Flower Pollination Algorithm Combined with Multiple Strategies of Opposition–Based Learning

Cácio L. N. A. Bezerra[1], **Fábio G. B. C. Costa**[1], **Lucas V. Bazante**[1], **Pedro V. M. Carvalho**[1] **e Fábio A. P. Paiva**[1]

[1] Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Campus Parnamirim – RN – Brasil

{cacio.lucas, fabio.gabriel, lucal.bazante}@escolar.ifrn.edu.br,

pedro.miranda@escolar.ifrn.edu.br, fabio.procopio@ifrn.edu.br

Abstract. *Flower Pollination Algorithm (FPA) has been widely used to solve optimization problems. However, it faces the problem of stagnation in local optimum. Several approaches have been proposed to deal with this problem. To improve the performance of the FPA, this paper presents a new variant that combines FPA and two variants of the Opposition Based Learning (OBL), such as Quasi OBL (QOBL) and Elite OBL (EOBL). To evaluate this proposal, 10 benchmark functions were used. In addition, the proposed algorithm was compared with original FPA and three variants such as FA–EOBL, SBFPA and DE–FPA. The proposal presented significant results.*

Resumo. *O Algoritmo de Polinização das Flores (FPA) vem sendo bastante utilizado para resolver problemas de otimização. Porém, ele enfrenta o problema da estagnação em ótimos locais. Diversas abordagens foram propostas para lidar com esse problema. Para melhorar o desempenho do FPA, este trabalho apresenta uma nova variante que combina FPA e duas variantes da Opposition Based Learning (OBL): Quasi OBL (QOBL) e Elite OBL (EOBL). Para avaliar esta proposta, foram usadas 10 funções de referências. Ademais, o algoritmo proposto foi comparado com o FPA original e com outras 3 variantes: FA–EOBL, SBFPA e DE–FPA. A proposta apresentou resultados significativos.*

1. Introdução

Otimização é um área que, invariavelmente, é usada na tomada de decisões em quase todos os domínios do mundo real [Shareef et al. 2015], inclusive ciência e engenharia. As circunstâncias e as necessidades nas tomadas de decisão aumentaram ainda mais a complexidade dos problemas de otimização.

O Algoritmo de Polinização das Flores (FPA – *Flower Pollination Algorithm*) [Yang 2012] é um método meta-heurístico, inspirado no processo de reprodução das flores, que tem sido usado para otimizar problemas de engenharia [Saxena and Kothari 2016, Meng et al. 2017]. No entanto, assim como muitas meta-heurísticas, o FPA enfrenta um problema quando estagna em ótimos locais, o que faz com que a busca global não seja bem sucedida.

Vários trabalhos, com diferentes abordagens, já foram apresentados a fim de melhorar o desempenho do FPA. O *Differential Evolution–Flower Pollination Algo-*

rithm (DE–FPA) [Chakraborty et al. 2014] combina o algoritmo de Evolução Diferencial com o FPA. A variante *Serendipity–Based Flower Pollination Algorithm* (SBFPA) [Paiva et al. 2017] utiliza Aprendizagem Baseada em Oposição (OBL) como uma alternativa para implementar a dimensão acaso, um conceito específico de serendipidade. FPA também foi combinado com o algoritmo Busca Tabu (TS – *Tabu Search*) para implementar a variante TS–FPA [Hezam et al. 2016].

Este trabalho propõe uma variante FPA a partir da combinação de duas estratégias de Aprendizagem Baseada em Oposição: *Quasi Opposition–Based Learning* (QOBL) e *Elite Opposition–Based Learning* (EOBL). O objetivo da variante proposta é aumentar a capacidade do algoritmo original em gerar novas soluções e, como resultado, melhorar o seu desempenho. Ao fim das simulações, observou-se que a nova proposta superou o FPA e, em muitos experimentos, ela também superou algumas variantes da literatura.

O trabalho está organizado como segue: na Seção 2, é apresentada uma breve introdução sobre a meta-heurística FPA e a aprendizagem baseada em oposição; na Seção 3, a nova variante que se propõe a melhorar a performance do FPA é apresentada. Na Seção 4, os experimentos computacionais são apresentados e os resultados são discutidos. Por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta algumas informações úteis para o entendimento da variante que será apresentada, como o Algoritmo de Polinização das Flores, *Quasi Opposition–Based Learning* e *Elite Opposition–Based Learning*.

2.1. Algoritmo de Polinização das Flores

Uma classe dos algoritmos meta-heurísticos que tem recebido muita atenção são os bioinspirados. Um desses algoritmos é inspirado no processo de polinização das flores e, por isso, é chamado de algoritmo de polinização das flores (FPA – *Flower Pollination Algorithm*) [Yang 2012]. FPA assume que cada planta possui apenas uma flor e que cada flor produz somente um gameta. O algoritmo implementa dois tipos de polinização: a global e a local. Na polinização global, os agentes carregam o pólen ao longo de grandes espaços de busca e o comportamento dos polinizadores pode ser modelado pelo voo de Lévy, em que é obedecida a distribuição de probabilidade de Lévy [Tran et al. 2014]. A flor mais saudável é representada por g_*. Matematicamente, a polinização global pode ser representada por:

$$x_i^{t+1} = x_i^t + L(g_* - x_i^t), \tag{1}$$

onde x_i^t é o pólen i (ou o vetor solução x_i) na iteração t. Já g_* é a melhor solução encontrada entre todas, até o momento. O parâmetro L é a força de polinização, que é um valor gerado pela distribuição de Lévy. O voo de Lévy é descrito na Equação 2:

$$L \sim \frac{\lambda\Gamma(\lambda)sen(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0), \tag{2}$$

onde Γ é a função gama padrão, cuja distribuição é válida para grandes passos $s > 0$ e $\lambda = 1.5$, como definido em [Yang 2012].

A geração de tamanhos de passos pseudo-aleatórios que obedeçam, corretamente, a Equação 2 não é uma tarefa trivial [Yang 2012]. Existem alguns métodos para geração

de números aleatórios e o mais eficiente é o algoritmo de Mantegna [Mantegna 1994], o qual usa duas distribuições gaussianas U e V, conforme equação a seguir:

$$s = \frac{U}{|V|^{1/\lambda}}, \tag{3}$$

com

$$U \sim N(0, \sigma^2), \;\; V \sim N(0, 1), \tag{4}$$

onde $U \sim N(0, \sigma^2)$ significa que as amostras são geradas a partir de uma distribuição normal gaussiana, com média zero e variância σ^2, calculada por

$$\sigma^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda \Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right]^{1/\lambda}. \tag{5}$$

Já a polinização local pode ser representada pela Equação 6, como segue:

$$x_i^{t+1} = x_i^t + \varepsilon(x_j^t - x_k^t), \tag{6}$$

onde x_j^t e x_k^t são os pólens de diferentes plantas da mesma espécie, na mesma iteração. Por fim, ε corresponde a uma distribuição normal uniforme $\in$ [0,1].

O Algoritmo 1 apresenta o pseudocódigo do FPA. Ele é iniciado com a geração aleatória da população de flores (linha 1). Na linha 2, a melhor solução da população inicial é selecionada. Na linha seguinte, a taxa de probabilidade é usada para escolher se a polinização a ser aplicada será a global ou a local.

Algoritmo 1 Pseudocódigo do FPA

```
1:  Inicialize a população de flores com soluções aleatórias
2:  Encontre a melhor solução g_* dentro da população inicial
3:  Defina uma taxa de probabilidade p ∈ [0, 1]
4:  enquanto (t ≤ num_max_iter) faça
5:      para i ← 1 até n faça
6:          se (rand < p) então
7:              Faça polinização global usando Equação 1
8:          senão
9:              Faça polinização local usando Equação 6
10:         fim se
11:         Avalie novas soluções
12:         Caso as novas soluções forem melhores, atualize-as
13:     fim para
14:     Selecione a melhor solução atual g_*
15: fim enquanto
```

Nas linhas 4–15, ocorre a evolução das flores ao longo do tempo e do espaço de busca. Na linha 6, a taxa de probabilidade p é comparada com um valor gerado aleatoriamente e, quando ele é menor que p, é realizada a polinização global (linha 7). Caso contrário, é realizada a polinização local (linha 9). Em seguida, as novas soluções são avaliadas e, se forem melhores, são atualizadas na população (linhas 11–12). Por fim, seleciona-se a melhor solução atual, g_*.

2.2. Aprendizagem Baseada em Oposição

Em geral, os algoritmos meta-heurísticos são iniciados com soluções aleatórias e, ao longo do tempo, eles procuram melhorá-las seguindo em direção da solução ótima. É comum que essas soluções estejam distantes daquela considerada ótima e o pior caso ocorre quando elas estão na posição oposta à da solução ótima.

Uma alternativa para essa situação é buscar, simultaneamente, em todas as direções ou, simplesmente, na direção oposta. Para lidar com esta situação, Aprendizagem Baseada em Quase Oposição (QOBL – *Quasi Opposition-Based Learning*) [Rahnamayan et al. 2007] e Aprendizagem Baseada em Oposição Elite (EOBL – *Elite Opposition-Based Learning*) [Zhou et al. 2012] podem ser usadas.

Antes de introduzir QOBL e EOBL, o conceito de Aprendizagem Baseada em Oposição (OBL – *Opposition-Based Learning*) [Tizhoosh 2005] será explanado. Na área de Inteligência Computacional, OBL tem sido empregado com o objetivo de aumentar a eficiência dos algoritmos. Para um determinado problema, OBL avalia uma solução x e a sua respectiva solução oposta $\tilde{x}$ possibilitando que uma solução candidata seja encontrada próxima à global.

O conceito de OBL pode ser aplicado não apenas para gerar soluções aleatórias iniciais, mas também a cada iteração do algoritmo no conjunto corrente de soluções [Rahnamayan et al. 2007]. A seguir, são definidos Número Oposto e Ponto Oposto.

Definição 1: Dado $x \in \mathbb{R}$, no intervalo $x \in [a, b]$, o número oposto $\tilde{x}$ é definido na Equação 7:

$$\tilde{x} = a + b - x. \tag{7}$$

A mesma definição pode ser estendida, de forma similar, para problemas multidimensionais.

Definição 2: Dado $P = (x_1, x_2, ..., x_n)$ 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:

$$\tilde{x_i} = a_i + b_i - x_i. \tag{8}$$

***Definição 3*:** Dado $x \in \mathbb{R}$, no intervalo $x \in [a, b]$, seu ponto *quasi opposition–based learning*, x_{qo}, é definido por

$$x_{qo} = rand(c, \tilde{x_i}), \tag{9}$$

onde c é o centro do intervalo $[a, b]$, calculado por $c = (a + b)/2$ e $rand(c, \tilde{x_i})$ é um número aleatório uniformemente distribuído entre c e $\tilde{x_i}$.

Para explicar EOBL, um exemplo será usado. Neste artigo, a flor que representa a melhor solução é vista como o indivíduo elite. Supondo que o indivíduo elite seja $X_e = (x_{e1}, x_{e2}, ...x_{en})$, a solução baseada em oposição elite de $X_i = (x_{i1}, x_{i2}, ..., x_{in})$ é definida como $\hat{X}_i = (\hat{x}_{i1}, \hat{x}_{i2}, ...\hat{x}_{in})$ e é obtida pela Equação 10:

$$\hat{x}_{ij} = \delta \cdot (da_j + db_j) - x_{ej}, i = 1, 2, ..., N; j = 1, 2, ..., n \tag{10}$$

onde N é o tamanho da população, n é a dimensão de X, $\delta \in (0, 1)$, (da_j, db_j) é o limite dinâmico da j-ésima variável de decisão. O limite dinâmico é obtido pela seguinte equação:

$$da_j = min(x_{ij}), \quad db_j = max(x_{ij}) \tag{11}$$

O limite dinâmico pode fazer com que $\hat{x}_{ij}$ salte para fora de (da_j, db_j). Se isso acontecer, a Equação 12 será usada para reiniciar $\hat{x}_{ij}$:

$$\hat{x}_{ij} = rand(da_j, db_j), \quad \hat{x}_{ij} < da_j \,||\, \hat{x}_{ij} > db_j. \tag{12}$$

3. FPA combinado com Quasi OBL e Elite OBL

O Algoritmo de Polinização das Flores é uma técnica capaz de encontrar bons resultados em um tempo de convergência rápido, porém ele pode não conseguir "escapar" de ótimos locais.

A variante apresentada, a qual combina *Quasi* OBL (QOBL) e *Elite* OBL (EOBL), tem como objetivo aumentar a velocidade de convergência do FPA e minimizar as possibilidade de ele "cair" em ótimos locais. Até onde esta pesquisa avançou, nenhuma variante FPA foi encontrada com a proposta de combinar QOBL e EOBL e, portanto, isso garante o ineditismo deste trabalho. A contribuição da pesquisa é apresentada nas linhas 5, 8, 9 e 10 do Algoritmo 2. Só serão comentadas as linhas que correspondem à contribuição apresentada, uma vez que as demais já foram comentadas na explicação do Algoritmo 1.

Na linha 5, é aplicado o processo de QOBL (ver equação 9) sobre a flor atual da iteração. Já na linha 8, após QOBL ter sido aplicada em todas as flores, aleatoriamente sorteia-se uma flor e, em seguida, na linha 9, aplica-se EOBL (ver equação 10). Por fim, na linha 10, a solução modificada por meio de EOBL é adicionada ao conjunto de soluções e o algoritmo segue o fluxo padrão.

Algoritmo 2 Pseudocódigo do FPA modificado

```
1:  Idem às linhas 1 – 3 do Algoritmo 1
2:  enquanto (t ≤ num_max_iter) faça
3:      para i ← 1 até n faça
4:          Idem às linhas 6 – 10 do Algoritmo 1
5:          flor_i ← QOBL(flor_i)
6:          Idem às linhas 11 – 12 do Algoritmo 1
7:      fim para
8:      flor_sort ← sorteiaFlorAleatoria()
9:      flor_sort ← EOBL(flor_sort)
10:     Adiciona flor_sort às soluções
11:     Selecione a melhor solução atual g_*
12: fim enquanto
```

4. Experimentos Computacionais

Esta seção apresenta as funções de referência usadas para validar o algoritmo proposto e, em seguida, as simulações computacionais e a análise dos resultados.

4.1. Funções de Referência e Configuração dos Experimentos

É comum o emprego de funções de referência com o pressuposto de que a dificuldade delas corresponde àquelas encontradas em aplicações reais. Para realizar os experimentos, foram escolhidas 10 funções de referência que são aplicadas em problemas de minimização e utilizadas em vários estudos de FPA [Kalra and Arora 2016, Ramadas and Kumar 2016, Paiva et al. 2017] e são apresentadas na Tabela 1.

Tabela 1. Funções de referência.

Fórmula	Espaço de Busca	Ótima
$f_1(x) = \sum_{i=1}^{d} x_i^2$	$-100 \leq x_i \leq 100$	0
$f_2(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-30 \leq x_i \leq 30$	1
$f_3(x) = \sum_{i=1}^{d} x_i^4$	$-100 \leq x_i \leq 100$	0
$f_4(x) = \sum_{i=1}^{d} \lvert x_i \rvert^{i+1}$	$-500 \leq x_i \leq 500$	0
$f_5(x) = x_i^2 + \sum_{i=2}^{d} x_i^2$	$-10 \leq x_i \leq 10$	0
$f_6(x) = -20\ exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + exp(1)$	$-32 \leq x_i \leq 32$	0
$f_7(x) = \sum_{i=1}^{d} [x_i^2 - 10cos(2\pi x_i) + 10]$	$-5.12 \leq x_i \leq 5.12$	0
$f_8(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \leq x_i \leq 600$	0
$f_9(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{d} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{d} x_i^2}$	$-100 \leq x_i \leq 100$	0
$f_{10}(x) = \sum_{i=1}^{d} \lvert x_i sen(x_i) + 0.1x_i \rvert$	$-10 \leq x_i \leq 10$	0

As funções de referência avaliadas nos experimentos são classificadas em dois grupos distintos. Abaixo, eles são apresentados e as funções agrupadas dentro deles:

1. Grupo 1 – formado por funções unimodais que, geralmente são usadas para testar a capacidade do algoritmo em buscas locais. As funções unimodais usadas nos experimentos são: Esfera (f_1), Rosenbrock (f_2), Schumer Steiglitz (f_3), *Powell Sum* (f_4) e Cigar (f_5).

2. Grupo 2 – formado por funções multimodais que possuem vários mínimos locais. Elas são usadas para verificar a habilidade do algoritmo para "escapar" de ótimos locais. São elas: Ackley (f_6), Rastrigin (f_7), Griewank (f_8), Salomon (f_9) e Alpine (f_{10}).

Todas as rotinas foram implementadas na linguagem de programação MATLAB R2014b. Os experimentos foram executados em um computador que utiliza processador Intel Core i7 com 2,4 GHz de frequência, 8 GB de memória RAM e sistema operacional Windows 10 *Home Single Language*, 64 bits. Não foram utilizadas técnicas de multiprocessamento.

Os experimentos computacionais foram realizados para 20 execuções independentes. Em cada execução, foram fixados a dimensão em 30D, o número de iterações em 2.000 e o número de flores em $n = 30$.

4.2. Simulações e Resultados

Para validar o algoritmo proposto, ele é comparado com o FPA original, SBFPA e FA–EOBL [Leite et al. 2018], uma variante do Algoritmo do Vagalume combinada com EOBL. A Figura 1 apresenta o comportamento de convergência dos algoritmos. A Tabela 2 mostra os resultados numéricos dos experimentos, enquanto a Tabela 3 apresenta os resultados do Teste de Wilcoxon.

As subfiguras 1(a)–(e) mostram os resultados dos algoritmos durante a otimização das funções unimodais. Apenas na figura 1(b), a variante proposta não conseguiu superar SBFPA, que foi a variante que apresentou o melhor comportamento neste experimento. Apesar disso, os gráfico das subfiguras 1(a), 1(c) – 1(e) mostram claramente a superioridade do algoritmo proposto quando comparado aos demais algoritmos. A nova variante apresenta uma taxa de convergência mais ativa que os outros algoritmos. Vale destacar que, na subfigura 1(c), a nova variante encontra a solução ótima perto da iteração de número 1.100, enquanto os demais algoritmos não encontram a solução ótima.

Já as subfiguras 1(f)–(j) apresentam o comportamento médio dos algoritmos para otimizar as funções multimodais. Novamente, percebe-se que a velocidade de convergência da nova variante supera as demais, exceto na subfigura 1(i). A subfigura 1(g) mostra que FA–EOBL e o algoritmo proposto encontraram a solução ótima, entretanto a velocidade de convergência da variante proposta é maior. Na subfigura 1(h), também é mostrado que o algoritmo proposto encontrou a solução ótima, ao passo que os demais algoritmos não encontraram o ótimo global da função.

Na Tabela 2, para cada uma das funções avaliadas, são apresentados a média das soluções e o desvio padrão. Os valores em negrito representam os melhores resultados. Na Tabela 3, são apresentados os *p-values* do Teste de Wilcoxon resultantes da comparação entre a) algoritmo proposto x FPA, b) algoritmo proposto x FA–EOBL e c) algoritmo proposto x SBFPA.

O Teste de Wilcoxon, com nível de significância de 0.05, foi aplicado a fim de comparar os resultados das soluções encontradas pelo algoritmo proposto, SBFPA e FA–EOBL. O teste é usado para verificar se os resultados da proposta apresentada são, estatisticamente, significativos quando comparados aos das outras duas variantes, isto é, se o *p-value* é menor que o nível de significância determinado.

Tabela 2. Comparação de desempenho entre FPA, FA–EOBL, SBFPA e o Algoritmo Proposto.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
FPA	5.98e+00 (3.12)	6.97e+02 (9.27e+02)	1.85e+02 (3.14e+02)	1.79e-11 (7.66e-11)	5.66e+04 (3.97e+04)	4.01e+00 (0.81e+00)	08.04e+01 (01.75e+01)	1.04e+00 (7.16e-02)	3.12e+00 (0.60e+00)	6.68e+00 (2.39e+00)
FA–EOBL	8.33e-16 (2.80e-15)	2.87e+01 (5.06e-04)	1.10e-30 (3.28e-30)	4.48e-14 (9.34e-14)	9.13e-12 (2.95e-11)	1.64e-09 (3.76e-09)	**0** **(0)**	2.53e-15 (1.12e-14)	**4.99e-03** **(2.23e-02)**	1.01e-09 (4.06e-09)
SBFPA	1.22e-17 (1.45e-17)	**6.19e+00** **(2.86e+00)**	2.00e-34 (4.29e-34)	3.40e-45 (1.52e-44)	1.18e-13 (1.30e-13)	1.36e-09 (1.27e-09)	2.84e-15 (1.27e-14)	2.16e-16 (8.42e-16)	9.98e-02 (4.27e-09)	4.64e-07 (7.29e-07)
Alg. Proposto	**2.19e-319** **(0)**	28.81e+01 (3.57e-02)	**0** **(0)**	**1.48e-60** **(5.63e-60)**	**2.27e-242** **(0)**	**8.88e-16** **(0)**	**0** **(0)**	**0** **(0)**	4.43e-02 (2.85e-02)	**4.29e-161** **(1.49e-160)**

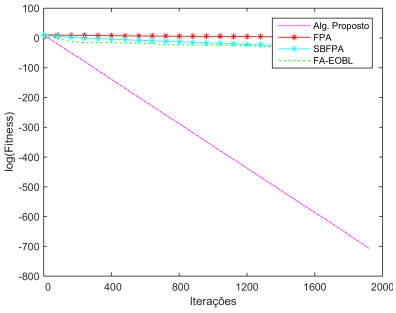
Tabela 3. Teste de Wilcoxon entre o Algoritmo Proposto e os outros algoritmos.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
FPA	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06
FA–EOBL	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	9.76e-04	1.00e+00	5.00e-01	2.09e-04	1.90e-06
SBFPA	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.90e-06	1.00e+00	1.00e+00	2.67e-05	1.90e-06

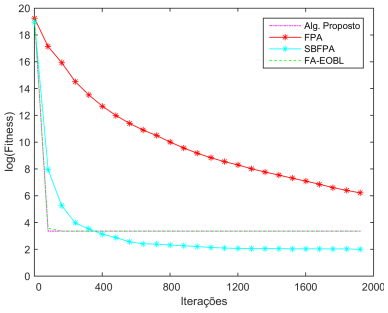
Os resultados da nova variante também foram comparados com a variante DE–FPA [Chakraborty et al. 2014] e também com o algoritmo de Evolução Diferencial (DE). Esses dois algoritmos não foram implementados em nossas simulações e, portanto, os seus resultados foram extraídos diretamente de [Chakraborty et al. 2014]. Das dez funções de referência usadas para validar a proposta apresentada, Chakraborty *et al.* (2014) realizaram simulações com apenas quatro dessas funções: Esfera (f_1), Rosenbrock (f_2), Rastrigin (f_7) e Griewank (f_8). Por esse motivo, a comparação considera apenas essas quatro funções. Para que nenhum algoritmo se beneficie de uma configuração que possa vir a ser a mais adequada, é usada a mesma configuração definida em [Chakraborty et al. 2014]: 30 flores, 30D e 2.000 iterações. A Tabela 4 apresenta a média das soluções e o desvio padrão encontrados.

Tabela 4. Comparação de desempenho entre DE, DE–FPA e o Alg. Proposto.

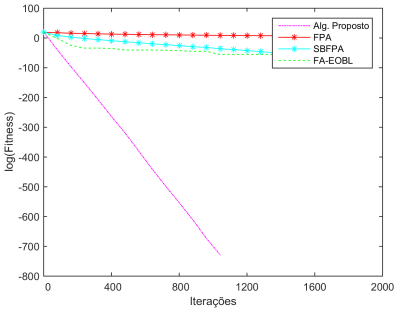
Função	DE	DE–FPA	Alg. Proposto
f_1	3.69e-11 (9.01e-11)	1.34e-14 (2.35e-14)	**2.19e-319 (0)**
f_2	42.35e+00 (28.09e+00)	2.72e+01 (2.09e+01)	2.88e+01 (3.57e-02)
f_7	6.95e-12 (1.24e-ll)	3.74e+01 (3.12e+00)	**0 (0)**
f_8	40.63e+00 (4.73e+00)	**0 (0)**	**0 (0)**



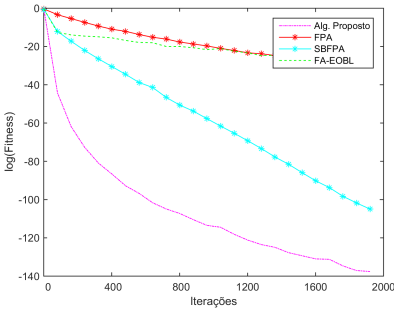
(a) f_1 – Esfera



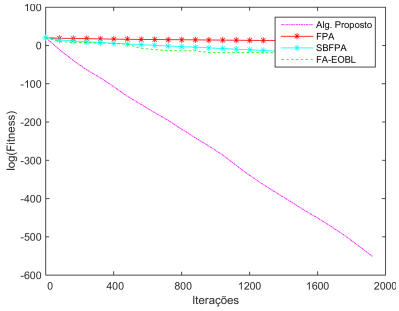
(b) f_2 – Rosenbrock



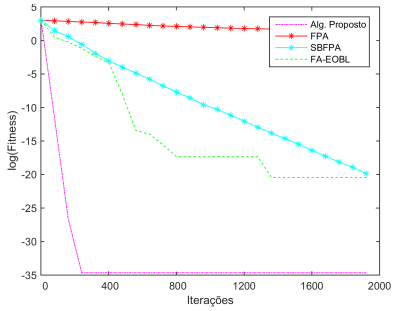
(c) f_3 – Schumer Steiglitz



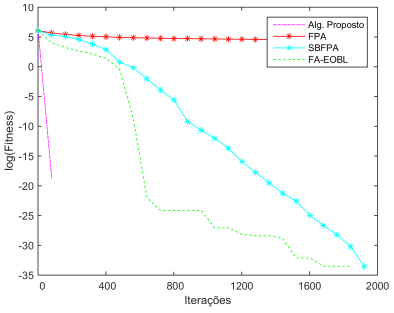
(d) f_4 – *Powell Sum*



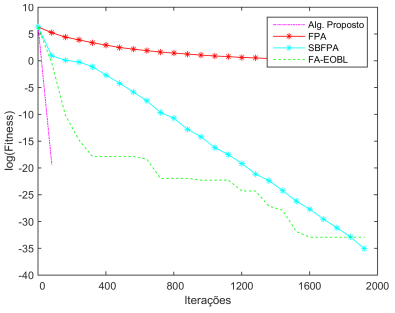
(e) f_5 – *Cigar*



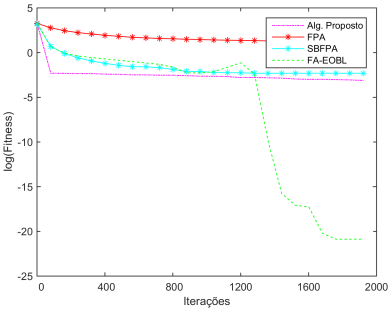
(f) f_6 – Ackley



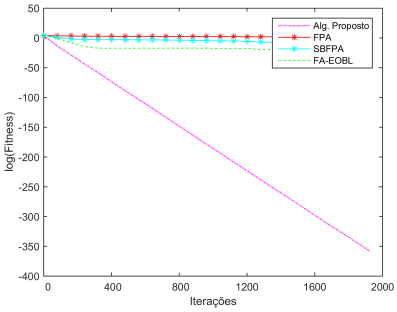
(g) f_7 – Rastrigin



(h) f_8 – Griewank



(i) f_9 – Salomon



(j) f_{10} – Alpine

Figura 1. Convergência média de FPA, FA–EOBL, SBFPA e Alg. Proposto.

5. Conclusões

Este trabalho propôs uma nova variante para o Algoritmo de Polinização das Flores (FPA) a partir de duas estratégias de aprendizagem baseada em oposição: *Quasi Opposition–Based Learning* (QOBL) e *Elite Opposition–Based Learning*. Observou-se que o algoritmo proposto reduziu o tempo de convergência do FPA e que as estratégias de aprendizagem baseada em oposição contribuíram para o algoritmo "escapar" das frequentes estagnações em ótimos locais.

O desempenho da variante proposta foi validado com o emprego de funções de referência, partindo do pressuposto de que elas se aproximam da complexidade de aplicações reais. Foram escolhidas dez funções de referência que possuem características de unimodalidade e de multimodalidade, aplicadas a problemas de minimização. Após a experimentação, a variante proposta mostrou 100% de superioridade em relação ao FPA e 95% em relação às outras variantes como DE–FPA, SBFPA e FA–EOBL.

Ainda realizaram-se outros experimentos que não foram reportados neste trabalho. As simulações consideraram QOBL e EOBL combinadas, individualmente, com o algoritmo FPA original. Depois de aplicar o Teste de Wilcoxon sobre as amostras apresentadas, observou-se que, em 100% dos experimentos, a contribuição da QOBL superou a da EOBL no resultado final do FPA.

Os resultados apresentados pela variante proposta são promissores. Como trabalhos futuros, pretende-se implementar as mesmas estratégias de Aprendizagem Baseada em Oposição, utilizadas neste trabalho, em uma outra meta-heurística inspirada no comportamento coletivo de cardumes de peixes conhecida como *Fish School Search*.

Referências

Chakraborty, D., Saha, S., and Dutta, O. (2014). De-fpa: A hybrid differential evolution-flower pollination algorithm for function minimization. In *International Conference on High Performance Computing and Applications (ICHPCA)*, pages 1–6. IEEE.

Hezam, I. M., Abdel-Baset, M., and Hassan, B. M. (2016). A hybrid flower pollination algorithm with tabu search for unconstrained optimization problems. *Inf. Sci. Lett*, 5:29–34.

Kalra, S. and Arora, S. (2016). Firefly algorithm hybridized with flower pollination algorithm for multimodal functions. In *Proceedings of the International Congress on Information and Communication Technology*, pages 207–219. Springer.

Leite, I. V., Marcone, M. H., and Paiva, F. A. (2018). Meta-heurística inspirada na bioluminescência dos vaga-lumes usando aprendizagem baseada em oposição elite. *Anais do Computer on the Beach*, pages 880–889.

Mantegna, R. N. (1994). Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E*, 49(5):4677.

Meng, O. K., Pauline, O., Kiong, S. C., Wahab, H. A., and Jafferi, N. (2017). Application of modified flower pollination algorithm on mechanical engineering design problem. In *IOP Conference Series: Materials Science and Engineering*, volume 165.

Paiva, F. A. P., Silva, C. R. M., Leite, I. V. O., Marcone, M. H., and Costa, J. A. F. (2017). Uma nova abordagem do conceito de serendipidade como base para a meta-

heurística inspirada no processo de polinização das flores. In *XIII Congresso Brasileiro em Inteligência Computacional (CBIC2017)*.

Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. (2007). Quasi-oppositional differential evolution. In *IEEE Congress on Evolutionary Computation*, pages 2229–2236.

Ramadas, M. and Kumar, S. (2016). An efficient hybrid approach using differential evolution and flower pollination algorithm. In *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pages 59–64.

Saxena, P. and Kothari, A. (2016). Linear antenna array optimization using flower pollination algorithm. *SpringerPlus*, 5(1):306.

Shareef, S. S., Mohideen, E. R., and Ali, L. (2015). Directed firefly algorithm for multimodal problems. In *Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–6.

Tizhoosh, H. R. (2005). 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.

Tran, T., Nguyen, T. T., and Nguyen, H. L. (2014). Global optimization using lÃ©vy flights. *arXiv preprint arXiv:1407.5739*.

Yang, X.-S. (2012). Flower pollination algorithm for global optimization. In *International Conference on Unconventional Computing and Natural Computation*, pages 240–249. Springer.

Zhou, X., Wu, Z., and Wang, H. (2012). Elite opposition-based differential evolution for solving large-scale optimization problems and its implementation on gpu. In *13th Int. Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 727–732.