Uma Nova Abordagem do Conceito de Serendipidade como base para a Meta-heurística Inspirada no Processo de Polinização das Flores

Fábio A. P. Paiva[1], Cláudio R. M. Silva[2], Izabele V. O. Leite[2], Marcos H. F. Marcone[2] e José A. F. Costa[2]

[1] Instituto Federal do Rio Grande do Norte, IFRN, Parnamirim, Brasil.
fabio.procopio@ifrn.edu.br
[2] Universidade Federal do Rio Grande do Norte, UFRN, Natal, Brasil.
{claudio.rmsilva, izaleite11, marcosmarcone48, jafcosta} @gmail.com

Resumo. Muitas vezes, os métodos meta-heurísticos se deparam com um problema conhecido como convergência prematura, o que faz com que eles percam a capacidade de gerar diversidade. Embora várias abordagens tenham sido propostas, a convergência prematura ainda é uma questão em aberto. Meta-heurísticas mais recentes como Algoritmo de Polinização das Flores (FPA) também enfrentam o problema. Este trabalho apresenta um conceito da área de sistemas de recomendação, chamado serendipidade, como uma nova abordagem para o contexto de otimização. O trabalho formaliza serendipidade e também propõe uma nova variante FPA que implementa duas dimensões desse conceito: acaso e sagacidade. O algoritmo proposto foi comparado com o FPA e com uma variante da literatura chamada DE–FPA. Os resultados obtidos mostraram a superioridade da variante proposta quando comparada ao FPA e à DE–FPA.

Palavras-chave: Meta-heurística, Algoritmo de Polinização das Flores, Serendipidade, Aprendizagem Baseada em Oposição.

1 Introdução

Os problemas de otimização estão presentes em diversos domínios do mundo real como, por exemplo, engenharia mecânica [1], gerenciamento de tráfego aéreo [2] e outros. As meta-heurísticas vêm sendo amplamente empregadas para resolver problemas como esses. Algumas delas se inspiraram na natureza como Algoritmos Genéticos, Algoritmo do Vaga-lume, Algoritmo do Morcego e outros.

Em [3], a reprodução das flores foi estudada a fim de propor uma nova meta-heurística conhecida como Algoritmo de Polinização das Flores (FPA – *Flower Pollination Algorithm*). Na modelagem desse algoritmo, uma flor pode ser usada para representar uma solução de um determinado problema. Assim como muitas meta-heurísticas, FPA também enfrenta um problema chamado convergência prematura, que ocorre quando o algoritmo perde a capacidade de gerar diversidade. Na área de sistemas de recomendação, há um conceito, conhecido como *serendipidade*, que pode ser utilizado como uma alternativa para lidar com o problema da convergência prematura.

Em geral, serendipidade é um termo que se refere a descobertas realizadas, aparentemente, por acaso. Na história da ciência, há vários exemplos de serendipidade como as descobertas da vaselina, da sacarina e do raio-X. As inovações podem ser consideradas casos de serendipidade, uma vez que elas são realizadas por indivíduos capazes de detectar padrões.

Nos últimos anos, o foco das pesquisas em sistemas de recomendação tem sido a procura pela exatidão das recomendações [4], apesar de se saber que, por mais exata que seja uma recomendação, isso pode ocasionar o problema *over-specialization* que consiste em o sistema recomendar apenas os itens adequados para o perfil do usuário, embora possam existir outros mais adequados. Quando um sistema de recomendação sugere apenas itens relacionados ao perfil de interesse do usuário, ele converge para recomendações que podem não atender as reais expectativas do usuário e, dessa forma, deixar de sugerir itens que podem ser mais adequados. De forma similar, quando um algoritmo meta-heurístico converge para um ótimo local sem levar em consideração outras soluções que são mais adequadas que a encontrada, é possível perceber a relação entre *over-specialization* e convergência prematura.

Na literatura, diversos trabalhos já foram propostos a fim de melhorar o desempenho do FPA. Em geral, eles são propostas híbridas. Em [5], a variante DE–FPA combina o algoritmo de Evolução Diferencial (DE – *Differential Evolution*) com o FPA. FA–FPA [6] é outra variante que combina FPA com o algoritmo do vaga-lume (FA – *Firefly Algorithm*). FPA também foi combinado com o algoritmo de Busca Tabu (TS – *Tabu Search*) para implementar a variante TS–FPA [7]. ABCFP [8] é uma outra variante resultante da combinação entre FPA e o algoritmo ABC (*Artificial Bee Colony*).

Este trabalho propõe uma variante FPA que utiliza uma nova abordagem do conceito de serendipidade. A abordagem tem como objetivo aumentar a capacidade do algoritmo original balancear a busca local e a busca global. A busca local é melhorada por meio da dimensão *sagacidade*, que é implementada a partir de inspeções nas adjacências da melhor solução atual. Já a busca global usa a dimensão *acaso*, que é implementada por meio da aprendizagem baseada em oposição. Os experimentos mostraram a superioridade da variante proposta quando comparada ao FPA original, ao algoritmo DE original e à variante DE–FPA.

A estrutura do trabalho está organizada com segue. Na Seção 2, é apresentada uma breve introdução sobre FPA e a aprendizagem baseada em oposição; na Seção 3, a nova variante que se propõe a melhorar a performance do FPA original é apresentada. Na Seção 4, os experimentos 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 do algoritmo apresentado na Seção 3, tais como o Algoritmo de Polinização das Flores e os conceitos de aprendizagem baseada em oposição e de serendipidade.

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*) [3]. 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 [9]. 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 [3].

A geração de tamanhos de passos pseudo-aleatórios que obedeçam, corretamente, a Equação 2 não é uma tarefa trivial [3]. Existem alguns métodos para geração de números aleatórios e o mais eficiente é o algoritmo de Mantegna [10], o qual usa duas distribuições gaussianas U e V, conforme equação a seguir:

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

com

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

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

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 de 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
```

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.

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. Nesse caso, aprendizagem baseada em oposição (OBL – *Opposition-Based Learning*) [11] pode ser aplicada. Este conceito é da área de inteligência computacional e 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 do ótimo global. A seguir, são definidos Número Oposto e Ponto Oposto.

***Definição 1*: Número Oposto** – Dado $x \in \Re$, no intervalo $x \in [a, b]$, o número oposto $\tilde{x}$ é definido como

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

***Definição 2*: Ponto Oposto** – Dado $P = (x_1, x_2, ..., x_n)$ como um ponto no espaço n-dimensional, com x_1, x_2, ..., $x_n \in \Re$ 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

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

2.3 Otimização Baseada em Serendipidade

O termo "serendipidade" é usada como a) algo achado sem, necessariamente, estar procurando-se por ele [12], b) surpresa agradável [13], c) acaso ou d) sorte. Campos e Figueiredo [14] foram um dos primeiros autores que apresentaram o conceito de serendipidade formalmente por meio da identificação de diferentes categorias. Para isso, foram utilizadas equações lógicas, chamadas Equações de Serendipidade, para apresentar quatro eventos que podem gerar serendipidade: a) pseudosserendipidade; b) serendipidade real; c) serendipidade sem uma metáfora de inspiração; e d) serendipidade com o conhecimento incorreto.

Lawley [15] propôs um modelo perceptivo de serendipidade que considera seis eventos que garantem a existência da serendipidade. São eles: 1) uma "mente preparada", 2) um evento não planejado e inesperado, 3) o reconhecimento da potencialidade de um significado no futuro, 4) uma ação para ampliar a potencialidade do efeito considerado no Evento 2, 5) efeitos que podem ser utilizados para amplificar ainda mais o benefício do Evento 2 e 6) ocorrência de serendipidade.

Na definição de serendipidade, existem duas importantes dimensões que devem ser consideradas [16]: *acaso* e *sagacidade*. A dimensão "acaso" representa um conjunto formado por ideias e intuições que, normalmente, não são combinadas. Já a dimensão "sagacidade" é o ambiente no qual essas ideias e intuições podem ser aproveitadas. Neste trabalho, o termo "acaso" deve ser entendido como uma possibilidade de algo acontecer, enquanto "sagacidade" é a qualidade de perceber um determinado evento ou algo relacionado a ele.

Considerando um espaço de busca S, diz-se que os elementos que representam as possíveis soluções de S pertencem ao conjunto P. O conjunto E, por sua vez, é formado por elementos que representam as soluções encontradas no processo de busca das possíveis soluções do espaço S, durante uma determinada iteração. Portanto, E é um subconjunto próprio de P, $E \subseteq P$.

Diz-se que e_{ij} é um elemento que representa uma solução i, encontrada na iteração j, pertencente ao conjunto E. A solução i, cujo valor da função objetivo é o melhor encontrado na iteração j, é representada pelo elemento $e^*_{ij} \in E$. Então, na iteração j, quando um elemento de P não pertence ao conjunto E, o elemento é considerado uma solução *ocasional*. Portanto, o conjunto que possui os elementos usados para representar as soluções ocasionais na iteração j é dado pelo complementar do conjunto E em relação a P, conforme Equação 9:

$$ACASO = \ P - E. \tag{9}$$

Um outro conjunto a ser considerado na formalização de serendipidade é o SRD. Ele é formado por elementos chamados *serendípetos* e é dado por

$$SRD = ACASO \ \cap \ SAGAZ, \tag{10}$$

onde $SAGAZ$ é um subconjunto próprio de $ACASO$, $SAGAZ \subseteq ACASO$, formado por elementos $sagaz_{ij}$ encontrados de forma sagaz. A sagacidade consiste na descoberta de soluções melhores que o elemento e^*_{ij}, conforme Equação 11:

$$SAGAZ = \{sagaz_{ij} \mid f(sagaz_{ij}) < f(e^*_{ij})\}, \tag{11}$$

onde $f(\cdot)$ é a função objetivo usada para medir o valor de *fitness*.

3 SBFPA: FPA Baseado em Serendipidade

Após a formalização do conceito de serendipidade, inserido no contexto de otimização, é apresentada a variante *Serendipity-Based Flower Pollination Algorithm* (SBFPA). Ela pode ser usada como uma alternativa para lidar com o problema da convergência prematura e, assim, melhorar o desempenho do FPA original.

A fim de gerar diversidade, SBFPA adota a abordagem que foi usada na implementação de uma variante do *Particle Swarm Optimization* [17–19]. Ela combina duas estratégias do domínio de sistemas de recomendação [20]: *blind luck* e *Princípio de Pasteur*. *Blind luck* é usada na implementação da dimensão acaso para complementar a exploração do espaço de busca e gerar diversidade adicional. Já o Princípio de Pasteur é usado na implementação da dimensão sagacidade para "reconhecer potenciais" e "aproveitar os momentos".

O pseudocódigo da variante proposta pode ser sumarizado no Algoritmo 2. Na linha 1, o algoritmo difere do original ao utilizar a abordagem baseada em oposição para gerar soluções iniciais. A linha 4 apresenta outra diferença em relação ao valor atribuído à taxa de probabilidade p. No algoritmo original, o valor padrão é fixado em 0.8, embora nos experimentos realizados, o valor atribuído tenha sido $p = 0.7$. No entanto, valores fixos não são suficientes para equilibrar a busca local e a global [21]. Portanto, na variante proposta, o valor de p é gerado dinamicamente, de acordo com os valores de *fitness* da iteração atual e ele é calculado conforme Equação 12:

$$p = e^{(max(fitness) - min(fitness))/max(fitness)}, \tag{12}$$

onde $max(fitness)$ e $min(fitness)$ são o maior e o menor valores de *fitness* da função objetivo, na iteração atual, respectivamente.

A implementação do conceito de serendipidade ocorre nas linhas 7–9. A dimensão acaso é iniciada com um sorteio aleatório de uma flor, $flor_s$. Na linha 8, OBL é aplicada sobre $flor_s$. Sobre a implementação da dimensão sagacidade, ela é realizada na linha 9, onde a pior flor, $flor_{pior}$, é deslocada para a vizinhança da melhor solução encontrada até o momento, conforme Equação 13:

$$flor_{pior} = g_* + \delta, \tag{13}$$

onde g_* é a melhor flor da iteração atual, δ é um valor obtido aleatoriamente no intervalo [0, 0.1 * $(lim_{max} - lim_{min})$], representando 10% do espaço de busca.

Algoritmo 2 Pseudocódigo da variante SBFPA

1: Gere uma população aleatória usando OBL
2: Encontre a melhor solução g_* dentro da população gerada
3: **enquanto** $(t \leq num\_max\_iter)$ **faça**
4: Calcule p usando a Equação 12
5: **para** i **de** 1 **até** n **faça**
6: Idem às linhas 6–12 do Algoritmo 1
7: Sorteie aleatoriamente uma flor, $flor_s$
8: $flor_s \leftarrow OBL(flor_s)$
9: Modifique pior flor usando Equação 13
10: **fim para**
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

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 [5, 6, 22] e são apresentadas na Tabela 1. Para cada uma delas, são apresentadas a sua formulação, as fronteiras do espaço de busca e a solução ótima.

4.2 Simulações e Resultados

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 comparam FPA com a variante SBFPA por meio de diversos experimentos computacionais, realizados em 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 $n = 30$.

As subfiguras 1(a) − 1(e) apresentam o comportamento médio dos algoritmos durante a otimização das funções unimodais. Para nenhuma delas, os algoritmos encontraram a solução ótima, apesar de os gráficos mostrarem claramente a superioridade da variante proposta quando comparada ao FPA original. Todavia, na subfigura 1(b), é observado que, após a iteração de número 800 a variante

Tabela 1: Funções de referência utilizadas para avaliar FPA e SBFPA.

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

proposta apresenta uma velocidade de convergência mais lenta, embora seja superior à velocidade do FPA.

Já as subfiguras 1(f) – 1(j) apresentam o comportamento médio dos algoritmos para otimizar as funções que possuem vários ótimos locais. Novamente, percebe-se a superioridade de convergência da variante SBFPA. Na subfigura 1(h), SBFPA encontrou a solução ótima da função Griewank próximo da iteração 1.950. Nas demais funções multimodais, as soluções ótimas não foram encontradas por nenhum dos dois algoritmos. Na subfigura 1(i), próximo da iteração 1.100, a variante não "escapou" de um ótimo local, embora mais uma vez, seu desempenho tenha superado o FPA original.

O Teste de Wilcoxon, com nível de significância de 0.05, foi aplicado a fim de comparar os resultados das soluções encontradas pela variante proposta e pelo FPA. O teste é usado para verificar se os resultados da variante são estatisticamente significativos quando comparados aos do FPA, isto é, se o *p-value* é menor que o nível de significância determinado.

A Tabela 2 compara o desempenho do FPA original e do SBFPA. Para cada uma das funções avaliadas, a tabela mostra a melhor solução, a pior solução, a

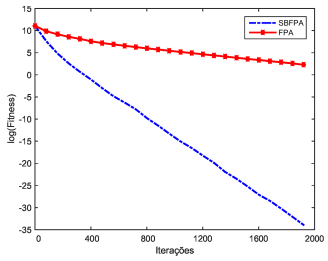
média das soluções, o desvio padrão e o *p-value* resultante do Teste de Wilcoxon. Os valores em negrito representam a melhor média e o desvio padrão dos experimentos.

Tabela 2: Comparação entre FPA e SBFPA para as funções avaliadas.

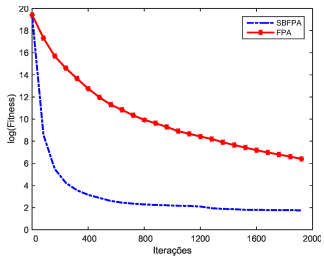
Função	**FPA**			**SBFPA**			**p-Value**
	Melhor	Pior	Média (Desvio)	Melhor	Pior	Média (Desvio)	
f_1	2.07	21.06	7.63 (4.54)	9.43e-18	2.01e-15	**4.11e-16 (6.04e-16)**	1.90e-06
f_2	34.42	1.06e+03	5.11e+02 (2.58e+02)	1.03	11.59	**5.67 (2.52)**	1.90e-06
f_3	7.35	1.10e+03	2.44e+02 (2.67e+02)	2.53e-36	8.81e-32	**8.21e-33 (2.08e-32)**	1.90e-06
f_4	3.01e-18	3.12e-12	6.06e-13 (8.89e-13)	1.91e-52	5.11e-46	**4.77e-47 (1.41e-46)**	1.90e-06
f_5	9.52e+03	2.50e+05	6.55e+04 (5.55e+04)	8.00e-15	2.78e-12	**5.41e-13 (7.55e-13)**	1.90e-06
f_6	2.74	6.69	4.13 (0.96)	1.03e-10	4.08e-09	**9.15e-10 (9.76e-10)**	1.90e-06
f_7	38.49	99.51	76.30 (15.87)	0	1.13e-13	**5.68e-15 (2.54e-14)**	1.90e-06
f_8	0.96	1.17	1.06 (6.18e-02)	0	0	**0 (0)**	1.90e-06
f_9	2.19	4.39	3.24 (0.58)	1.14e-02	9.98e-02	**9.54e-02 (1.97e-02)**	1.90e-06
f_{10}	3.49	10.79	7.87 (2.22)	8.94e-09	1.50e-05	**1.70e-06 (4.42e-06)**	1.90e-06

Tabela 3: Comparação de desempenho dos diferentes algoritmos.

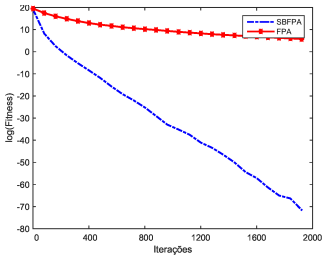
Função	**DE**	**FPA**	**DE–FPA**	**SBFPA**
Esfera	3.69e-11 (9.01e-11)	8.40e+00 (5.03e+00)	1.34e-14 (2.35e-14))	**4.11e-16 (6.04e-16)**
Rosenbrock	42.35 (28.09)	5.11e+02 (2.57e+02)	27.19 (20.93)	**5.67 (2.52)**
Griewank	6.95e-12 (1.24e-ll)	1.06e+00 (3.90e-02)	**0 (0)**	**0 (0)**
Rastrigin	40.63 (4.73)	8.64e+01 1.73e+01	37.38 (3.12)	**5.68e-15 (2.54e-14)**



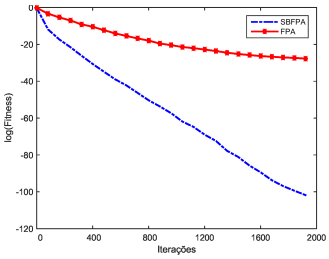
(a) Esfera (f_1)



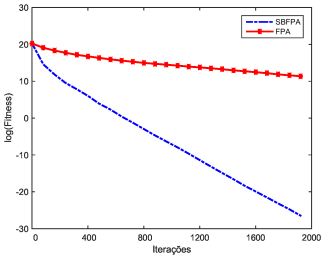
(b) Rosenbrock (f_2)



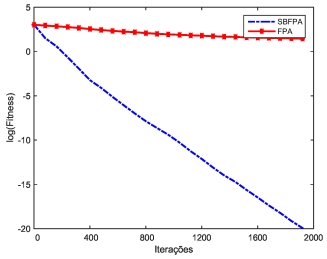
(c) Schumer Steiglitz (f_3)



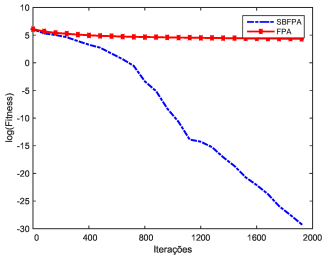
(d) *Powell Sum* (f_4)



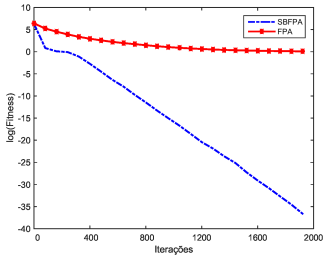
(e) Cigar (f_5)



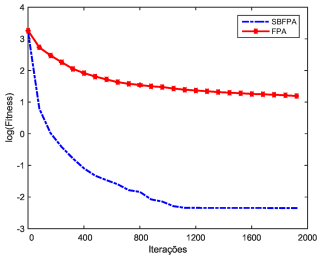
(f) Ackley (f_6)



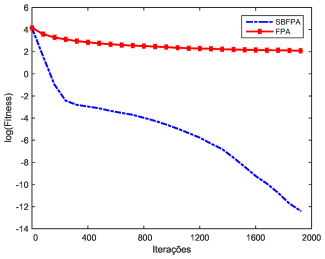
(g) Rastrigin (f_7)



(h) Griewank (f_8)



(i) Salomon (f_9)



(j) Alpine (f_{10})

Figura 1: Convergência média dos algoritmos nas funções avaliadas.

Os resultados da variante proposta também foram comparados com uma variante da literatura chamada DE–FPA [5] e também com o algoritmo de Evolução Diferencial (DE). Para que nenhuma das variantes se beneficie de uma configuração que possa vir a ser a mais adequada, nesta comparação, é usada a mesma configuração definida em [5]: 30 flores, 30D e 2.000 iterações. A Tabela 3 apresenta a média dos valores de *fitness* e o desvio padrão encontrados.

Os resultados apresentados na Tabela 2 demostram, claramente, que a variante SBFPA supera o FPA original. Já, na Tabela 3, SBFPA também supera a variante DE–FPA em todas as funções, exceto na Griewank porque ambas encontraram o ponto ótimo. Os resultados revelam que, quando comparada ao FPA e à DE–FPA, a variante proposta apresenta estabilidade, melhores soluções e maior velocidade de convergência.

5 Conclusões

Este trabalho apresentou o conceito de serendipidade como base para a criação de uma variante da meta-heurística inspirada no processo de polinização das flores. A variante foi denominada SBFPA e tem o objetivo de gerar diversidade de soluções e aumentar a velocidade de convergência do algoritmo.

O desempenho do SBFPA foi comparado com o FPA original, com o DE original e com uma variante híbrida chamada DE–FPA. A comparação foi realizada por meio de vários experimentos computacionais usando funções de referência clássicas da literatura. Após os experimentos, observou-se que, no que diz respeito à qualidade e à estabilidade das soluções, a variante baseada no conceito de serendipidade obteve resultados superiores aos do algoritmo FPA, do algoritmo DE e da variante híbrida DE–FPA.

Como trabalhos futuros, é interessante o desenvolvimento de uma variante que possa ser aplicada em problemas discretos, como o do Caixeiro-Viajante. A validação nesse problema é importante por ele ser bastante utilizado em cenários reais como roteamento de veículos, problemas de sequenciamento e outros.

Referências

1. E. K. da Silva, H. J. Barbosa, and A. C. Lemonge, "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization," *Optimization and Engineering*, vol. 12, no. 1, pp. 31–54, 2011.
2. A. Alonso-Ayuso, L. F. Escudero, and F. J. Martín-Campo, "Multiobjective optimization for aircraft conflict resolution. a metaheuristic approach," *European Journal of Operational Research*, vol. 248, no. 2, pp. 691–702, 2016.
3. X.-S. Yang, "Flower pollination algorithm for global optimization," in *International Conference on Unconventional Computing and Natural Computation*. Springer, 2012, pp. 240–249.
4. M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: evaluating recommender systems by coverage and serendipity," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 257–260.

5. D. Chakraborty, S. Saha, and O. Dutta, "De-fpa: A hybrid differential evolution-flower pollination algorithm for function minimization," in *Int. Conference on High Performance Computing and Applications*, 2014, pp. 1–6.
6. S. Kalra and S. Arora, "Firefly algorithm hybridized with flower pollination algorithm for multimodal functions," in *Proceedings of the International Congress on Information and Communication Technology*. Springer, 2016, pp. 207–219.
7. A.-B. M. Hezam, Ibrahim M. and B. M. Hassan, "A hybrid flower pollination algorithm with tabu search for unconstrained optimization problems," *Information Sciences Letters*, vol. 5, no. 1, Jan 2016.
8. E. El-sharkay, "A hybrid bee colony algorithm for solving unconstrained optimization problems," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 28, no. 9, pp. 480–484, Out 2015.
9. T. Tran, T. T. Nguyen, and H. L. Nguyen, "Global optimization using l\'evy flights," *arXiv preprint arXiv:1407.5739*, 2014.
10. R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of levy stable stochastic processes," *Physical Review E*, vol. 49, no. 5, p. 4677, 1994.
11. H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on*, vol. 1. IEEE, 2005, pp. 695–701.
12. J. H. Austin, *Chase, chance, and creativity: The lucky art of novelty*. Mit Press, 2003.
13. J. Tolson, "A word's eventful journey," *US News & World Report*, vol. 136, no. 4, p. 51, 2004.
14. J. Campos and A. D. Figueiredo, "Programming for serendipity," in *Proceedings of Fall Symposium on Chance Discovery - The Discovery and Management of Chance Events*, 2002, pp. 48–60.
15. J. Lawley, "Maximising serendipity: The art of recognising and fostering unexpected potential – a systemic approach to change," in *Neuro Linguistic Psychotherapy and Counselling Association 2013*, 2013.
16. S. Olma, "14 university as übungsraum," *Higher Education and the Creative Economy: Beyond the Campus*, p. 261, 2016.
17. F. A. P. Paiva, "Estudo do conceito de serendipidade como base para novas abordagens ao problema da convergência prematura," Ph.D. dissertation, 2016.
18. F. A. P. Paiva, J. A. F. Costa, and C. R. M. Silva, "A serendipity-based pso approach to delay premature convergence using scout particle," *International Journal of Innovative Computing, Information and Control*, vol. 12, no. 4, Ago 2016.
19. ——, "A serendipity-based approach to enhance particle swarm optimization using scout particles," *IEEE Latin America Transactions*, vol. 15, no. 6, pp. 1101–1112, June 2017.
20. E. Toms, "Serendipitous information retrieval," in *First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, 2000, pp. 11–12.
21. W. Zhang, Y. Yang, S. Zhang, D. Yu, and Y. Xu, "A new manufacturing service selection and composition method using improved flower pollination algorithm," *Mathematical Problems in Engineering*, vol. 2016, 2016.
22. M. Ramadas and S. Kumar, "An efficient hybrid approach using differential evolution and flower pollination algorithm," in *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*. IEEE, 2016, pp. 59–64.