}^{tar}$ para representar as posições de origem de V e v^{tar} respectivamente, e $i = 1, 2, \ldots, k$.

O algoritmo então manipula x_i para executar uma caminhada aleatória em direção a $v_{x_i}^{tar}$. Aqui utiliza-se uma máscara de dimensão para guiar o movimento. Cada aranha possui uma máscara de dimensão m, que é um vetor binário 0-1 de comprimento D e D é a dimensão do problema de otimização. Inicialmente, todos os valores na máscara são zero. Em cada iteração, a aranha x_i tem uma probabilidade de $1 - p_c^{cs}$ para mudar sua máscara onde $p_c \in (0, 1)$ é um atributo definido pelo usuário que descreve a probabilidade de mudança de máscara. Se a máscara for decidida a ser alterada, cada *bit* do vetor tem uma probabilidade de p_m para ser atribuído com um, e $1 - p_m$ para ser um zero. p_m é também um parâmetro controlado pelo usuário definido em (0, 1). Cada *bit* de uma máscara é alterado independentemente e não tem qualquer correlação com a máscara anterior. No caso que todos os *bit* são zeros, um valor aleatório da máscara é alterado para um. Da mesma forma, um *bit* aleatório é atribuído 0 se todos os valores forem 1.

Após a determinação da máscara de dimensão, é gerada uma nova posição seguinte $P_{x_i}^{fo}$ com base na máscara para s. O valor da j-ésima dimensão da seguinte posição $P_{x_i,j}^{fo}$ é gerado como se segue na Equação 164.

$$P_{x_i,j}^{fo} = \begin{cases} P_{x_i,j}^{tar}, m_{x_i,i} = 0, \\ P_{x_i,j}^{r}, m_{x_i,i} = 1, \end{cases} \tag{164}$$

onde r é um valor inteiro aleatório gerado em $[1, k]$ e $m_{x_i,i}$ representa a j-ésima dimensão da dimensão máscara m da aranha x_i. Aqui o número aleatório r para duas dimensões diferentes com $m_{x_i,i} = 1$ é gerado independentemente. Com o $P_{x_i}^{fo}$ gerado, x_i executa uma caminhada aleatória para a posição. Esta caminhada aleatória é conduzida utilizando a Equação 165.

$$\tilde{P}_{x_i} = P_{x_i} + (P_{x_i} - P_{x_{i,t-1}} * r + P_{x_i}^{fo} - P_{x_i}) * R, \tag{165}$$

onde R é um vetor de números aleatórios com distribuição uniforme entre 0 e 1. Antes de seguir $P_{x_i}^{fo}$, primeiro x_i se move ao longo de sua direção anterior, que é a direção de movimento na iteração anterior. A distância ao longo desta direção é uma porção aleatória do movimento. Então s se aproxima $P_{x_i}^{fo}$ ao longo de cada dimensão com fatores aleatórios gerados em (0, 1). Este fator aleatório para diferentes dimensões é gerado de forma independente. Após essa caminhada aleatória, x_i armazena seu movimento na iteração atual para a próxima iteração. Isso termina a sub-etapa de caminhada aleatória.

A abordagem de reflexão para a manipulação de restrições e produzir uma posição livre de restrição de limite $\tilde{P}_{x_i}$ pela Equação 166 :

$$\tilde{P}_{x_i} = \begin{cases} (max_j - P_{x_i}) * r, \text{se} P_{x_i} > max_j, \\ (P_{x_i} - min_j) * r, \text{se} P_{x_i} < min_j, \end{cases} \tag{166}$$

onde max_j é o limite superior do espaço de pesquisa na j-ésima dimensão e min_j é o limite inferior da dimensão correspondente. r é um número de ponto flutuante aleatório gerado em (0, 1).

Algoritmo de Libélulas

Criada por Mirjalili (2015), origina-se dos comportamentos de enxame estáticos e dinâmicos de libélulas na natureza. Duas fases essenciais de otimização, pesquisa e exploração, são projetados para modelar a interação social das libélulas na navegação, em busca de alimentos, e evitando inimigos com o enxame dinâmica ou estatisticamente (MIRJALILI, 2016a). No Algoritmo 33 é apresentado o pseudocódigo do Algoritmo de Libélulas (*Dragonfly Algorithm* - DA).

Há cinco fatores principais na atualização de posição de indivíduos em enxames. Cada um destes comportamentos é matematicamente modelado como seguinte (MIRJALILI, 2016a): (a) Separação: este fator refere-se ao impedimento de colisão entre os indivíduos na vizinhança. A Equação 167 apresenta a fórmula usada para o cálculo.

$$S_i = -\sum_{n=1}^{N} x_i - x_n, \tag{167}$$

onde x_i é a posição do indivíduo corrente, x_n mostra a posição n-ésimo indivíduo vizinho e N é o número de indivíduos vizinhos.

(b) Alinhamento: indica a correspondência de velocidade dos indivíduos com a de outros indivíduos na vizinhança.

$$A_i = \frac{\sum_{n=1}^{N} V_n}{N}, \tag{168}$$

Algoritmo 33 Pseudocódigo do Algoritmo de Libélulas

Entrada: Parâmetros comuns (Algoritmo 1).
Inicializa os vetores de passo (Δx_i (i = 1, 2, ..., n)).

início
  enquanto *(critério de parada não é satisfeito)* **faça**
    Calcula a aptidão de todas as libélula.
    Atualiza a fonte de alimento e inimigos.
    Atualiza w, s, a, c, f, e.
    Calcula S, A, C, F e E usando as Equação 167 a 171.
    Atualiza o raio dos vizinhos
    se *uma libélula tem pelo menos uma libélula vizinha* **então**
      Atualiza o vetor de velocidade usando a Equação 172.
      Atualiza o vetor de posição usando Equação 173.
    fim
    senão
      Atualiza a posição usando Equação 174.
    fim
    Trata qualquer violação de limitações.
    Avalia cada libélula de acordo com sua nova posição.
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

onde V_n mostra a velocidade do n-ésimo indivíduo vizinho.

(c) Coesão: refere-se a tendência dos indivíduos de manter um centro da massa da vizinhança.

$$C_i = \frac{\sum_{n=1}^{N} x_n}{N} - x_i, \tag{169}$$

onde x_i é a posição do indivíduo atual, N é o número de vizinhanças, e x_n mostra a posição n-ésimo indivíduo vizinho.

(d) Atração: refere-se a tendência dos indivíduos de irem na direção onde há mais comida.

$$F_i = x_{food} - x_i, \tag{170}$$

onde x_i é a posição do indivíduo atual e x_{food} mostra a posição da fonte de alimento.

(e) Distração: indica a evasão na presença de um predador.

$$F_i = x_{enemy} + x_i, \tag{171}$$

onde x_i é a posição do indivíduo atual, e x_{enemy} mostra a posição do inimigo.

Para atualizar a posição das libélulas artificiais em um espaço de busca e simular seus movimentos, dois vetores são considerados: passo (Δx) e a posição (x_i). O vetor passo é análogo ao vetor de velocidade em PSO, e o algoritmo DA é desenvolvido com base na estrutura do algoritmo PSO. O vetor Δx mostra a direção do movimento das libélulas e definido como segue na Equação 172:

$$\Delta \tilde{x}_i = (s * S_i + a * A_i + c * C_i + f * F_i + e * E_i) + w * \Delta x, \tag{172}$$

onde s mostra o peso de separação, S_i indica a separação do i-ésimo indivíduo, a é o peso de alinhamento, A_i é o alinhamento do i-ésimo indivíduo, c indica o peso de coesão, C_i é a coesão do i-ésimo indivíduo , f é o fator de alimento, F_i é a fonte de alimento do i-ésimo indivíduo, e é o fator inimigo, E_i é a posição de inimigo do i-ésimo indivíduo, w é o peso de inércia e t é o contador de iteração . Depois de calcular o vetor de passos, os vetores de posição são calculados como se segue na Equação 173:

$$\tilde{x}_i = x_i + \Delta\tilde{x}_i, \tag{173}$$

com t sendo a iteração corrente.

As libélulas são obrigadas a mudar seus pesos adaptativamente para o trânsito de exploração global para exploração local do espaço de busca. Também se supõe que as libélulas tendem a ver mais libélulas para ajustar o caminho de voo à medida que o processo de otimização progride. Em outras palavras, a área de vizinhança também é aumentada, de forma que o enxame se torna um grupo na fase final de otimização para convergir para o ótimo global. A fonte de alimento e o inimigo são escolhidos das melhores e piores soluções que o enxame inteiro é encontrado até agora. Isso faz convergir para áreas promissoras do espaço de pesquisa e divergência para fora regiões não promissoras do espaço de busca (MIRJALILI, 2016a). Para melhorar a aleatoriedade, o comportamento estocástico e exploração das libélulas artificiais, eles são obrigados a voar ao redor do espaço de busca usando uma caminhada aleatória (voo Levy) quando não há soluções vizinhas. Neste caso, a posição das libélulas é atualizada usando a Equação 174:

$$\tilde{x}_i = x_i + Levy(d) * x_i, \tag{174}$$

onde t é a iteração corrente e j é a dimensão dos vetores de posição. O voo Levy é calculado a partir da Equação 175:

$$Levy(d) = 0.01 * \frac{r_1 * \sigma}{|r_2|^{\frac{1}{\beta}}}, \tag{175}$$

onde r_1, r_2 são dois números aleatórios em [0,1], β é uma constante e σ é calculado pela Equação 176:

$$\sigma = \left(\frac{\Gamma(1+\beta) * sen(\frac{\pi*\beta}{2})}{\Gamma(\frac{1+\beta}{2}) * \beta * 2^{\frac{\beta-1}{2}}} \right), \tag{176}$$

onde $\Gamma(x) = (x-1)!$.

Otimização por Colônia de Bactérias

A Otimização por Colônia de Bactérias (*Bacterial Foraging Optimization* - BFO), proposta por Kevin Passino (2002), é uma técnica inspirada na estratégia de forrageamento da bactéria *E. coli*, que é governada pelos processos: quimiotaxia, a reprodução, dispersão, e *swin*. O Algoritmo 34 a seguir apresenta o pseudo-código do BFO básico, conforme descrito em (Chen et al., 2008). Nele é gerada uma população de S bactérias no espaço de busca, onde cada bactéria representa uma possível solução para o problema de otimização sendo tratado. São executadas N_c etapas de quimiotaxia (*chemotaxis*) ao fim das quais é executada a etapa de reprodução. Isso é repetido N_{re} vezes e, ao fim, é executada a etapa de dispersão, sendo repetido N_{ed} vezes (TEODORO; PARPINELLI; LOPES, 2010).

Por definição, o BFO é um algoritmo de minimização e tem os seguintes parâmetros de controle (TEODORO; PARPINELLI; LOPES, 2010):

- n: número de dimensões do espaço de busca;
- S: número de bactérias;
- N_c: número de etapas de quimiotaxia;
- N_s: número máximo de passos de *swin*;
- N_{re}: número de etapas de reprodução;
- N_{ed}: número de etapas de dispersão;
- P_{ed}: probabilidade de dispersão;
- $C(i), \text{com } i \in 1, 2, ..., S$: comprimento do passo nos movimentos da bactéria;
- Sr: número de bactérias replicadas a cada etapa de reprodução.

O processo de busca do BFO pode ser dividido pelos seguintes passos (TEODORO; PARPINELLI; LOPES, 2010):

(a) Quimiotaxia e *Swin*: No movimento de quimiotaxia a bactéria dá um passo em uma direção aleatória. Caso esse passo representa uma melhora na qualidade da aptidão da bactéria, inicia-se então o movimento *swin*, em que a bactéria se move na mesma direção enquanto a aptidão continuar diminuindo ou até alcançar o número máximo de passos de *swin* (N_s). Isto é feito para cada uma das bactérias, prefixando um $C(i)$. Este movimento da bactéria pode ser representada pela 177:

$$\theta_i(j+1,k,l) = \theta_i(j,k,l) + C(i) * \phi(j), \tag{177}$$

Algoritmo 34 Pseudocódigo da Otimização por Colônia de Bactérias

Entrada: Parâmetros comuns (Algoritmo 1).
Inicializa parâmetros $n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, C(i), \theta_i (i = 1, 2, ..., S)$.
início
  enquanto *(critério de parada não é satisfeito)* **faça**
    para l *até* N_{ed} **faça**
      para k *até* N_{re} **faça**
        para j *até* N_c **faça**
          para i *= 1 até* S **faça**
            Quimiotaxia
            -Calcula a aptidão da i-ésima bactéria.
            -Gere um vetor unitário aleatório entre 0 e 1: Δ_i.
            -Gera nova posição da bactéria (conforme Equação 177).
            -Calcula a aptidão da nova bactéria.
            $m = 0$
            enquanto $m < N_s$ *e* $J(i, j+1, k, l) < J_{last}$ **faça**
              Swin
              -Mova a bactéria mais um passo, de -comprimento C_i, na direção Δ_i (conforme Equação 177).
              -Calcula $J(i, j+1, k, l)$ com o novo $\theta_i(j+1, k, l)$.
              $m = m + 1$
            fim
          fim
        fim
        para i *= 1 até* S **faça**
          Calcula a saúde das bactérias (J_{health_i}), segundo Equação 178.
        fim
        Faz a reprodução substituindo as Sr bactérias com os maiores J_{health} por cópias das Sr bactérias com os menores J_{health}.
      fim
      Faz a dispersão com probabilidade P_{ed} (Equação 179), atribuindo uma posição aleatória para cada bactéria.
    fim
  fim
  Retorna a melhor posição e sua respectiva solução.
fim

onde $C(i)$ é o tamanho da etapa tomada na direção aleatória especificada pela unidade de comprimento de rotação, e $\phi(j)$ é uma direção aleatória.

(b) Reprodução: Na etapa de reprodução as Sr bactérias mais saudáveis são replicadas e introduzidas no mesmo local do espaço busca dos seus "pais", e as Sr bactérias menos saudáveis são eliminadas. A saúde das bactérias é o somatório das aptidões para todos os passos de quimiotaxia realizados. Quanto menor o valor, mais saudável é a bactéria e, consequentemente, melhor é a qualidade da solução que ela representa. O cálculo do J_{health_i} pode ser visto na Equação 178:

$$J_{health_i} = \sum_{j=1}^{N_c+1} J(i, j, k, l), \tag{178}$$

(c) Dispersão: Na etapa de dispersão, cada bactéria pode ser eliminada e substituída por outra gerada aleatoriamente, de acordo com a probabilidade P_{ed}, mostrada em 179.

$$P_{ed} = \frac{f(x_i)}{\sum_{i=1}^{S} f(x_i)}, \tag{179}$$

onde $f(x_i)$ é a aptidão da i-ésima bactéria.