A Serendipity-Based Approach to Enhance Particle Swarm Optimization Using Scout Particles

F. A. P. Paiva, J. A. F. Costa and C. R. M. Silva

***Abstract*— In metaheuristic algorithms, such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), it is common to deal with a problem known as premature convergence. It happens when a swarm loses diversity and starts converging too early towards a suboptimal solution for an optimization problem. There have been many approaches to this problem along to the latest two decades, but it is a understanding that the problem is still open. This work proposes a new approach based on a concept normally applied in the Recommender Systems context (serendipity-based approach). The paper presents a formalization for the concepts of serendipity and premature convergence, as well a Serendipity-Based PSO (SBPSO) algorithm prototype which implements the concept of serendipity by means of two dimensions: chance and sagacity. The algorithm was compared with the traditional PSO and some PSO variants. The results were successful and showed that SBPSO outperformed the traditional PSO. The experiments also compared SBPSO with some studies in the literature, considering a set of hard functions (such as Rosenbrock, HappyCat, etc) and a fixed number of particles and varying the problem dimensionality and the number of iterations. In all experiments, SBPSO also showed a better convergence behavior, outperforming the traditional PSO and some variants available in the literature regarding the solution quality, the ability to find global optimum, the solutions stability and the ability to restart the movement of the swarm in case of stagnation has been detected.**

***Keywords*— PSO, Swarm Intelligence, Serendipity, Scout Particles, Premature Convergence.**

I. INTRODUÇÃO

EM muitos problemas de engenharia, é comum o estudo de um tipo de processo que se comporta, via de regra, como um sistema dinâmico que pode ser modelado por um conjunto de equações que considera as mesmas variáveis e está sujeito a uma determinada evolução ao longo do tempo.

Um conjunto de técnicas que vem sendo aplicado com sucesso a esse tipo de problema é o que envolve o que conhecemos como técnicas de computação bio-inspirada [1]. Estas técnicas têm sido aplicadas a problemas do mundo real em diferentes domínios tais como: Telecomunicações [2][3], Sistemas de Potência [4][5], Sistemas de Predição [6][7], Sistemas de Detecção de Intrusão [8][9], entre outros.

No contexto da computação bio-inspirada, a Inteligência de Enxames consiste em um conjunto de meta-heurísticas baseado em alguns seres vivos cujos comportamentos emergentes podem resultar em uma capacidade de resolver problemas complexos [10]. O algoritmo *Particle Swarm Optimization* (PSO) [11], por exemplo, tem sido amplamente utilizado no contexto de problemas de otimização global para implementar técnicas que buscam inspiração no comportamento social dos pássaros e dos peixes para resolver problemas de otimização. Nestas técnicas, os indivíduos da população são representados por partículas do espaço de busca que se movimentam no espaço para convergir para uma posição de uma partícula ótima. O PSO possui algumas vantagens tais como facilidade de implementação e rápida convergência, porém muitas vezes enfrenta um problema em que as suas partículas ficam "presas" em ótimos locais. Esse problema é muitas vezes chamado de convergência prematura.

Em outra área de estudo, conhecida como Sistemas de Recomendação, há um conceito que pode ser usado como uma estratégia para retardar a convergência prematura. Esse conceito é chamado de *serendipidade*. De maneira geral, serendipidade é um termo que se refere a resultados afortunados que, aparentemente, ocorreram por acaso.

Os Sistemas de Recomendação surgiram como uma abordagem para resolver o problema da sobrecarga de informação. Eles são aplicações especiais do cotidiano que fornecem recomendações personalizadas pelas quais os usuários podem se interessar [12][13]. Nos últimos anos, o foco das pesquisas em sistemas de recomendação tem sido a precisão das recomendações [14]–[16]. No entanto, alta precisão pode causar um problema específico conhecido como *over-specialization*. Embora isso ocorra quando o sistema se refere a apenas itens relacionados ao perfil do usuário, podem existir itens mais apropriados. Serendipidade é uma estratégia que pode ser utilizada para resolver a *over-specialization*.

De acordo com [17], 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, deixa de sugerir itens que podem ser mais adequados. De forma similar, quando um algoritmo meta-heurístico converge para um ótimo local sem considerar outras soluções mais adequadas que a melhor encontrada, é possível observar a correlação entre *over-specialization* e convergência prematura. Algumas abordagens na literatura usam partículas escoteiras com o objetivo de melhorar o desempenho do algoritmo PSO tradicional [18][19]. Estas abordagens são essencialmente estocásticas, mas elas podem se beneficiar de combinações com outras técnicas como, por exemplo, a serendipidade.

Este trabalho propõe uma variante do PSO chamada *Serendipity-Based Particle Swarm Optimization* (SBPSO) que combina o conceito de serendipidade com o uso de partículas escoteiras para melhorar a exploração do algoritmo no espaço de busca.

F. A. P. Paiva, Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), Parnamirim, RN, Brasil, fabio.procopio@ifrn.edu.br
J. A. F. Costa, Universidade Federal do Rio Grande do Norte (UFRN), Natal, RN, Brasil, jafcosta@gmail.com
C. R. M. Silva, Universidade Federal do Rio Grande do Norte (UFRN), Natal, RN, Brasil, claudio.rmsilva@gmail.com
Corresponding author: Fábio Augusto Procópio Paiva

II. BACKGROUND

O Dicionário Cambridge define serendipidade como "*O fato de encontrar coisas interessantes ou importantes por acaso*". No entanto, ao contrário de muitas definições tradicionais que consideram o termo apenas como sinônimo de "acaso", serendipidade está mais perto de uma combinação de acaso e de sagacidade [20].

Em [21], o conceito de serendipidade é apresentado de uma maneira formal 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. Esses eventos são associados a diferentes tipos de serendipidade: a) pseudo-serendipidade; b) serendipidade real; c) serendipidade sem uma metáfora de inspiração e; d) serendipidade com o conhecimento incorreto.

No contexto de algoritmos de meta-heurística, para um determinado espaço de busca, diz-se que o conjunto E é formado por elementos que representam as soluções encontradas em uma determinada iteração de um espaço de busca. Assim, 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 de *fitness* é o melhor valor encontrado na iteração j é representada pelo elemento $e_{ij}^* \in E$.

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, esse elemento é considerado 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 $acaso_{ij}$ é um elemento *serendípeto*. O conjunto *SRD* é formado por elementos serendípetos que representam soluções ocasionais cujos valores de *fitness* são melhores que o valor de *fitness* do elemento e_{ij}^*, conforme Equação (2):

$$SRD = \{srd_{ij} \mid fitness(acaso_{ij}) < fitness(e_{ij}^*)\} \tag{2}$$

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*.

Nas duas subseções seguintes, são apreentados três métodos para detectar convergência prematura e também alguns trabalhos disponíveis na literatura.

A. Métodos para Detectar Convergência Prematura

O algoritmo PSO é uma técnica capaz de encontrar bons resultados em um tempo de convergência rápido, porém o algoritmo corre o risco de convergir prematuramente. Diz-se que o enxame convergiu prematuramente quando a solução proposta aproxima-se de um ótimo local ao invés do ótimo global do problema que está sendo otimizado.

A convergência prematura ocorre devido à diminuição da diversidade no espaço de busca o que leva o enxame a um estado de estagnação. Depois que a convergência prematura é iniciada, as partículas continuam o processo de convergência próximas umas das outras em uma região extremamente estreita no espaço de busca [22].

A fim de evitar a estagnação do enxame, em [23] são apresentados três métodos para detectar a convergência prematura:

1) *Maximum Swarm Radius* – consiste em avaliar a maior distância euclidiana entre uma partícula do enxame e a partícula *gBest*. A estagnação ocorre quando essa distância euclidiana é inferior a um *threshold* chamado δ_{stag}. O método é formalmente apresentado pela Equação (3):

$$\varphi_j = \frac{\max_{i \in \{1,\dots,s\}} \|e_{ij} - e_{ij}^*\|}{|\mu_{max} - \mu_{min}|}, \tag{3}$$

onde $\|e_{ij} - e_{ij}^*\|$, é a norma euclidiana entre uma partícula *i* do enxame e a partícula *gBest* na iteração *j*. μ_{max} e μ_{min} representam as extremidades máxima e mínima de cada dimensão da partícula *i*, respectivamente.

2) *Cluster Analysis* – consiste em avaliar o percentual do número de partículas que fazem parte de um *cluster C*. Uma partícula pertence a *C* se a distância entre ela e a partícula *gBest* é menor que um *threshold* δ_{min}. Então, quando um percentual de partículas em *C* atingir um *threshold* δ_{max}, assume-se que a convergência ocorreu;

3) *Objective Function Slope* – considera a posição das partículas no espaço de busca. Baseia-se exclusivamente na taxa de mudança da função objetivo e o seu valor normalizado é obtido por meio da Equação (4):

$$f_{ratio} = \frac{f(\hat{y}_j) - f(\hat{y}_{j-1})}{f(\hat{y}_j)} \tag{4}$$

onde $f(\hat{y}_j)$ denota o valor de *fitness* da função objetivo na iteração j e $f(\hat{y}_{j-1})$ na iteração $j-1$. Se o valor de f_{ratio} é inferior a um *threshold* pré-definido, um contador é incrementado. Quando o valor do contador atinge o *threshold* δ_{conv} é assumido que o enxame convergiu.

B. Outros Trabalhos Relacionados

Serendipidade é definida como o ato de encontrar algo bom ou útil em uma determinada ocasião apesar de, não necessariamente, se procurar por ele. O projeto de

mecanismos de recomendação que utiliza serendipidade é novo e também ainda é um problema de pesquisa em aberto.

Quando o conceito de serendipidade é adaptado ao contexto de algoritmos meta-heurísticos e suas aplicações de otimização, "algo bom ou útil" pode ser visto como uma solução candidata a substituir, após um certo número de iterações, a melhor solução atual.

O algoritmo PSO ganhou muita atenção das comunidades de pesquisa para resolver problemas de otimização do mundo real devido à sua facilidade de implementação e ao seu bom desempenho. PSO é uma abordagem estocástica usada para modelar o comportamento social dos pássaros. Neste tipo de algoritmo, a solução é representada por uma partícula que "voa" ao longo do espaço de busca à procura de uma solução ótima, durante um determinado número de iterações.

O movimento das partículas é baseado em duas informações: $gBest$ e $pBest$. O $gBest$ influencia o movimento da partícula para a direção da melhor posição encontrada pelo enxame, enquanto o *pBest* movimenta a partícula para a melhor posição encontrada por ela [24]. Após encontrar *gBest* e *pBest*, cada partícula atualiza sua velocidade e sua posição por meio das equações (5) e (6), respectivamente:

$$v_{id}^{(t+1)} = w v_{id}^{t} + c_1 r_1 (p_{id}^{t} - x_{id}^{t}) + c_2 r_2 (p_{gd}^{t} - x_{id}^{t}) \quad (5)$$

$$x_{id}^{(t+1)} = x_{id}^{t} + v_{id}^{(t+1)} \quad (6)$$

Na Equação (5), o termo w é chamado fator inercial e representa a velocidade inercial da partícula. v_{id}^{t} e x_{id}^{t} são a velocidade e a posição da partícula *i* no instante *t*, respectivamente. p_{id}^{t} e p_{gd}^{t} são o melhor valor de *fitness* encontrado pela partícula *i* e o melhor valor de *fitness* encontrado entre todas as partículas do enxame até o instante *t*, respectivamente. c_1 é um coeficiente que contribui com a autoexploração da partícula, enquanto c_2 contribui com o movimento da partícula no sentido do deslocamento global do enxame no espaço de busca. r_1 e r_2 são valores randômicos uniformemente distribuídos no intervalo [0, 1].

Diferentes variantes do algoritmo PSO tradicional surgiram para melhorar a performance e o comportamento de convergência do algoritmo padrão. Em [25], é apresentada uma variante do PSO que se baseia na teoria quântica, chamada *Quantum-Behaved PSO* (QPSO). Para avaliar a performance do algoritmo, os experimentos foram realizados com algumas funções de *benchmark* e mostraram que a proposta apresentou resultados melhores que o PSO.

Em [26], um QPSO melhorado chamado *Weighted QPSO* (WQPSO) é proposto. O método utiliza média ponderada da melhor posição, de acordo com os valores de *fitness* das partículas. A proposta foi testada em funções de *benchmark* e mostrou resultados satisfatórios em relação ao PSO tradicional e ao QPSO.

Em [27], é proposta uma variante do PSO que combina operadores de mutação e Teoria *Wavelet* aumentando, assim, o espaço de exploração. Algumas funções de *benchmark* e três aplicações industriais foram utilizados para avaliar o desempenho e a aplicabilidade do método proposto.

Em [28], é proposto um método que combina o processo de mutação, comumente usado em Algoritmos Genéticos, com o algoritmo PSO tradicional. O algoritmo é chamado *Hybrid PSO with Genetic Mutation* (HPSOM) e suas partículas tem uma probabilidade de sofrer mutação a cada iteração. Os experimentos mostraram que o HPSOM foi bem sucedido quando avaliado em funções unimodais e multimodais.

III. Serendipity-Based Particle Swarm Optimization

Na seção anterior, serendipidade foi apresentada como uma capacidade de realizar descobertas afortunadas por acaso e de forma sagaz. Neste trabalho, o termo "acaso" deve ser entendido como uma possibilidade de algo acontecer. O termo "sagacidade", por sua vez, deve ser entendido como a qualidade de perceber um evento (ou algo relacionado). Esse conceito de serendipidade é interessante e útil porque é o principal indicativo para definir onde serendipidade pode ser aplicada em um algoritmo PSO tradicional, além de qual tipo de sagacidade deve ser usado para melhorar o comportamento do PSO por meio de decisões inspiradas em serendipidade.

Há três abordagens naturais para implementar essa melhoria. A primeira é baseada em modificações nos pontos aleatórios de decisões disponíveis no algoritmo PSO tradicional. A segunda é baseada na inclusão de um método complementar para ser usado em conjunto com o PSO tradicional para implementar um algoritmo do tipo PSO com serendipidade. A terceira é uma mistura dos outros dois. Todas elas consideram "descobertas afortunadas" como pontos no espaço de busca que são úteis para um procedimento de otimização, uma vez que eles são melhores do que a melhor solução disponível encontrada até a iteração atual.

Este trabalho adota uma variante da terceira abordagem. Ele propõe uma estratégia usada em [17] para gerar serendipidade baseada em um modelo perceptivo [29] que combina duas estratégias usadas no domínio de sistemas de recomendação [30]: *blind luck* e Princípio de Pasteur. A estratégia *blind luck* é implementada por meio do uso de partículas escoteiras. Elas implementam a dimensão *acaso* e complementam o espaço exploratório do PSO tradicional gerando diversidade adicional. O princípio de Pasteur é usado para reconhecer potenciais e usar percepções para aproveitar o momento. O princípio usa partículas escoteiras a fim de inspecionar regiões no espaço de busca inexploradas pelo enxame, de acordo com as equações 5 e 6. Essa estratégia é a implementação do conceito de sagacidade ou, em outras palavras, o conceito de "mente preparada".

O novo algoritmo implementado é chamado *Serendipity-Based Particle Swarm Optimization* (SBPSO). O algoritmo procura por vales (ou picos) em torno dos pontos ótimos escolhidos em cada iteração. As partículas escoteiras retardam o tempo de convergência gerando diversidade adicional para o algoritmo de otimização.

O comportamento que as partículas escoteiras implementam para explorar o espaço de busca difere do

comportamento típico do enxame. Elas podem ser usadas para aumentar a capacidade de exploração do algoritmo. É importante assegurar que a inclusão de partículas escoteiras não comprometa o comportamento habitual do enxame. No algoritmo proposto, isso é transparente porque as escoteiras são usadas para identificar as soluções melhores que a melhor partícula do enxame. A velocidade de uma partícula escoteira *k* é dada pela Equação (7):

$$V_{kd}^{(t+1)} = w \cdot V_{kd}^{t} + c_3 \cdot r_3 (X_{kd}^{t} - x_{best}^{t}), \quad (7)$$

onde c_3 é o coeficiente de diversidade e r_3 é um valor randômico uniformemente distribuído no intervalo [-1, 1]. X_{kd}^{t} e x_{best}^{t} são, respectivamente, a posição da partícula escoteira *k* e a posição da melhor partícula do enxame no instante *t*. A posição de uma partícula escoteira *k* é dada pela Equação (8):

$$X_{kd}^{(t+1)} = -x_{best}^{t} + V_{kd}^{(t+1)} \quad (8)$$

Os passos iniciais do SBPSO, mostrado no Algoritmo 1, são similares ao PSO tradicional. A velocidade e a posição de cada partícula do enxame e de cada partícula escoteira são inicializadas aleatoriamente (linhas 1–2). Em seguida, para cada partícula do enxame, o valor de *fitness* é calculado para encontrar a melhor partícula (linhas 4–10). Isso também ocorre para as partículas escoteiras (linhas 11–17). Depois que a melhor partícula do enxame e a melhor partícula escoteira são encontradas, a melhor delas será a nova partícula *gBest* (linhas 19–25).

O próximo passo é iniciar o processo de inspeção nas proximidades da partícula $gBest$ (linhas 27–33). Para isso, o algoritmo cria n Pontos Adjacentes (AP), de acordo com a Equação (9):

$$AP_n = best + \alpha \cdot R, \quad (9)$$

onde *best* é a partícula $gBest$ atual, α é uma constante real positiva, R é uma matriz $1 \times dim$ cujos elementos são valores randômicos uniformemente distribuídos no intervalo [-1, 1] e dim é a dimensão da partícula $gBest$ atual.

De acordo com a Equação (10), para cada AP_n que foi criado, define-se um vetor $\vec{d}_n$ que representa o segmento orientado $\overline{bestAP_n}$:

$$\overrightarrow{d_n} = AP_n - best \quad (10)$$

Uma vez que $best$ e AP_n definem um único vetor $\overrightarrow{d_n}$ o qual representa uma direção, então existem infinitos Pontos de Inspeção (IP) cuja direção $\overline{bestIP_n^m}$ é a mesma de $\overline{bestAP_n}$. Assim, para cada AP_n criado, escolhem-se dois IP de acordo com a Equação (11):

$$IP_n^m = best \pm \left({\|\vec{u}\|}/{\|\vec{d}_n\|} \right) \cdot \lambda \cdot \vec{d}_n, \quad (11)$$

onde IP_n^m é um ponto de inspeção m definido sobre a reta que passa por $best$ e AP_n, $\vec{u}$ é o vetor $\vec{d}_n$ que possui a menor norma e λ é uma constante real não nula.

Algoritmo 1: Pseudocódigo do SBPSO usando partículas escoteiras.

```
01: inicializa velocidade e posição do enxame
02: inicializa velocidade e posição das escoteiras
03: while critério de parada não satisfeito do
04:   for each partícula i em S do
05:     calcula o valor de fitness
06:     if fit(x_id) < fit(pBest_i) then
07:       pBest_i ← x_id
08:     end if
09:   end for
10:   best_enxame ← getMelhorParticulaEnxame()
11:  for each escoteira k em S do
12:    calcula o valor de fitness
13:    if fit(escoteira_kd) < fit(pBest_k) then
14:      pBest_k ← escoteira_kd
15:    end if
16:  end for
17:  best_escoteira ← getMelhorParticulaEscoteira()
18:
19:  if fit(best_enxame) < fit(best_escoteira) and
        fit(best_enxame) < fit(gBest) then
20:     gBest ← best_enxame
21:  else
22:     if fit(best_escoteira) < fit(best_enxame) and
           fit(best_escoteira) < fit(gBest) then
23:        gBest ← best_escoteira
24:     end if
25:  end if
26:
27:  cria n pontos adjacentes (AP) usando Eq. (9)
28:  for each AP do
29:    define n vetores diretores d⃗ usando Eq. (10)
30:    for each vetor d⃗ do
31:      cria dois IP usando Eq. (11)
32:    end for
33:  end for
34:  gBest ← getMelhorPontoInspecao()
35:
36:  for each partícula i em S do
37:    calcula velocidade usando Eq. (5)
38:    atualiza posição usando Eq. (6)
39:  end for
40:  for each escoteira k em S do
41:    calcula velocidade usando Eq. (7)
42:    atualiza posição usando Eq. (8)
43:  end for
44:
45:   if (enxame estagnou) and
        (enxame convergiu) then
46:     reinicia posição e velocidade das escoteiras
47:   end if
48: end while
```

Depois da escolha dos IP, são definidos dois critérios usados para encontrar um ponto de inspeção que possa substituir a partícula $gBest$ atual. Os critérios são: 1) IP cujo valor de *fitness* é melhor que o da partícula $gBest$ atual e 2) IP cujo valor de *fitness* é melhor que os dos demais. No entanto, caso haja mais de um IP que atenda a esses dois critérios, é realizado um sorteio para escolher um deles.

Conforme o Princípio de Pasteur, a sorte favorece a mente preparada. Neste trabalho, o conceito de "mente preparada" é implementado por meio de inspeções em torno da partícula $gBest$ e, como resultado, tem-se a implementação da dimensão *sagacidade*. A Fig. 1 mostra um exemplo da escolha aleatória de Pontos de Inspeção em um espaço 2D.

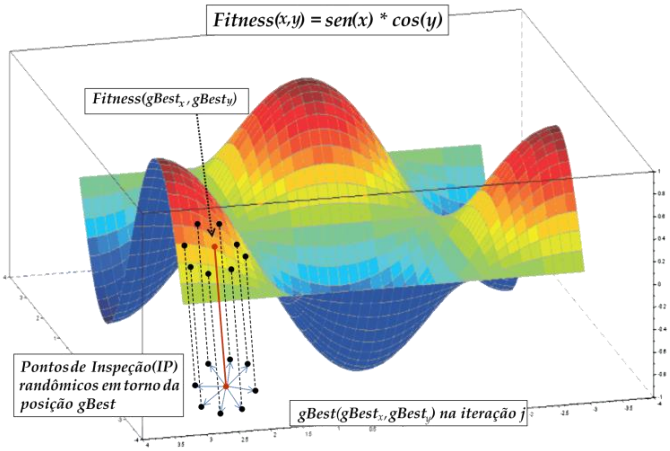

Figura 1. Exemplo da escolha aleatória de 8 Pontos Inspeção em torno da partícula *gBest*, em um espaço 2D. Isso é o resultado para cada um dos 4 Pontos Adjacentes definidos.

Embora o exemplo da Fig. 1 esteja no espaço 2D, isso também ocorre em um espaço *n*–dimensional, uma vez que qualquer espaço pode conter "cortes" implementados por retas no domínio da função de *fitness*. A direção de cada uma das retas é estocástica, uma vez que elas ligam o $gBest$ aos pontos de inspeção randômicos. Durante todas as iterações as retas apresentam uma percepção razoável do comportamento da função de *fitness*, mesmo em um espaço *n*–dimensional.

Mais uma vez, os passos do SBPSO são similares ao PSO tradicional. A velocidade das partículas do enxame é calculada e a posição é atualizada (linhas 36–39). Isso também ocorre para as partículas escoteiras (linhas 40–43).

Finalmente, o algoritmo avalia a situação do enxame em relação à estagnação e à convergência prematura (linha 45). Quando a condição abaixo é satisfeita, o SBPSO reinicia a posição e a velocidade das partículas escoteiras (linha 46):

$$(\varphi_j < \delta_{stag}) \wedge (cont \geq \delta_{conv}), \quad (12)$$

onde φ_j é o raio do enxame na iteração j e δ_{stag} é um *threshold* que avalia a estagnação do enxame, de acordo com a Equação (3); *cont* indica o número de iterações nas quais o enxame não apresenta melhorias significativas e δ_{conv} é o *threshold* que indica a convergência do enxame, de acordo com a Equação (4).

IV. Experimentos Computacionais

Na Subseção IV-A, são apresentadas algumas funções de *benchmark* usadas para avaliar o algoritmo proposto. Na Subseção IV-B, os parâmetros utilizados no algoritmo são apresentados. Na última subseção, o algoritmo é comparado com o PSO tradicional e também com algumas variantes como PSO-Scout, HWPSO, WQPSO, QPSO e HPSOM.

A. Funções de Benchmark

Foram escolhidas algumas funções que são frequentemente aplicadas em problemas de minimização para investigar a estagnação e a convergência do algoritmo [23]. Elas foram aplicadas em vários estudos de PSO como em [25]–[28] e em outros. As funções são descritas abaixo:

- Função Esfera – é caracterizada por ser simples, convexa e unimodal. Sua principal função é avaliar a taxa de convergência do processo de busca:

$$f_1(x) = \sum_{i=1}^{d} x_i^2$$

- Função Rosenbrock – posssui o mínimo global em um vale parabólico. Embora seja fácil encontrá-lo, a convergência é difícil [31]:

$$f_2(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

- Função Griewank – há vários mínimos locais regularmente distribuídos:

$$f_3(x) = \frac{1}{400}\sum_{i=1}^{d} x_i^2 - \prod \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

- Função Rastrigin – é uma função altamente multimodal, no entanto as localizações dos vários mínimos locais são regularmente distribuídos:

$$f_4(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

- Função HappyCat – é caracterizada por ser unimodal. A sua característica faz com que os algoritmos de otimização encontrem dificuldades para escapar de uma "ranhura negra" [32]:

$$f_5(x) = [(\|x\|^2 - d)^2]^{\frac{1}{8}} + \frac{1}{d}\left(\frac{1}{2}\|x\|^2 + \sum_{i=1}^{d} x_i\right) + \frac{1}{2}$$

- Função Ackley – há vários mínimos locais e a função se caracteriza por ter uma região externa quase plana e um grande orifício no centro:

$$f_6(x) = -20\, exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

Para cada uma das funções apresentadas, a Tabela I mostra as fronteiras do espaço de busca, as faixas de inicialização e o valor que representa o ótimo global.

TABELA I. CARACTERÍSTICAS DAS FUNÇÕES DE BENCHMARK

FUN	ESPAÇO DE BUSCA	FAIXA DE INICIALIZAÇÃO	ÓTIMO GLOBAL
f_1	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$	$f(x^*) = 0$
f_2	$-30 \leq x_i \leq 30$	$15 \leq x_i \leq 30$	$f(x^*) = 0$
f_3	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$	$f(x^*) = 0$
f_4	$-5.12 \leq x_i \leq 5.12$	$2.56 \leq x_i \leq 5.12$	$f(x^*) = 0$
f_5	$-2 \leq x_i \leq 2$	$1 \leq x_i \leq 2$	$f(x^*) = 0$
f_6	$-32.76 \leq x_i \leq 32.76$	$16.38 \leq x_i \leq 32.76$	$f(x^*) = 0$

B. Definição dos Parâmetros

Os valores associados aos parâmetros w, c_1 e c_2, definidos na Eq. (5), podem variar a performance do PSO tradicional e de suas variantes [33].

Para realizar uma comparação adequada, todos os parâmetros comuns ao PSO tradicional e ao SBPSO foram definidos com o mesmo valor: $c_1 = c_2 = 2.0$ e w (peso de inércia) que inicia em 0.9 e decai linearmente finalizando em 0.4. A velocidade máxima (V_{max}) de cada partícula foi definida como a metade do tamanho do espaço de busca em uma dimensão [27].

Existem outros parâmetros que são específicos do SBPSO. Eles são relacionados com a) a quantidade e a velocidade das partículas escoteiras, b) a criação de pontos adjacentes e de inspeção e c) a detecção de convergência prematura.

O parâmetro que define a quantidade de escoteiras foi definido em 10% do número total de partículas do enxame. Outros parâmetros também foram apresentados nas equações 7, 9 e 11. Respectivamente, esses parâmetros foram definidos da seguinte maneira: $c_3 = 1.5$, α corresponde a 1% do tamanho do espaço de busca em uma dimensão e $\lambda = 10^{-4}$.

Para detectar a convergência prematura, dois *thresholds* foram usados: δ_{stag} e δ_{conv}. Para δ_{stag} foi atribuído o valor 10^{-3} e o *threshold* δ_{conv} corresponde a 5% do número total de iterações. δ_{conv} foi usado para definir o número de iterações cujo valor de *fitness* da partícula *gBest* não foi melhorado significativamente.

Todos os valores atribuídos aos parâmetros descritos acima foram empiricamente escolhidos depois de várias simulações que apresentaram bons resultados nas funções avaliadas.

C. Resultados Experimentais

A performance do algoritmo proposto é avaliada da seguinte forma: primeiro, o SBPSO é comparado com o PSO tradicional e a variante PSO-Scout. Em seguida, o SBPSO é comparado com quatro variantes do PSO disponíveis na literatura: QPSO[25], WQPSO[26], HWPSO[27] e HPSOM[28].

O conceito de serendipidade, quando combinado com a utilização de partículas escoteira, revela que o comportamento do SBPSO é mais ativo do que o do PSO tradicional e o do PSO-Scout, uma variante implementada neste trabalho.

As figuras 2–7 apresentam o comportamento do enxame (com 5 partículas) nas funções Esfera, Rosenbrock, Griewank, Rastrigin, HappyCat e Ackley, em 30 dimensões, durante 2.000 iterações respectivamente. Nas figuras, o PSO tradicional e o PSO-Scout estagnaram antes da iteração 2.000, porém o SBPSO continua em atividade.

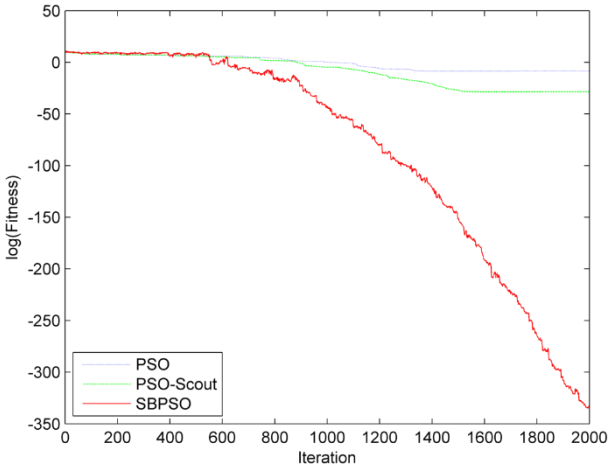

Figura 2. Na função Esfera (5 partículas), o PSO tradicional estagnou próximo à iteração 1.380 e o PSO-Scout próximo à iteração 1.500. Por outro lado, o SBPSO permance ativo próximo à iteração 2.000.

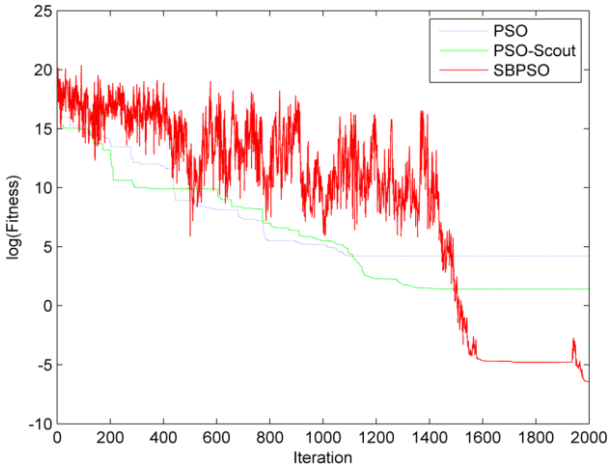

Figura 3. Na função Rosenbrock (5 partículas), o PSO tradicional estagnou próximo à iteração 1.100 e o PSO-Scout próximo à iteração 1.380. É observado que o SBPSO permance ativo próximo à iteração 2.000.

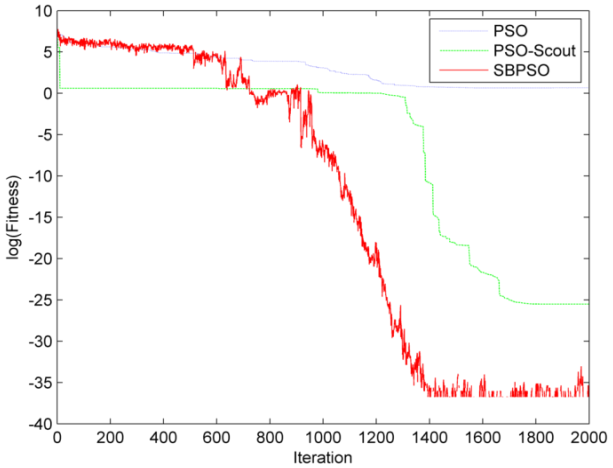

Figura 4. Na função Griewank (5 partículas), o PSO estagnou próximo à iteração 1.400, enquanto o PSO-Scout próximo à iteração 1.700. Observa-se que o SBPSO permance ativo próximo à iteração 2.000.

A escalabilidade do novo algoritmo também é investigada. O tamanho da população (20, 40 e 80), a dimensão das funções (10, 20 e 30) e o número máximo de iterações (1.000, 1.500 e 2.000) são variados para cada função de *benchmark*. Nos experimentos, a melhor média do valor de *fitness* e o desvio padrão obtidos são registrados.

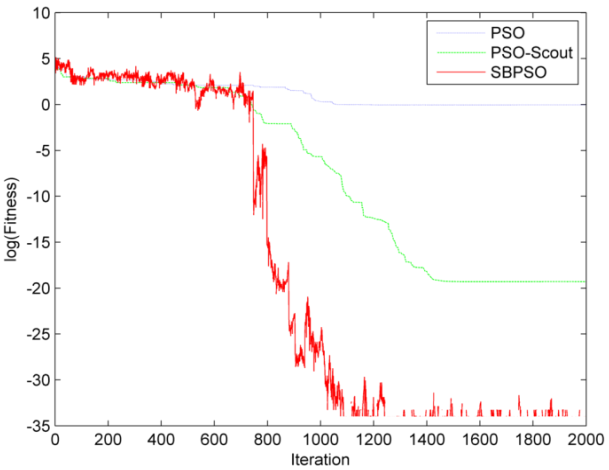

Figura 5. Na função Rastrigin (5 partículas), o PSO tradicional estagnou próximo à iteração 1.080 e o PSO-Scout próximo à iteração 1.400. É observado que o SBPSO permance ativo próximo à iteração 2.000.

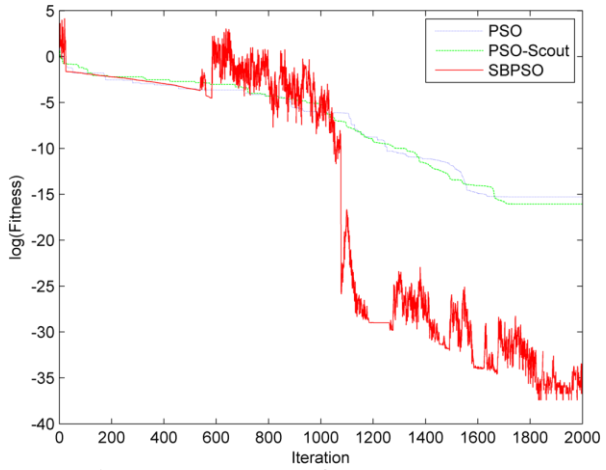

Figura 6. Na função HappyCat (5 partículas), o PSO tradicional estagnou próximo à iteração 1.600 e o PSO-Scout próximo à iteração 1.700. Por outro lado, o SBPSO permance ativo próximo à iteração 2.000.

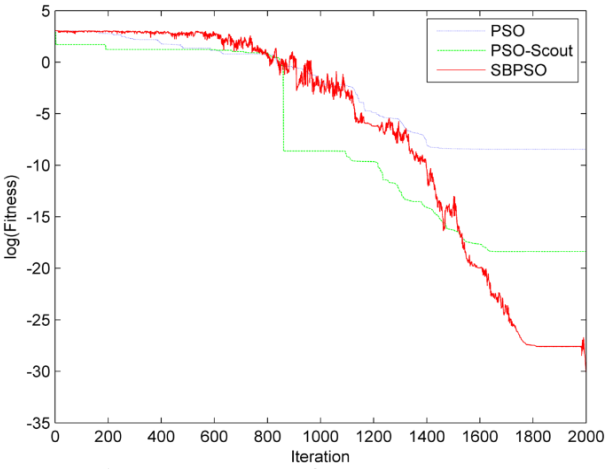

Figura 7. Na função Ackley (5 partículas), o PSO tradicional estagnou próximo à iteração 1.400 e o PSO-Scout próximo à iteração 1.600.

As figuras 8–13 mostram o processo de convergência medido em 100 execuções do algoritmo. O processo ocorre no espaço de busca das funções Esfera (Figura 8), Rosenbrock (Fig. 9), Griewank (Fig. 10), Rastrigin (Fig. 11), HappyCat (Fig. 12) e Ackley (Fig. 13) usando 20 partículas, 30 dimensões e 2.000 iterações. As tabelas II, III e IV mostram o tamanho da população, a dimensionalidade da função, o número de iterações, a melhor média do valor de *fitness* e o desvio padrão para cada função de *benchmark*.

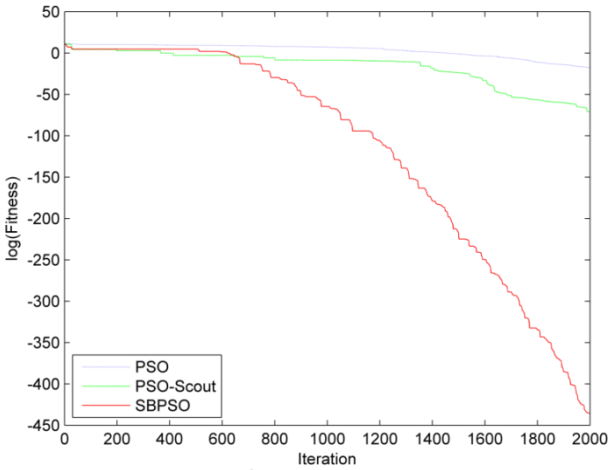

Figura 8. Processo de convergência do PSO, PSO-Scout e SBPSO para a função Esfera (20 partículas, 30 dimensões), medido em 100 execuções.

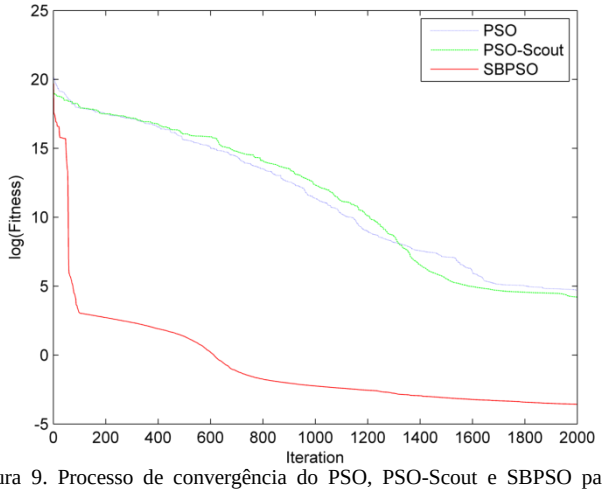

Figura 9. Processo de convergência do PSO, PSO-Scout e SBPSO para a função Rosenbrock (20 partículas, 30 dimensões), medido em 100 execuções.

O Teste de Wilcoxon é aplicado com nível de significância 0.05 para avaliar as soluções encontradas pelo PSO tradicional, PSO-Scout e SBPSO. Wilcoxon [34] é um teste estatístico não-paramétrico usado para comparar duas amostras independentes. A hipótese nula H_0 indica que duas amostras vem 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 menor do que o nível de significância, então decide-se rejeitar H_0, ou seja, há uma diferença significativa entre as amostras.

A Tabela V apresenta os *p-values* obtidos do Teste de Wilcoxon quando o SBPSO é comparado com a variante PSO-Scout. O teste é usado para verificar se os resultados do SBPSO são estatisticamente significativos. Todos os

experimentos apresentaram *p-values* inferiores ao nível de significância definido (0.05). Nesses casos, a hipótese H_0 é rejeitada. Isso garante que não há igualdade entre as soluções encontradas pelos métodos e que o SBPSO, estatisticamente, obteve melhores resultados que a variante PSO-Scout. O SBPSO também superou o PSO em todos os experimentos.

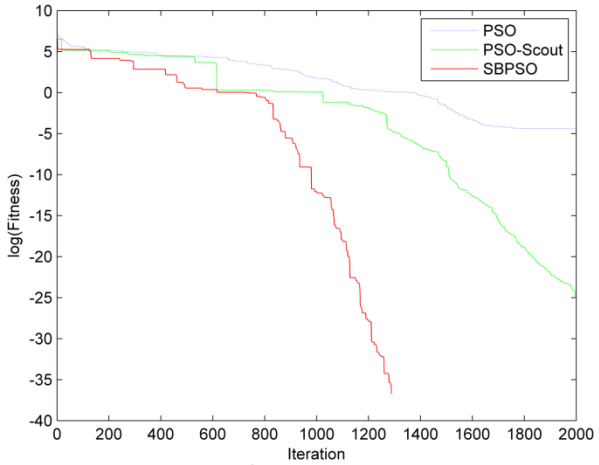

Figura 10. Processo de convergência do PSO, PSO-Scout e SBPSO para a função Griewank (20 partículas, 30 dimensões), medido em 100 execuções.

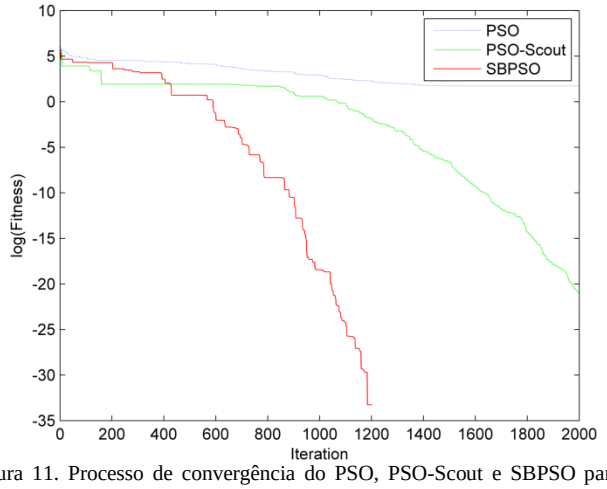

Figura 11. Processo de convergência do PSO, PSO-Scout e SBPSO para a função Rastrigin (20 partículas, 30 dimensões), medido em 100 execuções.

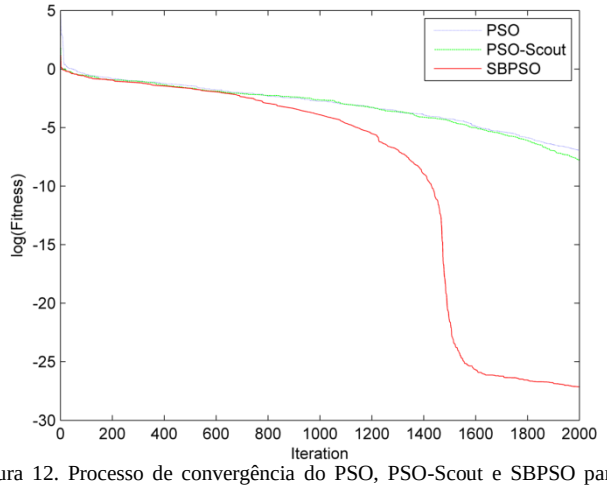

Figura 12. Processo de convergência do PSO, PSO-Scout e SBPSO para a função HappyCat (20 part., 30 dim.), medido em 100 execuções.

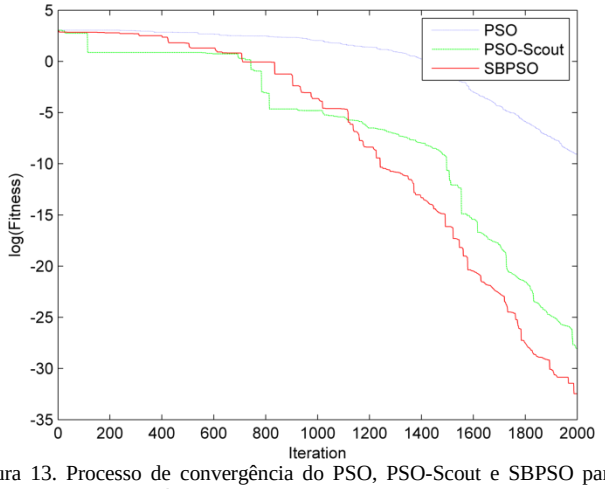

Figura 13. Processo de convergência do PSO, PSO-Scout e SBPSO para a função Ackley (20 partículas, 30 dimensões), medido em 100 execuções.

Os resultados do SBPSO também são comparados com algumas variantes disponíveis na literatura: QPSO, WQPSO, HWPSO e HPSOM. A comparação considera quatro funções de *benchmark*: Esfera, Rosenbrock, Griewank e Rastrigin. Na comparação, são variados as dimensões (10, 20 e 30) e o número máximo de iterações (1.000, 1.500 e 2.000). Apenas o tamanho da população é fixado em 20 partículas. A melhor média do valor de *fitness* e o desvio padrão obtidos são registrados.

A Tabela VI compara os resultados do algoritmo proposto com os das quatro variantes citadas acima. Nas funções unimodais Esfera e Rosenbrock, apesar de o ótimo global não ter sido encontrado ($f(x^*) = 0$), o algoritmo apresenta boa performance nos valores de *fitness* médios obtidos. Por outro lado, quando as funções multimodais Griewank e Rastrigin são avaliadas, o algoritmo encontra os seus ótimos globais. Os resultados apresentados na Tabela VI mostram que o SBPSO supera as outras variantes.

As funções Esfera e HappyCat foram usadas para testar a habilidade do algoritmo para realizar buscas locais. Já a função Rosenbrock foi usada para testar o algoritmo em buscas locais e globais. Por fim, as funções Griewank, Rastrigin e Ackley foram empregadas para testar a habilidade do algoritmo para realizar buscas globais.

V. Conclusão

Este trabalho apresentou uma variante do PSO tradicional chamada *Serendipity-Based Particle Swarm Optimization* (SBPSO) com o objetivo de retardar a convergência prematura em problemas de otimização. O algoritmo usou uma abordagem que considera duas dimensões do conceito de serendipidade: acaso e sagacidade. A dimensão acaso foi implementada por meio da utilização de partículas escoteiras para melhorar a exploração do espaço de busca. Já a dimensão sagacidade foi implementada por meio de inspeções nas adjacências da partícula $gBest$.

Depois da realização dos experimentos, observou-se que o SBPSO superou o PSO tradicional e uma variante chamada PSO-Scout. Para realizar essas comparações, dois critérios foram analisados: convergência e estagnação.

TABELA II. COMPARAÇÃO ENTRE PSO TRADICIONAL, PSO-SCOUT E SBPSO PARA AS FUNÇÕES ESFERA E ROSENBROCK.

Part	Dim	Iter	Esfera (f_1)			Rosenbrock (f_2)		
			PSO	PSO-Scout	SBPSO	PSO	PSO-Scout	SBPSO
20	10	1.000	1.2368E-020 (3.1403E-020)	1.1644E-22 (3.5357E-22)	3.7821E-156 (5.8921E-155)	57.7381 (130.8324)	39.8921 (101.8241)	1.4324E-04 (7.9193E-05)
	20	1.500	2.9396E-011 (1.8370E-010)	5.1239E-018 (5.0147E-017)	7.3821E-124 (9.6715E-123)	105.5410 (159.7832)	97.7863 (183.8732)	9.7916E-03 (7.7471E-04)
	30	2.000	3.8711E-007 (1.4253E-006)	3.9683E-20 (3.1021E-19)	3.6721E-081 (2.7621E-080)	150.7192 (235.0521)	106.5631 (148.6737)	2.8091E-02 (4.3134E-03)
40	10	1.000	1.9504E-022 (1.9105E-021)	2.0521E-027 (1.3162E-026)	3.9011E-129 (9.003E-128)	29.6739 (115.8193)	13.7843 (36.8161)	1.0476E-04 (3.7346E-04)
	20	1.500	9.0576E-015 (2.3939E-014)	2.2788E-024 (2.0506E-023)	7.8921E-119 (2.8321E-118)	85.8143 (140.7206)	47.6621 (59.0512)	7.1592E-03 (5.7066E-04)
	30	2.000	2.0422E-010 (3.7569E-010)	5.4840E-036 (3.6081E-035)	9.7813E-084 (8.6311E-083)	131.6743 (203.8510)	114.8821 (14.7843)	1.9692E-02 (1.3895E-03)
80	10	1.000	7.6959E-028 (2.9084E-028)	2.5243E-031 (8.8054E-031)	3.09160E-162 (2.7813E-161)	23.5431 (41.5023)	12.8312 (33.8817)	9.9319E-04 (6.3264E-05)
	20	1.500	1.5353E-017 (6.7061E-017)	1.5265E-027 (1.3953E-026)	8.9143E-120 (6.8931E-119)	64.9304 (108.8135)	50.7721 (91.5321)	5.6966E-03 (3.6902E-04)
	30	2.000	1.9450E-013 (3.7207E-013)	2.2743E-030 (2.2743E-029)	8.7810E-081 (7.9503E-080)	88.8442 (124.4092)	68.8216 (89.8842)	1.6086E-02 (1.9874E-03)

TABELA III. COMPARAÇÃO ENTRE PSO TRADICIONAL, PSO-SCOUT E SBPSO PARA AS FUNÇÕES GRIEWANK E RASTRIGIN.

Part	Dim	Iter	Griewank (f_3)			Rastrigin (f_4)		
			PSO	PSO-Scout	SBPSO	PSO	PSO-Scout	SBPSO
20	10	1.000	0.1692 (0.0431)	0.0365 (0.0274)	0 (0)	3.9673 (2.8921)	0.9514 (0.6391)	0 (0)
	20	1.500	0.0732 (0.0994)	9.6410E-04 (3.8202E-03)	0 (0)	20.6732 (11.7471)	0.0475 (0.3913)	0 (0)
	30	2.000	0.0185 (0.0142)	7.7481E-04 (8.6723E-03)	0 (0)	48.7790 (13.4018)	0.0253 (0.1905)	0 (0)
40	10	1.000	0.05362 (0.0532)	0.0673 (0.0192)	0 (0)	4.6732 (1.7291)	0.3781 (0.8965)	0 (0)
	20	1.500	0.0287 (0.0213)	4.8193E-04 (1.8421E-03)	0 (0)	14.6329 (1.9013)	5.8931E-04 (6.4832E-03)	0 (0)
	30	2.000	0.0532 (0.1852)	4.8921E-04 (1.0931E-03)	0 (0)	37.8871 (2.7654)	7.8932E-04 (8.8104E-03)	0 (0)
80	10	1.000	0.0659 (0.0325)	0.0298 (0.0497)	0 (0)	1.9591 (0.6492)	0.1738 (0.5821)	0 (0)
	20	1.500	0.0383 (0.0397)	2.8816E-04 (2.0753E-03)	0 (0)	11.6743 (2.5279)	4.6723E-04 (4.9129E-03	0 (0)
	30	2.000	0.0194 (0.0128)	3.9021E-04 (6.8921E-03)	0 (0)	33.6462 (8.8769)	5.8821E-04 (9.6721E-03)	0 (0)

TABELA IV. COMPARAÇÃO ENTRE PSO TRADICIONAL, PSO-SCOUT E SBPSO PARA AS FUNÇÕES HAPPYCAT E ACKLEY.

Part	Dim	Iter	HappyCat (f_5)			Ackley (f_6)		
			PSO	PSO-Scout	SBPSO	PSO	PSO-Scout	SBPSO
20	10	1.000	1.2226E-10 (1.2960E-10)	1.0736E-11 (1.6935E-11)	1.3323E-16 (9.2888E-17)	1.0284E-10 (1.6075E-10)	3.2633E-11 (7.3606E-11)	9.2371E-16 (3.5527E-16)
	20	1.500	2.0678E-05 (2.7529E-05)	2.1938E-05 (1.7937E-05)	1.6768E-13 (1.2280E-13)	2.1024 (6.1631)	1.2274E-07 (1.0225E-06)	8.8818E-16 (7.7843E-16)
	30	2.000	9.5609E-04 (6.4488E-04)	4.2530E-04 (2.1559E-04)	1.6382E-12 (7.1549E-12)	5.4425 (8.7257)	8.9242E-08 (4.6587E-07)	6.6543E-16 (3.7832E-16)
40	10	1.000	2.4402E-13 (4.2835E-13)	3.6771E-14 (2.5444E-14)	2.2204E-16 (1.3597E-16)	0.4030 (2.8358)	2.3100E-13 (5.7051E-13)	6.4310E-16 (3.5146E-16)
	20	1.500	1.8586E-06 (1.8930E-06)	9.8826E-07 (7.8267E-07)	4.5530E-14 (4.8989E-14)	1.2258 (4.8766)	6.5643E-10 (2.1161E-09)	9.7320E-16 (5.7781E-16)
	30	2.000	2.7231E-04 (1.0315E-04)	4.30201E-04 (4.7691E-04)	6.1959E-13 (3.6021E-13)	2.9268 (7.3008)	5.1107E-08 (2.4071E-07)	5.2401E-16 (7.0631E-16)
80	10	1.000	4.2327E-15 (3.8337E-14)	4.2327E-15 (4.1428E-14)	2.3592E-16 (1.3838E-16)	6.1342 (9.8780)	1.0511E-08 (2.0403E-08)	7.6102E-16 (3.8193E-16)
	20	1.500	1.7293E-07 (1.0147E-07)	1.8845E-07 (8.1058E-08)	5.2958E-15 (3.2068E-15)	0.40728 (2.8653)	1.1966E-11 (2.6224E-11)	9.4768E-16 (2.6347E-16)
	30	2.000	3.9029E-05 (1.3066E-05)	8.0394E-05 (4.4500E-05)	4.6358E-13 (3.2960E-13)	1.8144 (5.799)	7.9845E-10 (2.5092E-09)	8.6098E-16 (1.7291E-16)

TABELA V. RESULTADOS DO TESTE DE WILCOXON COM NÍVEL DE SIGNIFICÂNCIA 0.05.

Part	Dim	Iter	p-Value (PSO-Scout × SBPSO)					
			f_1	f_2	f_3	f_4	f_5	f_6
20	10	1.000	1.5777E-30	4.9211E-15	3.9173E-14	5.9021E-13	4.4672E-09	3.1554E-30
	20	1.500	1.5777E-30	2.8911E-15	2.9021E-14	7.9021E-13	6.6469E-08	1.2621E-29
	30	2.000	1.5777E-30	3.9021E-13	4.9021E-14	4.8921E-12	1.9484E-06	4.6200E-12
40	10	1.000	1.5777E-30	3.9011E-15	5.9021E-14	5.9013E-13	6.7876E-09	2.8464E-30
	20	1.500	1.5777E-30	7.0291E-15	8.9021E-14	7.1642E-13	2.9386E-08	6.3108E-29
	30	2.000	1.5777E-30	5.9917E-13	7.6729E-14	2.4321E-12	3.0459E-06	6.7837E-17
80	10	1.000	1.5777E-30	4.6820E-15	5.9021E-14	3.6284E-13	2.7064E-05	5.8302E-30
	20	1.500	1.5777E-30	2.9013E-15	5.9915E-14	2.3812E-13	7.9331E-07	3.1912E-29
	30	2.000	1.5777E-30	1.9747E-13	3.4320E-14	1.8652E-12	1.8108E-09	1.6155E-27

TABELA VI. COMPARAÇÃO ENTRE DIFERENTES VARIANTES DO PSO TRADICIONAL COM UMA POPULAÇÃO DE 20 PARTÍCULAS.

Função	Dim	Iter	PSO	HWPSO	WQPSO	QPSO	HPSOM	PSO-Scout	SBPSO
f_1	10	1.000	1.2368E-20 (3.1403E-20)	6.2868E-56 (1.506E-55)	2.2922E-56 (1.7300E-58)	1.3909E-41 (1.4049E-043)	2.2400E-56 (1.7300E-58)	1.1644E-22 (3.5357E-22)	3.7821E-156 (5.8921E-155)
	20	1.500	2.9396E-11 (1.8370E-10)	6.2830E-45 (2.0330E-44)	2.9451E-40 (2.8717E042)	3.5103E-022 (3.5452E-24)	2.1449E-49 (1.6891E-43)	5.1239E-18 (5.0147E-17)	7.3821E-124 (9.6715E-123)
	30	2.000	3.8711E-07 (1.4253E-06)	3.7940E-36 (1.4060E-35)	3.9664E-33 (3.8435E-35)	5.3183E-014 (5.3623E-16)	6.5764E-034 (5.6809E-36)	3.9683E-20 (3.1021E-19)	3.6721E-081 (2.7621E-080)
f_2	10	1.000	57.7381 (130.8324)	36.4736 (0.1844)	35.8436 (0.2843)	51.9761 (0.4737)	6.9688 (0.2730)	39.8921 (101.8241)	1.4324E-04 (7.9193E-05)
	20	1.500	105.5410 (159.7832)	65.6678 (0.5870)	62.7696 (0.4860)	136.8782 (0.6417)	17.3033 (0.2210)	97.7863 (183.8732)	9.7916E-03 (7.7471E-04)
	30	2.000	150.7192 (235.0521)	70.7275 (0.4813)	70.9525 (0.4283)	157.4707 (0.8287)	27.5645 (0.2939)	106.5631 (148.6737)	2.8091E-02 (4.3134E-03)
f_3	10	1.000	0.1692 (0.0431)	0.13333 (0.33993)	5.6353E-04 (5.5093E-04)	5.5093E-04 (0.0657)	4.32057E-11 (3.1216E-11)	0.0365 (0.0274)	0 (0)
	20	1.500	0.0732 (0.0994)	2.9333 (2.7439)	2.1318E-04 (1.0402E-04)	1.0402E-04 (0.0211)	4.00370E-11 (3.13852E-11)	9.6410E-04 (3.8202E-03)	0 (0)
	30	2.000	0.0185 (0.0142)	9.2333 (6.1455)	2.1286E-04 (1.2425E-04)	1.2425E-04 (0.0110)	5.1756E-11 (3.0143E-11)	7.7481E-04 (8.6723E-03)	0 (0)
f_4	10	1.000	3.9673 (2.8921)	4.6100 (2.5364)	4.0567 (0.0094)	4.8274 (0.0015)	4.1558E-11 (3.2202E-11)	0.9514 (0.6391)	0 (0)
	20	1.500	20.6732 (11.7471)	19.6670 (6.7661)	12.1102 (0.0287)	16.0519 (0.0414)	4.1403E-11 (3.2402E-11)	0.0475 (0.3913)	0 (0)
	30	2.000	48.7790 (13.4018)	44.7230 (13.9680)	23.5593 (0.0713)	33.7218 (0.0114)	4.8007E-11 (3.1883E-11)	0.0253 (0.1905)	0 (0)

No critério convergência, quando as funções Griewank e Rastrigin foram avaliadas, o algoritmo SBPSO encontrou as soluções globais mais rapidamente que os outros algoritmos. Para as funções Esfera, Rosenbrock, HappyCat e Ackley, nenhum dos algoritmos encontrou as soluções ótimas, no entanto, o SBPSO encontrou soluções bastante satisfatórias quando comparadas às encontradas pelos outros algoritmos.

A avaliação do critério estagnação na função Esfera mostrou que, quando o SBPSO foi comparado com o PSO tradicional houve um atraso de, pelo menos, 31% do número máximo de iterações; já em relação ao PSO-Scout, o SBPSO retardou a estagnação em 25%, pelo menos. Para a função Rosenbrock, o retardo da estagnação foi, pelo menos, 45% do número de iterações em relação ao PSO tradicional; já em relação ao PSO-Scout, o atraso foi de, pelo menos, 31%. Para a função Griewank, o SBPSO retardou a estagnação em, pelo menos, 30% quando comparado com o PSO tradicional e em, pelo menos, 15% comparado ao PSO-Scout. Para a função Rastrigin, a estagnação foi retardada em, pelo menos, 46% e 30% em relação ao PSO tradicional e ao PSO-Scout, respectivamente. Na função HappyCat, quando o SBPSO foi comparado com o PSO tradicional houve um atraso de, pelo menos, 20% do número máximo de iterações; já em relação ao PSO-Scout, o SBPSO retardou a estagnação em 15%, pelo menos. Por fim, na função Ackley, o SBPSO retardou a estagnação em, pelo menos, 30% e 20%, respectivamente, em relação ao PSO tradicional e ao PSO-Scout.

Os experimentos também foram realizados para comparar o SBPSO com alguns estudos da literatura, considerando o mesmo tamanho da população (20 partículas) e o mesmo número de variáveis (10, 20 e 30) e iterações (1.000, 1.500 e 2.000). Em todos os experimentos, o SBPSO também apresentou um melhor comportamento de convergência, superando algumas variantes em relação à qualidade da solução, a capacidade de encontrar o ótimo global, a estabilidade das soluções e a capacidade de reiniciar o movimento do enxame em caso de estagnação.

Em geral, os resultados se mostraram promissores para a aplicação em problemas de otimização, uma vez que a convergência prematura foi realmente retardada nos experimentos. No entanto, observou-se que o protótipo necessita de alguns ajustes para melhorar o seu tempo de execução e testar a viabilidade da utilização de outras estratégias de implementação da serendipidade.

REFERÊNCIAS

[1] C.-M. Pintea, *Advances in Bio-inspired Computing for Combinatorial Optimization Problems*, vol. 57 of *Intelligent Systems Reference Library*. Springer, 2014.

[2] C. R. M. Silva, H. W. C. Lins, S. R. Martins, E. L. F. Barreto, and A. G. D'Assunção, "A multiobjective optimization of a uwb antenna using a self organizing genetic algorithm," *Microwave and Optical Technology Letters*, vol. 54, no. 8, pp. 1824–1828, 2012.

[3] E. Dosciatti, W. Godoy, and A. Foronda, "Tq/pso - a new scheduler to optimize the time frame with pso in wimax networks," *IEEE Latin America Transactions*, vol. 13, pp. 365–376, Jan 2015.

[4] A. Esmin, G. Lambert-Torres, and A. de Souza, "A hybrid particle swarm optimization applied to loss power minimization," *Power Systems, IEEE Transactions on*, vol. 20, pp. 859–866, May 2005.

[5] J. Perez, O. Hernandez, R. Caporal, J. de J R Magdaleno, and H. Barreto, "Parameter identification of a permanent magnet synchronous machine based on current decay test and particle swarm optimization," *IEEE Latin America Transactions*, vol. 11, pp. 1176–1181, Sept 2013.

[6] A.-P. Chen, C.-H. Huang, and Y.-C. Hsu, "A novel modified particle swarm optimization for forecasting financial time series," in *IEEE International Conference on Intelligent Computing and Intelligent Systems.*, vol. 1, pp. 683–687, Nov 2009.

[7] L. Jin and Y. Huang, "A particle swarm optimization-neural network prediction model for typhoon intensity based on isometric mapping algorithm," in *Fifth International Joint Conference on Computational Sciences and Optimization*, pp. 857–861, June 2012.

[8] A. Boukerche, K. R. L. Jucá, M. S. M. A. Notare, and J. B. M. Sobral, "Biological inspired based intrusion detection models for mobile telecommunication systems.," in *Handbook of Bioinspired Algorithms and Applications*, Chapman and Hall/CRC, 2005.

[9] R. B. Machado, A. Boukerche, J. B. M. Sobral, K. R. L. Jucá, and M. S. M. A. Notare, "A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations.," in *IPDPS*, IEEE Computer Society, 2005.

[10] E. Silva and C. Bastos Filho, "Pso efficient implementation on gpus using low latency memory," *IEEE Latin America Transactions*, vol. 13, pp. 1619–1624, May 2015.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948, Nov 1995.

[12] F. A. P. Paiva, J. A. F. Costa, and C. R. M. Silva, "A hierarchical architecture for ontology-based recommender systems," in *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC)*, pp. 362–367, 2013.

[13] F. A. P. Paiva, J. A. F. Costa, and C. R. M. Silva, "An ontology-based recommender system architecture for semantic searches in vehicles sales portals," in *Hybrid Artificial Intelligence Systems* (M. Polycarpou, A. de Carvalho, J.-S. Pan, M. Wo´zniak, H. Quintian, and E. Corchado, eds.), vol. 8480 of *Lecture Notes in Computer Science*, pp. 537–548, Springer International Publishing, 2014.

[14] 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*, RecSys '10, (New York, NY, USA), pp. 257–260, ACM, 2010.

[15] K. Oku and F. Hattori, "Fusion-based recommender system for improving serendipity.," in *DiveRS@RecSys*, vol. 816 of *CEUR Workshop Proceedings*, pp. 19–26, CEUR-WS.org, 2011.

[16] P. Adamopoulos and E. Tuzhilin, "On unexpectedness in recommender systems: Or how to expect the unexpected," in *Proc. of RecSys'11 Intl. Workshop on Novelty and Diversity in Recommender Systems*, 2011.

[17] F. A. P. Paiva, J. A. F. Costa, and C. R. M. Silva, "Uma metaheurística alternativa de inteligência de enxames baseada em serendipidade guiada," in *2nd LA-CCI (Latin American) and 12th CBIC Brazilian Congress on Computational Intelligence*, 2015.

[18] A. Silva, A. Neves, and T. Gonçalves, "Using scout particles to improve a predator-prey optimizer," in *Adaptive and Natural Computing Algorithms, 11th International Conference, ICANNGA 2013, Lausanne, Switzerland, April 4-6, 2013.* pp. 130–139, 2013.

[19] Y.-L. Wu, T.-F. Ho, S. J. Shyu, and B. M. T. Lin, "Discrete particle swarm optimization with scout particles for library materials acquisition," *The Scientific World Journal*, vol. 2013, pp. 1–11, 2013.

[20] S. Catellin, *Sérendipité: Du conte au concept*. Paris: Seuil, 2014.

[21] J. Campos and A. D. Figueiredo, "Programming for serendipity," in *Proceedings of Fall Symposium on Chance Discovery - The Discovery and Management of Chance Events*, pp. 48–60, 2002.

[22] G. I. Evers and M. B. Ghalia, "Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks.," in *SMC*, pp. 3901–3908, IEEE, 2009.

[23] F. Van Den Bergh, *An Analysis of Particle Swarm Optimizers*. PhD thesis, Pretoria, South Africa, South Africa, 2002.

[24] H. Leitão, W. Lopes, and F. Madeiro, "Pso algorithm applied to codebook design for channel-optimized vector quantization," *IEEE Latin America Transactions*, vol. 13, pp. 961–967, April 2015.

[25] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, pp. 325–331, June 2004.

[26] M. Xi, J. Sun, and W. Xu, "An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position.," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 751–759, 2008.

[27] S.-H. Ling, H. H. C. Iu, K. Y. Chan, H.-K. Lam, C. W. Yeung, and F. H. F. Leung, "Hybrid particle swarm optimization with wavelet mutation and its industrial applications.," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 3, pp. 743–763, 2008.

[28] A. Esmin and S. Matwin, "Hpsom: A hybrid particle swarm optimization algorithm with genetic mutation," *International Journal of Innovative Computing, Information and Control*, vol. 9, pp. 1919–1934, May 2013.

[29] 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.

[30] E. Toms, "Serendipitous information retrieval," in *First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, pp. 11–12, 2000.

[31] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.

[32] H.-G. Beyer and S. Finck, "Happycat - a simple function class where well-known direct search algorithms do fail.," in *PPSN (1)*, vol. 7491 of *Lecture Notes in Computer Science*, pp. 367–376, Springer, 2012.

[33] H. Wang, H. Li, Y. Liu, C. Li, and S. Zeng, "Opposition-based particle swarm algorithm with cauchy mutation," in *IEEE Congress on Evolutionary Computation*, pp. 4750–4756, Sept 2007.

[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006.

Fábio Augusto Procópio de Paiva recebeu seu título de Doutor em Engenharia Elétrica e da Computação pela Uiversidade Federal do Rio Grande do Norte (UFRN), em 2016. É também professor do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), campus Parnamirim. Suas áreas de interesse incluem Inteligência Computacional, Sistemas de Recomendação e Banco de Dados.

José Alfredo Ferreira da Costa recebeu seu título de Doutor em Engenharia Elétrica e da Computação pela Universidade Estadual de Campinas (UNICAMP), em 1999. Atualmente,é Professor Associado da UFRN, Departamento de Engenharia Elétrica. Suas áreas de interesse incluem Redes Neurais e Sistemas Auto-Organizáveis, Visualização de Dados e *Clustering*, Mineração de Dados e *Business Intelligence*.

Cláudio Rodriguez Muniz da Silva recebeu seu título de Doutor em Engenharia Elétrica e da Computação pela Universidade Estadual de Campinas (UNICAMP), em 2003. Atualmente é Professor Associado da UFRN, Departamento de Comunicações. Suas áreas de interesse incluem aplicações de Inteligência Computacional, Sistemas de Recomendação e plataformas Industriais de Internet das Coisas (IIoT).