Uma Meta-Heurística Alternativa de Inteligência de Enxames Baseada em Serendipidade Guiada

Fábio A. Procópio de Paiva
Depto. Acadêmico, Campus Parnamirim
Instituto Federal do RN, Parnamirim/Brasil
fabio.procopio@ifrn.edu.br

José Alfredo F. Costa
Depto. de Eng. Elétrica
Univ. Federal do RN, Natal/Brasil
jafcosta@gmail.com

Cláudio R. M. Silva
Depto. de Eng. de Comunicações
Univ. Federal do RN, Natal/Brasil
claudio.rmsilva@gmail.com

Resumo – No estudo das técnicas de meta-heurística, é muito comum lidar com um problema conhecido como convergência prematura. Este problema é mais conhecido no contexto dos algoritmos genéticos, mas tem sido observado em outros métodos de meta-heurística como *Particle Swarm Optimization* (PSO). A maioria das abordagens para o problema considera a geração e/ou o posicionamento de indivíduos no espaço de busca de forma aleatória. Este trabalho aborda o problema usando o conceito de serendipidade e sua adaptação neste novo contexto. Várias técnicas que implementam serendipidade foram avaliadas com o objetivo de construir uma variante do PSO baseada nesse conceito. Os resultados foram comparados com o PSO tradicional e levaram em consideração a qualidade das soluções encontradas e a capacidade de localizar ótimos globais. O protótipo apresentou resultados promissores com relação aos critérios citados anteriormente, embora demonstre a necessidade de ajustes adicionais para diminuição do tempo de execução.

Palavras-chave: Otimização, Serendipidade, Inteligência de Enxames, Meta-Heurística

I. Introdução

Uma característica comum à maioria dos métodos de meta-heurística é a definição de mecanismos para criação de diversidade que evitem a convergência prematura para ótimos locais. Nos algoritmos genéticos, por exemplo, esse mecanismo é implementado por meio de operações genéticas (reprodução, *crossover* ou mutação) que geram novos indivíduos [1]. Já no *Particle Swarm Optimization* (PSO), que explora a inteligência de enxames, o mecanismo é implementado pela atribuição de velocidades aleatórias na direção de um ótimo global durante o processo de convergência. Na área de sistemas de recomendação, existe um conceito que pode ser usado como uma estratégia para evitar a convergência prematura. Esse conceito é chamado de *serendipidade*.

De modo geral, serendipidade é um termo que se refere a descobertas afortunadas realizadas, aparentemente, por acaso. Em um estudo exaustivo, Van Andel [2] mostrou que a serendipidade tem uma grande contribuição para o progresso da ciência, da tecnologia e da arte. Além disso, Kuhn [3] acredita ainda que, na ciência e na tecnologia, a frequência de descobertas serendípetas é maior do que se imagina. Na história da ciência, por exemplo, há vários exemplos de serendipidade como as descobertas 1) da vaselina, em 1859; 2) da sacarina, em 1878; 3) do raio-X, em 1895; 4) da radioatividade, em 1896; 5) da penicilina, em 1928; 6) do forno micro-ondas, em 1945, além de tantas outras. As inovações, por exemplo, podem ser consideradas casos de serendipidade uma vez que, via de regra, elas são realizadas por indivíduos capazes de detectar padrões onde outros veem aleatoriedade.

Nos últimos anos, o foco das pesquisas em sistemas de recomendação tem sido a procura pela exatidão das recomendações [4][5][6][7][8], apesar de se saber que, por mais exata que seja uma recomendação, isso pode ocasionar um problema específico conhecido como *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 correlação entre *over-specialization* e convergência prematura.

Este trabalho propõe uma variante do método PSO baseado em um modelo perceptivo de serendipidade [9] com o objetivo de resolver o problema da convergência prematura e evitar que partículas do enxame fiquem presas em ótimos locais. O protótipo serve como uma prova conceitual da viabilidade de se utilizar o conceito de serendipidade no contexto de algoritmos meta-heurísticos. Ele foi testado em várias funções de *benchmark* com características de unimodalidade e de multimodalidade. Os resultados mostram que o método proposto supera o PSO nos critérios qualidade das soluções e capacidade de encontrar ótimos globais, apesar da necessidade de redução do seu tempo de execução.

A estrutura do trabalho está organizada da seguinte maneira: na Seção II, é apresentado o conceito de serendipidade; na III, os trabalhos relacionados são apresentados e analisados; na Seção IV, a abordagem da otimização baseada em serendipidade é apresentada; na V, os experimentos e os resultados são apresentados em detalhes; e por fim, na Seção VI, são apresentadas as considerações finais do trabalho.

II. Conceito de Serendipidade

O que há de comum entre o raio-X, o *nylon* e a vacina? Uma possível resposta é que todos eles foram descobertos por acaso [10], ou seja, as descobertas foram realizadas por meio de serendipidade.

Em 1754, o romancista Horace Walpole escreveu uma carta destinada a um amigo na qual era narrada a história persa de um conto intitulado *The Three Princes of Serendip*. Na história, os príncipes realizam descobertas cujos resultados não eram esperados. Por serem observadores e sagazes, os três rapazes descobrem, por acaso, soluções para alguns dilemas. Naquela carta, Walpole cunhou o termo serendipidade com o objetivo de expressar descobertas que acontecem por acaso.

Uma definição para serendipidade é que ela consiste na arte de encontrar o que não se está procurando [11]. O dicionário *Cambridge* define-a como *"The fact of finding interesting or valuable things by chance"*. No entanto, para Catellin [12], ao contrário do que muitas pessoas imaginam, a serendipidade não se refere a apenas descobertas acidentais, mas sim a uma mistura de sagacidade e de acaso.

Campos e Figueiredo [10] foram um dos primeiros pesquisadores a expressar o conceito de serendipidade de uma maneira formal por meio da identificação de diferentes categorias. Para isso, os autores utilizaram equações lógicas, chamadas de *Equações de Serendipidade*, para apresentar quatro eventos que podem gerar serendipidade: a) pseudo-serendipidade; b) serendipidade real; c) serendipidade sem uma metáfora de inspiração e; d) serendipidade com o conhecimento incorreto.

Recentemente, Lawley e Tompkins propuseram um modelo perceptivo de serendipidade [9] formado por seis componentes, os quais representam eventos que ocorrem durante o processo de serendipidade: Ev_{t-1}, Ev, Ev_{t+1}, Ev_{t+2}, Ev_{t+3} e Ev_{t+4}. No modelo proposto, Ev é um evento não planejado e ocasional, Ev_{t-1} ocorre no tempo anterior a Ev, Ev_{t+1} ocorre após Ev, e assim por diante.

No contexto meta-heurístico, para um determinado espaço de busca, o conjunto E é formado por elementos que representam as soluções encontradas em uma iteração. Assim, dizemos que e_{ij} é um elemento que representa uma solução i, encontrada na iteração j, pertencente ao conjunto E. A solução i cujo valor de *fitness* é o melhor valor encontrado na iteração j é representada pelo elemento e^*_{ij}.

S é um conjunto cujos elementos representam as possíveis soluções de um espaço de busca e, portanto E é um subconjunto próprio de S. Assim, na iteração j, quando um elemento de S não pertence ao conjunto E, considera-se que esse elemento é uma solução *ocasional*. Então, o conjunto contendo os elementos que representam as soluções *ocasionais* na iteração j é dado pelo complemento relativo de E em S:

$$ACASO = S - E \tag{1}$$

Quando um elemento $acaso_{ij}$ apresentar um valor de *fitness* melhor que o do elemento e^*_{ij}, diz-se que a solução representada por esse elemento é *serendípeta*. O conjunto SRD é formado por elementos que representam soluções ocasionais cujos valores de *fitness* são melhores que o do *gBest* corrente, conforme equação (2):

$$SRD = \{srd_{ij} | fitness(acaso_{ij}) < fitness(e^*_{ij})\} \tag{2}$$

O conceito de serendipidade apresentado neste trabalho consiste em dois aspectos essenciais: *aceitabilidade* e *ocasionalidade*. Assim um elemento do conjunto SRD pode ser usado para representar uma solução serendípeta porque ele é a) *aceitável*, por representar uma solução cujo valor de *fitness* é melhor que o *fitness* do elemento e^*_{ij} e b) *ocasional*, por ser um elemento que não pertence ao conjunto E na iteração j.

III. Trabalhos Relacionados

Uma classe meta-heurística que tem recebido bastante atenção é formada por algoritmos bio-inspirados. Um dos conceitos bastante difundidos na computação bio-inspirada é o de algoritmos baseados em Inteligência de Enxames [13]. Esses algoritmos são inspirados no comportamento de alguns seres vivos como pássaros, peixes, morcegos e outros. A auto-organização e o controle descentralizado são características marcantes desse tipo de algoritmo. Nesse contexto, o PSO é um dos algoritmos bastante conhecidos.

O PSO surgiu da modelagem do comportamento social observado em espécies de pássaros e de peixes, inclusive o comportamento das pessoas também. No PSO [14], os indivíduos da população são representados por pontos, chamados de partículas, que percorrem o espaço de busca. O movimento das partículas é baseado em duas informações: g_{best} e p_{best}. O g_{best} influencia o movimento da partícula para a direção da melhor posição encontrada pelo enxame, enquanto o p_{best} movimenta a partícula para a melhor posição encontrada por ela. Depois de encontrar os valores g_{best} e p_{best}, cada partícula atualiza sua velocidade e sua posição por meio de (3) e (4), respectivamente:

$$v^{(t+1)}_{id} = wv^t_{id} + c_1 r_1 (p^t_{id} - x^t_{id}) + c_2 r_2 (p^t_{gd} - x^t_{id}) \tag{3}$$

$$x^{(t+1)}_{id} = x^t_{id} + v^{(t+1)}_{id} \tag{4}$$

Para resolver o problema de estagnação em mínimos locais, Esmin *et al*. [15] propuseram um método chamado *Hybrid Swarm Optimizer with Mutation* (HPSOM) o qual consiste em adicionar o processo de mutação ao PSO padrão que, a cada iteração, existe uma probabilidade das partículas sofrerem mutação. Yu *et al*. [16] apresentaram um novo PSO híbrido (HPSO) combinando *Space Transformation Search* (STS) com um novo modelo de velocidade modificada.

Já Mirjalili e Hashim [17] hidridizaram PSO com o *Gravitational Search Algorithm* (GSA). A ideia é integrar a capacidade de comportamento social do PSO com a habilidade de busca local do GSA. No trabalho de Li *et al*. [18], uma hibridização combinando PSO com um método modificado de Broyden-Fletcher-Goldfarb-Shanno (BFGS) é proposta. O método BFGS modificado é integrado no contexto do PSO para melhorar a habilidade de busca local das partículas.

Na área de sistemas de recomendação, muitos trabalhos propuseram o uso de serendipidade a fim de aumentar a

diversidade dos resultados. Entre eles, Murakami *et al.* [4] apresentaram duas métricas para avaliar a serendipidade das listas de recomendação. Com base nessas duas métricas, Ge *et al.* [5] propuseram uma nova medida para o cálculo da serendipidade. Oku e Hattori [6] apresentaram um método baseado em fusão para gerar serendipidade. Os autores se inspiraram na ideia que a mistura de características de elementos distintos (por exemplo, a mistura de cores) produzem novos elementos.

Na literatura, não foram encontradas variantes do PSO que utilizassem o conceito de serendipidade, principalmente, com o propósito de gerar diversidade para evitar a convergência prematura.

IV. Otimização Baseada em Serendipidade

Este trabalho apresenta o algoritmo *Serendipity-Based Particle Swarm Optimization* (SBPSO) para resolver o problema da convergência prematura e evitar que partículas do enxame fiquem presas em ótimos locais. Para isso, é inserido o conceito de serendipidade no algoritmo PSO.

A novidade do trabalho consiste em uma estratégia que utiliza *serendipidade guiada* para inspecionar regiões do espaço de busca que não foram exploradas pelo PSO em uma determinada iteração. O termo "guiada" é usado para indicar que a geração de diversidade das soluções ótimas não é feita de forma aleatória, mas sim por meio da procura e da detecção de padrões potenciais de vales (ou de picos, caso o problema seja de maximização). O algoritmo é baseado nos eventos Ev_{t+1} e Ev_{t+2} do modelo perceptivo de serendipidade [9]. O algoritmo detecta "padrões potenciais" durante o evento Ev_{t+1}, enquanto que no evento Ev_{t+2} ele escolhe a ação apropriada para preservar e ampliar os padrões potenciais já detectados.

A utilização de serendipidade objetiva a identificação de soluções melhores que a partícula *gBest*, mas que não foram localizadas como tal. Essas soluções podem ser representadas como elementos do conjunto $ACASO$, definido na Seção II. A tentativa de detectar regiões do espaço de busca que apresentem padrões potenciais de vales que sejam melhores que a partícula *gBest* é realizada por meio de inspeções em suas adjacências.

A estratégia proposta é iniciada assim que o *gBest* da iteração j é obtido, conforme Algoritmo 1. A partir da equação (5), o algoritmo define m vizinhos para a partícula *gBest*, denominados de *Unidade Auxiliar de Prospecção* (UAP):

$$uap_{mj} = e^*_{ij} + rand() \times v_i, \tag{5}$$

onde e^*_{ij} é um elemento do conjunto E, apresentado na Seção II, que representa a partícula i cujo valor de *fitness* é o melhor da iteração j; $rand()$ é uma função que gera um número aleatório uniformemente distribuído no intervalo [-1, 1] e; v_i é o i-ésimo elemento do vetor V, usado para definir a vizinhança de e^*_{ij}.

O grupo formado pela partícula e^*_{ij} e suas m UAPs é chamado de *Unidade de Prospecção* (UP). A escolha de um valor que será atribuído ao parâmetro m é essencial para um bom desempenho do algoritmo uma vez que o tempo de execução do algoritmo pode ser comprometido se um valor alto for atribuído a esse parâmetro.

```
Inicializa partículas
while critério de parada não atingido do
    obtém pBest e gBest;
    //****** Uso da serendipidade guiada ******
    define m UAPs usando equação (5);
    foreach UAP_m do
        calcula fitness(UAP_m);
        if fitness(UAP_m) < fitness(gBest) then
            UAP_m se torna um elemento de SRD;
        end
    end
    gBest ← melhor elemento do conjunto SRD;
    //*******************************
    calcula velocidade das partículas usando (3);
    calcula a posição das partículas usando (4);
end
```

Algoritmo 1: Pseudo-código do SBPSO

Depois que a UP é formada, os valores de *fitness* de cada UAP são calculados. Se na iteração j houver UAPs cujos valores de *fitness* são melhores que o da partícula representada por e^*_{ij}, então elas são definidas como elementos $srd_{ij} \in SRD$. O algoritmo então identifica srd^*_{ij}, que é o elemento cujo valor de *fitness* é o melhor entre todos os elementos do conjunto SRD, e define-o como sendo o novo elemento e^*_{ij}. Em seguida, o fluxo normal de execução do PSO é continuado com os cálculos das velocidades e das respectivas posições das partículas.

V. Experimentos Computacionais e Resultados

O SBPSO e o PSO padrão foram implementados na linguagem de programação Scilab. Os experimentos computacionais foram executados em um computador que utiliza processador Intel Core i5 com 2,4 GHz de frequência, 6 Gb de memória RAM e sistema operacional Windows 8.1 Pro, 64 bits. Nos experimentos, não foram utilizadas técnicas de multiprocessamento.

A fim de avaliar o desempenho do algoritmo proposto, quatro funções de *benchmark* foram escolhidas. Todas elas são aplicadas a problemas de minimização. As funções Schwefel 1.2 e Rosenbrock são unimodais, ao passo que Griewank e Rastrigin são multimodais com vários mínimos locais. Essas funções tem sido aplicadas em vários estudos de *Particle Swarm Optimization* [15][16][17][18] e são descritas abaixo:

- Schwefel 1.2 – é uma função unimodal cuja solução ótima encontra-se em uma região do espaço bastante restrita:
  $f_1(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$
- Rosenbrock – é uma função com interação entre algumas variáveis, cujo mínimo global encontra-se em

um vale parabólico. No entanto, segundo Picheny *et al.* [19], apesar de ser fácil encontrar o vale parabólico a convergência para o mínimo é difícil:
$f_2(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$

- Griewank – é uma função com interações entre as variáveis. A função apresenta muitos mínimos locais generalizadas distribuídos regularmente:
$f_3(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod cos(\frac{x_i}{\sqrt{i}}) + 1$
- Rastrigin – é uma função com vários mínimos locais cujas localizações são regularmente distribuídas:
$f_4(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$

Para cada uma das quatro funções, são apresentadas as fronteiras do espaço de busca e da região de inicialização das posições das partículas, conforme Tabela I.

Tabela I
FRONTEIRAS DAS FUNÇÕES DE BECHMARK

Função	Espaço de Busca	Inicialização das Posições
f_1	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$
f_2	$-30 \leq x_i \leq 30$	$15 \leq x_i \leq 30$
f_3	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$
f_4	$-5,12 \leq x_i \leq 5,12$	$2,56 \leq x_i \leq 5,12$

Para garantir que o algoritmo encontre boas soluções sem comprometer o seu tempo de execução, para cada uma das funções de *benchmark* estudadas foram realizados vários experimentos com o objetivo de encontrar um valor ideal para m. A partir de um conjunto de configurações pré-definidas, a melhor média e o desvio padrão foram os critérios definidos para avaliar o melhor valor de m. O conjunto de configurações consiste na função de *benchmark*, no tamanho da população, no número de iterações e no próprio valor de m. Após a realização de vários experimentos variando esssas configurações, m = 6 foi o valor que apresentou melhores resultados nos critérios melhor média e desvio padrão. Quando valores superiores a 6 foram atribuídos a m, não foram encontradas melhorias significativas no valor da melhor média. Além disso, o tempo médio de execução do algoritmo passou a ser comprometido.

As figuras 1, 2, 3 e 4 apresentam o comportamento das partículas *gBest* do PSO e do SBPSO para as funções Schwefel 1.2 (f_1), Rosenbrock (f_2), Rastrigin (f_3) e Griewank (f_4), respectivamente, durante 100 iterações. As figuras mostram que o PSO entra em estagnação antes do SBPSO o qual continua em busca das melhores soluções.

Os experimentos compararam o PSO com o SBPSO por meio de diversas configurações variando o tamanho da população, as dimensões das funções e a quantidade de iterações. Todos os experimentos foram executados 100 vezes. Os parâmetros dos dois algoritmos foram definidos com os seguintes valores: c_1 = c_2 = 2.0, peso de inércia w com um decrescimento linear iniciando em 0,9 e finalizando em 0,4 e v_{max} = 0,4.

Os tamanhos definidos para as populações são 20, 40 e 80, os números máximos de iterações são 1.000 e 1.500,

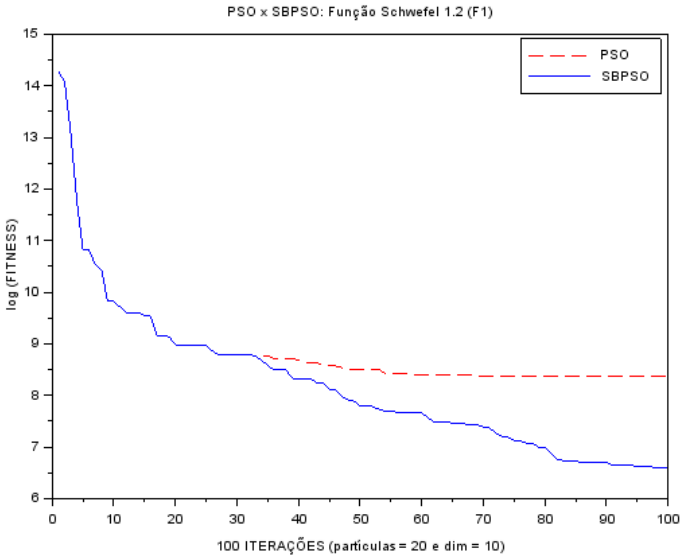

Figura 1. Na função f_1, o comportamento do PSO é estagnado perto da iteração 55, enquanto que no SBPSO a estagnação ocorre próximo à iteração 100.

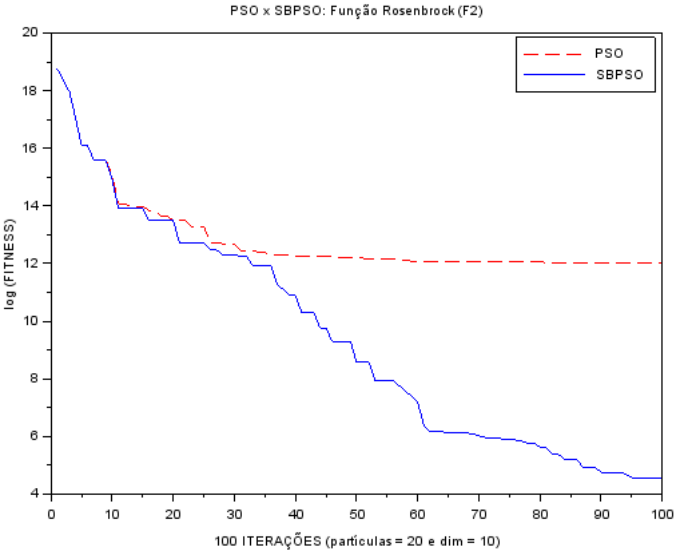

Figura 2. Na função f_2, o PSO entra em estagnação perto da iteração 80, enquanto o SBPSO nas proximidades da iteração 95.

correspondendo às dimensões 10 e 20 para cada uma das quatro funções, respectivamente.

A primeira função avaliada é a Schwefel 1.2 que possui apenas uma solução ótima e normalmente é usada para testar a habilidade do algoritmo na realização de buscas locais. Na função Rosenbrock, o mínimo global se encontra em um vale parabólico, em uma região de difícil convergência. Assim, ela é usada normalmente para testar a habilidade do algoritmo para realizar buscas locais e globais. Já as funções Rastrigin e Griewank são utilizadas para verificar a habilidade do algoritmo em buscas globais.

As tabelas II, III, IV e V comparam os algoritmos PSO

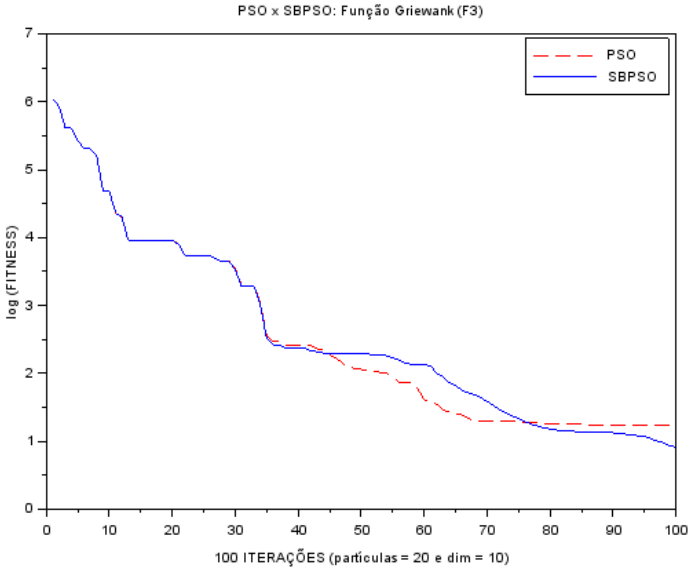

Figura 3. Na função f_3, o comportamento do PSO é estagnado perto da iteração 70, enquanto que o SBPSO, na iteração 100, ainda continua ativo.

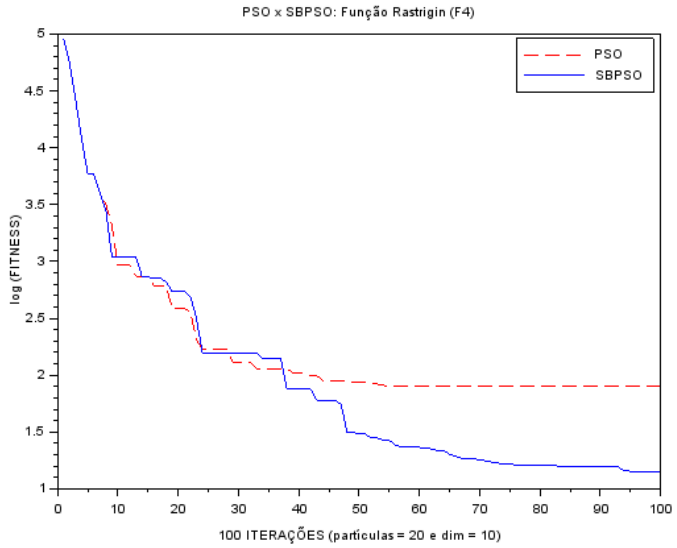

Figura 4. Na função f_4, o PSO entra em estagnação perto da iteração 55, enquanto o SBPSO nas proximidades da iteração 95.

Tabela II
VALORES DE FITNESS PARA A FUNÇÃO f_1

Part.	Dim.	Iter.	PSO		SBPSO	
			Melhor Média	Desvio Padrão	Melhor Média	Desvio Padrão
20	10	1.000	2,65E-04	9,27E-02	0	6,00E-06
	20	1.500	0	6,47E-02	0	0
40	10	1.000	0	9,04E-02	0	8,91E-04
	20	1.500	2,79E-03	7,36E-02	0	0
80	10	1.000	0	9,99E-02	0	3,25E-04
	20	1.500	2,37E-03	7,26E-02	0	0

Tabela III
VALORES DE FITNESS PARA A FUNÇÃO f_2

Part.	Dim.	Iter.	PSO		SBPSO	
			Melhor Média	Desvio Padrão	Melhor Média	Desvio Padrão
20	10	1.000	0	9,69E-02	0	0
	20	1.500	9,00E-06	9,33E-02	0	6,29E-04
40	10	1.000	5,80E-05	9,05E-02	0	3,51E-03
	20	1.500	9,40E-05	8,95E-02	0	5,83E-04
80	10	1.000	0	9,48E-02	0	2,54E-03
	20	1.500	1,40E-04	9,08E-02	0	1,88E-03

Tabela IV
VALORES DE FITNESS PARA A FUNÇÃO f_3

Part.	Dim.	Iter.	PSO		SBPSO	
			Melhor Média	Desvio Padrão	Melhor Média	Desvio Padrão
20	10	1.000	2,33E-02	3,61E-02	0	2,19E-02
	20	1.500	0	5,03E-02	0	4,39E-02
40	10	1.000	9,82E-03	4,23E-02	0	1,67E-02
	20	1.500	0	5,95E-02	0	3,77E-02
80	10	1.000	1,11E-02	3,68E-02	0	2,23E-02
	20	1.500	0	5,78E-02	0	4,07E-02

Tabela V
VALORES DE FITNESS PARA A FUNÇÃO f_4

Part.	Dim.	Iter.	PSO		SBPSO	
			Melhor Média	Desvio Padrão	Melhor Média	Desvio Padrão
20	10	1.000	0	5,39E-02	0	4,75E-02
	20	1.500	1,09E-02	5,48E-02	1,30E-02	2,00E-06
40	10	1.000	0	7,18E-02	0	4,68E-02
	20	1.500	0	6,96E-02	0	6,52E-03
80	10	1.000	0	7,68E-02	0	5,96E-02
	20	1.500	0	8,92E-02	0	3,00E-06

e SBPSO, quando avaliados nas funções f_1, f_2, f_3 e f_4, respectivamente. Para cada uma delas, as tabelas mostram a quantidade utilizada de partículas nos experimentos, a dimensão da função, a quantidade de iterações, a melhor média do valor de *fitness* e o desvio padrão dos dois algoritmos quando executados 100 vezes.

O Teste de Wilcoxon [20], com nível de significância de 0,05 para o *p-value*, foi aplicado a fim de comparar os soluções encontradas pelos algoritmos quando avaliados pelas funções Schwefel 1.2, Rosenbrock, Rastrigin e Griewank, conforme Tabela VI.

Wilcoxon é um teste estatístico não-paramétrico utilizado para comparar duas amostras independentes. A hipótese nula H_0 afirma que as duas amostras vêm da mesma população, enquanto a hipótese alternativa H_1 afirma que uma delas possui valores maiores que a outra. Quando o *p-value* é menor que o nível de significância, decide-se rejeitar H_0, isto é, há diferença significativa entre as amostras.

A Tabela VI apresenta os *p-values* resultantes do teste de Wilcoxon quando são comparadas as soluções encontradas pelos algoritmos PSO e SBPSO, em cada uma das configurações, nas quatro funções avaliadas. O teste foi usado para verificar se os resultados do SBPSO foram estatisticamente diferentes dos encontrados pelo PSO.

Na Tabela II, com configuração 80 partículas, 10 dimen-

Tabela VI
AVALIAÇÃO DO TESTE DE WILCOXON NAS 4 FUNÇÕES

Part.	Dim.	Iter.	p-Value			
			f_1	f_2	f_3	f_4
20	10	1.000	1,57E-30	2,68E-26	2,33E-26	0,04
	20	1.500	1,57E-26	5,06E-27	9,30E-03	9,12E-09
40	10	1.000	8,84E-22	2,13E-19	4,27E-25	**0,28***
	20	1.500	1,57E-30	2,99E-29	5,80E-03	5,96E-13
80	10	1.000	**0,59***	5,31E-17	1,80E-23	**0,78***
	20	1.500	1,57E-30	3,04E-22	6,93E-04	**0,82***

sões e 1.000 iterações, observa-se que o SBPSO apresentou melhores resultados que o PSO. A mesma coisa é observada na Tabela V, nas seguintes configurações: a) 40 partículas, 10 dimensões e 1.000 iterações; b) 80 partículas, 10 dimensões e 1.000 iterações e; c) 80 partículas, 20 dimensões e 1.500 iterações. Apesar de as soluções encontradas pelo SBPSO terem superado as do PSO nas quatro configurações citadas acima, a hipótese nula H_0 não foi rejeitada (valores marcados com * e em negrito, na Tabela VI) o que, estatisticamente, significa que os resultados encontrados pelos dois algoritmos não vêm de populações distintas para essas configurações. No entanto, nas demais, o valor *p-value* da comparação foi inferior ao do nível de significância 0,05 estabelecido no teste e, assim, a hipótese H_0 foi rejeitada garantindo que não foram encontradas igualdades entre as soluções apresentadas pelos dois algoritmos.

VI. CONCLUSÕES

Este trabalho implementou uma variante do PSO que consiste em uma estratégia para geração de serendipidade no contexto de métodos meta-heurísticos. O novo algoritmo é baseado em um modelo perceptivo de serendipidade cuja essência é fundamentada no Princípio de Pasteur, usado na área de sistemas de recomendação para indução de serendipidade nos resultados listados.

A abordagem proposta pode ser generalizada com relação ao número usado de variáveis independentes em cada função objetivo e com relação ao número de funções objetivos consideradas. No primeiro caso, a função de adequação acompanha a dimensionalidade do problema, ou seja, cada variável independente implica em uma dimensão a mais no problema. Isto vale também para a função que analisa a vizinhança de cada ponto considerado. No segundo caso, a função de adequação final pode ser idealizada como sendo uma função composta da soma ponderada das funções de adequação considerada.

A proposta foi comparada com o PSO por meio de quatro funções de *benchmark*, das quais duas são unimodais e duas multimodais. Os experimentos mostraram que o SBPSO apresentou a capacidade de escapar de ótimos locais e de superar o PSO na qualidade das soluções, embora o PSO tenha apresentado um melhor desempenho quando o tempo de execução foi avaliado.

Para confirmar os resultados apresentados, o teste de Wilcoxon foi utilizado e, na maioria dos experimentos, as análises indicam que há uma significância estatística entre os resultados obtidos pelo SBPSO e pelo PSO, conforme Tabela VI, mostrada na seção anterior.

Os resultados obtidos se mostraram promissores para o contexto de métodos meta-heurísticos, visto que é uma alternativa simples que objetiva oferecer a esses métodos estratégias para que eles se beneficiem da diversidade de soluções ótimas por meio do uso de serendipidade.

REFERÊNCIAS

[1] Smith, S. S., Using multiple genetic operators to reduce premature convergence in genetic assembly planning, Computers in Industry, Vol. 54, Issue 1, 2004, pp 35–49.

[2] Van Andel, P., Anatomy of the Unsought Finding. Serendipity: Origin, History, Domains, Traditions, Appearances, Patterns and Programmability, The British Journal for the Philosophy of Science Vol. 45, No. 2, 1994, pp. 631-648.

[3] Kuhn, T. S., The Structure of Scientific Revolutions, 3 ed., The University of Chicago Press, 1996.

[4] Murakami, T., Mori, K., and Orihara, R., Metrics for evaluating the serendipity of recommendation lists. New Frontiers in Artificial Intelligence, pp. 40–46, 2008.

[5] Ge, M, Delgado-Battenfeld, C., and Jannach, D, "Beyond accuracy: evaluating recommender systems by coverage and serendipity", Proc. 4th ACM conference on Recommender Systems, Barcelona, Spain, 2010.

[6] Oku, K., and Hattori, F., "Fusion-based recommender system for improving serendipity." Proc. 5th ACM International Conference on Recommender Systems, Chicago, Illinois, USA, 2011.

[7] Iaquinta, L. Gemmis, M. Lops, P. Semeraro, and G. Molino, P. , Can a Recommender System Induce Serendipitous Encounters?. In-Tech, Vienna, Austria, 2010.

[8] Adamopoulos, P., and Alexander Tuzhilin. "On unexpectedness in recommender systems: Or how to better expect the unexpected.", Proc. 5th ACM International Conference on Recommender Systems, Chicago, Illinois, USA, 2011.

[9] Lawley, J, T., Penny, Maximising Serendipity: The art of recognising and fostering unexpected potential – A Systemic Approach to Change. Proc. Neuro Linguistic Psychotherapy and Counselling Association, Kettering, Northamptonshire, United Kingdom, 2013.

[10] Campos, J. and Figueiredo, A. Dias de, "Programming for Serendipity", Proc. Symposium on Chance Discovery, The Discovery and Management of Chance Events, North Falmouth, Massachusetts, 2002, pp. 48–60.

[11] Quéau, P., Éloge de la Simulation, Seissel, Haute-Savoie, Champ Valon, 1986.

[12] Catellin, S., "Sérendipité: Du conte au concept", Seuil, Paris, Île-de-France, 2014.

[13] Zhang, M., Zhang, B. and Zheng, Y., "Bio-Inspired Meta-Heuristics for Emergency Transportation Problems", Algorithms, Vol. 7, No. 1, 2014, pp.15–31.

[14] Kennedy, J., and Eberhart, R. C., "Particle swarm optimization", Proc. IEEE International Conference on Neural Networks (ICNN), Vol.4, Perth, Australia, 1995, pp.1942–1948.

[15] Esmin, A. A. A., Torres, G. L., and Alvarenga, G. B., "Hybrid Evolutionary Algorithm Based on PSO and GA Mutation". Proc. 6th International Conference on Hybrid Intelligent, Wroclaw, Poland, 2006.

[16] Yu, S., Wu, Z., Wang, H., Chien, Z., and Zhong H., "A Hybrid Particle Swarm Optimization Algorithm Based on Space Transformation Search and a Modified Velocity Model", Int. Journal of Numerical Analysis and Modeling, Vol. 9, Dec-2012, pp. 371–377.

[17] Mirjalili, S., and Hashim S. Z. M., "A New Hybrid PSOGSA Algorithm for Function Optimization." Proc. Int. Conf. on Computer and Information Application, Tianjin, China, 2010.

[18] Li, S., Tan, M., Tsang, I. W., and Kwok, J. T., "A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions", IEEE Transaction on Systems, Man, and Cybernetics, Vol. 41(4), Aug-2012, pp. 1003–1014.

[19] Picheny, V., Wagner, T., and Ginsbourger, D., "A benchmark of kriging-based infill criteria for noisy optimization". Structural and Multidisciplinary Optimization, Vol.48, 2013, pp.607–628.

[20] Demsar, J., "Statistial Comparisons of Classifiers over Multiple Data Sets", Journal of Mahine Learning Research, Vol. 7, No. 1, Dec-2006, pp. 1–30.