Metaheurística Inspirada no Comportamento do Corvo com Multiestratégias de Aprendizagem Baseada em Oposição

Fábio Augusto Procopio de Paiva*
IFRN, Campus Parnamirim, Brasil
fabio.procopio@ifrn.edu.br

Alexya Ferreira de Santana Brito
IFRN, Campus Parnamirim, Brasil
alexya.s@academico.ifrn.edu.br

Emanuel Lázaro Porpino Campos
IFRN, Campus Parnamirim, Brasil
emanuel.lazaro@academico.ifrn.edu.br

Rafaela Ribeiro Marques dos Santos
IFRN, Campus Parnamirim, Brasil
marques.rafaela@academico.ifrn.edu.br

RESUMO

Frequently, metaheuristics algorithms are used in many scientific and engineering problems optimization. However, these metaheuristics come across an impossibility to escape from optimal solutions. To equilibrate the balancing between exploitation and exploration, several proposals, inspired in the behavior of animal groups, have been developed. A recent metaheuristic is Crow Search Algorithm (CSA). It is inspired in the intelligent behavior of crows. This paper presents a novel CSA variant combined with two strategies opposed to based-learning called Elite Opposition-Based Learning and Generalized Opposition-Based Learning. To evaluate the performance of the proposed algorithm we executed two experiment groups that utilized twelve reference functions. In the first group, the proposal was compared to a CSA variant and other three algorithms. For the second experiment group, we compared to other four algorithms and more one variant. After the execution of experiments, we observed that the obtained numerical results showed a superiority when compared to the other algorithms.

KEYWORDS

Busca Corvo, *Crow Search Algorithm*, CSA, EOBL, GOBL

1 INTRODUÇÃO

A otimização consiste na busca pela melhor solução dentre todas aquelas que são consideradas viáveis. Para resolver problemas dessa espécie, as metaheurísticas vêm sendo vastamente utilizadas em aplicações do cotidiano como computação quântica [1], estratégias de mercado [2], roteamento de veículos [3] etc.

Diversas metaheurísticas são inspiradas na natureza e, nos últimos anos, muitas delas têm estudado o comportamento de animais que vivem em grupos como o *Particle Swarm Optimization* (PSO) [4], o *Fish School Search* (FSS) [5], o *Firefly Algorithm* (FA) [6], o *Grey Wolf Optimizer* (GWO) [7] e outras.

O Algoritmo de Busca Corvo (*Crow Search Algorithm* – CSA) [8] é uma dessas metaheurísticas que é inspirado no comportamento do pássaro corvo. CSA é um algoritmo populacional que trabalha com a ideia de que os corvos armazenam o excesso de comida em esconderijos e, quando a comida é necessária, eles retornam à fonte do alimento para recuperá-la. O CSA tem sido utilizado em aplicações como ressonância magnética [9], diagnóstico da doença de Alzheimer [10] e diversos problemas de engenharia [11].

Quando uma metaheurística é projetada, é importante considerar que deve existir um equilíbrio eficiente entre a diversificação (exploração do espaço de busca) e a intensificação (exploração das melhores soluções obtidas). Nesse sentido, uma alternativa interessante é construir algoritmos que possam ser combinados com outras estratégias de busca, a fim de melhorar o desempenho geral.

Algumas variantes híbridas foram apresentadas a fim de melhorar a performance do CSA. O *Sine Cosine Crow Search Algorithm* (SCCSA) [12] é o resultado da combinação entre CSA com o *Sine Cosine Algorithm* [13] mesclando os conceitos e operadores de cada algoritmo. O *Grey Wolf Optimization and Crow Search Algorithm* (GWOCSA) [14] é uma combinação do CSA com o GWO dando origem à hibridação baseada no comportamento de ambos os grupos. Além dessas variantes híbridas, em [15] é proposta a aplicação dos Mapas Caóticos para balancear a intensificação e a diversificação do algoritmo, alterando o valor de perturbação aleatória e a probabilidade de consciência. Uma outra estratégia, que também vem sendo utilizada em metaheurísticas, é a Aprendizagem Baseada em Oposição (*Opposition Based Learning* – OBL) [16], além de suas variantes.

Este trabalho apresenta uma nova variante do CSA que consiste na inclusão da *Elite Opposition Based-Learning* (EOBL) [17] e da *Generalized Opposition Based-Learning* (GOBL) [18] no algoritmo original. Considerando que uma das características marcantes das metaheurísticas é o balancemento entre diversificação e intensificação, GOBL foi utilizada para melhorar a exploração do espaço de busca, enquanto EOBL para a exploração das melhores soluções obtidas. Dez funções de referência tradicionais são utilizadas para comparar a proposta apresentada com outros algoritmos como *Genetic Algorithm* (GA) [19], *Bat Algorithm* (BA) [20], *Dragonfly Algorithm* (DA) [21], GWO, PSO, CSA e as variantes GWOCSA e *Crow Particle Optimization Algorithm* (CPO) [22]. Após os experimentos, os resultados da variante proposta apresentaram resultados promissores.

O trabalho está organizado como segue: na Seção 2, são apresentados o CSA, a OBL e suas variantes; na seção seguinte, a proposta deste trabalho é apresentada, que consiste na combinação do CSA com EOBL e GOBL. A Seção 4 apresenta os experimentos realizados e a discussão dos resultados. Na última seção, são apresentadas as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para um melhor entendimento da variante proposta, esta seção tem como objetivo apresentar informações sobre o Algoritmo Busca Corvo e a Aprendizagem Baseada em Oposição.

2.1 Algoritmo de Busca Corvo

O corvo é uma ave da família *Corvidae* que vive em bando nas zonas temperadas de todos os continentes. Ele é considerado uma das aves mais inteligentes da natureza [23] por conseguir esconder o excesso de comida em vários lugares e, quando necessário, recuperá-la. Na busca por uma fonte de alimento escondida, quando percebe que está sendo seguido, o corvo tenta enganar aquele que o persegue deslocando-se para outra região do ambiente.

Do ponto de vista de otimização, os corvos se assemelham aos agentes de busca, a região ao espaço de busca, as posições do ambiente às soluções no espaço de busca, a qualidade da comida à função objetivo e a melhor fonte de comida à solução global. Baseado nessas similaridades, o CSA surgiu como uma proposta de algoritmo metaheurístico inspirado no comportamento inteligente dos pássaros corvo.

Sob essa ótica, o espaço de busca possui d dimensões e o tamanho do bando é representado por N. Assim, cada corvo possuirá uma posição i na iteração t e é representado por $x^{i,t} = [x_1^{i,t}, x_2^{i,t}, ..., x_d^{i,t}]$.

Cada corvo possui uma memória, representada por $m^{i,t}$, que armazena a posição do seu esconderijo. Assim, na iteração t, se o corvo i desejar seguir o corvo j para descobrir o esconderijo de sua fonte de alimento, ambos estarão susceptíveis a mudar sua posição atual. Portanto dois casos podem acontecer:

Estado 1: se o corvo j não perceber que está sendo observado pelo corvo i, o esconderijo de j será descoberto. Na sequência, o corvo i atualizará sua posição como segue:

$$x^{i,t+1} = x^{i,t} + r_i \times fl^{i,t} \times (m^{j,t} - x^{i,t}), \tag{1}$$

onde r_i é um número aleatório distribuído uniformemente no intervalo [0,1], $fl^{i,t}$ é a duração do voo e $m^{j,t}$ é a memória de j.

Estado 2: se o corvo j perceber que está sendo observado pelo i, j se move aleatoriamente a fim de enganar o corvo i.

Assim, baseado nesses dois estados, pode-se representá-los matematicamente por

$$x^{i,t+1} = \begin{cases} x^{i,t} + r_i * fl^{i,t} * (m^{j,t} - x^{i,t}), & se \;\; r_j \geq AP^{j,t} \\ \text{posição aleatória} & \text{, caso contrário} \end{cases} \tag{2}$$

onde $AP^{j,t}$ é a probabilidade de conscientização.

Os passos do CSA são apresentados no Algoritmo 1. Na linha 1, a população de corvos é gerada aleatoriamente. Na linha seguinte, a posição de cada corvo é avaliada. Na linha 3, os corvos memorizam os seus respectivos esconderijos de fonte de comida. A partir da linha 4, haverá a condição do critério de parada que circunda todas as iterações do algoritmo. Dessa forma, para cada posição de todos os corvos, aleatoriamente, um corvo j será escolhido para ser seguido, como referenciado na linha 6. Na linha 7, o valor de AP será definido para sinalizar os próximos passos do algoritmo.

Algoritmo 1 Pseudocódigo do Algoritmo Busca Corvo

```
1:  Inicialize aleatoriamente as posições do bando de corvos
2:  Avalie a posição de cada corvo
3:  Inicialize a memória de cada corvo
4:  enquanto critério de parada não for atingido faça
5:      para i ← 1 até qtde-de-corvos faça
6:          Escolha aleatoriamente um corvo j para seguir
7:          Defina um valor para AP
8:          se r_j ≥ AP^{j,t} então
9:              x^{i,t+1} = x^{i,t} + r_i × fl^{i,t} × (m^{j,t} − x^{i,t})
10:         senão
11:             x^{i,t+1} = rand(espaço de busca)
12:         fim se
13:     fim para
14:     Verifique as novas posições
15:     Avalie a nova posição de cada corvo
16:     Atualize a memória de cada corvo
17: fim enquanto
18: Ordene os corvos e selecione o melhor deles
```

Na linha 8, caso o corvo j não perceba que está sendo seguido por outro corvo, o seu esconderijo será descoberto e ocorrerá a intensificação na linha 9. Caso o corvo j perceba que está sendo seguido por outro corvo, haverá a diversificação do algoritmo, como mostrado na linha 11. Nas linhas 14 – 15, ocorre uma nova verificação e avaliação para cada corvo do bando. Na linha 16, as memórias dos corvos são atualizadas novamente. Por fim, na linha 18, o melhor corvo é selecionado como sendo a solução do problema.

2.2 Aprendizagem Baseada em Oposição

Os algoritmos metaheurísticos são iniciados com soluções aleatórias e, ao longo do tempo, eles seguem em direção da solução ótima. Quando as soluções iniciais estão próximas do ponto ótimo, a convergência é rápida, porém, se estão distantes, a convergência se torna mais lenta. O pior caso ocorre quando elas estão posicionadas do lado oposto da solução ótima, o que pode demandar muito tempo ou, definitivamente, torna-se impraticável localizá-la.

Uma solução para esse problema é buscar, simultaneamente, em todas as direções ou, simplesmente, na direção oposta. Para lidar com esta situação, a Aprendizagem Baseada em Oposição (OBL – *Opposition-Based Learning*)[16] vem sendo utilizada por outras metaheurísticas [24][25][26]

Para melhorar o desempenho da OBL, diversas variantes foram propostas na literatura. Algumas delas são *Quasi Opposition-Based Learning* (QOBL) [27], *Opposition-Based Learning using the Current Optimum* (COOBL) [28], *Elite Opposition-Based Learning* (EOBL) [17] e *Generalized Opposition-Based Learning* (GOBL) [18], além de outras.

Nos próximos parágrafos, serão apresentadas definições de número oposto, ponto oposto, EOBL e GOBL.

Definição 1: Dado $x \in \mathbb{R}$, no intervalo $x \in [a, b]$, o número oposto $\tilde{x}$ é definido a seguir.

$$\tilde{x} = a + b - x. \tag{3}$$

Definição 2: Dado $P = (x_1, x_2, ..., x_i)$ como sendo um ponto no espaço n-dimensional, com $x_1, x_2, ..., x_n \in \mathbb{R}$ e $x_i \in [a_i, b_i]$, $\forall i \in \{1,$

2, ..., n}. Assim, o ponto oposto $\tilde{P} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$ é definido pelos seus componentes:

$$\tilde{x}_i = a_i + b_i - x_i. \tag{4}$$

Definição 3: Dada $X_e^t = (x_{e1}^t, x_{e2}^t, ..., x_{en}^t)$ a melhor solução da iteração t e $X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{in}^t)$ a i–ésima solução de t. Então a solução oposta elite de X_i^t, é definida por $\hat{X}_i = (\hat{x}_{i1}, \hat{x}_{i2}, ..., \hat{x}_{in})$ e obtida como segue:

$$\hat{x}_{ij} = \delta(da_j + db_j) - x_{ej}, \tag{5}$$

onde $i \in (1, 2, ..., N)$, N é o número de soluções em t, $j \in (1, 2, ..., n)$, δ é um coeficiente aleatório distribuído uniformemente dentro do intervalo [0, 1], $x_{ej} \in [a_i, b_i]$ e $[da_j, db_j]$ é o limite dinâmico, definido por:

$$da_j = min(x_{ij}), \quad db_j = max(x_{ij}). \tag{6}$$

No entanto, é possível que uma solução transformada por meio da oposição elite "salte" para fora do intervalo $[a_i, b_i]$, isto é, $\hat{x}_{ij} < da_j$ ou $\hat{x}_{ij} > db_j$, Então, caso isso ocorra, $\hat{x}_{ij}$ reiniciará com:

$$\hat{x}_{ij} = rand(da_j, db_j). \tag{7}$$

Definição 4: O ponto oposto generalizado de x_i é definido por

$$x^g = \delta \cdot (a_i + b_i) - x_i \tag{8}$$

Como ocorre na EOBL, também é possível que uma solução GOBL "salte" para fora do intervalo $[a_i, b_i]$, isto é, $x^g < a_i$ ou $x^g > b_i$. Então, se isso ocorrer

$$x^g = rand(a_i, b_i). \tag{9}$$

3 CSA COMBINADO COM ESTRATÉGIAS OBL

Até onde esta pesquisa evoluiu, nenhuma melhoria para o CSA foi proposta, considerando o algoritmo original combinado às estratégias EOBL e GOBL. Portanto, a contribuição do trabalho apresentado é o CSA modificado por meio das variantes EOBL e GOBL.

EOBL foi utilizada para melhorar a busca local do algoritmo original, uma vez que ela tem como característica aproximar uma solução S da melhor solução encontrada até a iteração atual, isto é, a solução elite. Portanto, após a linha 9 do Algoritmo 1, foi incluída a aplicação da EOBL, como visto na linha 10 do Algoritmo 2. Nesse algoritmo, o corvo que apresentar o melhor valor da função objetivo será considerado o corvo elite. Já a variante GOBL tem como objetivo substituir o processo de diversificação da linha 11 do Algoritmo 1, a fim de melhorar a sua busca global, como se vê na linha 12 do Algoritmo 2.

Por fim, a combinação do CSA com as estratégias EOBL e GOBL é apresentada no Algoritmo 2, cujas modificações estão nas linhas 10 e 12, em negrito.

Algoritmo 2 Pseudocódigo do CSA-Proposto

1: Inicialize aleatoriamente as posições do bando de corvos
2: Avalie a posição de cada corvo
3: Inicialize a memória de cada corvo
4: **enquanto** critério de parada não for atingido **faça**
5: **para** i ← 1 **até** qtde-de-corvos **faça**
6: Escolha aleatoriamente um corvo j para seguir
7: Defina um valor para AP
8: **se** $r_j \leq AP^{i,t}$ **então**
9: $x_i^{t+1} = x_i^t + r_i \times fl_i^t \times (m_j^t - x_i^t)$
10: $x_i^{t+1} =$ **EOBL**(x_i^{t+1})
11: **senão**
12: $x_i^{t+1} =$ **GOBL**(x_i^{t+1})
13: **fim se**
14: **fim para**
15: Verifique as novas posições
16: Avalie a nova posição de cada corvo
17: Atualize a memória de cada corvo
18: **fim enquanto**
19: Ordene os corvos e selecione o melhor deles

4 EXPERIMENTOS E RESULTADOS

Nesta seção, são apresentadas as funções de referência utilizadas nos experimentos para validação do algoritmo proposto. Além disso, as configurações do ambiente experimental e dos resultados obtidos a partir das simulações.

4.1 Funções de otimização

Ao avaliar novos algoritmos de otimização, é comum o emprego de funções de referência (ou *benchmark*) porque pressupõe-se que a dificuldade encontrada para otimizá-las corresponde às complexidades de aplicações reais. Assim, para avaliar a performance da proposta apresentada, foram realizados vários experimentos em subconjuntos variados de problemas de otimização de funções. Alguns estudos de variantes do CSA [12][14][22] também utilizaram essas funções.

Neste estudo, as funções são divididas em três categorias considerando a dimensionalidade e a modalidade, conforme a Tabela 1. Na primeira categoria, as funções unimodais (f_1, f_2, f_3, f_7, f_8 e f_9) investigam a velocidade de convergência do algoritmo e a sua capacidade de busca local. Na segunda categoria, as funções multimodais (f_4, f_5 e f_6) avaliam a capacidade do algoritmo encontrar o ótimo global, quando a quantidade de soluções subótimas aumenta exponencialmente com a dimensão do problema. Por fim, nas funções multimodais de dimensão fixa (f_{10}, f_{11} e f_{12}), o número de dimensões não pode ser alterado e, além disso, elas possuem um espaço de busca diferente, quando comparadas às funções multimodais.

4.2 Avaliação dos Resultados

Todos os algoritmos foram implementados em MATLAB R2018a. Os experimentos foram executados em um computador que utiliza um processador Intel Core i5, com 1,8 GHz de frequência, 4 GB de memória RAM e Windows 10 Pro (64 bits).

Tabela 1: Funções de referência utilizadas nos experimentos.

Função	Formulação	Dimensão	Espaço de Busca	Melhor Fitness
Esfera	$f_1(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$	10, 30	$[-100, 100]$	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{d-1} [100(x_i - x_{i+1})^2 + (x_i - 1)^2]$	10, 30	$[-30, 30]$	0
Schwefel 2.22	$f_3(\mathbf{x}) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	10, 30	$[-10, 10]$	0
Griewank	$f_4(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$	10, 30	$[-600, 600]$	0
Ackley	$f_5(x) = -20\ exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + exp(1)$	10, 30	$[-32, 32]$	0
Rastrigin	$f_6(x) = 10d + \sum_{i=1}^{d}(x_i^2 - 10cos(2\pi x_i))$	10, 30	$[-5.12, 5.12]$	0
Noisy Quartic	$f_7(\mathbf{x}) = \sum_{i=1}^{d} i x_i^4 + \text{random}(0, 1)$	30	$[-1.28, 1.28]$	0
Schwefel 1.2	$f_8(\mathbf{x}) = \sum_{i=1}^{d} (\sum_{j-1}^{i} x_j)^2$	30	$[-100, 100]$	0
Schwefel 2.21	$f_9(\mathbf{x}) = \max(\lvert x_i \rvert, 1 \leq i \leq d)$	30	$[-100, 100]$	0
Shekel 5	$f_{10}(\mathbf{x}) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
Shekel 7	$f_{11}(\mathbf{x}) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.4028
Shekel 10	$f_{12}(\mathbf{x}) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

Para avaliar o desempenho do algoritmo proposto, foram realizados dois conjuntos de experimentos, que serão apresentados a seguir.

4.2.1 Conjunto de Experimentos 1. Nesta experimentação, a proposta foi comparada com os algoritmos CSA, PSO, GSA [29] e a variante CPO [22]. Para cada algoritmo utilizado, os valores dos parâmetros iniciais foram definidos de acordo com o trabalho original da variante CPO.

Os experimentos foram realizados com 10 dimensões e ao longo de 2.000 iterações, utilizando 20 agentes. Os resultados apresentam a média de 30 execuções independentes.

A Tabela 2 mostra os resultados numéricos obtidos na execução do primeiro conjunto de experimentos e contém a média dos valores de *fitness* da melhor solução encontrada por cada um dos algoritmos. Os valores das colunas CSA, PSO, GSA e CPO foram, fielmente, extraídos da referência na qual a variante CPO foi proposta [22]. Para cada função de referência otimizada, o melhor resultado é destacado em negrito. Observa-se que a performance da proposta apresentada é superior a dos outros algoritmos em mais de 83%, ou seja, cinco das seis funções de referência avaliadas no conjunto de experimentos.

Tabela 2: Resultado do primeiro conjunto de experimentos.

Fun	CSA	PSO	GSA	CPO	Proposta
f_1	1,50E-09	1,27E-04	1,09E-25	1,60E-21	**0,00E+00**
f_2	4,70E+00	7,99E+00	5,61E+00	**4,62E+00**	7,98E+00
f_3	4,11E-10	6,83E-01	3,02E-12	1,59E-13	**0,00E+00**
f_4	4,04E-02	2,84E-01	2,08E-01	2,56E-02	**0,00E+00**
f_5	3,37E-09	6,03E-06	4,78E-13	2,25E-11	**8,88E-16**
f_6	5,03E+00	8,64E+00	6,70E+00	3,87E+00	**0,00E+00**

4.2.2 Conjunto de Experimentos 2. Nestes experimentos, a proposta apresentada foi comparada com BA, CSA, DA, GA, GWO, PSO e a variante híbrida GWOCSA. A definição dos valores iniciais para cada parâmetro desses algoritmos estão em conformidade com a referência [14].

Os algoritmos foram testados com 300 iterações e 30 agentes. Sete funções de referência foram otimizadas em 30 dimensões e três funções em 4 dimensões porque, como dito anteriormente, elas são fixas. Como na primeira experimentação, os resultados apresentam a média de 30 execuções independentes.

A Tabela 3 apresenta os resultados numéricos dos experimentos. Nela contém a média dos valores de *fitness* e o desvio padrão das melhores soluções encontradas. Apenas os resultados de GWOCSA foram extraídos diretamente da referência na qual a proposta foi apresentada [14]. Para cada uma das dez funções de referência otimizadas, o melhor resultado é destacado em negrito. Observa-se que a performance do algoritmo proposto neste estudo é superior a dos outros algoritmos em 80%, isto é, oito das dez funções avaliadas. É possível notar na função f_2 uma inferioridade da proposta, quando comparada ao GWO e ao GWOCSA. Embora essa diferença exista, não há uma discrepância entre os valores comparados, e o resultado da proposta aproxima-se de forma considerável dos valores obtidos pelo GWO e pelo GWOCSA. Na função f_4, há uma superioridade relevante da proposta, quando comparada aos demais algoritmos, porém não existe diferença entre a média do GWOCSA e a da proposta apresentada. Contudo ambos encontraram a solução ótima da função.

As subfiguras 1(a) – 1(c) e 1(g) – 1(i) demonstram o comportamento médio de convergência para otimizar as funções unimodais dos algoritmos avaliados. Já as subfiguras 1(d) – 1(f) representam a otimização das funções multimodais e as 1(j) – 1(l) as funções multimodais com dimensão fixa.

Para as funções unimodais, observa-se que o algoritmo proposto converge rapidamente quando comparado aos demais. Ao otimizar a função f_2, percebe-se que o algoritmo "cai" em uma solução subótima, próximo da iteração de número 50, e não consegue "escapar". No entanto, em mais de 83% das funções unimodais, a proposta alcança ótimos resultados, demonstrando sua forte capacidade de

Tabela 3: Resultado numérico da execução do segundo conjunto de experimentos.

Função	BA	CSA	DA	GA	GWO	PSO	GWOCSA	CSA-Proposto
f_1	2,85E+04	1,26E+02	1,89E+04	3,93E+03	3,88E-15	1,67E-01	1,01E-28	**2,11E-61**
	(5,62E+03)	(4,00E+01)	(4,61E+03)	(9,23E+02)	(3,93E-15)	(7,51E-02)	(1,26E-28)	**(1,08E-60)**
f_2	4,24E+07	4,18E+03	1,82E+07	1,58E+06	2,73E+01	1,27E+02	**2.70E+01**	2,87E+01
	(2,44E+07)	(1,90E+03)	(1.01E+07)	(7,09E+05)	(7,97E-01)	(2,94E+01)	**(5.00E-01)**	(5,07E-04)
f_3	1,75E+05	7,80E+00	6,65E+01	2,19E+01	1,36E-09	1,03E+00	1,50E-17	**4,40E-33**
	(4,17E+05)	(1,47E+00)	(3,03E+01)	(2,70E+00)	(6,57E-10)	(4,04E-01)	(1,25E-17)	**(1,01E-32)**
f_4	2,68E+02	2,10E+00	1,69E+02	3,67E+01	3,95E-03	9,17E-03	**0,00E+00**	**0,00E+00**
	(7,24E+01)	(3,44E-01)	(3,84E+01)	(8,82E+00)	(9,17E-03)	(6,27E-03)	(**0,00E+00**)	(**0,00E+00**)
f_5	1,83E+01	6,31E+00	1,74E+01	1,20E+01	1,08E-08	8,71E-01	1,37E-14	**1,00E-15**
	(1.36E+00)	(1.17E+00)	(1,13E+00)	(8,60E-01)	(4,94E-09)	(4,44E-01)	(3,53E-15)	(**6,48E-16**)
f_6	1,03E+02	5,84E+01	2,55E+02	1,01E+02	7,82E+00	5,87E+01	1,19E+00	**0,00E+00**
	(3,81E+01)	(1,68E+01)	(6,54E+01)	(1,19E+01)	(5,81E+00)	(1,635E+01)	(3,32E+00)	(**0,00E+00**)
f_7	1,90E+01	1,30E-01	9,50E+00	2,24E+00	4,31E-03	2,64E-01	1,92E-03	**2,86E-04**
	(8.09E+00)	(6,29E-02)	(4,57E+00)	(5,91E-01)	(2,43E-03)	(9,18E-02)	(9,88E-04)	**(2,44E-04)**
f_8	7,57E+04	1,28E+03	3,95E+04	2,12E+04	4,31E-03	7,31E-02	5,18E-04	**7,12E-55**
	(3.69E+04)	(3.66E+02)	(1,73E+04)	(4,23E+03)	(2,43E-03)	(1,39E-02)	(1,07E-03)	**(3,80E-54)**
f_9	6,50E+01	1,12E+01	4,93E+01	3,81E+01	7,74E-04	8,32E-01	2,07E-07	**7,92E-34**
	(6.81E+00)	(2,02E+00)	(7,27E+00)	(3,37E+00)	(7,72E-04)	(5,36E-01)	(3,00E-07)	**(1.95E-33)**
f_{10}	-5,82E+00	-7,82E+00	-4,70E+00	-5,80E+00	-9,05E+00	-5,06E+00	-6,80E+00	**-1,02E+01**
	(3,53E+00)	(3,40E+00)	(1,87E+00)	(3,27E+00)	(2,26E+00)	(1,32E-07)	(2,23E+00)	**(1,54E-07)**
f_{11}	-5,50E+00	-9,24E+00	-5,76E+00	-6,91E+00	-1,02E+01	-5,09E+00	-8,76E+00	**-1,04E+01**
	(3,23E+00)	(2,68E+00)	(2,46E+00)	(3,57E+00)	(9,69E-01)	(2,72E-15)	(6,47E-01)	**(1,24E-01)**
f_{12}	-4,85E+00	-9,13E+00	-4,32E+00	-5,50E+00	-9,75E+00	-5,13E+00	-8,82E+00	**-1,05E+01**
	(3,29E+00)	(2,88E+00)	(2,06E+00)	(3,39E+00)	(2,39E+00)	(3,45E-15)	(5,52E-01)	**(1,25E-05)**

intensificação durante o processo de busca. Portanto, quando utilizado para resolver as funções de otimização apresentadas, pode-se concluir que o desempenho do algoritmo apresentado é superior a dos outros algoritmos, na maioria dos experimentos.

Já nas funções multimodais, em 100% dos experimentos, a proposta alcança bons resultados e comprova sua boa capacidade de exploração do espaço de busca. Percebe-se também que nas funções f_4 e f_6, o algoritmo proposto encontrou a solução ótima. GWOCSA obteve também o ótimo global. Nas funções com dimensões fixas, a proposta apresenta, novamente, uma rápida convergência em busca da solução ótima e, em todos os experimentos, o valor médio de *fitness* supera os demais algoritmos. Observa-se que nas funções f_{11} e f_{12}, a proposta apresentada encontrou a solução ótima, embora quando o seu desempenho foi comparado ao do CSA original não foi identificada uma significância estatística. Todavia é notório que o desempenho da proposta apresentada supera os dos demais algoritmos quando comparados às funções multimodais estudadas em 30D e com dimensões fixas.

Em geral, os testes estatísticos são utilizados em pesquisas que objetivam comparar determinadas condições experimentais. Eles podem ser divididos em testes paramétricos e não-paramétricos [30]. Nos paramétricos, os valores da variável estudada devem ter distribuição ou aproximação normais. Já os não-paramétricos não exigem o conhecimento da distribuição da variável da população.

O Teste de Wilcoxon (*Signed Rank Test*) [31] é não-paramétrico e, neste estudo, foi utilizado para verificar se os resultados obtidos da comparação entre a variante CSA proposta com o BA, o CSA, o DA, o GA, o GWO e o PSO apresentam uma melhora significativa.

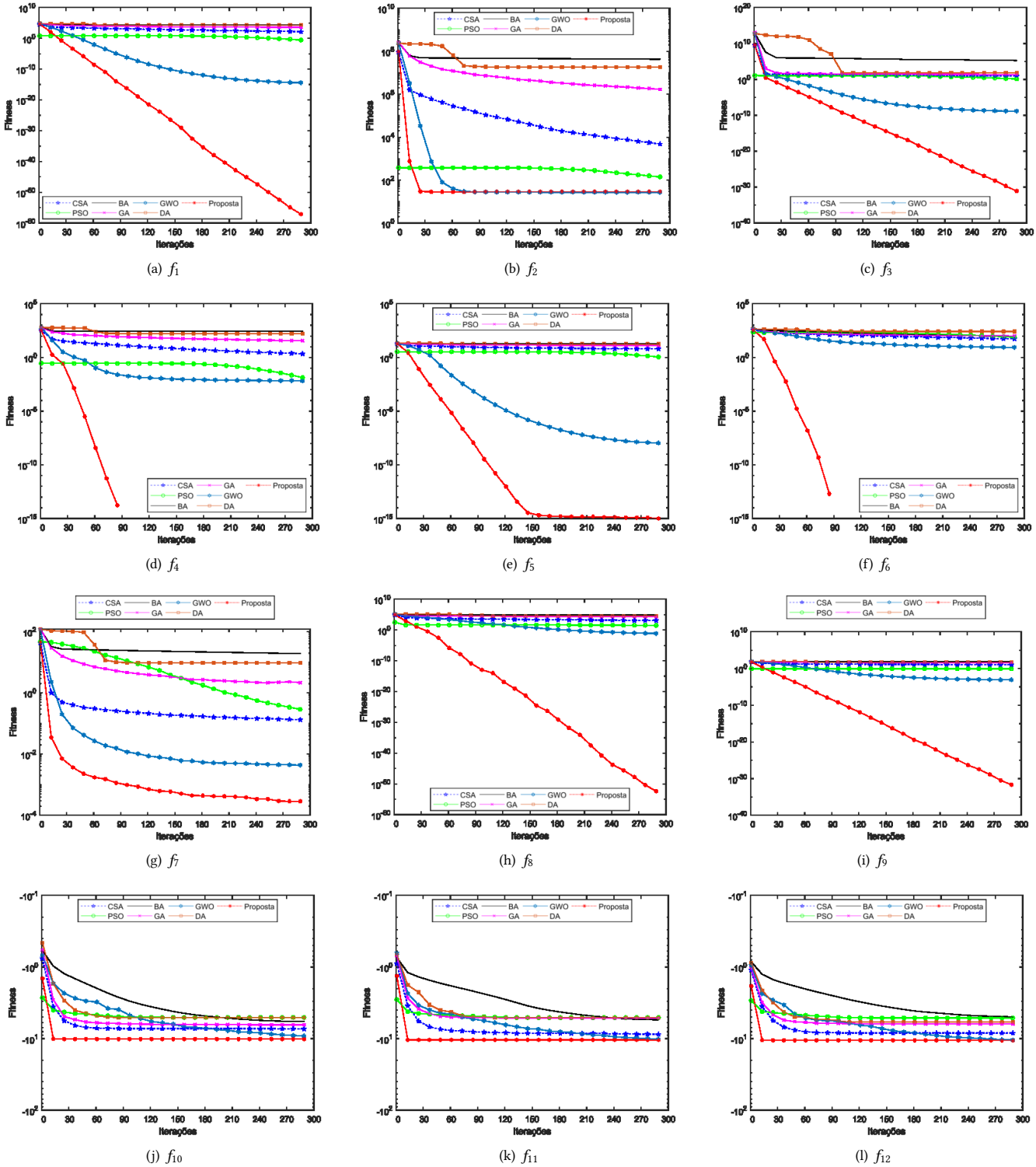
A hipótese nula, H_0, indica que duas amostras vêm da mesma população, ao passo que a hipótese alternativa, H_1, indica que uma população tem valores maiores que a outra. Quando *p-value* é inferior ao nível de significância, então decide-se rejeitar H_0, ou seja, há uma diferença significativa entre as amostras.

A Tabela 4 representa os resultados obtidos pelo teste, considerando um nível de significância de 0.05. Quando o *p-value* resultante é inferior a 0.05 e a média dos valores de *fitness* do algoritmo proposto é

(1) melhor que a do algoritmo em comparação, o resultado é representado por um ícone de "positivo";
(2) pior que a do algoritmo em comparação, o resultado é representado por um ícone de "negativo".

Tabela 4: Resultado do Teste de Wilcoxon.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
BA	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍
CSA	👍	👍	👍	👍	👍	👍	👍	👍	👍	—	—	—
DA	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍
GA	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍
GWO	👍	👎	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍
PSO	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

(g) f_7 (h) f_8 (i) f_9

(j) f_{10} (k) f_{11} (l) f_{12}

Figura 1: Convergência média dos algoritmos nas funções avaliadas.

Ainda na Tabela 4, é observado que em mais de 93% dos resultados do teste estatístico aplicado, o algoritmo proposto supera os demais algoritmos com significância estatística. Nas funções f_{10} – f_{12}, embora os resultados numéricos tenham sido melhores que os dos demais, a proposta apresentada não indicou significância estatística, quando comparada ao CSA original. No entanto, ao compará-lo com o GWO na função f_2, seus resultados são inferiores e com significância estatística apontada.

Nos *boxplots* da Figura 2, alguns padrões foram encontrados. Observam-se algumas variações nas médias do DA e do BA nas subfiguras 2(a), 2(b), 2(d), 2(g) e 2(h), embora o GA apresente uma variação menor. Nas subfiguras 2(j) – 2(l) há uma variação excessiva dos algoritmos CSA, BA, GA e DA, alguns deles com *outliers*. Por fim, em todos os *boxplots* apresentados, a proposta apresentou uma variação mínima em seus valores de média, indicando um desempenho estável comparado aos demais algoritmos.

5 CONCLUSÕES

Este trabalho apresentou uma nova proposta do Algoritmo Busca Corvo combinando-o com duas variantes da aprendizagem baseada em oposição, conhecidas como EOBL e GOBL, a fim de aumentar a velocidade de convergência do algoritmo original, bem como os seus resultados numéricos.

Para validar a proposta apresentada, diversos testes computacionais foram realizados com 12 funções de referência, das quais a) 6 são unimodais, b) 3 multimodais e c) 3 multimodais com dimensões fixas. Os testes foram realizados em dois conjuntos de experimentos considerando os problemas em 4, em 10 e em 30 dimensões. Após os experimentos, observou-se a superioridade da variante proposta quando comparada a outros algoritmos da literatura. Quando o teste de Wilcoxon foi aplicado, a proposta apresentada obteve resultados significativos na maioria dos problemas testados. Portanto considerando os resultados obtidos, acredita-se que a proposta apresentará desempenho satisfatório quando empregada em aplicações reais.

Como trabalhos futuros, pretende-se validar o algoritmo apresentado em uma aplicação de telecomunicações no contexto de síntese de *arrays* de antenas lineares. Por fim, também é interessante considerar uma adaptação desse algoritmo à otimização de problemas discretos.

REFERÊNCIAS

[1] Kuk-Hyun Han, Kui-Hong Park, Ci-Ho Lee, and Jong-Hwan Kim. Parallel quantum-inspired genetic algorithm for combinatorial optimization problem. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 2, pages 1422–1429. IEEE, 2001.

[2] JD Weber and TJ Overbye. A two-level optimization problem for analysis of market bidding strategies. In *1999 IEEE Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No. 99CH36364)*, volume 2, pages 682–687. IEEE, 1999.

[3] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. An improved ant colony optimization for vehicle routing problem. *European journal of operational research*, 196(1):171–176, 2009.

[4] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[5] Carmelo JA Bastos Filho, Fernando B de Lima Neto, Anthony JCC Lins, Antonio IS Nascimento, and Marilia P Lima. A novel search algorithm based on fish school behavior. In *2008 IEEE international conference on systems, man and cybernetics*, pages 2646–2651. IEEE, 2008.

[6] Xin-She Yang. Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178, 2009.

[7] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[8] Alireza Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, 169:1–12, 2016.

[9] Diego Oliva, Salvador Hinojosa, Erik Cuevas, Gonzalo Pajares, Omar Avalos, and Jorge Gálvez. Cross entropy based thresholding for magnetic resonance brain images using crow search algorithm. *Expert Systems with Applications*, 79: 164–180, 2017.

[10] Deepak Gupta, Shirsh Sundaram, Ashish Khanna, Aboul Ella Hassanien, and Victor Hugo C De Albuquerque. Improved diagnosis of parkinson's disease using optimized crow search algorithm. *Computers & Electrical Engineering*, 68:412–424, 2018.

[11] Primitivo Díaz, Marco Pérez-Cisneros, Erik Cuevas, Omar Avalos, Jorge Gálvez, Salvador Hinojosa, and Daniel Zaldivar. An improved crow search algorithm applied to energy problems. *Energies*, 11(3):571, 2018.

[12] Seyed Hamid Reza Pasandideh and Soheyl Khalilpourazari. Sine cosine crow search algorithm: a powerful hybrid meta heuristic for global optimization. *arXiv preprint arXiv:1801.08485*, 2018.

[13] Seyedali Mirjalili. Sca: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, 96:120–133, 2016.

[14] Sankalap Arora, Harpreet Singh, Manik Sharma, Sanjeev Sharma, and Priyanka Anand. A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. *IEEE Access*, 7:26343–26361, 2019.

[15] Dunia S Tahir and Ramzy S Ali. A chaotic crow search algorithm for high-dimensional optimization problems. *Basrah Journal for Engineering Science*, 17 (1):16–25, 2017.

[16] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *Int. Conf. on Computational intelligence for modelling, control and automation and Int. Conf. on intelligent agents, web technologies and internet commerce*, volume 1, pages 695–701, 2005.

[17] Xinyu Zhou, Zhijian Wu, and Hui Wang. Elite opposition-based differential evolution for solving large-scale optimization problems and its implementation on gpu. In *13th Int. Conf. on Parallel and Distributed Computing, Applications and Technologies*, pages 727–732, 2012.

[18] Hui Wang, Zhijian Wu, Yong Liu, Jing Wang, Dazhi Jiang, and Lili Chen. Space transformation search: a new evolutionary technique. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 537–544. ACM, 2009.

[19] David E Goldenberg. Genetic algorithms in search, optimization and machine learning, 1989.

[20] Xin-She Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[21] Seyedali Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, 27(4):1053–1073, 2016.

[22] Ko-Wei Huang and Ze-Xue Wu. Cpo: A crow particle optimization algorithm. *International Journal of Computational Intelligence Systems*, 12(1):426–435, 2018.

[23] Amelia Stymacks. Porque é que os Corvos e as Gralhas São as Aves Mais Inteligentes do Planeta? https://www.natgeo.pt/animais/2018/05/porque-e-que-os-corvos-e-gralhas-sao-aves-mais-inteligentes-do-planeta, 2018. [Online; acessado em 13-August-2020].

[24] Xian Shan, Kang Liu, and Pei-Liang Sun. Modified bat algorithm based on lévy flight and opposition based learning. *Scientific Programming*, 2016, 2016.

[25] MengChu Zhou, Zeyu Zhao, Caifei Xiong, and Qi Kang. An opposition-based particle swarm optimization algorithm for noisy environments. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6. IEEE, 2018.

[26] Cácio LNA Bezerra, Fábio GBC Costa, Gabriel M Nascimento, Pedro VM Carvalho, and Fábio AP Paiva. Uma variante melhorada do algoritmo busca cuco usando uma estratégia de quasi opposition–based learning. *Anais do Computer on the Beach*, pages 060–069, 2018.

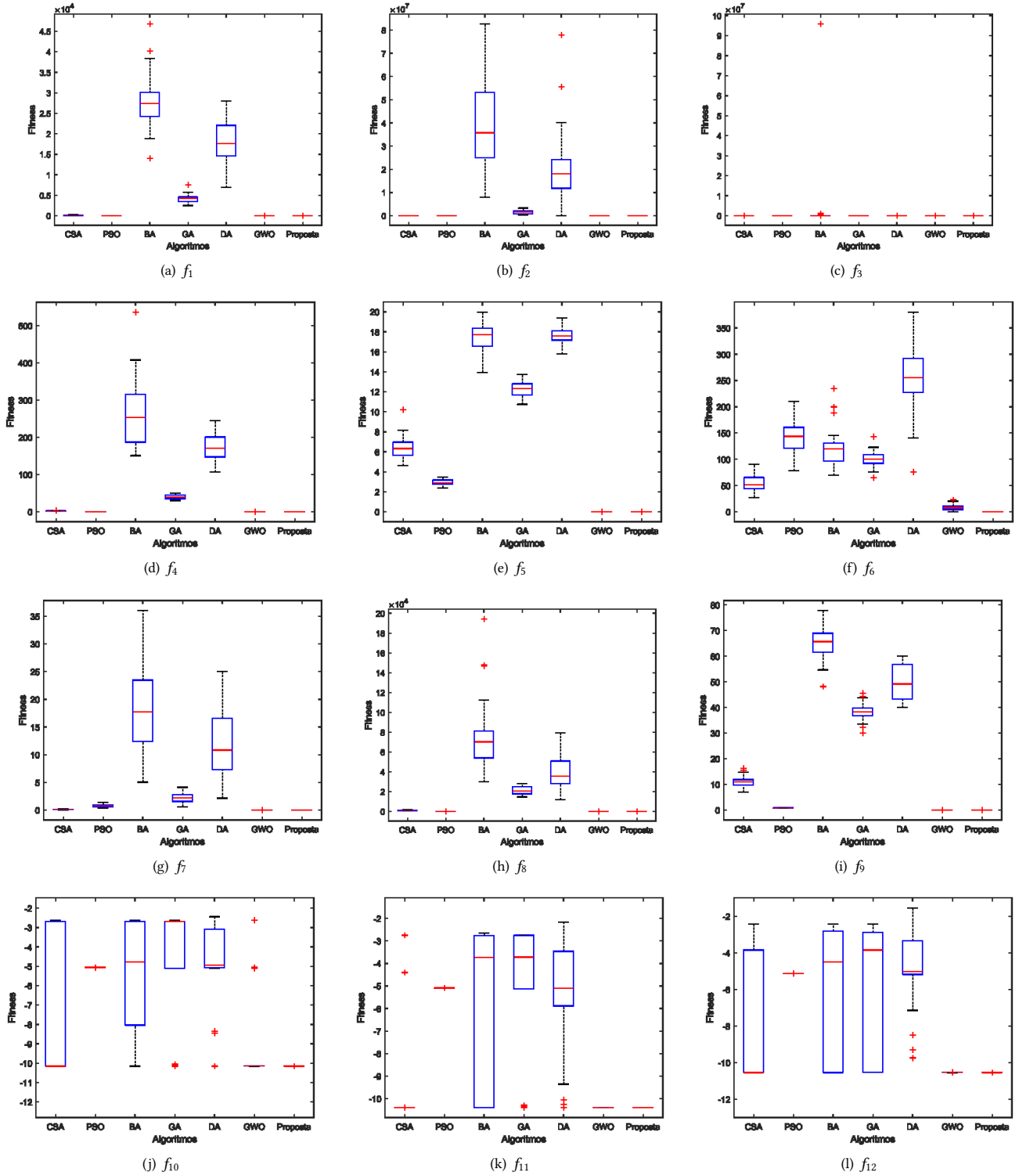
[27] Shahryar Rahnamayan, Hamid R Tizhoosh, and Magdy MA Salama. Quasi-oppositional differential evolution. In *IEEE Congress on Evolutionary Computation*, pages 2229–2236, 2007.

[28] Qingzheng Xu, Lei Wang, Baomin He, and Na Wang. Modified opposition-based differential evolution for function optimization. *J. of Comp. Information Systems*, 7(5):1582–1591, 2011.

[29] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248, 2009.

[30] Sidia M Callegari-Jacques. *Bioestatística: princípios e aplicações*. Artmed Editora, 2009.

[31] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

(g) f_7 (h) f_8 (i) f_9

(j) f_{10} (k) f_{11} (l) f_{12}

Figura 2: Boxplot dos melhores valores de *fitness* para as funções avaliadas.