Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

Ômega Grey Wolf Optimizer: Uma melhoria para o Grey Wolf Optimizer adicionando lobos Ômega à caçada

Kevin Andrews Marques Silva

Trabalho de Graduação

Recife
12 de Maio de 2022

Universidade Federal de Pernambuco
Centro de Informática

Kevin Andrews Marques Silva

Ômega Grey Wolf Optimizer: Uma melhoria para o Grey Wolf Optimizer adicionando lobos Ômega à caçada

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Prof. Dr. Paulo Salgado Gomes de Mattos Neto*

Recife
12 de Maio de 2022

A Deus
Aos meus pais, Marcos e Jussara.
À minha irmã, Ana Lívia.
À minha avó, Ivonete.
À minha namorada, Dara.

Agradecimentos

Gostaria de agradecer primeiro a Deus, por me guiar nessa jornada e em minha vida.

Agradeço a minha família, meus pais Marcos e Jussara, por todo suporte e apoio, por todo cuidado que tiveram por mim, à minha avó Ivonete, por me acolher em seu lar durante os períodos difíceis, a minha irmã, Ana Lívia, pelas alegrias nos momentos de dificuldade e a todos os demais familiares que sempre acreditaram em mim.

Agradeço a minha namorada Dara, minha amiga e confidente, que compartilhou comigo os momentos difíceis, foi meu porto seguro e que sempre me incentivou a seguir em frente.

Ao meu orientador, Paulo Salgado, que me inspirou desde o inicio da minha jornada na universidade, sua ajuda foi essencial para escrever esse trabalho.

Aos amigos que fiz na universidade e nos diversos projetos que participei, aos colegas da DINE, do Projeto CIN/Motorola, do Summer Job e das monitorias. Agradeço a Ricardo, Nicholas, Gabriel, Carol, Amanda, Rafael, Pedro, Lívia e aos meus amigos de fora da universidade que me ajudaram a manter a cabeça no lugar, em especial João e Iris.

Não é o mais forte que sobrevive, nem o mais inteligente, mas o que melhor se adapta as mudanças

—CHARLES DARWIN

Resumo

A computação bioinspirada é um campo de estudo que busca compreender padrões da natureza e utilizá-los para resolução de problemas. O *Grey Wolf Optimizer* (GWO) é um algoritmo inspirada pelo comportamento de caça dos lobos cinzas e que pode ser utilizada em problemas de otimização global, ajuste de parâmetros de técnicas de aprendizagem de máquina, processamento de imagens, aplicações médicas e de bioinformática, entre outras.

Este trabalho propõe uma melhoria na área de exploração desse algoritmo, a partir de uma versão modificada do GWO, chamada Ômega Grey Wolf Optimizer (OGWO) que utiliza lobos ômega nos grupos de caça dos lobos cinzas. São executados testes com 22 benchmarks e os resultados são comparados com outras 5 heurísticas, verifica-se que os resultados são competitivos e por vezes superiores aos algoritmos usados para comparação.

Palavras-chave: técnicas de otimização, algoritmos heurísticos, meta-heurísticas, GWO.

Abstract

Bio-inspired computing is a field of study that seeks to understand patterns in nature and use them to solve problems. The *Gray Wolf Optimizer* (GWO) is an algorithm inspired by the hunting behavior of gray wolves and that can be used in global optimization problems, parameter adjustment of machine learning techniques, image processing, medical applications and of bioinformatics, among others.

This work proposes an improvement in the exploration area of this algorithm, from a modified version of the GWO, called Omega Gray Wolf Optimizer (OGWO) that uses omega wolves in the gray wolf hunting groups. Tests are performed with 22 benchmarks and the results are compared with another 5 heuristics, it is verified that the results are competitive and sometimes superior to the algorithms used for comparison.

Keywords: optimization, optimization techniques, heuristic algorithm, metaheuristics, GWO.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Contexto e Motivação

O algoritmo *Grey Wolf Optimizer* (GWO) [1] é inspirado no padrão de comportamento de lobos cinzas, utilizando seu método de caça e sua hierarquia. Uma hierarquia com quatro categorias de lobos é utilizada em um processo de caçar, buscar, cercar e atacar a presa. O GWO usa desse comportamento para encontrar soluções de diferentes problemas, entre eles problemas nas áreas de *Feature Selection* [2], *Global Optimization* , *Machine Learning*, engenharia e design, processamento de imagens, etc [3].

O desempenho do GWO é notável [1], com uma boa fase de exploração, e resultados de explotação na média com outros algoritmos como *Particle Swarm Optimization* (PSO) [4] e *Gravitational Search Algorithm* (GSA) [5], com base nessa premissa, esse trabalho propõe modificações que buscam melhorar o desempenho nessa área sem aumentar o custo computacional.

Este trabalho irá compreender o funcionamento do GWO e propor uma versão com melhorias para combater elitismo de soluções, melhorar a capacidade de exploração do algoritmo e a diversidade de população.

A proposição de um lobo Ômega na hierarquia de caça dos lobos, adiciona um fator extra a exploração, mantendo a diversidade de soluções antes de convergir para a presa, atrelado a isso, um peso relativo na decisão de cada membro da hierarquia será avaliado, em busca de um balanço entre explotação e exploração. Devido a isso o algoritmo será denominado *Ômega Grey Wolf Optimizer* (OGWO). Essas modificações buscarão desenvolver uma outra camada ao GWO, buscando resultados melhores em problemas que exijam uma boa exploração sem perder desempenho, isso será visualizado através de benchmarks.

1.2 Objetivos gerais

O objetivo geral deste trabalho é estudar a heurística do GWO e propor uma versão com melhorias no seu desempenho, principalmente no seu fator de exploração, denominada OGWO.

1.3 Objetivos específicos

- Desenvolver e apresentar a melhoria proposta do GWO;
- Simular o GWO e o OGWO;

- Gerar resultados em benchmarks do GWO, OGWO, e outras heurísticas para comparação;
- Comparar resultados das heurísticas e fazer uma avaliação do desempenho da melhoria proposta;

1.4 Organização do trabalho

- Capítulo 2: No segundo capítulo mergulharemos em uma revisão de trabalhos acadêmicos sobre o GWO e as melhorias que já foram exploradas, bem como as oportunidades que temos.
- Capítulo 3: Será explicado o funcionamento do GWO.
- Capítulo 4: Apresenta o OGWO, explicando como ele funciona e mostrando sua implementação.
- Capítulo 5: Será apresentado e discutido os resultados gerados a partir de benchmarks, também será comparado com os resultados de outras heurísticas.
- Capítulo 6: Apresentação das conclusões do trabalho e discussão das perspectivas de trabalhos futuros.

CAPÍTULO 2

Revisão da Literatura

Inteligência computacional é a teoria, desenvolvimento e aplicação de algoritmos inspirados linguística e biologicamente, a inteligência computacional possui muitas formas, como Computação Neural [6], Computação Nebulosa [7], Computação Evolucionária [8] e Inteligência de Enxame [9].

Algoritmos de inteligência de enxame simulam o comportamento de comunidades ou sistemas como cardumes de peixes, revoada de pássaros, colônias de insetos e rebanhos de animais. Os algoritmos se baseiam em seus comportamentos de bando para se mover dentro do espaço de busca e localizar fontes de comida.

Existem muitos algoritmos de Inteligência de enxame como *Ant Colony Optimization* (ACO) [10], *Particle Swarm Intelligence* (PSO) [11], *Artificial Bee Colony* (ABC) [12], *Krill Herd* (KH) [13], *Grey Wolf Optimizer* (GWO) [1], *Whale optimization Algorithm* (WOA) [14] e muitos outros. Esses Algoritmos tem sido muito usados para resolver problemas contínuos e discretos de otimização [2].

GWO é um desses algoritmos, proposto recentemente por Mirjalili et al. [1] em 2014. O GWO é um algoritmo inspirado no comportamento de caça e na hierarquia social do lobos cinzas na natureza. O processo de busca do GWO ocorrer a partir da criação de uma população de lobos cinzas, uma hierarquia é definida onde as três melhores soluções representam os lobos Alpha, Beta e Delta, e as outras soluções são os lobos Ômega. Cada lobo representa uma solução candidata que é atualizada no processo de busca.

O GWO também apresenta uma simplicidade devido a poucos parâmetros de controle e uma implementação fácil, devido ao seu sucesso o GWO motivou outros pesquisadores a aplicar o algoritmo para diversos problemas de otimização como problemas de otimização global [15], problemas de engenharia elétrica [16], reconhecimento de padrão [17], seleção de variável [18], previsão de vento [19] entre outros.

Os algoritmos de inteligência de enxame também podem possuir pontos fracos, alguns problemas comuns são, ficar preso em máximos locais, convergência prematura, perda de diversidade nas soluções e por isso modificações são propostas por diversos pesquisadores para abordar esses pontos fracos. No caso do GWO, um número de variantes foram propostas para abordar essas fraquezas, o *Grey Wolf Optimizer Evolutionary Population Dynamics* (GWO-EPD) [20] integra um dinâmica evolucionária de população (EPD) dentro do algoritmo do GWO, o EPD é usado para remover as piores soluções e posicionar elas próximo das três melhores, ela apresenta resultados competitivos mas também sofre de perda de diversidade.

O *Weigthed Distance Grey Wolf Optimizer* wdGWO [21] foi proposto por Malik et al. onde uma média ponderada das melhores soluções é utilizada ao invés de uma média simples.

O *Hybrid Grey Wolf Optimizer* HGWO [22] propõe um GWO que utiliza mutação e ope-

radores de crossover. Mesmo ainda sofrendo com desbalanceamento entre exploração e explotação e não apresentando os melhores resultado quando utilizado em funções compostas, ele apresenta uma ótima performance.

O *Improved Grey Wolf Optimizer* I-GWO [23] melhora a estratégia de caça utilizando *Dimension Learning-based Hunting* (DLH), gerando duas soluções antes de mover o lobo de sua posição para uma posição melhor.

Algumas outras variantes foram desenvolvidas com propósitos em mente, para evitar ótimos locais e acelerar a velocidade de convergência como o *Quasi-Oppositional Grey Wolf Optimizeer* QOGWO [24] e o *Exploration-Enhanced Grey Wolf Optimizer* EEGWO [25].

CAPÍTULO 3

Grey Wolf Optimizer

O Grey Wolf Optimizer (GWO) é uma técnica de inteligência de enxame com inspiração no comportamento social dos lobos cinzas, principalmente na sua hierarquia social e na sua técnica de caçada.

Em cada matilha existe uma hierarquia a ser seguida, sendo o Alpha o lobo que lidera essa matilha na caçada, o segundo membro mais importante é o Beta, que age como líder quando o Alpha está incapacitado, essa hierarquia procede com o Delta até o Ômega.

Sua técnica de caçada é dividida em três principais passos, o primeiro tem relação com rastrear a presa, se aproximando dela, o segundo passo é perseguir essa presa, à cercando até que pare de mover, e o passo final é atacar a presa que foi cercada.

Esse comportamento foi modelado matematicamente de forma a poder ser utilizado no algoritmo, a hierarquia representa as soluções com o melhor fitness, assim a solução com o melhor fitness representa o Alpha (α), a segunda melhor solução é representada pelo Beta (β), a terceira melhor solução é representada pelo Delta (δ), o resto das soluções serão o Omega (ω).

Além disso o comportamento de caçada pode ser dividido em alguns passos que serão apresentados a seguir.

3.1 Cercar a presa

A fase de cercar a presa pode ser modelada da seguinte forma:

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \tag{3.1}$$

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|, \tag{3.2}$$

em que $\vec{X}_p$ posição vetorial da presa, $\vec{X}$ indica a posição vetorial do lobo, t é a iteração atual e $\vec{C}$ e $\vec{A}$ são vetores de coeficientes.

Os vetores A e C são calculados da seguinte forma:

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r_1} - \vec{a}, \tag{3.3}$$

$$\vec{C} = 2 \cdot \vec{r_2}, \tag{3.4}$$

em que $\vec{r_1}$ e $\vec{r_2}$ são vetores aleatório em [0,1], e o elementos do vetor $\vec{a}$ decrescem linearmente de 2 até 0.

3.2 Caçar a presa

Para modelar o comportamento de caçada supõe-se que os lobos α, β e γ possuem o melhor conhecimento sobre a posição da presa, então os outros lobos são guiados pelos três, seguindo as seguintes equações.

$$\vec{D_\alpha} = |\vec{C_1} \cdot \vec{X_\alpha} - \vec{X}(t)|, \tag{3.5}$$

$$\vec{D_\beta} = |\vec{C_2} \cdot \vec{X_\beta} - \vec{X}(t)|, \tag{3.6}$$

$$\vec{D_\delta} = |\vec{C_3} \cdot \vec{X_\delta} - \vec{X}(t)|, \tag{3.7}$$

em que $\vec{C_1}$, $\vec{C_2}$, $\vec{C_3}$ são calculados pela equação (3.4).

$$\vec{X_1}(t) = \vec{X_\alpha}(t) - \vec{A_1} \cdot \vec{D_\alpha}(t), \tag{3.8}$$

$$\vec{X_2}(t) = \vec{X_\beta}(t) - \vec{A_2} \cdot \vec{D_\beta}(t), \tag{3.9}$$

$$\vec{X_3}(t) = \vec{X_\delta}(t) - \vec{A_3} \cdot \vec{D_\delta}(t), \tag{3.10}$$

em que $\vec{X_\alpha}$, $\vec{X_\beta}$ e $\vec{X_\delta}$ são as três melhores soluções da iteração t, $\vec{A}$ é calculado como na equação (3.3) e $\vec{D}$ é definido pela equação (3.5).

$$\vec{X}(t+1) = \frac{\vec{X_1}(t) + \vec{X_2}(t) + \vec{X_3}(t)}{3}, \tag{3.11}$$

até que finalmente $\vec{X}(t+1)$ atualiza sua posição baseada na posição das melhores soluções como localizado na equação (3.7).

3.3 Atacar a presa

Por último, o processo de atacar a presa ocorre quando a constante a que decresce linearmente se aproxima de 0. Durante esse passo os lobos mudam sua posição para uma posição aleatória entre a posição atual deles e a suposta posição da presa.

Esses passos são repetidos utilizando a população inicial gerada aleatoriamente, a função de fitness avalia a posição dos lobos até que a condição de parada seja satisfeita, nesse caso a condição de parada será chegar a um número predefinido de iterações.

Algorithm 1 Pseudocódigo do algoritmo GWO

Inicializa a população de lobos cinzas $X_i(i = 1,2...n)$
Inicializa a, A e C
Calcula o fitness de cada agente de busca
X_α =a melhor solução
X_β =a segunda melhor solução
X_δ =a terceira melhor solução
while t < Nº máximo de iterações **do**
    for cada agente **do**
        Atualiza a posição do agente atual com a equação (3.7)
    end for
    Atualiza a, A e C
    Calcula o fitness de todos os agentes de busca
    Atualiza X_α, X_β e X_δ
    $t = t + 1$
end while
Retorna X_α

CAPÍTULO 4

OGWO

Mesmo que o GWO seja simples e possua diversas aplicações, ele sofre com a falta de diversidade populacional, desbalanceamento entre exploração e explotação e convergência prematura.

Como os lobos que lideram a matilha são sempre as três melhores soluções, isso cria um ambiente de elitismo que pode ocasionar em um aprisionamento em um ótimo local, pensando nisso algumas modificações são propostas, e a principal dela é a inclusão de um Ômega como um dos guias da matilha, vamos chamar a versão modificada de OGWO (Omega Grey Wolf Optimizer), as melhorias do OGWO inclui além desta já mencionada, a adição de pesos a decisão dos lobos que guiam a matilha.

No momento de caçar a presa, um membro é selecionado aleatoriamente para compor o grupo de lobos para cada lobo ômega, representando um 4º membro significativo para aquele lobo, isso ajuda a compor variedade a solução e dar aos lobos um senso de individualidade maior.

A seguinte equação representa a adição do 4º lobo:

$$\vec{D_\omega} = |\vec{C_4} \cdot \vec{X_\omega} - \vec{X}(t)|, \tag{4.1}$$

em que $\vec{C_4}$ é calculado pela equação (3.4),

$$\vec{X_4}(t) = \vec{X_\omega}(t) - \vec{A_4} \cdot \vec{D_\omega}(t), \tag{4.2}$$

em que $\vec{X_\omega}$ é uma solução escolhida aleatoriamente na iteração t, $\vec{A}$ é calculado como a equação (3.3) e $\vec{D}$ é definido pela equação (3.5).

A segunda modificação relaciona-se com o peso das decisões dos lobos que estão guiando a matilha, apesar da adição de um novo membro ao grupo de lobos guias, a decisão do Alpha ainda deve ter mais relevância, então a equação (3.7) passa a ser fornecida a partir de uma media ponderada do fitness das soluções $\vec{X_\alpha}$, $\vec{X_\beta}$, $\vec{X_\delta}$ e $\vec{X_\omega}$ como mostrado na equação (4.3)

$$\vec{X}(t+1) = \frac{F_1 \cdot \vec{X_1}(t) + F_2 \cdot \vec{X_2}(t) + F_3 \cdot \vec{X_3}(t) + F_4 \cdot \vec{X_4}(t)}{F_1 + F_2 + F_3 + F_4}, \tag{4.3}$$

sendo F_1,F_2,F_3 e F_4 o fitness relativo a cada uma dessas soluções.

Algorithm 2 Pseudocódigo do algoritmo OGWO

Inicializa a população de lobos cinzas $X_i(i = 1,2...n)$
Inicializa a, A e C
Calcula o fitness de cada agente de busca
X_α =a melhor solução
X_β =a segunda melhor solução
X_δ =a terceira melhor solução
X_ω =qualquer solução
while t < Nº máximo de iterações **do**
    for cada agente **do**
        Atualiza a posição do agente atual com a equação (4.3)
    end for
    Atualiza a, A e C
    Calcula o fitness de todos os agentes de busca
    Atualiza X_α, X_β e X_δ
    $t = t + 1$
end while
Retorna X_α

CAPÍTULO 5

Validação e Experimentos

Nessa seção, iremos avaliar a performance que foi proposta no OGWO através de vários testes de função.

A implementação utilizará a suíte Evolopy [26] que contem 23 funções e outros 14 algoritmos de otimização, as funções incluem unimodal, multimodal e multimodal de dimensão fixa.

As funções de benchmark serão avaliadas em 20 execuções independentes, em cada execução o máximo de iterações será 100 com população de tamanho 100.

Os experimentos forma performados em uma CPU, Intel(R) Core(TM) i5-8300H CPU de 2.30GHz com 16gb de memória ram e Python 3.9.0 foi utilizado para programação. Os gráficos utilizados representam o fitness médio das 20 execuções de cada um dos algoritmos que foi usado para medir a performance e podem ser encontrados nas imagens 5.1, 5.2, 5.3.

Os resultados do OGWO foram comparados com as meta-heurísticas como GA, PSO, MVO, BAT e GWO, utilizando os parâmetros recomendados como visto na tabela Tabela 5.1, esses valores já foram experimentados em estudos prévios como [27] [28].

Das 23 funções clássicas de benchmark [29] [30] , 22 foram utilizadas, elas incluem funções unimodais (F1 – F7), multimodais (F8 – F12), multimodais com dimensão fixa (F14 – F23), as funções são apresentadas na tabela Tabela 5.2.

5.1 Desempenho de Exploração e Explotação

A partir dos resultados presentes na tabela Tabela 5.3 nota-se que o OGWO tem a melhor performance em 7 das funções de teste, porém tendo resultados bem competitivos nos outros casos.

Nas funções unimodais (F1-F7) ele obtém o melhor resultado em 2 das 7 funções, e tem o segundo lugar nas outras 5 funções. As funções unimodais são ideais para testar convergência, pois elas não tem soluções locais.

Observando as curvas de convergência, principalmente a Tabela 5.1(c) e a Tabela 5.1(d) nota-se que o OGWO consegue convergir mais rapidamente que o GWO, ficando com o melhor resultado nas funções F3 e F5, nas outras funções, seu resultado supera todos os outros algoritmos com exceção do GWO..

Podemos concluir que, o OGWO tem resultados competitivos e uma convergência acelerada, porém ele não se mantém com os melhores resultados em todas as funções, isso revela um potencial que pode ser explorado no algoritmo.

Algoritmo	Parâmetros
GA	Probabilidade de Crossover = 0.9
	Probabilidade de Mutação = 0.01
	Mecanismo de Seleção = Roleta
PSO	Constantes de aceleração = [2.1,2.1]
	Pesos de Inércia [0.9,0.6]
MVO	Mínima probabilidade de existência de buraco de minhoca = 0.2
	Máxima probabilidade de buraco de minhoca = 1.0
BAT	Volume = 0.5
	Taxa de Pulso = 0.5
	Frequência mínima = 0
	Frequência máxima = 1
GWO	a diminui linearmente de 2 à 0
OGWO	a diminui linearmente de 2 à 0

Tabela 5.1 Parâmetros de configuração

5.2 Prevenção de ótimos locais

As funções multimodais apresentam múltiplas soluções locais, o que nos permite verificar a performance dos algoritmos em termos de prevenção de ótimos locais, também é possível verificar o balanceamento entre exploração e explotação, os resultados de (F8 - F23), nota-se que o OGWO teve suas melhores performances nas funções F9, F16, F18, F19 e F23, performando melhor ou igual o GWO em 1/3 dos casos, já comparado com os outros algoritmos ele performar melhor 75% das vezes.

Nas funções multimodais o OGWO tem resultados consideráveis, ele consegue suprir alguns pontos fracos do GWO em funções específicas, como por exemplo na função F9 onde o OGWO permanece com melhores resultados durante quase toda execução. Observando também os resultados, pode-se inferir que o balanceamento entre exploração e explotação do OGWO pode ser um ponto de melhoria, para alcançar resultados finais mais precisos.

5.3 Convergência

A partir das figuras 5.1, 5.2, 5.3, é possível notar que o OGWO converge mais rápido que o GWO na maioria das vezes, isso acontece principalmente nas funções unimodais, nas funções multimodais o OGWO converge mais rápido 80% das vezes.

Sua convergência possui uma alta flutuação durante as iterações, convergindo rapidamente nas iterações iniciais, isso representa a população de lobos colaborando para encontrar os resultados atualizando suas posições, a alta flutuação representa uma exploração forte, porém o lobo Ômega adicionado ao grupo de caça tem um impacto negativo na iterações finais, contudo

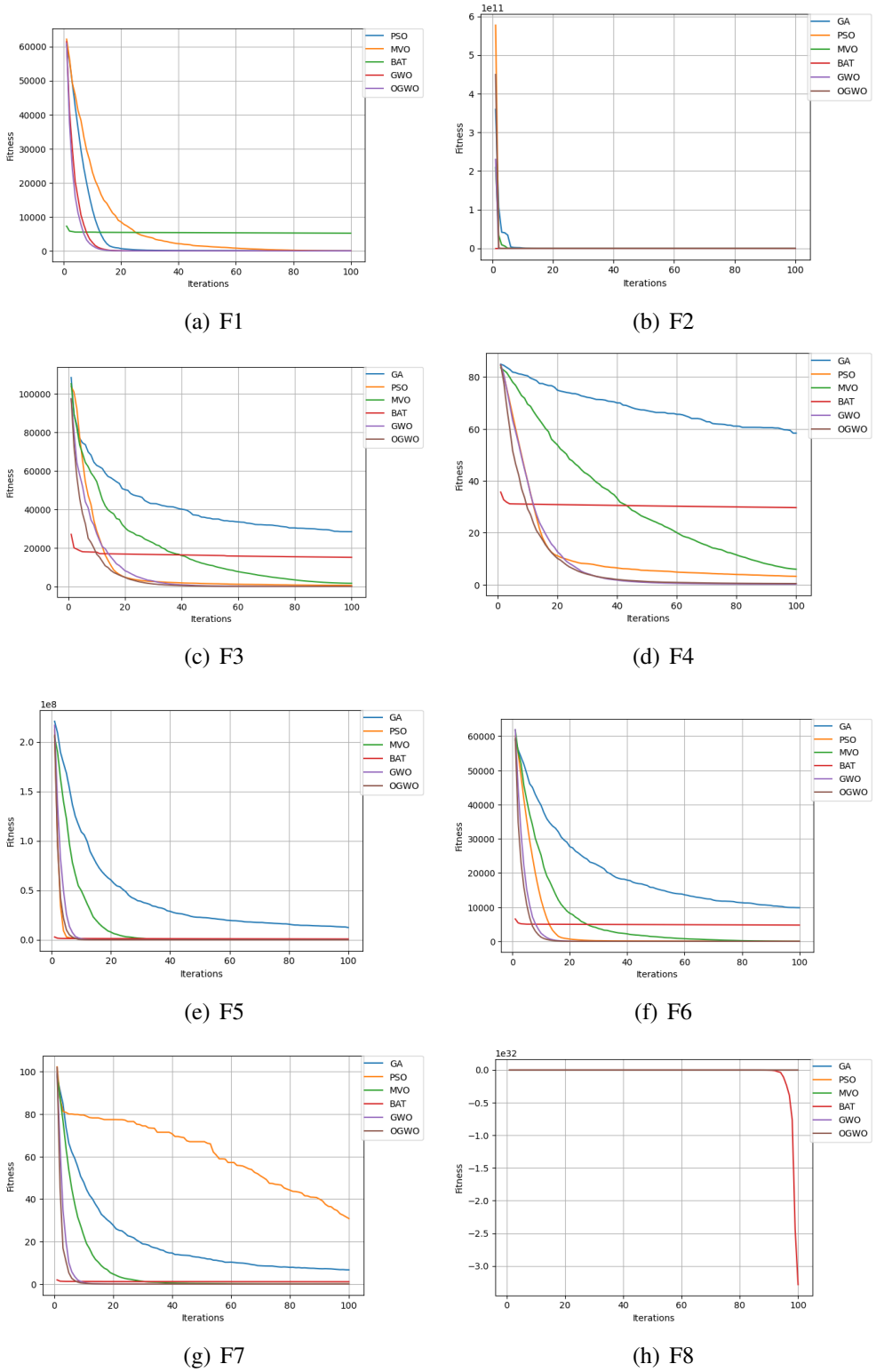
o OGWO apresenta um bom comportamento de convergência.

F	Função	Dim	$f$min
F1	$\sum_{i=1}^{n} {x_i}^2$	30	0
F2	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x+i \rvert$	30	0
F3	$\sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	0
F4	$max_i\{\lvert x+i \rvert, 1 <= i <= n\}$	30	0
F5	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	0
F6	$\sum_{i=1}^{n} ([x_i + 0.5])^2$	30	0
F7	$\sum_{i=1}^{n} i x_i^4 + random[0,1]$	30	0
F8	$\sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$	30	-4189.829
F9	$\sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	30	0
F10	$-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i) + 20 + e)$	30	0
F11	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	30	0
F12	$\frac{\pi}{n}\{10sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_i + 1)] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	0
F14	$(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$	2	1
F15	$\sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	0.00030
F16	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	-1.0316
F17	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cosx_1 + 10$	2	0+3981
F18	$[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_i^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 + 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 46x_2 - 36x_1x_2 + 27x_2^2)]$	2	3
F19	$-\sum_{i=1}^{4} C_i exp(-\sum_{j=i}^{3} a_{ij}(x_j - p_{ij})^2)$	3	-3.86
F20	$-\sum_{i=1}^{4} C_i exp(-\sum_{j=i}^{6} a_{ij}(x_j - p_{ij})^2)$	6	-3+32
F21	$-\sum_{i=10}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	-10.1532
F22	$-\sum_{i=10}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	-10.4028
F23	$-\sum_{i=10}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	-10.5363

Tabela 5.2 Funções de benchmark

F	GA	PSO	MVO	BAT	GWO	OGWO
F1	1.037E+04	2.976E+01	1.171E+01	4.723E+04	**5.152E-06**	1.943E-04
F2	5.642E+01	2.643E+01	2.409E+01	2.811E+01	**3.230E-04**	5.536E-04
F3	2.842E+04	5.846E+02	1.662E+03	1.512E+04	1.835E+01	**1.709E+01**
F4	5.836E+01	3.221E+00	5.958E+00	2.967E+01	**2.131E-01**	4.449E-01
F5	1.250E+07	1.596E+04	2.553E+02	8.649E+05	3.036E+01	**2.858E+01**
F6	9.837E+03	2.995E+01	9.980E+00	4.750E+03	**6.764E-01**	1.322E+00
F7	6.688E+00	3.081E+01	5.562E-02	1.088E+00	**5.592E-03**	6.394E-03
F8	-6.730E+03	**-3.869E+03**	-7.665E+03	-3.285E+03	-5.888E+03	-5.041E+03
F9	1.562E+02	2.946E+02	1.24E+02	5.826E+01	4.122E+01	**3.532E+01**
F10	1.762E+01	5.226E+00	2.663E+00	1.126E+01	**4.705E-04**	2.91E-03
F11	9.591E+01	4.783E+00	1.089E+00	4.541E+01	**1.309E-02**	1.85E-02
F12	1.007E+07	1.272E+00	3.14E+00	7.672E+04	**4.241E-02**	1.255E-01
F14	1.074E+00	1.196E+00	**1.045E+00**	5.024E+00	2.725E+00	2.325E+00
F15	2.108E-03	2.345E-03	2.734E-03	2.985E-03	**1.557E-03**	2.242E-03
F16	-1.026E+00	**-1.031E+00**	**-1.031E+00**	-9.908E-01	**-1.031E+00**	**-1.031E+00**
F17	4.035E-01	3.978E-01	3.978E-01	3.978E-01	3.98E-01	**3.981E-01**
F18	3.166E+00	**3E+00**	**3E+00**	**3E+00**	**3E+00**	**3E+00**
F19	-	-3.862E+00	-3.862E+00	-3.862E+00	**-3.861E+00**	-3.862E+00
F20	-3.150E+00	**-3.276E+00**	-3.23E+00	-3.256E+00	-3.268E+00	-3.253E+00
F21	-	-5.771E+00	-5.114E+00	-6.252E+00	**-9.40E+00**	-8.57E+00
F22	-	-9.387E+00	-8.912E+00	-6.441E+00	**-9.751E+00**	-9.649E+00
F23	-	-8.838E+00	-8.178E+00	-7.726E+00	**-1.025E+01**	**-1.025E+01**

Tabela 5.3 Comparação da média dos resultados obtidos com as funções de benchmark, em negrito são destacados os melhores resultados



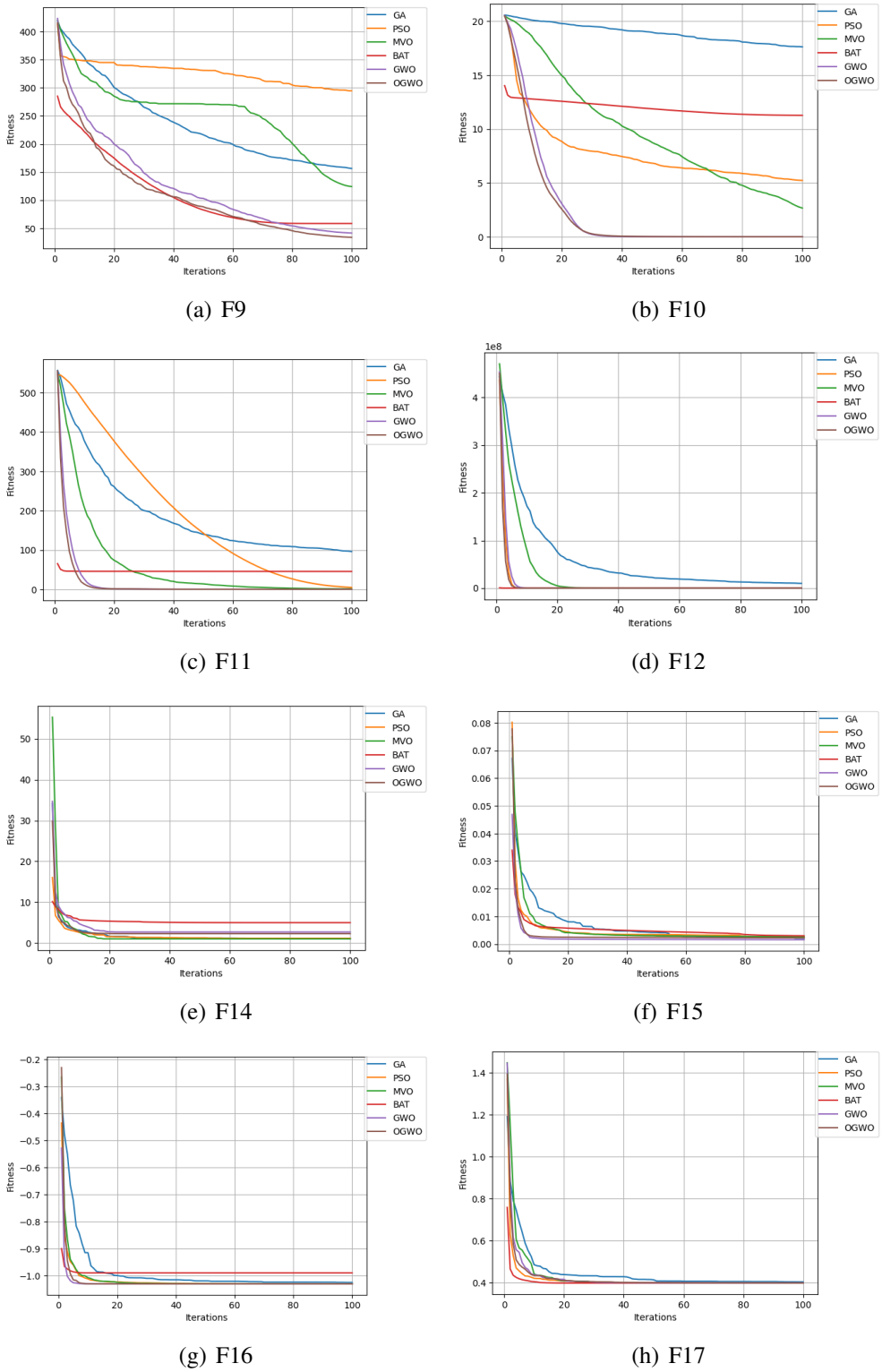
(a) F1 (b) F2

(c) F3 (d) F4

(e) F5 (f) F6

(g) F7 (h) F8

Figura 5.1 Curva de convergência para funções de benchmark F1 - F8



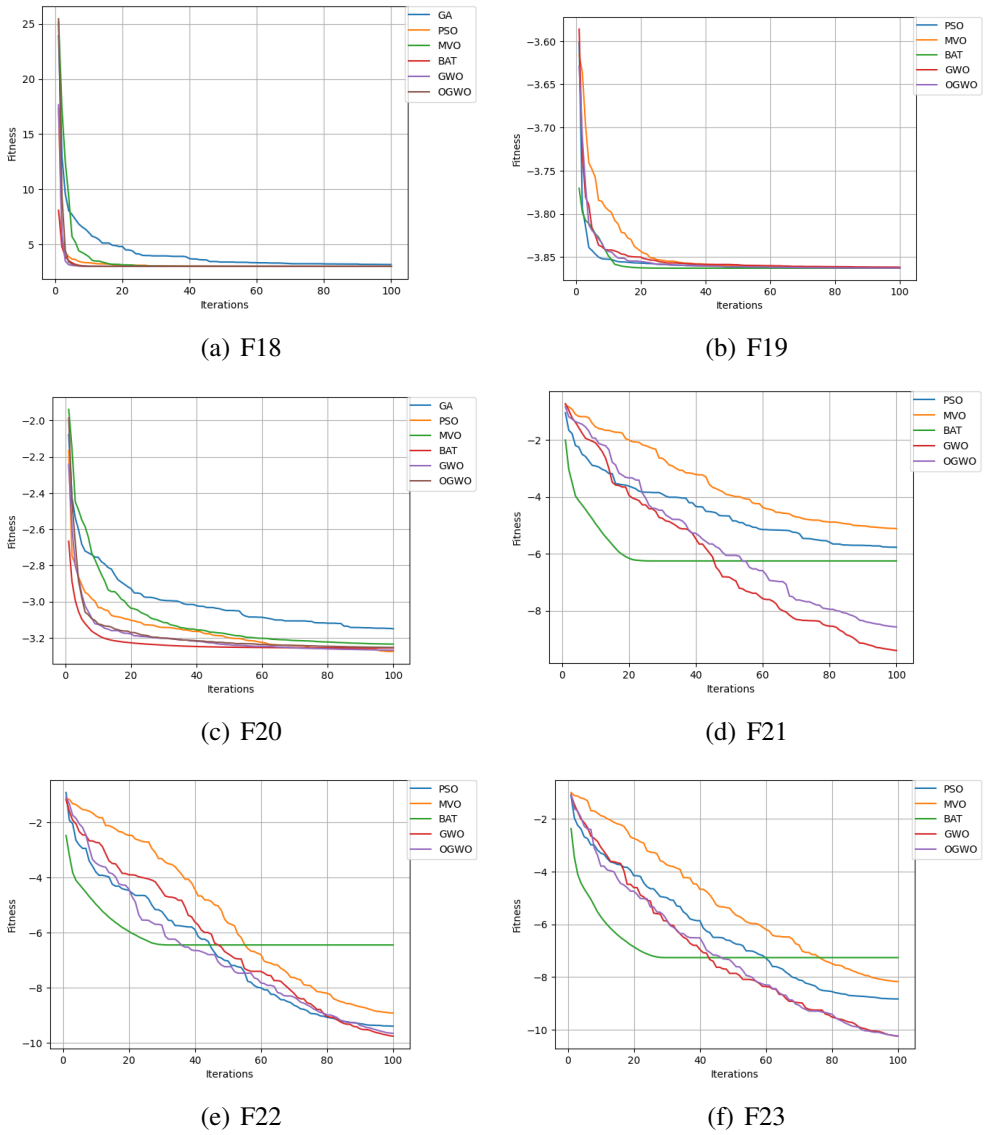
(a) F9 (b) F10

(c) F11 (d) F12

(e) F14 (f) F15

(g) F16 (h) F17

Figura 5.2 Curva de convergência para funções de benchmark F9 - F17



(a) F18 (b) F19

(c) F20 (d) F21

(e) F22 (f) F23

Figura 5.3 Curva de convergência para funções de benchmark F18 - F23

CAPÍTULO 6

Conclusão

Esse trabalho propôs uma melhoria na performance do trabalho do GWO. O principal objetivo foi buscar um maior equilíbrio entre exploração e explotação, melhorando a velocidade de convergência e evitando máximos locais. Após a proposta de modificações no código do GWO, o OGWO foi comparado com outras meta-heurísticas em 22 funções de benchmark. Os resultados foram discutidos e analisados quanto a exploração, explotação, prevenção de ótimos locais e velocidade de convergência.

Os resultados mostraram que, o OGWO mostrou uma velocidade de convergência maior e resultados competitivos porém seus resultados finais não superaram os do GWO 70% das vezes, mostrando que ainda há um desbalanceamento entre exploração e explotação, isso nos permite ver um potencial para o algoritmo que pode ser melhorado para aproveitar melhor seus resultados.

6.1 Trabalhos Futuros

Para trabalhos futuros, um estudo mais aprofundado do impacto de cada uma das mudanças utilizadas pelo GWO pode nos trazer uma melhor percepção do desempenho do algoritmo, buscando também ajustar os parâmetros incluídos como os pesos da média ponderada, e o número de adições de lobo ômega ao grupo de caça.

A aplicação do OGWO em um problema real expande o conhecimento das capacidades do algoritmo e também nos permite provar a aplicabilidade do mesmo.

Referências Bibliográficas

[1] Seyedali Mirjalili, Seyed Mohammad Mirjalili e Andrew Lewis. "Grey wolf optimizer". Em: *Advances in engineering software* 69 (2014), pp. 46–61.

[2] Qasem Al-Tashi et al. "A review of grey wolf optimizer-based feature selection methods for classification". Em: *Evolutionary machine learning techniques* (2020), pp. 273–286.

[3] Hossam Faris et al. "Grey wolf optimizer: a review of recent variants and applications". Em: *Neural computing and applications* 30.2 (2018), pp. 413–435.

[4] Riccardo Poli, James Kennedy e Tim Blackwell. "Particle swarm optimization". Em: *Swarm intelligence* 1.1 (2007), pp. 33–57.

[5] Esmat Rashedi, Hossein Nezamabadi-Pour e Saeid Saryazdi. "GSA: a gravitational search algorithm". Em: *Information sciences* 179.13 (2009), pp. 2232–2248.

[6] Simon Haykin. *Redes neurais: princıpios e prática*. Bookman Editora, 2001.

[7] Lotfi A Zadeh. "Fuzzy logic". Em: *Computer* 21.4 (1988), pp. 83–93.

[8] YJ Cao e QH Wu. "Evolutionary programming". Em: *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*. IEEE. 1997, pp. 443–446.

[9] James Kennedy. "Swarm intelligence". Em: *Handbook of nature-inspired and innovative computing*. Springer, 2006, pp. 187–219.

[10] Marco Dorigo, Mauro Birattari e Thomas Stutzle. "Ant colony optimization". Em: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.

[11] Russell Eberhart e James Kennedy. "Particle swarm optimization". Em: *Proceedings of the IEEE international conference on neural networks*. Vol. 4. Citeseer. 1995, pp. 1942–1948.

[12] Dervis Karaboga. "Artificial bee colony algorithm". Em: *scholarpedia* 5.3 (2010), p. 6915.

[13] Amir Hossein Gandomi e Amir Hossein Alavi. "Krill herd: a new bio-inspired optimization algorithm". Em: *Communications in nonlinear science and numerical simulation* 17.12 (2012), pp. 4831–4845.

[14] Seyedali Mirjalili e Andrew Lewis. "The whale optimization algorithm". Em: *Advances in engineering software* 95 (2016), pp. 51–67.

[15] Aijun Zhu et al. "Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC". Em: *Journal of Systems Engineering and Electronics* 26.2 (2015), pp. 317–328.

[16] Ahmed Fathy e Almoataz Y Abdelaziz. “Grey wolf optimizer for optimal sizing and siting of energy storage system in electric distribution network”. Em: *Electric Power Components and Systems* 45.6 (2017), pp. 601–614.

[17] Belkacem Mahdad e K Srairi. “Blackout risk prevention in a smart grid based flexible optimal strategy using Grey Wolf-pattern search algorithms”. Em: *Energy Conversion and Management* 98 (2015), pp. 411–429.

[18] Eid Emary et al. “Feature subset selection approach by gray-wolf optimization”. Em: *Afro-European conference for industrial advancement*. Springer. 2015, pp. 1–13.

[19] Jingjing Song, Jianzhou Wang e Haiyan Lu. “A novel combined model based on advanced optimization algorithm for short-term wind speed forecasting”. Em: *Applied energy* 215 (2018), pp. 643–658.

[20] Shahrzad Saremi, Seyedeh Zahra Mirjalili e Seyed Mohammad Mirjalili. “Evolutionary population dynamics and grey wolf optimizer”. Em: *Neural Computing and Applications* 26.5 (2015), pp. 1257–1263.

[21] Mahmad Raphiyoddin S Malik, E Rasul Mohideen e Layak Ali. “Weighted distance grey wolf optimizer for global optimization problems”. Em: *2015 IEEE international conference on computational intelligence and computing research (ICCIC)*. IEEE. 2015, pp. 1–6.

[22] T Jayabarathi et al. “Economic dispatch using hybrid grey wolf optimizer”. Em: *Energy* 111 (2016), pp. 630–641.

[23] Mohammad H. Nadimi-Shahraki, Shokooh Taghian e Seyedali Mirjalili. “An improved grey wolf optimizer for solving engineering problems”. Em: *Expert Systems with Applications* 166 (2021), p. 113917. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113917. URL: https://www.sciencedirect.com/science/article/pii/S0957417420307107.

[24] Dipayan Guha, Provas Kumar Roy e Subrata Banerjee. “Load frequency control of large scale power system using quasi-oppositional grey wolf optimization algorithm”. Em: *Engineering Science and Technology, an International Journal* 19.4 (2016), pp. 1693–1713.

[25] Wen Long et al. “An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization”. Em: *Engineering Applications of Artificial Intelligence* 68 (2018), pp. 63–80.

[26] Hossam Faris et al. “EvoloPy: An Open-source Nature-inspired Optimization Framework in Python.” Em: *IJCCI (ECTA)*. 2016, pp. 171–177.

[27] Seyedali Mirjalili. “How effective is the Grey Wolf optimizer in training multi-layer perceptrons”. Em: *Applied Intelligence* 43.1 (2015), pp. 150–161.

[28] Gai-Ge Wang et al. “Hybridizing harmony search algorithm with cuckoo search for global numerical optimization”. Em: *Soft Computing* 20.1 (2016), pp. 273–285.

[29] Jason G Digalakis e Konstantinos G Margaritis. “On benchmarking functions for genetic algorithms”. Em: *International journal of computer mathematics* 77.4 (2001), pp. 481–506.

[30] Marcin Molga e Czesław Smutnicki. “Test functions for optimization needs”. Em: *Test functions for optimization needs* 101 (2005), p. 48.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).