Otimização por enxame de partículas

usando uma adaptação de serendipidade

Fábio Augusto Procópio de Paiva

editoraifrn

Otimização por enxame de partículas

usando uma adaptação de serendipidade

Fábio Augusto Procópio de Paiva

editoraifrn
Natal, 2018

Prefixo editorial: 94137
Linha Editorial: Acadêmica
Disponível para *download* em:
http://memoria.ifrn.edu.br

editoraifrn

Contato
Endereço: Rua Dr. Nilo Bezerra Ramalho, 1692, Tirol. Natal-RN.
CEP: 59015-300. Telefone: (84) 4005-0763 l E-mail: editora@ifrn.edu.br

Paiva, Fábio Augusto Procópio de.
P142o Otimização por enxame de partículas / Fábio Augusto Procópio de Paiva; projeto gráfico, diagramação e capa Hanna Andreza Fernandes Sobral; revisão linguística Rodrigo Luiz Silva Pessoa. – Natal: IFRN, 2018.
96 p : il. color.

ISBN: 978-85-94137-45-6

1. Serendipidade. 2. Algoritmos Bioinspirados. 3. *Serendipity-Based Particle Swarm Optimization*. I. Paiva, Fábio Augusto Procópio de.II. Título.

CDU 004.421

Catalogação da publicação na fonte elaborada pela Bibliotecária
Patrícia da Silva Souza Martins – CRB: 15/502

Esta obra foi submetida e selecionada por meio de edital específico para publicação pela Editora IFRN, tendo sido analisada por pares no processo de editoração científica.

Eu dedico este livro à minha esposa Raissa e aos meus filhos Heitor e Henrique.

“

A sorte favorece a
mente bem preparada.

●●●

Louis Pasteur

”

AGRADECIMENTOS

Este livro é uma adaptação da minha tese, desenvolvida enquanto aluno de doutorado do Programa de Pós-graduação em Engenharia Elétrica e de Computação (PPgEEC), na Universidade Federal do Rio Grande do Norte (UFRN).

Ao longo do curso, estudei conteúdos de diversas áreas do conhecimento, deparando-me com o cansaço, com o medo e com a vontade de desistir. Por outro lado, aprendi a conviver com desafios que pareciam insuperáveis e, para isso, eu precisei da ajuda de várias pessoas a quem devo os meus sinceros agradecimentos.

Professor José Alfredo, por investir o seu tempo em discussões relacionadas ao tema da minha pesquisa e pelo compromisso que assumiu em ser o meu orientador.

Cláudio Muniz, mais que um professor doutor, um mestre. Enquanto assumia o desafio de me coorientar, vivenciou, de perto, as minhas angústias.

Leandro Pasa, pelas sugestões de melhorias nos artigos que produzi durante o curso de doutorado.

Também devo minha gratidão aos meus amigos, familiares e colegas de trabalho. É impossível citar todos, mas as palavras de apoio de cada um deles permanecem incessantemente em minha memória.

Pró-Reitoria de Pesquisa e Inovação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN) pela construção do edital nº 03/2018, o qual viabilizou a publicação desta obra por meio da Editora IFRN.

Minha esposa e os meus filhos, pelo amor e pela companhia de todos os dias.

Minha avó Eunice (*in memoriam*), meu pai e minha mãe pelo maior presente que me ofereceram: a educação.

Por fim, Deus, indiscutivelmente, o autor principal desta obra.

Sumário

Introdução

Os problemas de otimização estão presentes em diversas áreas do mundo real, como medicina, planejamento de transportes, roteamento de veículos, planejamento da produção, projetos de engenharia, gerenciamento de tráfego aéreo, sistemas de potência, telecomunicações e muitos outros domínios.

A otimização consiste em um processo que objetiva encontrar a melhor solução para determinado problema. Assim, há uma necessidade constante da construção de algoritmos de otimização eficientes e robustos, capazes de resolver problemas em aplicações científicas e do mundo real. Recentemente, os algoritmos de otimização meta-heurísticos vêm sendo amplamente empregados para resolver problemas dessa natureza. As meta-heurísticas[1] são utilizadas por serem mais poderosas que os métodos convencionais, que se baseiam nas lógicas formais ou na programação matemática, além de o tempo necessário para execução ser menor que o dos algoritmos exatos.

Um conjunto de algoritmos inspirado na natureza vem sendo frequentemente aplicado com sucesso na área de otimização. A maioria desses algoritmos se baseia em algumas características de sucesso do sistema biológico e, portanto, são chamados de algoritmos bioinspirados. Uma classe especial deles se inspira na inteligência dos enxames e, por isso, são conhecidos como algoritmos baseados em inteligência de enxames.

A inteligência de enxames diz respeito ao comportamento coletivo de vários agentes que interagem entre si e que seguem algumas regras. Um agente, quando é observado de forma individual, não pode ser considerado inteligente, porém um sistema com vários deles apresenta um comportamento auto-organizável e utiliza a inteligência coletiva.

1 São algoritmos gerais adaptáveis a diversos problemas de otimização.

Esses algoritmos consistem em um conjunto de meta-heurísticas que se baseia em alguns seres vivos cujos comportamentos emergentes podem resultar em uma capacidade de resolver problemas complexos. A Otimização por Enxame de Partículas (PSO – *Particle Swarm Optimization*), por exemplo, tem sido amplamente utilizada no contexto de problemas de otimização para implementar técnicas que buscam inspiração no comportamento social das aves e dos peixes. Os indivíduos da população são representados por partículas no espaço de busca, as quais se movimentam com o objetivo de convergir para a posição de uma partícula ótima.

Segundo Gandomi e Alavi (2012), a PSO é uma das meta-heurísticas mais utilizadas para resolução de problemas de otimização. Ela ganhou muita atenção devido a algumas de suas características, como aplicabilidade a pesquisas científicas e a problemas do mundo real; exploração mais rápida do espaço de busca, quando comparada com algumas meta-heurísticas tradicionais; e facilidade de implementação.

Apesar dessas características, muitas vezes, a implementação da PSO enfrenta um problema em que as suas partículas ficam "presas" em soluções subótimas, também chamadas de ótimos locais. Esse problema enfrentado durante o processo de otimização é conhecido como convergência prematura e ocorre quando o algoritmo perde a sua capacidade de gerar diversidade de soluções.

Muitas abordagens foram propostas para acelerar a velocidade de convergência da PSO e evitar a convergência prematura. Operadores que são frequentemente utilizados nos algoritmos genéticos[2], tais como seleção e mutação, também têm sido incor-

2 Algoritmos de otimização global que se baseiam nos mecanismos da genética e da seleção natural.

porados na PSO por serem eficientes em melhorar a diversidade do enxame. Outras abordagens, como sistema quântico e teoria *wavelet*, também foram propostas para acelerar a velocidade de convergência da PSO.

Essas abordagens são essencialmente estocásticas, porém elas podem se beneficiar de combinações com outras técnicas para diminuir a aleatoriedade e melhorar a performance da meta-heurística. Para esse propósito, uma abordagem interessante é a adaptação e a implementação do conceito de serendipidade, a fim de adicioná-lo em um algoritmo meta-heurístico.

O termo "serendipidade", muitas vezes, é utilizado como: algo que foi achado sem, necessariamente, estar sendo procurado; sinônimo de "qualquer surpresa agradável"; acaso; ou sorte. Pek Van Andel (1994), em um estudo exaustivo, mostrou que a serendipidade tem grande contribuição para o progresso da ciência, da tecnologia e da arte. Na ciência e na tecnologia, as descobertas casuais ocorrem com frequência. Na história da ciência, por exemplo, existem vários casos de serendipidade, como a descoberta da vaselina, em 1859, pelo químico britânico Chesebrough; da sacarina, em 1878, pelo químico russo Fahlberg; do raio-X, em 1895, pelo físico alemão Röntgen; da radioatividade, em 1896, pelo cientista francês Beckerel; do forno micro-ondas, em 1945, pelo engenheiro americano Spencer, além de tantas outras.

A fim de criar mecanismos que possibilitem o enxame escapar de soluções subótimas, este livro estuda o conceito de serendipidade e apresenta uma adaptação desse conceito à área dos algoritmos de inteligência de enxames, em especial à PSO. Com base na adaptação do conceito de serendipidade, uma nova variante da PSO é implementada a fim de melhorar os resultados da PSO original.

Duas estratégias comumente empregadas em uma área de estudo conhecida como sistemas de recomendação são utilizadas para implementar as dimensões de serendipidade: acaso e sagacidade. A dimensão acaso é implementada por meio da estratégia chamada *blind luck* (sorte cega), já a implementação da dimensão sagacidade é realizada usando a estratégia chamada princípio de Pasteur.

Neste livro, a estratégia *blind luck* consiste em usar o comportamento exploratório de um conjunto de partículas (que, aqui, são chamadas partículas escoteiras) para implementar a dimensão acaso. Já a dimensão sagacidade consiste em inspeções nas adjacências da melhor partícula do enxame, também chamada de *gBest*.

Por fim, a nova variante apresentada considera um conjunto de padrões que são relacionados ao comportamento de convergência da PSO. Eles são utilizados para implementar a dimensão sagacidade. A vantagem em se considerar esses padrões é o fato de que eles representam o comportamento do valor de *fitness* das partículas, isto é, pontos sobre uma reta que passam pela partícula *gBest*.

1

Processo de Otimização

As pessoas e a natureza estão sempre tentando otimizar processos. As companhias aéreas, por exemplo, escalam tripulações e aeronaves de modo que o custo das viagens seja minimizado. Os investidores criam carteiras que evitem riscos excessivos quando o objetivo é obter altas taxas de retorno. Por outro lado, na natureza, os sistemas físicos tendem a atingir um estado de energia mínima. Os raios de luz, por exemplo, propagam-se por caminhos pelos quais o seu tempo de viagem tende a ser minimizado.

Otimizar consiste na tarefa de encontrar o melhor resultado possível sob determinado conjunto de restrições. Antes de se iniciar um processo de otimização, é necessário identificar um *objetivo*, isto é, uma medida quantitativa que represente o desempenho do sistema a ser otimizado. O objetivo pode ser, por exemplo, o lucro de um investimento, a tensão de um sistema elétrico ou qualquer quantidade (ou uma combinação delas) que possa ser representada por um único número. O objetivo está em função de um conjunto de características do sistema que são chamadas de *variáveis*. Em geral, as variáveis devem ser tratadas sob algumas restrições. Por exemplo, o tempo de permanência de uma aeronave no solo não pode ser inferior a zero.

O processo para identificar os objetivos, as variáveis e as restrições de certo problema é muitas vezes conhecido como *modelagem*. A construção de um modelo para representar determinado problema deve ser o primeiro passo no processo de otimização.

Depois que o modelo de otimização é definido, um algoritmo pode ser utilizado para encontrar a sua *solução*. Existem vários algoritmos com esse propósito, e cada um deles é adequado para cada tipo de problema de otimização. A escolha de um desses algoritmos pode determinar se a resolução do problema será rápida (ou lenta) e se a solução considerada ótima será encontrada.

O modelo matemático básico de um problema de otimização consiste em minimizar uma função objetivo $f(x)$, a qual está sujeita a restrições em suas variáveis. Nesse modelo, pode-se adotar a seguinte notação: x é o vetor de variáveis (ou parâmetros); f é a função objetivo, ou seja, uma função de x que se deseja minimizar; e c é o vetor contendo as restrições que as variáveis x devem satisfazer. O problema de otimização pode ser escrito como segue:

$$\min_{x \in R^n} f(x) \; sujeito\ a \begin{cases} c_i(x) = 0, i \in \varepsilon \\ c_i(x) \geq 0, i \in \varrho \end{cases} \quad (1)$$

onde f e cada c_i são funções com valores escalares das variáveis x; enquanto ε e ϱ são os conjuntos de índices.

No caso de o problema de otimização consistir em maximizar a função objetivo $f(x)$, isso equivale a, simplesmente, minimizar a função $-f(x)$. Em outras palavras, considerando o ponto x^* como o valor mínimo de $f(x)$, esse mesmo ponto corresponde ao valor máximo da função objetivo $-f(x)$.

1.1 OTIMIZAÇÃO LOCAL E OTIMIZAÇÃO GLOBAL

Em uma região A, de determinado espaço de busca, um minimizador local x_A^* pode ser definido da seguinte maneira:

$$f(x_A^*) \leq f(x), \forall\ x \in A, \quad (2)$$

onde $A \subset S \subseteq \mathbb{R}^n$ e S representa o espaço de busca. Observa-se que $S = \mathbb{R}^n$ quando o problema estudado não possui restrições. Observa-se também que A é um subconjunto próprio de S. Um determinado espaço de busca S pode conter várias sub-regiões A_i de

modo que $A_i \cap A_j \neq \emptyset$, para $i \neq j$. Então $x^*_{\mathrm{A}_i} \neq x^*_{\mathrm{A}_j}$ tal que o minimizador de cada sub-região A_i seja único. Qualquer $x^*_{\mathrm{A}_i}$ pode ser considerado um minimizador de A, embora eles sejam simplesmente minimizadores locais. Não há restrições para os valores que a função pode assumir, então $f\left(x^*_{\mathrm{A}_i}\right) = f\left(x^*_{\mathrm{A}_j}\right)$ é permitido. Assim, o valor $f\left(x^*_{\mathrm{A}_i}\right)$ é chamado de mínimo local. Já o minimizador global x^* pode ser definido como segue:

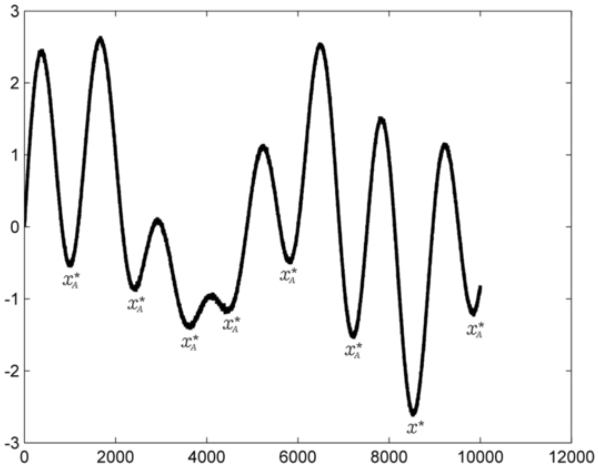
$$f\left(x^*\right) \leq f\left(x\right), \forall\, x \in S, \tag{3}$$

onde S é o espaço de busca. Para problemas sem restrição é comum escolher $S = \mathbb{R}^n$, onde n é a dimensão de x. Existem algumas definições para os algoritmos de otimização global que são diferentes do que foi apresentado na Equação 3. Nessas definições, o algoritmo deve ser capaz de encontrar um minimizador local de $A \subset S$ independentemente da posição atual z_0. Esses algoritmos consistem de dois processos: busca local e busca global.

Em geral, a busca local é a aplicação de um algoritmo de minimização local, enquanto a busca global objetiva garantir que o algoritmo se moverá em uma região A_j, a partir do ponto em que a busca local é capaz de encontrar o minimizador de A_i. Esses algoritmos também são capazes de encontrar o minimizador global, uma vez que a posição de partida z_0 é escolhida corretamente.

A apresenta um exemplo de função que possui vários minimizadores locais x^*_{A} e o minimizador global x^*.

Figura 1 – Função com mínimo global e vários mínimos locais.



Fonte: Autoria própria (2018).

1.2 META-HEURÍSTICA

Um problema complexo, que é submetido a um processo de otimização, pode ser solucionado a partir de um método exato ou aproximado. Os métodos exatos garantem que a solução global ótima será encontrada. Eles são formados pelos seguintes algoritmos: programação dinâmica, programação por restrições, família *branch* (*branch and bound*, *branch and cut* e *branch and price*) e família A* (A* e IDA*). Por outro lado, os métodos aproximados produzem soluções de alta qualidade em tempo razoável, porém não há garantia de que a solução ótima global será encontrada. Os métodos aproximados podem ser classificados em algoritmos heurísticos e algoritmos de aproximação.

Os algoritmos heurísticos encontram boas soluções para problemas complexos. Eles obtêm desempenho e custo aceitáveis

em vários problemas. Normalmente, esses algoritmos não garantem aproximação das soluções obtidas. Eles podem ser classificados em duas subclasses: heurística específica e meta-heurística. As heurísticas específicas são adaptadas e projetadas para resolver um problema específico. Por outro lado, as meta-heurísticas são de uso geral e podem ser aplicadas para resolver qualquer problema de otimização.

A origem do termo "heurística" vem do grego *heuriskein*, que significa "a arte de descobrir novas estratégias para resolução de problemas". O sufixo *meta* é também de origem grega e significa "metodologia de alto nível". O termo "meta-heurística" foi utilizado pela primeira vez por Glover, em 1986.

As meta-heurísticas são formadas por uma família de técnicas de otimização que ganharam popularidade nas últimas duas décadas. O motivo da sua popularização é o fato de elas oferecerem soluções aceitáveis para problemas complexos de ciência e de engenharia em tempo razoável. Os métodos de busca meta-heurística podem ser definidos como metodologias gerais de alto nível e utilizados como estratégias no projeto de heurísticas para resolver problemas específicos de otimização.

A eficiência e a eficácia das meta-heurísticas são comprovadas por meio de sua grande utilização para resolver problemas em diferentes domínios como telecomunicações, gerenciamento de tráfego aéreo, processamento de imagens, automação, aerodinâmica, mercado financeiro, planejamento de robôs, aprendizagem de máquina e outros.

Existem dois componentes presentes nos algoritmos meta-heurísticos bastante significativos: diversificação e intensificação. Na diversificação, as regiões não exploradas devem ser visitadas para garantir que todas as regiões do espaço de busca

sejam uniformemente observadas e que a busca não seja restrita a um número reduzido delas. Por outro lado, na intensificação, as regiões promissoras são exploradas mais profundamente a fim de obter melhores soluções. Algumas meta-heurísticas são mais voltadas à intensificação, enquanto outras possuem características mais fortes de diversificação, porém uma bom equilíbrio entre os dois componentes, em geral, garante o sucesso de uma meta-heurística.

Existem várias formas de se classificar os algoritmos meta-heurísticos, e uma delas é baseá-los em trajetória e em população. Os algoritmos baseados em trajetória trabalham com uma única solução, adicionando mecanismos para escapar de ótimos locais durante o processo de busca. Nesse tipo de algoritmo, a busca se caracteriza por apresentar uma trajetória dentro do espaço de soluções possíveis. Já os algoritmos baseados em população funcionam com um conjunto de soluções. Para criar novos recursos, determinada solução é mesclada implícita ou explicitamente. Eles utilizam fatores de aprendizagem à medida que tentam compreender a correlação entre as variáveis de projeto para identificar as regiões do espaço de busca com soluções de alta qualidade.

Para Zang et al. (2010), os meta-heurísticos possuem várias fontes de inspiração, no entanto a principal delas é a natureza. Os algoritmos inspirados na natureza vêm sendo bastante utilizados no desenvolvimento de sistemas para resolver problemas. Uma das principais categorias desse tipo de algoritmo é a bioinspiração.

1.3 INTELIGÊNCIA DE ENXAMES

Percebe-se claramente a organização estrutural de muitos grupos de animais, como peixes e aves. O comportamento dos

agentes que formam o grupo é tão integrado que, mesmo que eles mudem a direção, parecem mover-se como se fossem apenas um.

Vários estudos foram feitos a fim de analisar o comportamento dinâmico de diferentes seres vivos, em especial o dos insetos sociais, como vespas, aranhas, borboletas, grilos, vaga-lumes, cupins e outros. A tentativa de imitar o comportamento desses seres, por meio de simulações computacionais, resultou em uma área de estudo conhecida como inteligência de enxames.

A expressão "inteligência de enxames" foi utilizada pela primeira vez por Beni e Wang (1989), em trabalho que tinha como proposta um conjunto de bibliotecas para otimização global no domínio de enxame de robôs. A inteligência de enxames pode ser definida como uma disciplina da inteligência artificial, que é utilizada para modelar o comportamento coletivo de enxames sociais da natureza.

Os sistemas baseados nessa inteligência são tipicamente constituídos por uma população de agentes simples, que interagem localmente uns com os outros e com o seu próprio ambiente. Embora esses agentes sejam relativamente simples e possuam capacidades limitadas, eles se relacionam em conjunto com certos padrões de comportamento, a fim de atingir tarefas necessárias para a sua sobrevivência.

Para que um algoritmo de enxame apresente comportamento inteligente, é importante que os seguintes princípios sejam considerados:

1. **Proximidade** – o enxame deve ser capaz de realizar cálculos simples de espaço e de tempo;
2. **Qualidade** – o enxame deve ser capaz de responder às características de qualidade do seu ambiente (por exemplo, qualidade da comida);

3. **Diversidade** – o enxame não deve comprometer a sua atividade ao longo dos canais excessivamente estreitos;
4. **Estabilidade** – o enxame não deve mudar o seu comportamento por causa de toda e qualquer variação ambiental; e
5. **Adaptabilidade** – o enxame deve ser capaz de se adequar, quando for necessário, a variações ambientais.

Os algoritmos de inteligência de enxames têm sido aplicados com sucesso em vários domínios de problemas de otimização. Podem ser citados como exemplos: roteamento de veículos; composições musicais; aplicações de eletromagnetismo; bioinformática; problemas do caixeiro-viajante; treinamentos de redes neurais; processamentos paralelos; otimização de funções, entre outros. Alguns modelos de inteligência de enxames, que já foram implementados computacionalmente, são: PSO, colônia artificial de abelhas (ABC – *Artificial Bee Colony*) , otimização por colônia de formigas (ACO – *Ant Colony Optimization*), algoritmo do vaga-lume (FA – *Firefly Algorithm*), algoritmo do morcego (BA – *Bat Algorithm*) e outros.

Dentre esses diversos algoritmos, a ACO e a PSO são as técnicas que mais se destacam. Elas podem obter soluções aproximadas em tempo computacional razoável. Neste livro, para adaptar o conceito de serendipidade, embora seja possível de ser implementado em outros algoritmos da inteligência de enxames, apenas a PSO será considerada.

2

Otimização por Enxame de Partículas

A maioria das aves possui olhos dos dois lados da cabeça, o que possibilita que enxerguem objetos simultaneamente em ambos os lados. Em geral, elas são atraídas por fontes de alimento e se agrupam com o objetivo de voar longas distâncias em busca de sua alimentação.

As aves também possuem uma interação social que viabiliza voos sem o risco de haver colisões, mesmo quando há mudanças repentinas de direção; espalhamento e rápido reagrupamento quando ameaças externas são percebidas; e capacidade de evitar predadores (KENNEDY; EBERHART, 1995).

As interações entre elas refletem o movimento compartilhado do bando, como mostra a Figura 2. Esse movimento (ou seja, a posição e a velocidade) não é realizado por um coordenador, mas por meio do princípio da proximidade, que se baseia nessas interações.

Figura 2 – Uma ave pode orientar a direção do seu voo de acordo com a posição de seus vizinhos

Fonte: adaptada de Avise (2011).

No contexto dos algoritmos meta-heurísticos, a PSO (KENNEDY; EBERHART, 1995) é uma abordagem estocástica, proposta pelo psicólogo social James Kennedy e pelo engenheiro eletricista Russell Eberhart, para modelar o comportamento social das aves. Ela foi originalmente utilizada para resolver problemas contínuos de otimização não lineares. No entanto, ela vem sendo empregada em muitas aplicações práticas do mundo real.

O termo "partícula" denota um indivíduo de um enxame ou, de forma análoga, uma ave que é membro de um bando. Em outras palavras, o conjunto dessas partículas é conhecido como *enxame*. Quando uma partícula descobre um bom caminho que a leve ao alimento, as outras também são capazes de se deslocar para aquela direção, mesmo que ele esteja distante do resto do enxame. Uma partícula representa uma solução para determinado problema que está sendo otimizado. Ela "sobrevoa" o espaço de busca procurando uma solução ótima, durante certo número de iterações.

Cada partícula é composta de três vetores: a sua posição no espaço de busca em D dimensões $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$; a sua melhor posição encontrada até o momento $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$; e a sua velocidade para uma trajetória $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. O pseudocódigo da PSO é sumarizado no Algoritmo 1.

No início da execução do algoritmo, a posição e a velocidade de cada partícula são inicializadas aleatoriamente (linha 2). Em seguida, o valor de *fitness* de cada uma delas é avaliado (linha 6) por uma função objetivo, a fim de definir os dois "melhores" valores (linhas 7-12). Eles são conhecidos como *pBest* (linha 8) e *gBest* (linha 11).

O *gBest*[3] influencia o movimento da partícula em direção à melhor posição encontrada pelo enxame, enquanto o *pBest*[4] movimenta a partícula para a melhor posição encontrada por ela mesma. Cada partícula é atraída para a localização na qual se encontra a *gBest*. A velocidade e a posição de cada uma são atualizadas, respectivamente, por meio das Equações 4 e 5:

$$v_{id} = w \cdot v_{id} + c_1 \cdot r_1 \left(p_{id} - x_{id}\right) + c_2 \cdot r_2 \left(p_{gd} - x_{id}\right) \quad (4)$$

$$x_{id} = x_{id} + v_{id} \quad (5)$$

Na Equação 4, o termo w é chamado fator de inércia e representa a velocidade inercial da partícula. Os termos v_{id} e x_{id} são, respectivamente, a velocidade e a posição da partícula i. Já p_{id} é a posição em que foi encontrado o melhor valor de *fitness* da partícula i nas iterações anteriores e p_{gd} é a posição em que foi encontrado o melhor valor de *fitness* entre todas as partículas do enxame até o momento. O termo 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 aleatórios uniformemente distribuídos no intervalo [0, 1], gerados a cada iteração, por meio de uma função densidade de probabilidade, para atualizar cada dimensão individual $d \in (1, 2, \ldots, D)$.

3 É a posição da i–ésima partícula cujo valor da função objetivo é o melhor encontrado em todo o enxame.

4 É a posição da i–ésima partícula cujo valor da função objetivo é o seu último melhor valor encontrado.

Algoritmo 1 – Pseudocódigo da PSO original

01: **para cada** *partícula* i *em* S **faça**
02: inicializa aleatoriamente velocidade e posição
03: **fim para**
04: **enquanto** *critério de parada não satisfeito* **faça**
05: **para cada** *partícula* i *em* S **faça**
06: calcula o valor de *fitness*
07: **se** $fit\,(x_{id}) < fit\,(pBest_i)$ **então**
08: $pBest_i \leftarrow x_{id}$
09: **fim se**
10: **se** $fit\,(pBest_i) < fit\,(gBest)$ **então**
11: $gBest \leftarrow pBest_i$
12: **fim se**
13: **fim para**
14: **para cada** *partícula* i *em* S **faça**
15: calcula velocidade usando Equação 4
16: atualiza posição usando Equação 5
17: **fim para**
18: **fim enquanto**

Apesar de a PSO ser uma técnica capaz de encontrar bons resultados em um tempo de convergência rápido, ela corre o risco de convergir prematuramente. A fim de evitar a estagnação do enxame, há três métodos que podem ser utilizados para detectar a convergência prematura (VAN DEN BERGH, 2006):

1) *Maximum swarm radius* – consiste em avaliar a maior distância euclidiana entre uma partícula do enxame e a $gBest$. A es-

tagnação ocorre quando essa distância é inferior a um limite chamado δ_{stag}. O método é formalmente apresentado na Equação 6:

$$\varphi_j = \frac{\max\limits_{i \in \{1, ..., S\}} \|e_{ij} - e^*_{ij}\|}{|\mu_{max} - \mu_{min}|} \quad (6)$$

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 limite δ_{min}. Então, quando um percentual de partículas em C atingir um limite 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 7:

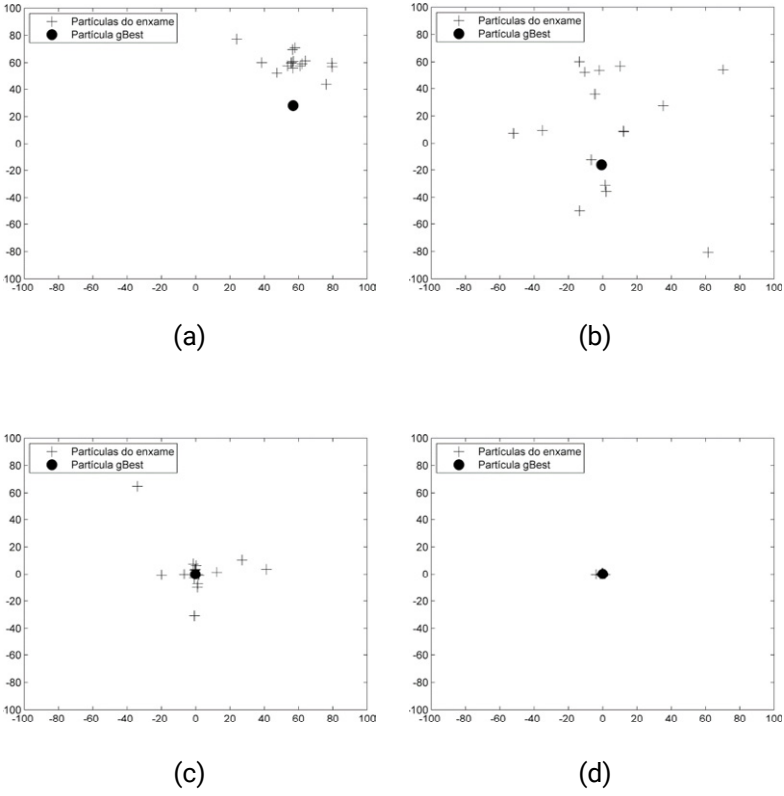
$$f_{ratio} = \frac{f(\hat{y}_j) - f(\hat{y}_{j\text{-}1})}{f(\hat{y}_j)} \quad (7)$$

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

Para Van Den Bergh (2006), o método *objective function slope* é melhor do que os outros, porém apresenta algumas desvantagens. Uma abordagem alternativa para a obtenção de melhores resultados é a combinação do método *objective function slope* com o *cluster analysis* ou com o *maximum swarm radius*.

Para exemplificar o princípio de funcionamento da PSO, usou-se uma função de referência bastante recorrente nos algoritmos de otimização, chamada Esfera, que será apresentada no Capítulo 6. O funcionamento da PSO durante várias iterações é representado nas subfiguras 3a, 3b, 3c e 3d. Em 3a, na primeira iteração, 20 partículas são inicializadas, de forma aleatória, no espaço de busca; em 3b, na iteração de número 40, as partículas são guiadas pelas experiências social e individual adquiridas ao longo das iterações anteriores; em 3c, na iteração 80, as partículas estão se movendo para uma posição que apresenta melhor valor de *fitness* e se aproximando da solução ótima; e, em 3d, na iteração 120, quase todas as partículas atingiram a melhor solução do espaço de busca. No experimento apresentado, o algoritmo convergiu na iteração 193.

Figura 3 – Comportamento exploratório da PSO na busca para encontrar a solução que representa o ótimo global



Fonte: Autoria própria (2018).

3

Conceito de Serendipidade: Uma breve introdução

O que existe em comum entre o raio-X, o *nylon* e a vacina? Uma possível resposta para tal questionamento é que todos esses elementos foram descobertos por acaso, ou seja, por meio de serendipidade. Uma frase frequentemente encontrada na literatura relacionada a isso é de Louis Pasteur: "A sorte favorece a mente bem preparada".

Em 1754, em carta destinada a um amigo, o romancista Horace Walpole cunhou o termo "serendipidade". Na carta, falou de uma descoberta fundamental que havia realizado. Segundo ele, ela era baseada em um conto persa do século XVI, intitulado *The three princes of Serendip*. Naquela história, por serem observadores e sagazes, os príncipes descobriram a solução para determinado problema. Mas, para isso, eles precisaram observar e combinar um conjunto de eventos fortuitos.

O conceito de serendipidade já foi estudado em diversos domínios, como telecomunicações, ciências sociais, medicina, marketing, direito penal, sistemas de recomendação e outros. Na ciência, um exemplo bastante conhecido de serendipidade é a descoberta da penicilina por Alexander Fleming, em 1928. Enquanto limpava seu laboratório, Fleming, por acaso, percebeu que o fungo da penicilina havia contaminado um de seus experimentos. A sagacidade surgiu no processo da descoberta porque, por muitos anos, Fleming vinha realizando experiências com as propriedades antibacterianas de substâncias comuns e, portanto, tinha conhecimento suficiente sobre fungos para discernir as implicações potenciais.

Há também a ideia de pseudosserendipidade, que consiste na procura por algo, no entanto a forma pela qual é descoberto é casual e imprevista. Em 1826, por exemplo, Charles Goodyear testou dezenas de substâncias misturadas à borracha, inclusive

o enxofre. Em uma dessas experiências, ele respingou uma parte da mistura entre a borracha e o enxofre na chapa quente de um fogão e percebeu que ela não fundia na madeira, por isso resolveu pesquisar mais sobre misturas com enxofre. Depois de vários experimentos, a borracha vulcanizada foi descoberta.

Campos e Figueiredo (2002) apresentaram o conceito de serendipidade formalmente a partir da identificação de diferentes categorias. Para isso, foram utilizadas equações lógicas, chamadas "equações de serendipidade", para apresentar quatro eventos que podem gerá-la. Esses eventos são associados a diferentes tipos de serendipidade, sendo eles: pseudosserendipidade; serendipidade real; serendipidade sem uma metáfora de inspiração; e serendipidade com o conhecimento incorreto. A formalização de pseudosserendipidade é apresentada na Equação 8:

$$\begin{matrix} P_1 \subset (KP_1) \\ M \subset (KM) \end{matrix} \quad \Rightarrow \quad S_1 \subset (KP_1, KM, KN), \tag{8}$$

onde P_1 é um problema criado, KP_1 é o domínio do conhecimento no qual P_1 foi criado, M é uma metáfora inspiradora, KM é o domínio do conhecimento de M, S_1 é a solução para P_1 – no domínio do conhecimento $(KP_1, KM, KN) - KN$ é o conhecimento adicional adquirido durante o processo de formulação e de resolução de P_1. Já a serendipidade real é formalizada de acordo com a Equação 9:

$$\begin{matrix} P_1 \subset (KP_1) \\ M \subset (KM) \end{matrix} \quad \Rightarrow \quad \begin{matrix} P_2 \subset (KP_2) \\ S_2 \subset (KP_2, KM, KN), \end{matrix} \tag{9}$$

onde P_2 é um novo problema inspirado a partir de M, KP_2 é o domínio de conhecimento do novo problema, e S_2 é a solução para P_2, a qual é reconhecida simultaneamente com a identificação de P_2. Serendipidade também pode ocorrer sem inspirar-se em uma metáfora M:

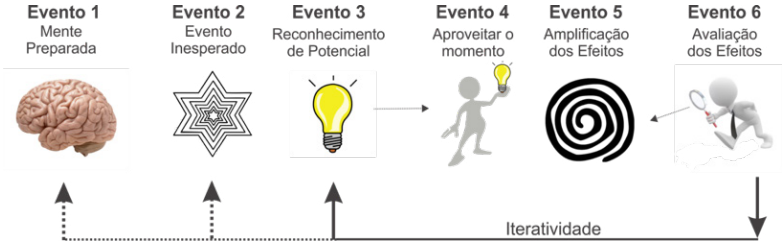
$$P_1 \subset (KP_1) \quad \Rightarrow \quad \begin{matrix} P_2 \subset (KP_2) \\ S_2 \subset (KP_2, KN). \end{matrix} \tag{10}$$

Por fim, a metáfora também pode ocorrer quando o conhecimento é representado de forma incorreta. Considerando a representação do conhecimento incorreto EP_1, tem-se que:

$$P_1 \subset (KP_1, EP_1) \quad \Rightarrow \quad \begin{matrix} P_2 \subset (KP_2) \\ S_2 \subset (KP_2, KN). \end{matrix} \tag{11}$$

Lawley e Tompkins (2008) propuseram um modelo perceptivo que considera seis eventos para garantir a existência da serendipidade clássica ou da pseudosserendipidade, como ilustrado na Figura 4. São eles: uma “mente preparada”, um evento não planejado e inesperado, o reconhecimento da potencialidade de um significado no futuro, uma ação para ampliar a potencialidade do efeito considerado no Evento 2, efeitos que podem ser utilizados para amplificar ainda mais o benefício do Evento 2 e ocorrência de serendipidade.

Figura 4 – Modelo perceptivo de serendipidade



Fonte: adaptado de Lawley e Tompkins (2008).

Ao contrário de muitas definições tradicionais que consideram "serendipidade" apenas como um sinônimo de "acaso", o termo está mais perto de uma combinação entre "sagacidade" e "acaso" (CATELLIN, 2014). Muitas descobertas são feitas por indivíduos que "veem pontes onde outros veem abismos". Nesse contexto, *ver pontes* consiste em fazer a combinação de eventos baseados em determinado significado, ao invés de simplesmente associá-los a algo casual.

Dentre as duas importantes dimensões que devem ser consideradas, "acaso" representa um conjunto formado por ideias e intuições que, normalmente, não são combinadas. Para alguns filósofos, essa dimensão é vista como uma relação capaz de criar um potencial, a partir do qual pode surgir algo novo. Já a dimensão "sagacidade" consiste no ambiente em que esse potencial pode ser aproveitado. Sem essa segunda dimensão, serendipidade não pode ser viabilizada.

4

Serendipidade no contexto da Inteligência de Enxames

Neste capítulo, é apresentada a formalização do conceito de serendipidade no contexto da inteligência de enxames, especificamente, na PSO. Além disso, também é apresentada uma nova variante baseada na adaptação do conceito de serendipidade. A nova variante possui a capacidade de equilibrar a busca local e a busca global durante o processo de otimização.

4.1 ADAPTAÇÃO E FORMALIZAÇÃO DO CONCEITO DE SERENDIPIDADE

No capítulo anterior, o conceito de serendipidade foi apresentado como uma capacidade de realizar descobertas afortunadas por acaso e de forma sagaz. Neste livro, o termo "acaso" deve ser entendido como uma possibilidade de algo acontecer, enquanto "sagacidade" denota a qualidade de perceber determinado evento ou algo relacionado a ele.

Tal conceito é interessante e útil porque ele é o principal indicativo para definir onde a serendipidade pode ser aplicada em um algoritmo de inteligência de enxames, além de qual tipo de sagacidade deve ser utilizado para melhorar o comportamento desse algoritmo, por meio de decisões inspiradas em serendipidade.

No contexto dos algoritmos de inteligência de enxames, em um determinado 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 determinada iteração. Portanto, 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. i, cujo va-

lor da função objetivo é o melhor encontrado na iteração j, é representado por $e_{ij}^{*} \in E$. Então, na iteração j, quando um elemento de P não pertence ao conjunto E, ele é considerado uma solução *ocasional*. Portanto, o conjunto que possui os membros utilizados para representar as soluções ocasionais na iteração j é dado pelo complementar do conjunto E em relação a P, isto é, os elementos que P possui a mais que E, de acordo com a Equação 12:

$$ACASO = \mathrm{C}_{\mathrm{P}}^{\mathrm{E}} = P - E. \quad (12)$$

Outro conjunto a ser considerado na formalização de serendipidade é o SRD. Ele é formado por elementos chamados *serendípetos* e é dado pela seguinte equação:

$$SRD = ACASO \cap SAGAZ \quad (13)$$

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 cujos valores de *fitness* são melhores que o valor de *fitness* do elemento e_{ij}^{*}, conforme a Equação 14:

$$SAGAZ = \{sagaz_{ij} \mid fit\,(sagaz_{ij}) < fit\,(e_{ij}^{*})\} \quad (14)$$

Após a adaptação e a formalização do conceito de serendipidade, a fim de implementá-lo no contexto da inteligência de enxames, escolheu-se a meta-heurística PSO para desenvolvimento de uma nova variante. O que justifica a escolha do algoritmo são as características que o tornaram uma das meta-heurísticas mais populares da área de otimização.

Há três abordagens que podem ser utilizadas para implementar esse conceito. Na PSO, a primeira delas é baseada em modificações nos pontos de decisões, de acordo com as linhas 2, 8, 11, 15 e 16 do Algoritmo 1. A segunda é baseada na inclusão de um método complementar para implementar um algoritmo baseado em serendipidade. A terceira abordagem é uma combinação das outras duas. Todas elas consideram as "descobertas afortunadas" como pontos no espaço de busca, que são úteis para um processo de otimização, uma vez que eles possuem valores da função objetivo melhores do que a melhor solução encontrada até a iteração atual.

Uma vez que a terceira abordagem atende às duas dimensões fundamentais do conceito de serendipidade, acaso e sagacidade, este livro adota uma variante PSO, propondo uma alternativa para gerar serendipidade (PAIVA; COSTA; SILVA, 2016), que combina duas estratégias utilizadas no domínio de sistemas de recomendação (TOMS, 2000): *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 a exploração do espaço de busca, gerando diversidade adicional. O princípio de Pasteur é empregado para reconhecer potenciais e usar percepções para aproveitar o momento. Ele combina partículas escoteiras com a detecção de comportamentos, a fim de inspecionar regiões no espaço de busca inexploradas pelo enxame. Essa estratégia é a implementação do conceito de sagacidade ou, em outras palavras, o conceito de "mente preparada".

4.2 OTIMIZAÇÃO POR ENXAMES DE PARTÍCULAS USANDO SERENDIPIDADE

Para validar a adaptação do conceito de serendipidade no contexto da inteligência de enxames e, posteriormente, avaliar os resultados encontrados, este capítulo apresenta a implementação de uma nova variante da PSO chamada *Serendipity-Based Particle Swarm Optimization* (SBPSO), na qual as estratégias *blind luck* e princípio de Pasteur são empregadas.

A nova variante utiliza partículas adicionais cujo comportamento difere do comportamento típico do enxame. Elas são chamadas de partículas escoteiras, cujo papel é gerar diversidade adicional para o algoritmo, a fim de aumentar a sua capacidade de exploração. No entanto, é importante assegurar que a inclusão delas não comprometerá o comportamento habitual do enxame. Na variante SBPSO, isso é transparente porque elas são utilizadas para identificar as soluções que apresentam valores da função objetivo melhores que o da melhor partícula do enxame.

A velocidade de uma partícula escoteira k, representada por V_{kd} , é dada pela Equação 15, que é uma adaptação da que foi apresentada na implementação da variante *Discrete Particle Swarm Optimization* (DPSO) (WU et al, 2013):

$$V_{kd} = w \cdot V_{kd} + c_3 \cdot r_3 \, (X_{kd} - x_{best}), \tag{15}$$

onde c_3 é o coeficiente de geração de diversidade, e r_3 é um valor aleatório uniformemente distribuído no intervalo [0, 1], gerado a cada iteração, por meio de uma função densidade de probabilidade. Os termos w, X_{kd} e x_{best} são, respectivamente, o fator de inércia, a posição da partícula escoteira k e a posição da melhor partícula do enxame até o momento. A posição de uma partícula escoteira

é dada pela Equação 16:

$$X_{kd} = -x_{best} + V_{kd}, \quad (16)$$

onde x_{best} é a posição da melhor partícula do enxame até o momento.

Os passos iniciais da SBPSO, mostrados no Algoritmo 2, são similares aos da PSO. A velocidade e a posição de cada partícula do enxame e de cada escoteira são inicializadas aleatoriamente (linhas 01 e 02). A seguir, para cada partícula, o valor de *fitness* é calculado para encontrar a melhor delas (linhas 04-10). Isso também ocorre para as escoteiras (linhas 11-17). Depois que as melhores partículas, do enxame e escoteira, são encontradas, a melhor delas se tornará a nova $gBest$ (linhas 19-25).

O próximo passo é iniciar o processo de inspeção nas proximidades da $gBest$ (linhas 27-28). Para isso, o algoritmo cria, de forma aleatória, um ponto adjacente (AP), de acordo com a Equação 17:

$$AP = best + \alpha \cdot R, \quad (17)$$

onde $best$ é a partícula $gBest$ atual, α é uma constante real positiva, R é um vetor com n elementos aleatórios distribuídos uniformemente no intervalo [-1, 1], e n é a dimensão de $best$.

Então, para o AP que foi criado, o algoritmo define um vetor $\vec{d}$ que representa o segmento orientado $\overline{bestAP}$, de acordo com a Equação 18:

$$\vec{d} = AP - best, \quad (18)$$

onde $best$ e AP foram apresentados na Equação 17.

Algoritmo 2 – Pseudocódigo da variante SBPSO usando partículas escoteiras.

01: inicializa velocidade e posição do enxame
02: inicializa velocidade e posição das partículas escoteiras
03: **enquanto** *critério de parada não satisfeito* **faça**
04: **para cada** *partícula i em S* **faça**
05: calcula o valor de $fitness$
06: **se** $fit(x_{id}) < fit(pBest_i)$ **então**
07: $pBest_i \leftarrow x_{id}$
08: **fim se**
09: **fim para**
10: $best_{enxame} \leftarrow identificaMelhorParticulaEnxame\ (\ )$
11: **para cada** *escoteira k em S* **faça**
12: calcula o valor de $fitness$
13: **se** $fit(escoteira_{kd}) < fit(pBest_k)$ **então**
14: $pBest_k \leftarrow escoteira_{kd}$
15: **fim se**
16: **fim para**
17: $best_{escoteira} \leftarrow identificaMelhorParticulaEscoteira\ (\ )$
18:
19: **se** $fit(best_{enxame}) < fit(best_{escoteira})$ **e** $fit(best_{enxame}) < fit(gBest)$ **então**
20: $gBest \leftarrow best_{enxame}$
21: **senão**
22: **se** $fit(best_{escoteira}) < fit(best_{enxame})$ **e** $fit(best_{escoteira}) < fit(gBest)$ **então**
23: $gBest \leftarrow best_{escoteira}$
24: **fim se**
25: **fim se**
26:
27: cria Ponto Adjacente a $gBest$ usando Equação 17
28: define um vetor $\vec{d}$ usando Equação 18
29: executa ações definidas na Tabela 1
30:
31: **para cada** *partícula i em S* **faça**
32: calcula velocidade usando Equação 4
33: atualiza posição usando Equação 5
34: **fim para**
35: **para cada** *escoteira k em S* **faça**
36: calcula velocidade usando Equação 15
37: atualiza posição usando Equação 16
38: **fim para**
39:
40: **se** *enxame estagnou* **e** *enxame convergiu* **então**
41: substitui a pior partícula escoteira pela pior partícula do enxame
42: **fim se**
43: **fim enquanto**

Uma vez que $best$ e AP definem um vetor simples $\vec{d}$, que representa uma direção, existem infinitos Pontos de Inspeção (IP) cuja direção $\overline{bestIP}$ é a mesma de $\overline{bestAP}$. Para o AP criado, dois IP podem ser escolhidos, de acordo com a Equação 19:

$$IP = best \pm \lambda \cdot \vec{d}, \tag{19}$$

onde λ é uma constante real não nula.

Depois de definir os dois IP, um conjunto de regras é avaliado (linha 29). Sua implementação tem o objetivo de tratar os padrões de comportamento para definir pontos de inspeção nas proximidades da $gBest$ da iteração corrente. Essas novas soluções podem ser obtidas a partir da geração de IP, de acordo com as regras apresentadas na Tabela 1.

Mais uma vez, os passos da SBPSO são similares aos da PSO. A velocidade das partículas do enxame é calculada e a posição é atualizada (linhas 32 e 33). Isso também ocorre para as escoteiras (linhas 36 e 37).

Finalmente, o algoritmo avalia a situação do enxame em relação à estagnação e à convergência (linhas 40-42). Quando a condição apresentada na Equação 20 é satisfeita, a SBPSO substitui a partícula escoteira que apresenta o pior valor de *fitness* pela pior partícula do enxame.

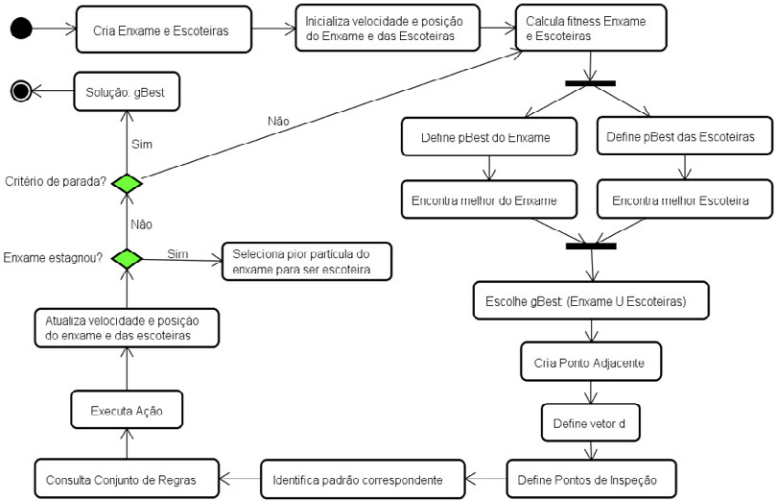
$$(\varphi_j < \delta_{stag}) \wedge (cont \geq \delta_{conv}) \tag{20}$$

onde φ_j é o raio do enxame na iteração j e δ_{stag} é um valor limite que avalia a estagnação dele, de acordo com a Equação 6. O número de iterações nas quais o enxame não apresenta melhorias significativas é denotado por $cont$ e δ_{conv} é o valor limite que indi-

ca a convergência, conforme a descrição da Equação 7.

Outra alternativa para apresentar o SBPSO é utilizando um diagrama da *Unified Modeling Language* (UML), bastante conhecido e chamado diagrama de atividades. Em sua forma básica, ele ilustra o que ocorre em um fluxo de trabalho, as atividades que podem ser realizadas em paralelo e a existência de caminhos alternativos no fluxo de execução. A Figura 5 é um diagrama de atividades que foi utilizado para representar a variante SBPSO.

Figura 5 – Diagrama de Atividades do algoritmo SBPSO



Fonte: Autoria própria (2018).

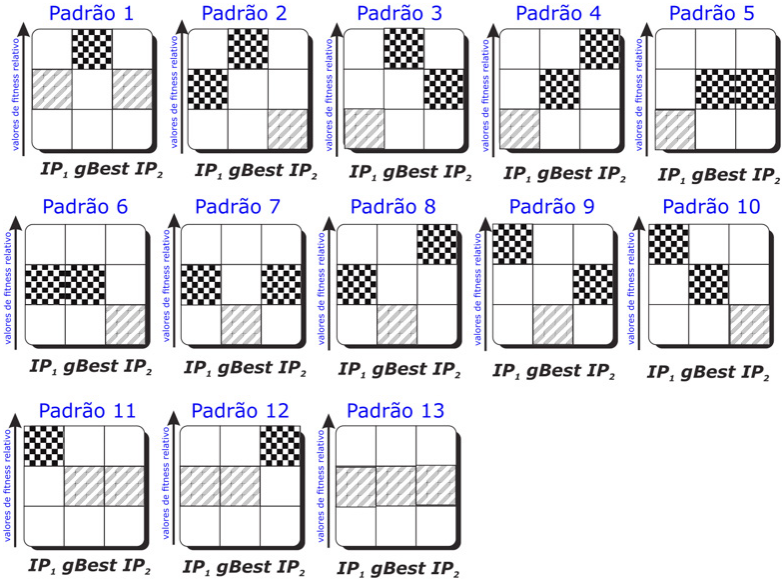
De acordo com o princípio de Pasteur, “a sorte favorece a mente bem preparada”. Neste livro, uma “mente bem preparada” é implementada utilizando um conjunto de padrões de comportamento, os quais são considerados para a dimensão sagacidade. Eles podem ser utilizados para representar o comportamento do

valor de *fitness* ao longo dos pontos em uma reta, que passam pela partícula $gBest$ (dentro do domínio da função objetivo), durante uma iteração j. Eles consideram várias possibilidades no formato de matrizes, por exemplo, uma matriz M_3.

Para mostrar um exemplo simplificado, um conjunto de padrões de comportamento é representado por meio de matrizes de ordem 3. A fim de implementar a dimensão sagacidade, são utilizados 13 padrões que representam o comportamento dos valores de *fitness* da função objetivo. A Figura 6 mostra cada um dos padrões, considerando os valores relacionados a três pontos que passam sobre uma reta que liga $gBest$ a AP. Os três pontos são IP_1, $gBest$ e IP_2.

Cada padrão, representado pelas matrizes da Figura 6, corresponde a um comportamento identificado após a definição dos dois pontos de inspeção. Os elementos que estão em destaque na matriz (hachurado e quadriculado) representam o valor de *fitness* calculado para IP_1, $gBest$ e IP_2. O(s) elemento(s) hachurado(s) corresponde(m) ao(s) menor(es) valor(es) de *fitness* calculado(s). No Padrão 1, por exemplo, os valores de *fitness* de IP_1 e de IP_2 são iguais, porém menores que o do $gBest$. No Padrão 2, é observado que o valor do IP_2 é menor que os de IP_1 e de $gBest$. Já no Padrão 3, o valor do IP_1 é o menor entre os três pontos. Outro comportamento é representado pelo Padrão 8, considerando a situação em que, embora tenham sido definidos pontos de inspeção, o valor de *fitness* do $gBest$ permanece o menor. Os outros padrões representam as situações que não foram exemplificadas aqui.

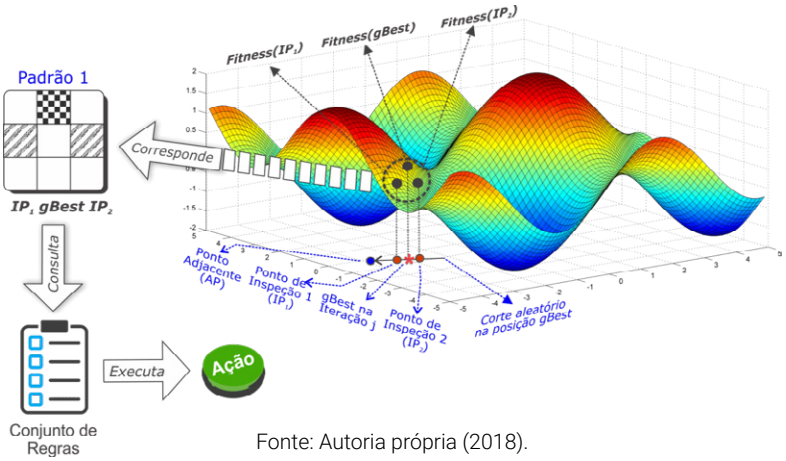
Figura 6 – Padrões de comportamento utilizados para implementar a dimensão sagacidade



Fonte: Autoria própria (2018).

Quando o algoritmo identifica um dos 13 padrões predefinidos, um conjunto de regras é consultado, a fim de determinar qual ação deve ser executada. A Figura 7 ilustra a situação em que o Padrão 1 foi identificado. Quando o conjunto de regras é consultado (verificar Tabela 1), o Padrão 1 pode ser representado pela Regra 3. Nessa situação, como os valores de *fitness* de IP_1 e de IP_2 são iguais, é realizado um sorteio para, assim, um dos dois IP ser definido como o novo $gBest$.

Figura 7 – Padrões de comportamento utilizados para implementar a dimensão sagacidade



Fonte: Autoria própria (2018).

Tabela 1 – Conjunto de regras para tratar padrões de comportamento na definição de pontos de inspeção próximos à partícula *gBest*

Regra	Condição	Padrão	Ação
01	$fit(IP_1) < fit(IP_2) \wedge fit(IP_2) < fit(gBest)$	3	$gBest \leftarrow IP_1$
02	$fit(IP_2) < fit(IP_1) \wedge fit(IP_1) < fit(gBest)$	2	$gBest \leftarrow IP_2$
03	$fit(IP_1) == fit(IP_2) \wedge fit(IP_1) < fit(gBest)$	1	$gBest \leftarrow sorteia\ (IP_1, IP_2)$
04	$fit(IP_1) < fit(gBest) \wedge fit(gBest) \leq fit(IP_2)$	4 e 5	Idem para a regra 01
05	$fit(IP_2) < fit(gBest) \wedge fit(gBest) \leq fit(IP_1)$	6 e 10	Idem para a regra 02
06	$fit(gBest) \leq fit(IP_1) \wedge fit(IP_1) \leq fit(IP_2)$	7, 8, 12 e 13	*gBest* permanece
07	$fit(gBest) \leq fit(IP_2) \wedge fit(IP_2) < fit(IP_1)$	9 e 11	*gBest* permanece

5

Funções de Referência

É comum o emprego de funções de referência com o pressuposto de que a dificuldade delas corresponde às encontradas em aplicações do mundo real. Por conta disso, elas são usadas para validar e comparar a performance de algoritmos de otimização.

Segundo Dieterich e Hartke (2012), qualquer nova abordagem para otimização global deve ser validada a partir de um conjunto de funções de referência. No entanto, para que os novos algoritmos de otimização sejam avaliados imparcialmente, é importante que essas funções possuam diversas propriedades, como:

1. **Continuidade** – uma função f é contínua em $x = a$, se e somente se

   $$\lim_{x \to a} f(x) = f(a);$$

2. **Diferenciabilidade** – se x_0 é um ponto do domínio de uma função f, então f é diferenciável em x_0 se existe a derivada $f'(x_0)$;

3. **Separabilidade** – é uma medida que pode ser utilizada para considerar a dificuldade de uma função. A condição geral de separabilidade é dada por

   $$\frac{\partial f(\bar{x})}{\partial x_i} = g(\bar{x}_i)\, h(\bar{x})$$

   onde $g(\bar{x}_i)$ significa qualquer função apenas do x_i, e $h(\bar{x})$ significa qualquer função de qualquer $\bar{x}$;

4. **Escalabilidade** – uma função é considerada escalável quando ela pode ser expandida;

5. **Modalidade** – uma função pode ser unimodal ou multimodal. A unimodal possui apenas um vale (quando o processo de otimização é aplicado em um problema de minimização) ou um pico (quando o problema é de maximização). As multimodais possuem mais de um vale (ou pico).

Para realização dos experimentos, 16 funções de referência, que são frequentemente aplicadas em problemas de minimização, foram escolhidas. Elas podem ser utilizadas para investigar a estagnação e a convergência dos algoritmos de otimização e vêm sendo aplicadas em vários estudos de inteligência de enxames.

As funções usadas nos experimentos são descritas a seguir e as Figuras 8-23 apresentam os mapas 3D correspondentes a cada uma delas:

1. **Função Esfera** – é caracterizada por ser convexa e unimodal.

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

2. **Função Rosenbrock** – possui o mínimo global em um vale parabólico.

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

3. **Função Griewank** – possui vários mínimos locais regularmente distribuídos.

$$f_3(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

4. **Função Rastrigin** – é uma função altamente multimodal, no entanto, as localizações dos vários mínimos locais são regularmente distribuídas.

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

5. **Função Ackley** – há vários mínimos locais, e a função tem uma região externa quase plana e um grande orifício no centro.

$$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

6. **Função Noisy Quartic** – é uma função unimodal simples e preenchida com um ruído *gaussiano*.

$$f_6(x) = \sum_{i=1}^{d} x_i^4 + rand(0,1)$$

7. **Função Zakharov** – é uma função unimodal simples, que possui forma de placa.

$$f_7(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} + 0.5i\, x_i\right)^2 + \left(\sum_{i=1}^{d} + 0.5i\, x_i\right)^4$$

8. **Função Hiper Elipsoide Rotacionada** – é uma função unimodal e convexa, cuja solução ótima se encontra em uma região do espaço bastante restrita. Ela é uma extensão da função Soma dos Quadrados.

$$f_8(x) = \sum_{i=1}^{d} \sum_{j=1}^{i} x_j^2$$

9. **Função Alpine** – é uma função multimodal com apenas um ótimo global. Em relação à origem, a superfície do espaço de busca não é completamente simétrica.

$$f_9(x) = \sum_{i=1}^{d} |x_i sen(x_i) + 0.1x_i|$$

10. **Função Weierstrass** – é uma função contínua em todos os lugares, porém nenhum deles é diferenciável.

$$f_{10}(x) = \sum_{i=1}^{d} \left(\sum_{k=0}^{kmax} [\alpha^k cos(2\pi\beta^k (x_i + 0.5))] \right) - d \sum_{k=0}^{kmax} [\alpha^k cos(2\pi\alpha^k \cdot 0.5))]$$

onde $\alpha = 0.5$, $\beta = 3$ e $kmax = 20$.

11. **Função Salomon** – é uma função não separável e altamente multimodal. Sua superfície se assemelha a uma lagoa com várias ondulações.

$$f_{11}(x) = 1 - cos\left(2\pi \sqrt{\sum_{i=1}^{d} x_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{d} x_i^2}$$

12. **Função Csendes** – é uma função multimodal definida como segue:

$$f_{12}(x) = \sum_{i=1}^{d} x_i^6 \left[2 + sen\left(\frac{1}{x_i}\right)\right]$$

13. **Função de Xin-She Yang 01** – é uma função estocástica genérica multimodal e é dada de acordo com a formulação a seguir:

$$f_{13}(x) = \sum_{i=1}^{d} rand(0,1) \cdot |x_i|^i$$

14. **Função Soma dos Quadrados** – essa função não possui mínimo local, exceto o global. Ela é contínua e convexa.

$$f_{14}(x) = \sum_{i=1}^{d} ix_i^2$$

15. **Função de Schumer Steiglitz** – é uma função que não possui mínimos locais, apenas o mínimo global.

$$f_{15}(x) = \sum_{i=1}^{d} x_i^4$$

16. **Função de Powell Sum** – é uma função unimodal definida conforme formulação a seguir:

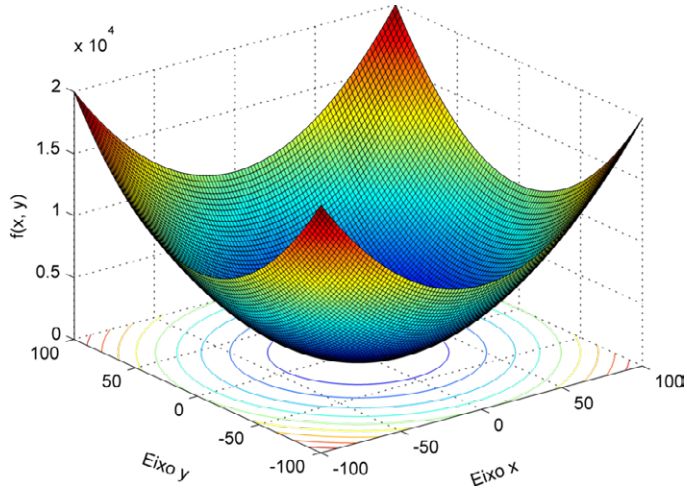
$$f_{16}(x) = \sum_{i=1}^{d} |x_i|^{i+1}$$

Para cada uma das funções apresentadas, a Tabela 2 mostra as fronteiras do espaço de busca e o valor que representa o ótimo global. Além disso, também são apresentadas as suas características no que diz respeito às propriedades de continuidade (CON), diferenciabilidade (DIF), separabilidade (SEP), escalabilidade (ESC) e modalidade (MOD).

Tabela 2 – Características das funções de referência avaliadas

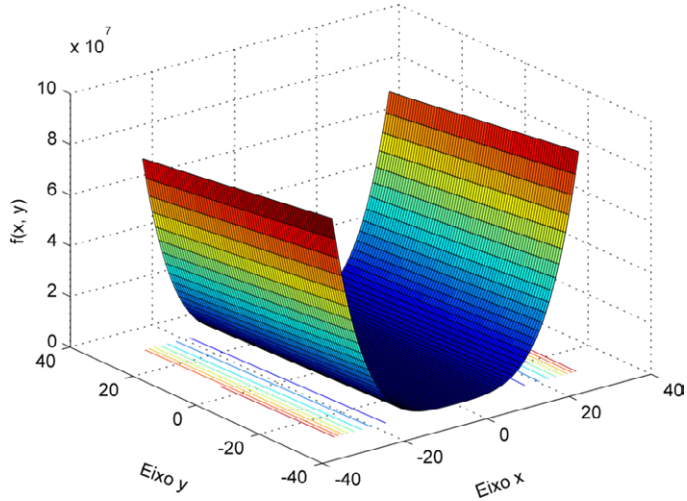
Função	Espaço de Busca	Ótimo Global	CON	DIF	SEP	ESC	MOD
f_1	$-100 \leq x_i \leq 100$	0	●	●	●	●	Uni
f_2	$-30 \leq x_i \leq 30$	0	●	●		●	Uni
f_3	$-600 \leq x_i \leq 600$	0	●			●	Multi
f_4	$-5.12 \leq x_i \leq 5.12$	0	●	●		●	Multi
f_5	$-32.76 \leq x_i \leq 32.76$	0	●	●		●	Multi
f_6	$-1.28 \leq x_i \leq 1.28$	0	●	●		●	Multi
f_7	$-5 \leq x_i \leq 10$	0	●	●		●	Multi
f_8	$-65.53 \leq x_i \leq 65.53$	0	●	●	●	●	Uni
f_9	$-10 \leq x_i \leq 10$	0	●		●		Multi
f_{10}	$-5 \leq x_i \leq 5$	0	●	●	●	●	Multi
f_{11}	$-100 \leq x_i \leq 100$	0	●	●		●	Multi
f_{12}	$-1 \leq x_i \leq 1$	0	●	●	●	●	Multi
f_{13}	$-5 \leq x_i \leq 5$	0			●		Multi
f_{14}	$-5.12 \leq x_i \leq 5.12$	0	●	●	●	●	Uni
f_{15}	$-100 \leq x_i \leq 100$	0	●	●	●	●	Uni
f_{16}	$-500 \leq x_i \leq 500$	0	●	●	●	●	Uni

Figura 8 – Plotagem da função Esfera



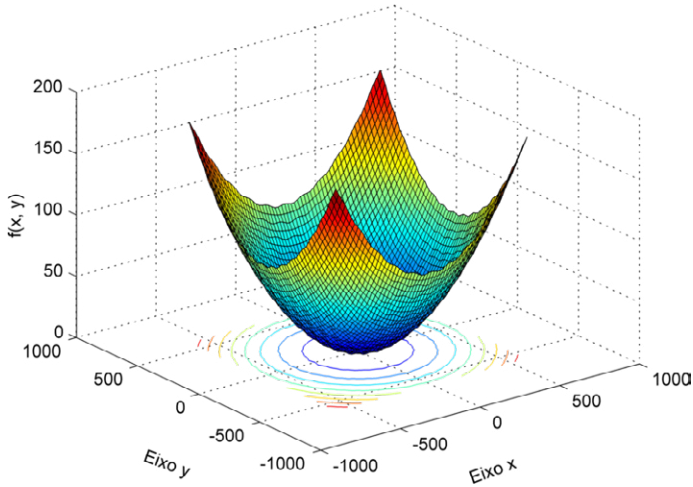
Fonte: Autoria própria (2018).

Figura 9 – Plotagem da função Rosenbrock



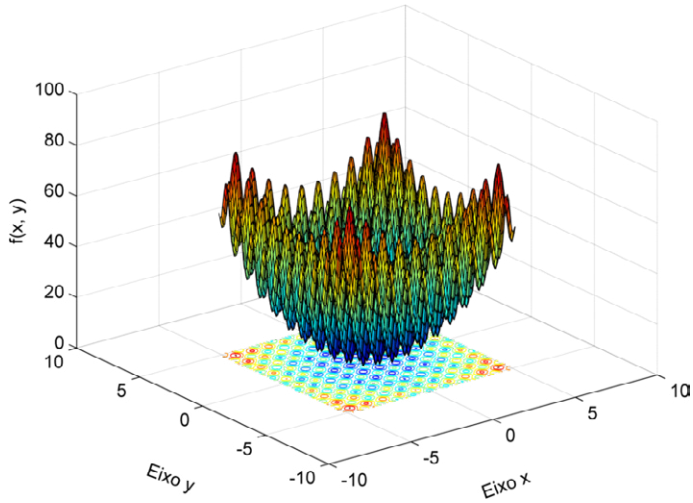
Fonte: Autoria própria (2018).

Figura 10 – Plotagem da função Griewank



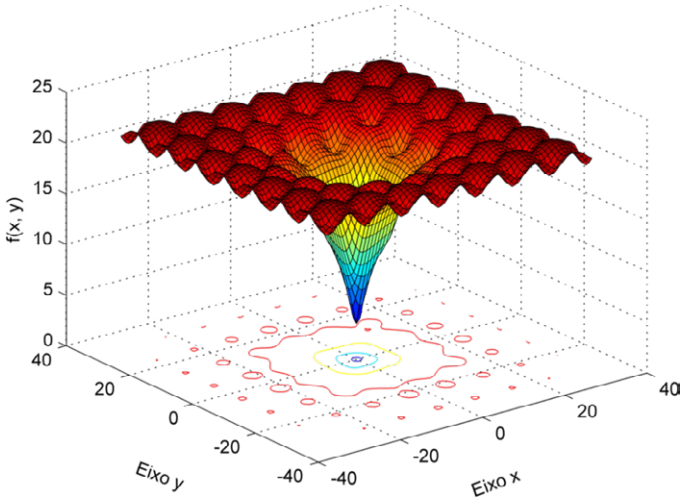
Fonte: Autoria própria (2018).

Figura 11 – Plotagem da função Rastrigin



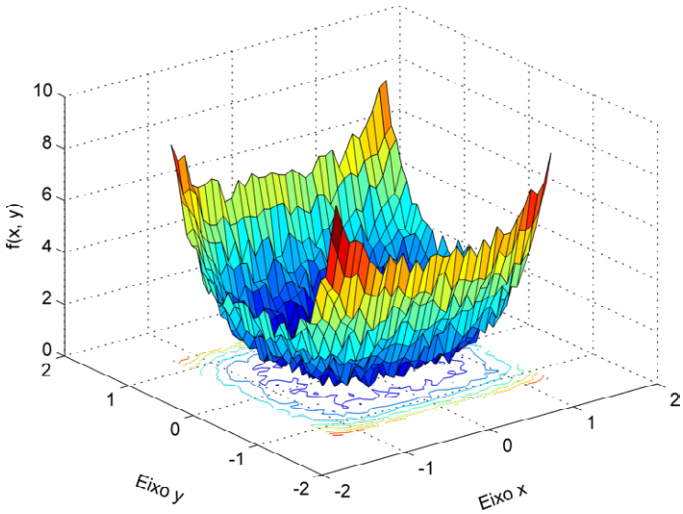
Fonte: Autoria própria (2018).

Figura 12 – Plotagem da função Ackley



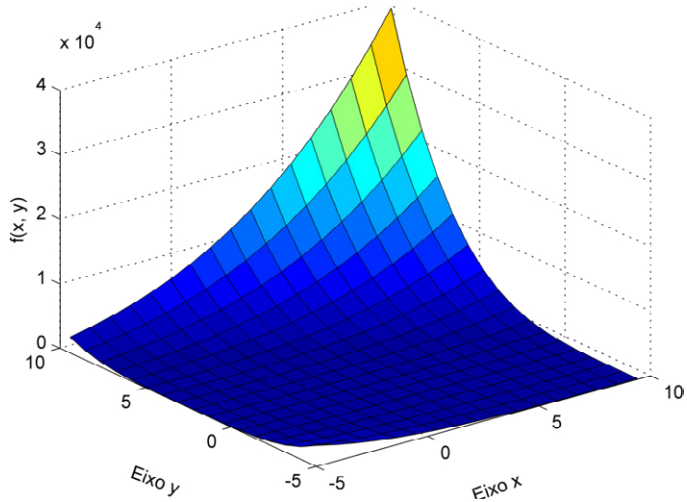
Fonte: Autoria própria (2018).

Figura 13 – Plotagem da Função Noisy Quartic



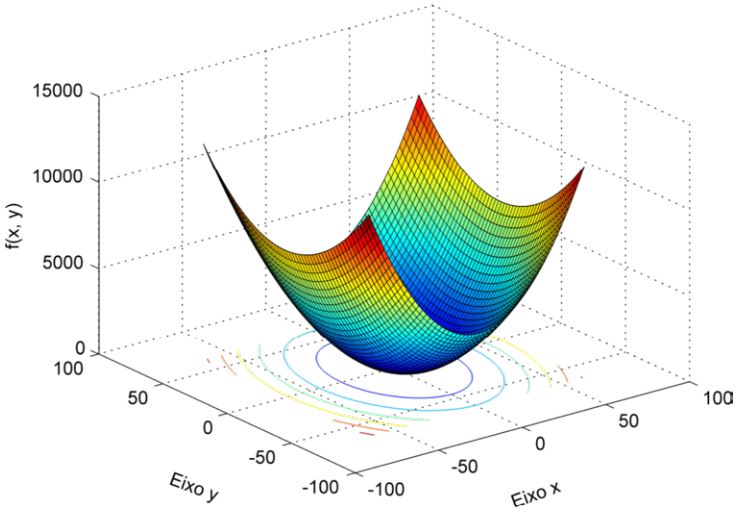
Fonte: Autoria própria (2018).

Figura 14 – Plotagem da função Zakharov



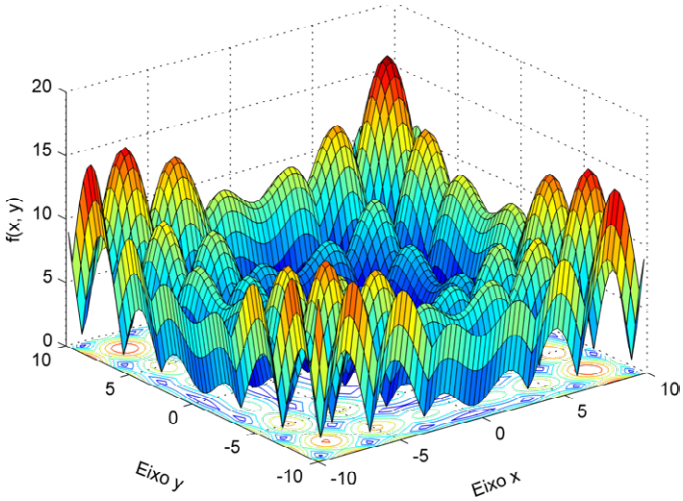
Fonte: Autoria própria (2018).

Figura 15 – Plotagem da função Hiper Elipsoide Rotacionada



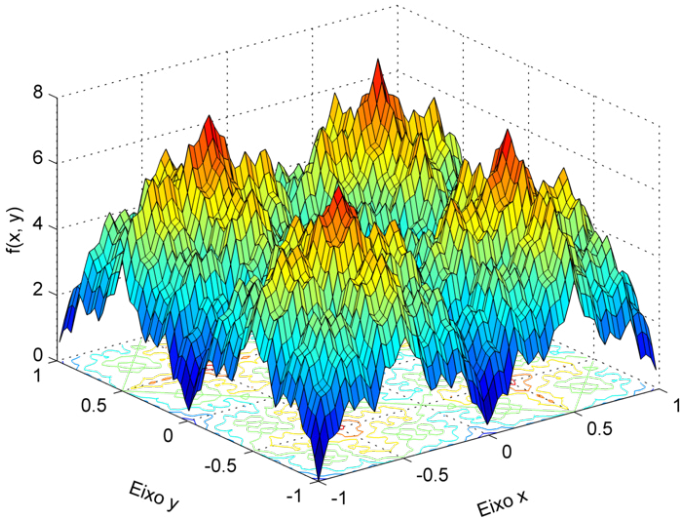
Fonte: Autoria própria (2018).

Figura 16 – Plotagem da função Alpine



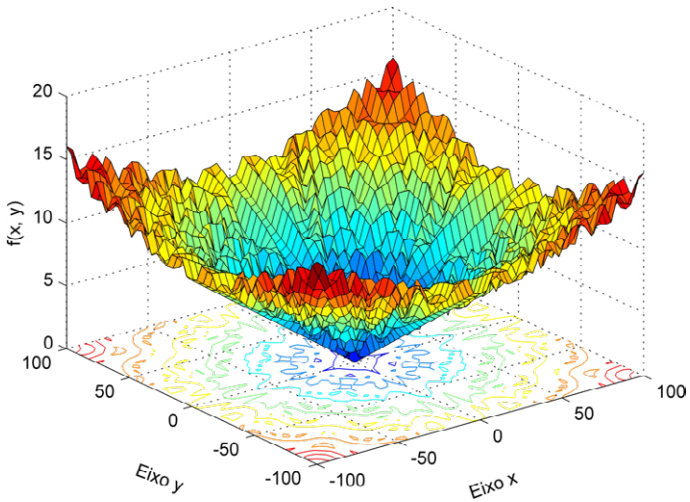
Fonte: Autoria própria (2018).

Figura 17 – Plotagem da função Weierstrass



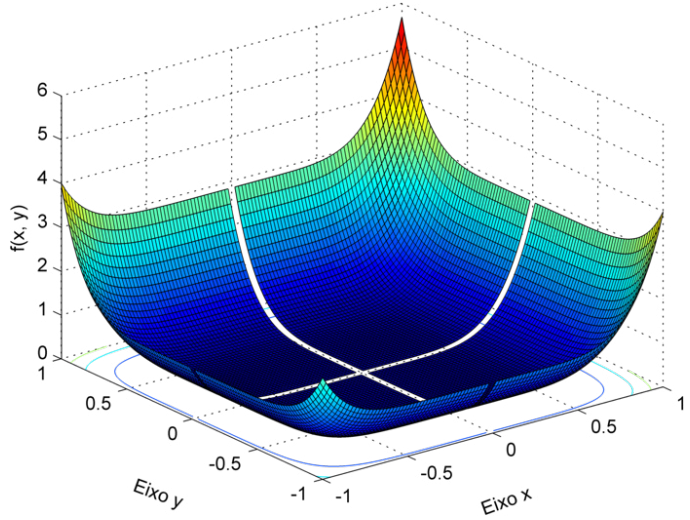
Fonte: Autoria própria (2018).

Figura 18 – Plotagem da função Salomon



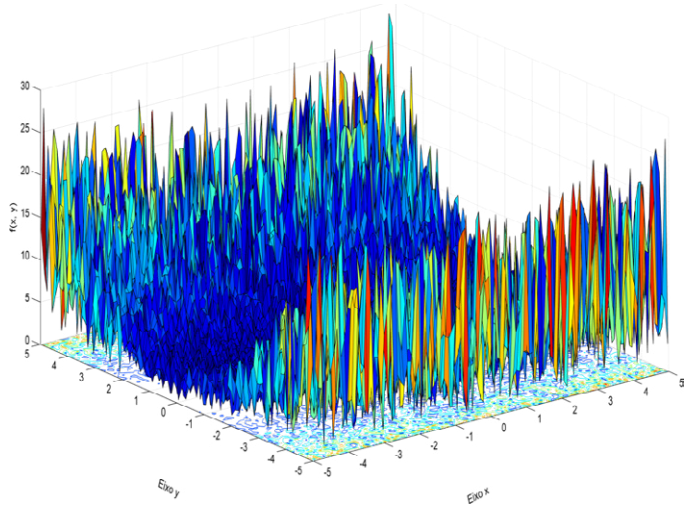
Fonte: Autoria própria (2018).

Figura 19 – Plotagem da função Csendes



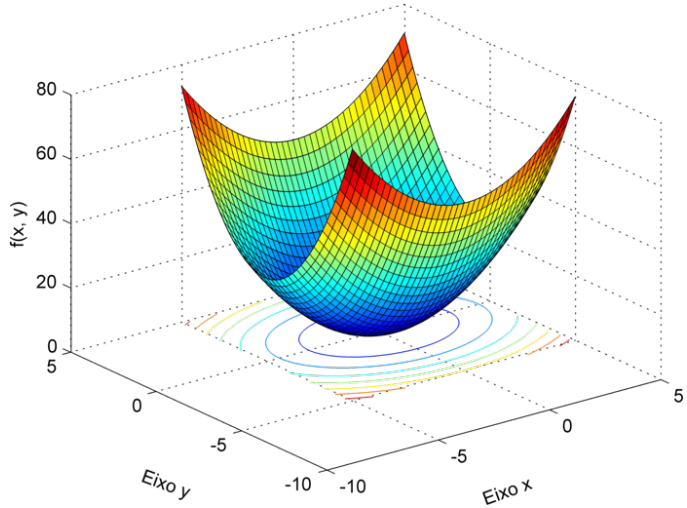
Fonte: Autoria própria (2018).

Figura 20 – Plotagem da função de Xin-She Yang 01



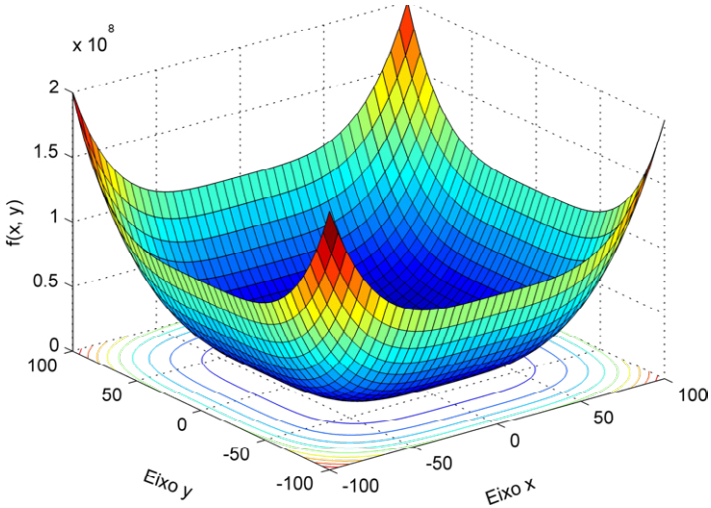
Fonte: Autoria própria (2018).

Figura 21 – Plotagem da função Soma dos Quadrados



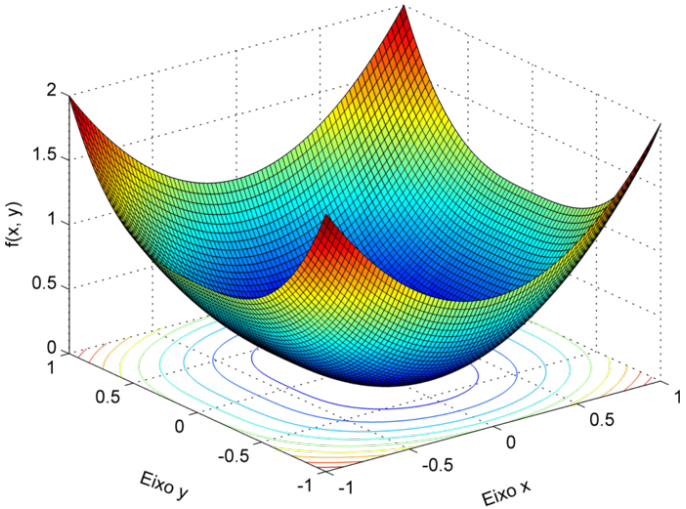
Fonte: Autoria própria (2018).

Figura 22 – Plotagem da função de Schumer Steiglitz



Fonte: Autoria própria (2018).

Figura 23 – Plotagem da função *Powell Sum*



Fonte: Autoria própria (2018).

6

Experimentos com a PSO baseada em Serendipidade

Neste capítulo, a variante PSO baseada em serendipidade é avaliada por meio de vários experimentos computacionais com o objetivo de analisar a sua performance. O capítulo está dividido em duas seções. Na Seção 6.1, os valores atribuídos aos parâmetros utilizados nos experimentos são apresentados. Na 7.2, a performance da SBPSO é comparada com a da PSO padrão e com as de algumas variantes encontradas na literatura.

6.1 CONFIGURAÇÃO DOS PARÂMETROS

Os valores associados aos parâmetros w, c_1 e c_2, definidos na Equação 4, podem variar a performance da PSO e de suas variantes. Para uma comparação adequada, todos os parâmetros comuns à PSO e às suas variantes foram definidos com o mesmo valor: a) $c_1 = c_2 = 2{,}0$; b) o peso de inércia, w, que decai linearmente iniciando em 0,9 e finalizando em 0,4; e c) a velocidade máxima, V_{max}, de cada partícula é definida como a metade do tamanho do espaço de busca em uma dimensão.

Existem outros parâmetros que são específicos da SBPSO, e os valores atribuídos a eles são apresentados a seguir. Os parâmetros estão relacionados a: quantidade e velocidade das partículas escoteiras; criação de pontos adjacentes e de inspeção, próximos à partícula $gBest$; e detecção de estagnação do enxame. Todos eles foram escolhidos empiricamente, depois de várias simulações que apresentaram bons resultados, os quais estão relacionados à qualidade e à estabilidade das soluções encontradas após a avaliação de todas as funções de referência utilizadas nos experimentos.

O parâmetro que define o número de partículas escoteiras corresponde a 10% do total de partículas do enxame. Outros parâmetros também foram apresentados nas equações 15, 17 e 19.

Eles foram definidos da seguinte maneira, respectivamente: $c_2 = 1{,}6$, α correspondendo a 1% do tamanho do espaço de busca em uma dimensão e $\lambda = 10^{-4}$. Quando valores inferiores a 10^{-4} foram atribuídos a λ, não se encontraram melhorias significativas no valor da média. Além disso, o tempo médio de execução do algoritmo foi comprometido.

Para detectar a convergência prematura, dois limites foram utilizados: δ_{stag} e δ_{conv}. O δ_{stag} foi definido em 10^{-3}, e o δ_{conv} corresponde a 5% do número total de iterações. O valor limite δ_{conv} foi usado para definir o número de iterações cujo valor de *fitness* da partícula $gBest$ não foi melhorado significativamente.

6.2 EXPERIMENTOS COMPUTACIONAIS

A performance do algoritmo proposto é avaliada por meio de dois conjuntos de experimentos que, aqui, foram chamados de Conjunto de Experimentos 1 e Conjunto de Experimentos 2. Para cada um deles, os resultados obtidos são discutidos separadamente.

No primeiro, são utilizadas quatro funções de referência para comparar a SBPSO, com a PSO e a PSO-Scout. Em seguida, SBPSO é comparada com quatro variantes da PSO, disponíveis na literatura. Já no segundo, outras doze funções de referência são utilizadas para comparar SBPSO, PSO e PSO-Scout.

PSO-Scout é outra variante que também foi implementada neste livro. Ela se caracteriza por utilizar um conjunto de partículas escoteiras. Elas são usadas como mecanismos para melhorar a performance da PSO e também como uma abordagem simples para incorporar conhecimentos específicos a fim de resolver problemas. Seu comportamento pode ser controlado usando as equações 15 e 16, apresentadas na Seção 4.

Como dito anteriormente, o número de partículas escoteiras utilizado nessa variante equivale a 10% do total das do enxame. Após vários experimentos, observou-se que a PSO-Scout, em algumas situações, pode melhorar a performance da PSO usando o comportamento exploratório das partículas escoteiras.

O teste de Wilcoxon (*signed rank test*), com nível de significância de 0,05, é utilizado para verificar se os resultados obtidos da comparação entre PSO-Scout e SBPSO são estatisticamente diferentes. A hipótese nula, H_0, indica que duas amostras vêm da mesma população, ao passo que a hipótese alternativa, H_1, indica que uma população tem valores maiores que a outra. Quando *p-value* é inferior ao nível de significância, então decide-se rejeitar H_0, ou seja, há uma diferença significativa entre as amostras.

Todas as rotinas utilizadas nos experimentos computacionais foram implementadas na linguagem de programação MATLAB (*matrix laboratory*). 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 para a realização dos experimentos.

6.2.1 CONJUNTO DE EXPERIMENTOS 1

Para investigar a escalabilidade da variante SBPSO nesse conjunto de experimentos, quatro funções de referência são usadas: Esfera (f_1), Rosenbrock (f_2), Griewank (f_3) e Rastrigin (f_4). Para cada uma delas, define-se nove configurações que consistem em variar o tamanho do enxame, a dimensionalidade do problema e o número máximo de iterações.

O tamanho do enxame é definido com 20, 40 e 80 partículas. Já a dimensionalidade do problema é definida em 10, 20 e 30 di-

mensões. Por último, a variação do número máximo de avaliações da função objetivo é definida em 1.000, 1.500 e 2.000 iterações. Após os experimentos, a média do valor de *fitness* e o desvio-padrão obtidos são registrados.

As Figuras 24-27 representam os processos de convergência médios da PSO, PSO-Scout e SBPSO, medidos em um enxame com 20 partículas, em 30 dimensões, durante 2.000 iterações, após 100 execuções independentes. Em todas as simulações, ficou evidente a superioridade da velocidade de convergência da SBPSO em relação à PSO e à PSO-Scout. As otimizações foram realizadas nos espaços de busca das funções f_1, f_2, f_3 e f_4. Os valores obtidos ao fim desses experimentos estão apresentados nas Tabelas 3-6.

Na Figura 24, função Esfera, observa-se que a velocidade de convergência da SBPSO é bastante superior à PSO e à variante PSO-Scout. Embora a solução ótima não tenha sido encontrada pela SBPSO durante as 2.000 iterações, ela atingiu um valor médio de *fitness* bastante significativo: 2.9522E-156. Na função Rosenbrock, nenhuma das três variantes atingiu o ponto ótimo, conforme a Figura 25. Como pode ser observado, a velocidade de convergência da PSO e da PSO-Scout são aproximadas, porém com uma sutil superioridade da segunda. Mais uma vez, a velocidade de convergência da SBPSO superou as outras duas.

Na Figura 26, função Griewank, SBPSO encontrou a solução ótima próximo da iteração 1.380. A PSO e a PSO-Scout convergiram perto da iteração 1.700, porém a solução ótima não foi localizada. Por fim, na Figura 27, função Rastrigin, SBPSO encontrou a solução ótima por volta da iteração 1.300. Já a PSO estagnou próximo da iteração 1.500, ao passo que o movimento da PSO-Scout continuou ativo da iteração 2.000.

As Tabelas 3 - 6 mostram o tamanho da população, a di-

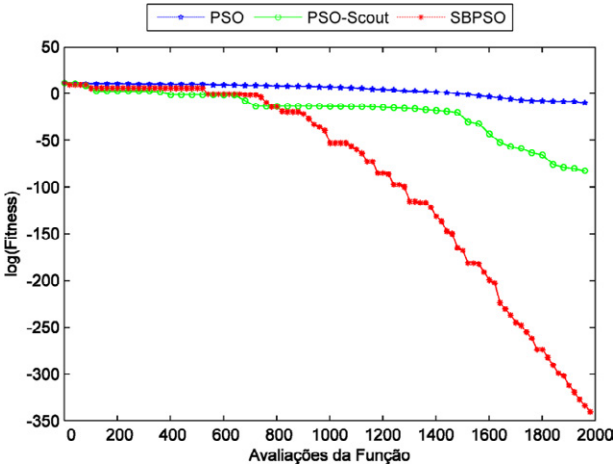
mensionalidade da função, o número de iterações, a média do valor de *fitness* e o desvio-padrão para essas funções. A coluna *p-value* apresenta os valores obtidos pelo teste de Wilcoxon quando a SBPSO é comparada com a PSO-Scout que, naturalmente, apresenta resultados melhores que a PSO. O teste é usado para verificar se os resultados produzidos pela SBPSO são, estatisticamente, significativos quando comparados com os da PSO-Scout.

Ainda em relação às Tabelas 3 - 6, é possível observar que – em todos os experimentos realizados no espaço de busca das funções Esfera, Rosenbrock, Griewank e Rastrigin – a coluna *p-value* apresenta um valor inferior ao nível de significância definido (0,05) e, nesses casos, a hipótese H_0 foi rejeitada. Isso garante que, quando comparadas SBPSO x PSO-Scout, não houve igualdade entre as soluções encontradas. Portanto, estatisticamente, os resultados da SBPSO superaram os da PSO-Scout.

Os resultados obtidos pela SBPSO também são confrontados com as variantes QPSO (SUN, FENG; XU, 2004), WQPSO (XI, SUN; XU, 2008), HPSOWM (LING et al., 2008) e HPSOM (ESMIN; MATWIN, 2013). Além disso, a comparação considera as funções f_1, f_2, f_3 e f_4, durante 100 execuções dos algoritmos.

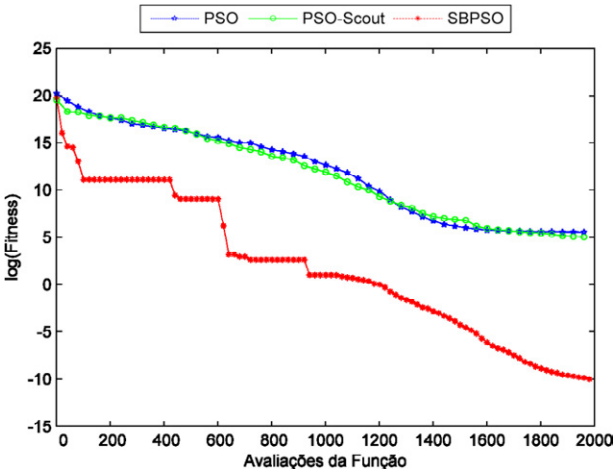
Para comparar os resultados, variam-se as dimensões (10, 20 e 30) e o número máximo de iterações (1.000, 1.500 e 2.000). Apenas o tamanho da população é fixado em 20 partículas. Após esse conjunto de experimentos, a média dos valores de *fitness* e o desvio-padrão obtidos são registrados na Tabela 7. Os resultados numéricos mostram que a SBPSO também superou quatro variantes da PSO. Nas funções unimodais Esfera e Rosenbrock, o algoritmo obteve boas soluções com alta precisão. Ela também mostrou um bom desempenho nas funções multimodais Griewank e Rastrigin, ao atingir as soluções ótimas.

Figura 24 – Convergência média na função Esfera, com 20 partículas, em 30D.



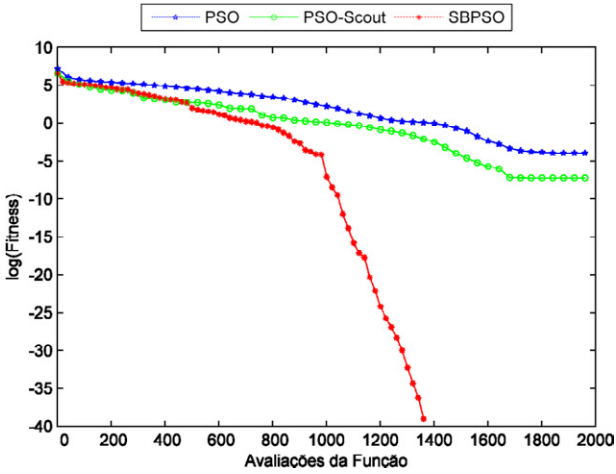
Fonte: Autoria própria (2018).

Figura 25 – Convergência média na função Rosenbrock, com 20 partículas, em 30D.



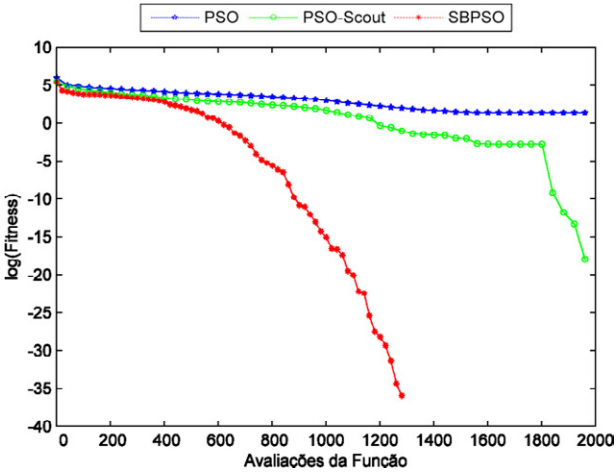
Fonte: Autoria própria (2018).

Figura 26 – Convergência média na função Griewank, com 20 partículas, em 30D.



Fonte: Autoria própria (2018).

Figura 27 – Convergência média na função Rastrigin, com 20 partículas, em 30D.



Fonte: Autoria própria (2018).

Tabela 3 – Comparação entre a PSO, PSO-Scout e SBPSO (melhores valores em negrito).

Fun	Part.	Dim	Iter	PSO	PSO-Scout	SBPSO	p-value
Esfera	20	10	1000	1.2368E-020 (3.1403E-020)	1.0056E-022 (5.5127E-022)	**1.3900E-081 (1.2772E-080)**	1.57E-30
		20	1500	2.9396E-011 (1.8370E-010)	3.4132E-018 (1.9818E-017)	**3.1139E-124 (2.6196E-123)**	1.57E-30
		30	2000	4.6804E-008 (1.3386E-007)	5.0619E-021 (5.0275E-020)	**2.9522E-156 (2.9520E-155)**	1.57E-30
	40	10	1000	2.2365E-024 (1.3369E-023)	1.3691E-027 (1.0608E-026)	**3.1786E-084 (3.1719E-083)**	1.57E-30
		20	1500	8.1334E-015 (2.5881E-014)	2.9916E-028 (2.9840E-027)	**2.2798E-119 (1.9630E-118)**	1.57E-30
		30	2000	8.0544E-011 (1.3452E-010)	5.4840E-034 (2.4924E-033)	**5.5811E-129 (2.3349E-128)**	1.57E-30
	80	10	1000	1.5394E-028 (4.8125E-028)	1.5496E-030 (1.0748E-029)	**9.7470E-081 (9.7469E-080)**	1.57E-30
		20	1500	5.1700E-018 (1.4221E-017)	2.3584E-026 (1.9487E-025)	**5.6513E-120 (5.6513E-119)**	1.57E-30
		30	2000	1.4817E-013 (2.9377E-013)	2.2470E-031 (2.2468E-030)	**1.1080E-162 (1.1113E-161)**	1.57E-30

Tabela 4 – Comparação entre a PSO, PSO-Scout e SBPSO (melhores valores em negrito).

Fun	Part.	Dim	Iter	PSO	PSO-Scout	SBPSO	p-value
Rosenbrock	20	10	1000	58.3417 (133.7896)	40.6746 (103.9401)	**9.5785E-08 (5.7832E-07)**	1.67E-17
		20	1500	104.9516 (162.9876)	99.4901 (186.9198)	**2.9085E-07 (8.6492E-06)**	1.06E-07
		30	2000	151.5238 (239.0893)	104.9869 (151.4036)	**2.0347E-05 (7.6421E-04)**	5.83E-15
	40	10	1000	30.80349 (114.8610)	15.6310 (38.1135)	**3.7120E-08 (8.8402E-07)**	5.83E-15
		20	1500	81.5949 (141.4203)	48.1244 (60.5719)	**6.8410E-07 (6.0376E-06)**	9.59E-14
		30	2000	132.6704 (204.0919)	112.7430 (13.9928)	**3.7810E-05 (1.9304E-04)**	3.27E-17
	80	10	1000	20.5744 (43.8337)	13.9724 (34.8846)	**4.8522E-08 (4.3964E-07)**	4.43E-16
		20	1500	65.7612 (105.0775)	52.7690 (92.4566)	**6.9654E-07 (3.0942E-06)**	1.63E-15
		30	2000	86.87974 (123.5531)	69.8944 (92.4215)	**3.8413E-05 (7.7475E-04)**	1.31E-17

Tabela 5 – Comparação entre a PSO, PSO-Scout e SBPSO (melhores valores em negrito).

Fun	Part.	Dim	Iter	PSO	PSO-Scout	SBPSO	p-value
Griewank	20	10	1000	0.1012 (0.0516)	0.0216 (0.0456)	**0 (0)**	1.57E-30
		20	1500	0.0334 (0.0336)	4.2475E-04 (5.1756E-03)	**0 (0)**	1.57E-30
		30	2000	0.0146 (0.0171)	2.4458E-04 (1.4592E-03)	**0 (0)**	1.57E-30
	40	10	1000	0.0845 (0.0438)	0.0192 (0.0396)	**0 (0)**	1.57E-30
		20	1500	0.0276 (0.0291)	7.9024E-04 (3.7688E-03)	**0 (0)**	6.31E-30
		30	2000	0.0592 (0.1690)	1.7253E-04 (1.2263E-03)	**0 (0)**	1.57E-30
	80	10	1000	0.0772 (0.0396)	0.0188 (0.0339)	**0 (0)**	1.57E-30
		20	1500	0.0357 (0.0325)	3.7898E-04 (1.4178E-03)	**0 (0)**	4.03E-28
		30	2000	0.0127 (0.0133)	1.0129E-04 (9.8582E-03)	**0 (0)**	1.57E-30

Tabela 6 – Comparação entre a PSO, PSO-Scout e SBPSO (melhores valores em negrito).

Fun	Part.	Dim	Iter	PSO	PSO-Scout	SBPSO	p-value
Rastrigin	20	10	1000	4.5782 (2.1132)	0.9514 (0.6391)	**0 (0)**	3.63E-06
		20	1500	22.8061 (10.0912)	0.0475 (0.3913)	**0 (0)**	3.38E-21
		30	2000	49.7192 (13.7956)	0.0253 (0.1905)	**0 (0)**	2.58E-23
	40	10	1000	3.9224 (1.5118)	0.2854 (0.4404)	**0 (0)**	6.10E-06
		20	1500	16.1082 (1.5722)	8.7266E-04 (8.7266E-03	**0 (0)**	9.53E-07
		30	2000	38.6344 (2.2039)	(8.7266E-03 (9.5142E-03)	**0 (0)**	2.22E-16
	80	10	1000	2.2854 (0.4780)	0.1141 (0.3123)	**0 (0)**	3.12E-06
		20	1500	12.3630 (2.2804)	3.5527E-04 (6.6714E-03	**0 (0)**	1.42E-06
		30	2000	31.5714 (6.2924	7.6115E-04 (3.2228E-03)	**0 (0)**	2.77E-17

Tabela 7 – Comparação entre diferentes variantes da PSO com 20 partículas em 10D (1.000 iterações), 20D (1.500 iterações) e 30D (2.000 iterações). Os melhores valores estão em negrito.

Fun	D	HPSOWM	WQPSO	QPSO	HPSOM	SBPSO
Esfera	10	6.2868E-56 (1.5060E-55)	2.2922E-56 (1.7300E-58)	1.3909E-41 (1.4049E-043)	2.2400E-56 (1.7300E-58)	**1.3900E-81 (1.2772E-80)**
	20	6.2830E-45 (2.0330E-44)	2.9451E-40 (2.8717E042)	3.5103E-022 (3.5452E-24)	2.1449E-49 (1.6891E-43)	**3.1139E-124 (2.6196E-123)**
	30	3.7940E-36 (1.4060E-35)	3.9664E-33 (3.8435E-35)	5.3183E-014 (5.3623E-16)	6.5764E-034 (5.6809E-36)	**2.9522E-156 (2.9520E-155)**
Rosenbrock	10	36.4736 (0.1844)	35.8436 (0.2843)	51.9761 (0.4737)	6.9688 (0.2730)	**9.5785E-08 (5.7832E-07)**
	20	65.6678 (0.5870)	62.7696 (0.4860)	136.8782 (0.6417)	17.3033 (0.2210)	**2.9085E-07 (8.6492E-06)**
	30	70.7275 (0.4813)	70.9525 (0.4283)	157.4707 (0.8287)	27.5645 (0.2939)	**2.0347E-05 (7.6421E-04)**
Griewank	10	0.13333 (0.33993)	5.6353E-04 (5.5093E-04)	5.5093E-04 (0.0657)	4.32057E-11 (3.1216E-11)	**0 (0)**
	20	2.9333 (2.7439)	2.1318E-04 (1.0402E-04)	1.0402E-04 (0.0211)	4.00370E-11 (3.13852E-11)	**0 (0)**
	30	9.2333 (6.1455)	2.1286E-04 (1.2425E-04)	1.2425E-04 (0.0110)	5.1756E-11 (3.0143E-11)	**0 (0)**
Rastrigin	10	4.610 (2.5364)	4.0567 (0.0094)	4.8274 (0.0015)	4.1558E-11 (3.2202E-11)	**0 (0)**
	20	19.6670 (6.7661)	12.1102 (0.0287)	16.0519 (0.0414)	4.1403E-11 (3.2402E-11)	**0 (0)**
	30	44.7230 (13.9680)	23.5593 (0.0713)	33.7218 (0.0114)	4.8007E-11 (3.1883E-11)	**0 (0)**

6.2.2 CONJUNTO DE EXPERIMENTOS 2

Nesse segundo conjunto de experimentos, foram avaliadas outras doze funções de referência. Assim como as do conjunto anterior, essas doze também possuem características distintas em relação à continuidade, à diferenciabilidade, à separabilidade, à escalabilidade e à modalidade. As funções avaliadas são: Ackley (f_5), Noisy Quartic (f_6), Zakharov (f_7), Hiper Elipsoide Rotacionada (f_8), Alpine (f_9), Weierstrass (f_{10}), Salomon (f_{11}), Csendes (f_{12}), Xin-She Yang 01 (f_{13}), Soma dos Quadrados (f_{14}), Schumer Steiglitz (f_{15}) e Powell Sum (f_{16}).

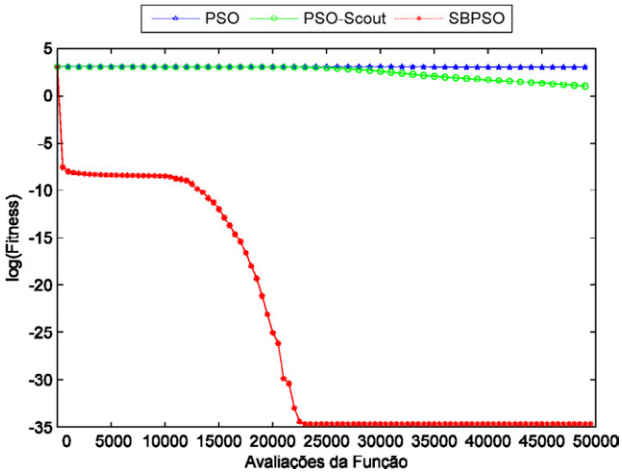
Os experimentos também investigaram a escalabilidade da SBPSO durante o processo de otimização, porém em alta dimensionalidade e com número maior de avaliações da função objetivo, quando comparados ao conjunto de experimentos anterior.

No Conjunto de Experimentos 1, foram variados 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). Porém, nesse conjunto de experimentos, para cada uma das doze funções de referência, o tamanho da população foi fixado em 50 partículas, a dimensão das funções em 300D e o número de avaliações da função objetivo em 50.000 vezes. A Tabela 8 mostra a média dos valores de *fitness*, o desvio-padrão e o *p-value* resultante do teste de Wilcoxon da comparação entre SBPSO e PSO-Scout.

Como consequência da alta dimensionalidade atribuída às funções avaliadas, o tempo necessário para realizar o Conjunto de Experimentos 2 é relativamente alto. Então, a fim de diminuir a duração dos experimentos, decidiu-se reduzir o número de execuções independentes dos algoritmos para 30, em vez de 100, como no conjunto anterior.

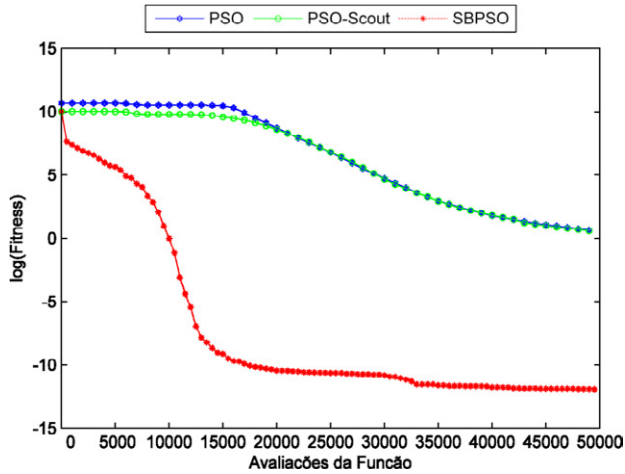
As Figuras 28 - 39 mostram o processo de convergência da PSO, PSO-Scout e SBPSO que foi medido, com 50 partículas, em 300 dimensões e com 50.000 iterações. O processo ocorre no espaço de busca das funções Ackley, Noisy Quartic, Zakharov, Hiper Elipsoide Rotacionada, Alpine, Weierstrass, Salomon, Csendes, Xin-She Yang 01, Soma dos Quadrados, Schumer Steiglitz e Powell Sum.

Figura 28 – Convergência média na função Ackley, com 50 partículas, em 300D.



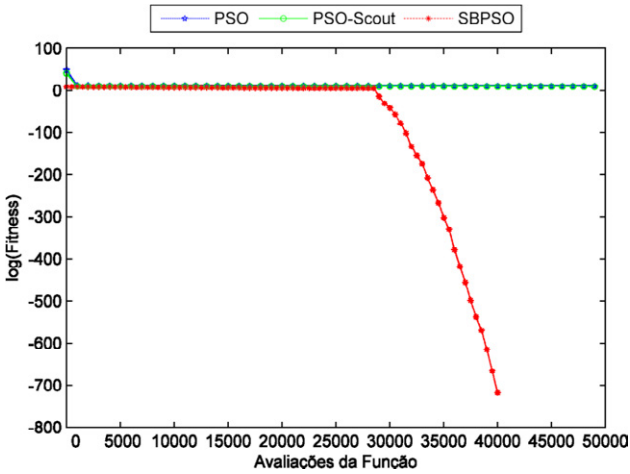
Fonte: Autoria própria (2018).

Figura 29 – Convergência média na função Noisy Quartic, com 50 partículas, em 300D.



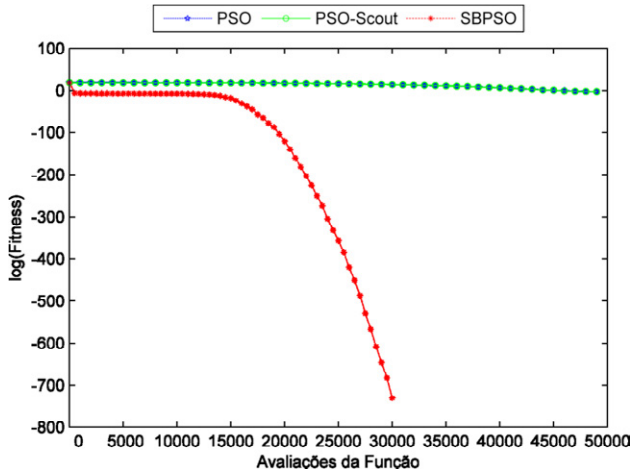
Fonte: Autoria própria (2018).

Figura 30 – Convergência média na função Zakharov, com 50 partículas, em 300D.



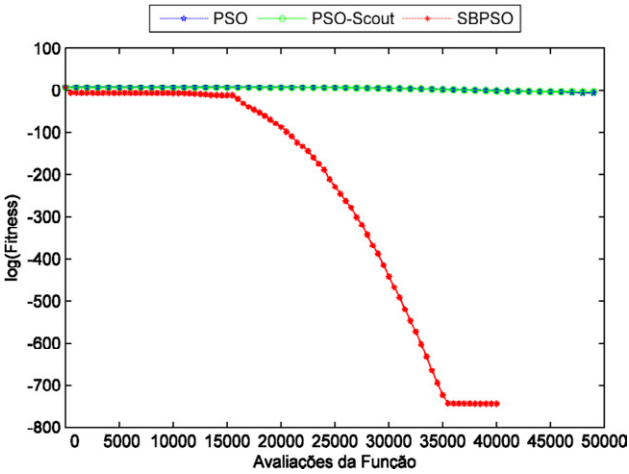
Fonte: Autoria própria (2018).

Figura 31 – Convergência média na função Hiper Elipsoide Rotacionada, com 50 partículas, em 300D.



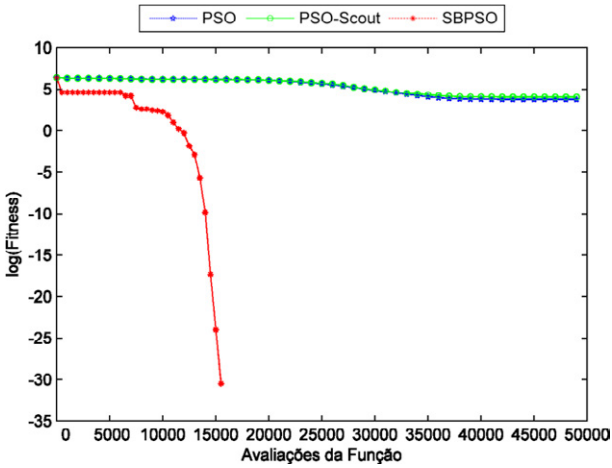
Fonte: Autoria própria (2018).

Figura 32 – Convergência média na função Alpine, com 50 partículas, em 300D.



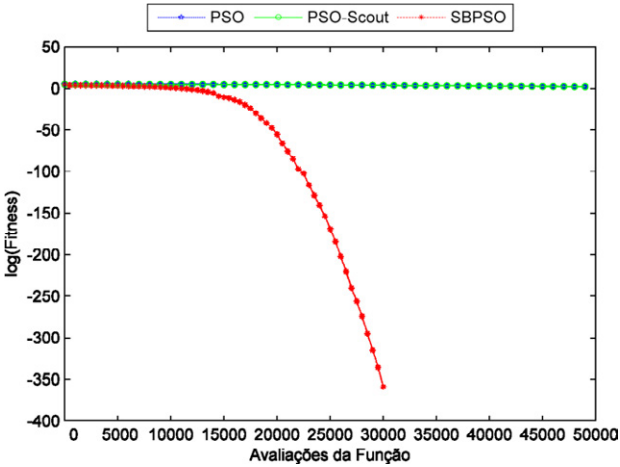
Fonte: Autoria própria (2018).

Figura 33 – Convergência média na função Weierstrass, com 50 partículas, em 300D.



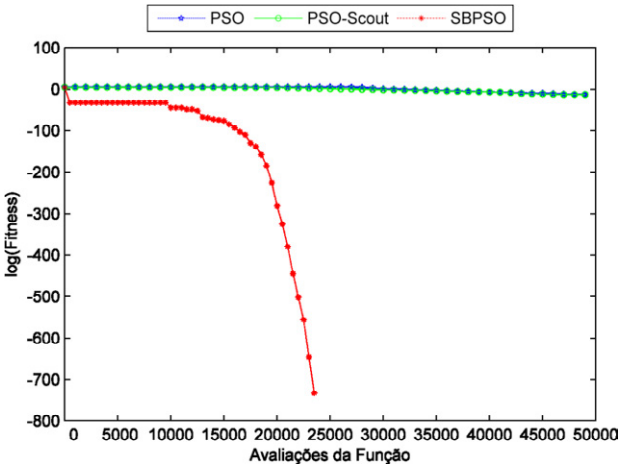
Fonte: Autoria própria (2018).

Figura 34 – Convergência média na função Salomon, com 50 partículas, em 300D.



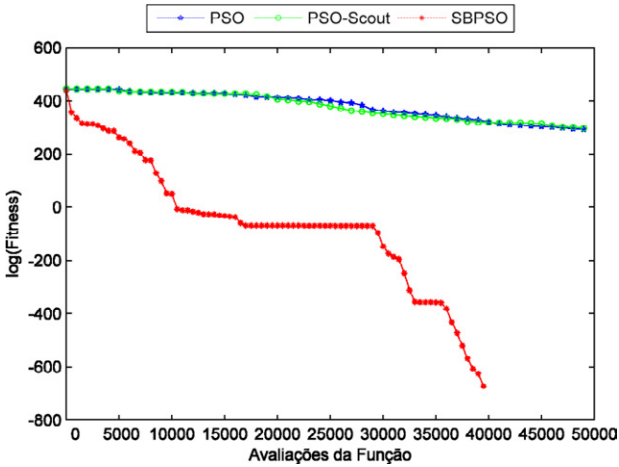
Fonte: Autoria própria (2018).

Figura 35 – Convergência média na função Csendes, com 50 partículas, em 300D.



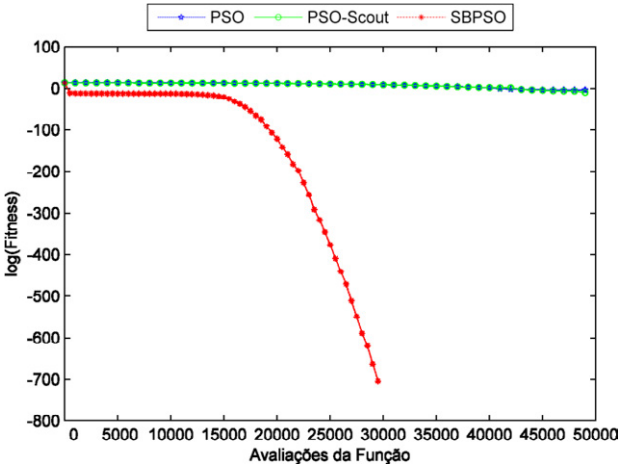
Fonte: Autoria própria (2018).

Figura 36 – Convergência média na função de Xin-She Yang 01, com 50 partículas, em 300D.



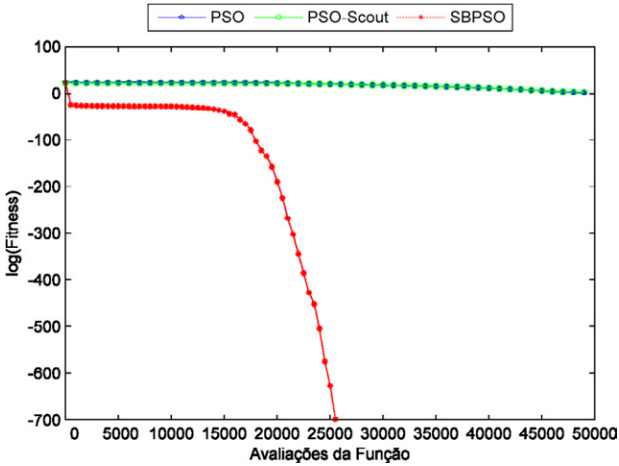
Fonte: Autoria própria (2018).

Figura 37 – Convergência média na função Soma dos Quadrados, com 50 partículas, em 300D.



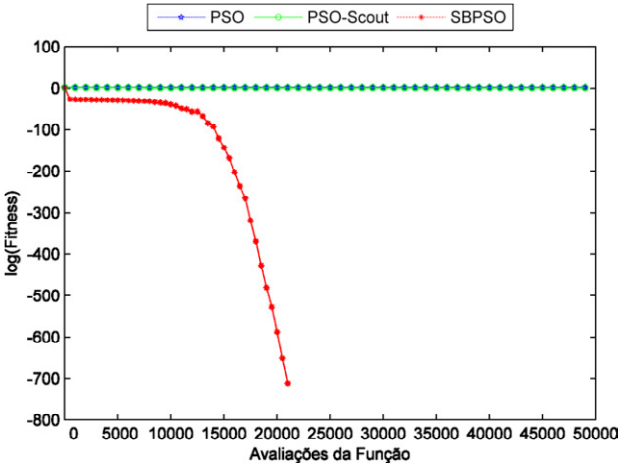
Fonte: Autoria própria (2018).

Figura 38 – Convergência média na função de Schumer Steiglitz, com 50 partículas, em 300D.



Fonte: Autoria própria (2018).

Figura 39 – Convergência média na função Powell Sum, com 50 partículas, em 300D.



Fonte: Autoria própria (2018).

Tabela 8 – Comparação entre a PSO, PSO-Scout e SBPSO para as funções f_5-f_{16}, em 300 dimensões com 50 partículas. Os melhores valores estão em negrito.

Função	PSO	PSO-Scout	SBPSO	p-value
f_5	19.9408 (0.0218)	2.4100 (2.3123)	**8.88E-16 (0)**	1.86E-09
f_6	1.7646 (4.0936)	1.6635 (0.3021)	**6.58E-06 (5.35E-06)**	1.86E-09
f_7	2.17E04 (4.55E03)	8.18E03 (1.00E03)	**0 (0)**	1.86E-09
f_8	2.35E-03 (4.27E-03)	4.84E-02 (0.24)	**0 (0)**	1.86E-09
f_9	1.13E-03 (3.09E-03)	0.0176 (0.0938)	**0 (0)**	1.86E-09
f_{10}	46.63 (6.18)	48.53 (7.38)	**0 (0)**	1.86E-09
f_{11}	7.12 (0.89)	7.02 (0.82)	**0 (0)**	1.86E-09
f_{12}	2.85E-06 (1.45E-06)	2.70E-06 (1.34E-06)	**0 (0)**	1.86E-09
f_{13}	8.24E125 (2.72E126)	8.97E125 (4.64E126)	**0 (0)**	1.86E-09
f_{14}	1.08E-05 (1.93E-05)	3.34E-05 (1.41E-04)	**0 (0)**	1.86E-09
f_{15}	0.38 (0.44)	0.76 (1.66)	**0 (0)**	1.86E-09
f_{16}	4.76 (0.65)	2.64 (0.69)	**0 (0)**	1.86E-09

Na Figura 28, função Ackley, observa-se que a SBPSO convergiu próximo da iteração 22.500, ao passo que a PSO, nas primeiras iterações, entra em processo de estagnação. Quanto à PSO-Scout, após 50.000 iterações, ela não convergiu, e a qualidade das soluções encontradas foi bastante inferior à da SBPSO, embora melhor que a da PSO. Para a função Noisy Quartic, representada pela Figura 29, o comportamento de convergência da PSO-Scout é muito parecido com o da PSO, apesar de a qualidade da solução encontrada ter sido inferior à da segunda. Embora SBPSO tenha encontrado boas médias para o valor de *fitness* das funções Ackley (8.88E-16) e Noisy Quartic (6.58E-06), a solução ótima não foi encontrada.

Na Figura 30, função Zakharov, PSO e PSO-Scout convergiram perto da iteração 2.000, e os valores médios de *fitness* foram, respectivamente, 2.17E04 e 8.18E03. Novamente, o comportamento da PSO superou o da PSO-Scout. Já a SBPSO encontrou a solução ótima próximo à iteração 40.000. Para a função Hiper Elipsoide Rotacionada, representada pela Figura 31, SBPSO encontrou a solução ótima aproximadamente depois de 30.000 avaliações da função objetivo. A velocidade de convergência da PSO e da variante PSO-Scout são aproximadas, porém a primeira é superior.

A Figura 32 representa o comportamento médio da função Alpine. A média do valor de *fitness* encontrada pela PSO foi 1.13E-03 e pela PSO-Scout 0.0176. Os algoritmos não encontraram a solução ótima, porém, na iteração 40.000, SBPSO localizou a solução global. Na Figura 33, função Weierstrass, a SBPSO encontrou a solução ótima próximo da iteração 15.000. Enquanto isso, a PSO apresentou uma ligeira superioridade em relação à PSO-Scout, convergindo próximo da iteração 40.000, mas sem encontrar o ótimo global.

A Figura 34 ilustra o processo de convergência dos algoritmos no espaço de busca da função Salomon. SBPSO localizou a solução global próximo da iteração 30.000. PSO-Scout superou PSO no que diz respeito à qualidade das soluções encontradas, apesar da estagnação de ambas nas primeiras iterações. Novamente, o comportamento de convergência da PSO e de sua variante foi parecido com a otimização da função Csendes, como visto na Figura 35. SBPSO encontrou a solução ótima perto da iteração 24.000.

Na Figura 36, função de Xin-She Yang 01, SBPSO encontrou o ótimo global próximo da iteração 40.000. No entanto, observa-se que houve um período de estagnação entre as iterações 15.000 e 30.000, aproximadamente. Nessa simulação, depois de o algoritmo ter detectado a estagnação do enxame, ele conseguiu fazê-lo movimentar-se novamente. A solução ótima foi encontrada nas proximidades da iteração 40.000. Por outro lado, PSO e PSO-Scout apresentaram uma velocidade de convergência lenta, além de o ótimo global não ter sido localizado. Já o comportamento de convergência delas na otimização da função Soma dos Quadrados, como visto na Figura 37, novamente, é aproximado. Na maior parte do número de iterações, os algoritmos permaneceram estagnados. Por outro lado, SBPSO atingiu o ponto ótimo ao redor da iteração 30.000.

Na Figura 38, função Schumer Steiglitz, SBPSO encontrou a solução ótima próximo da iteração 25.500. Já a PSO superou a PSO-Scout, apesar de o comportamento médio de convergência apresentado por elas ser bastante próximo. Por fim, na Figura 39, a velocidade de convergência da PSO e da PSO-Scout é parecida, mas a segunda é superior. SBPSO encontrou a solução global próximo à iteração 22.000.

Em todas as simulações realizadas nesse conjunto de experimentos, bem como no anterior, observou-se que o comportamento de convergência da SBPSO superou o da PSO e o da variante PSO-Scout. Os valores médios obtidos ao fim dos experimentos foram apresentados na Tabela 8. Além disso, os resultados do teste de Wilcoxon foram inferiores ao nível de significância predefinido (0,05) e, consequentemente, a hipótese H_0 foi rejeitada.

7

Considerações Finais

Este livro estudou o conceito de serendipidade a fim de adaptá-lo ao contexto da inteligência de enxames. Para validar a adaptação do conceito nesse novo contexto, uma variante da PSO, com abordagem baseada em serendipidade (SBPSO), foi implementada com o objetivo de obter melhores resultados no processo de convergência.

A implementação da SBPSO considerou as duas dimensões definidas no conceito de serendipidade, acaso e sagacidade. A primeira foi implementada por meio do uso de partículas escoteiras com o objetivo de melhorar a exploração no espaço de busca. Já a dimensão sagacidade foi implementada usando um conjunto de padrões que representam o comportamento dos valores de *fitness* da função objetivo avaliada.

Para avaliar a performance da variante SBPSO, dezesseis funções de referência foram utilizadas. Elas possuem propriedades importantes que devem ser observadas durante o processo: continuidade, diferenciabilidade, separabilidade, escalabilidade e modalidade. A partir disso, foram realizados dois conjuntos de experimentos. No primeiro, quatro funções de referência foram usadas para comparar SBPSO, PSO, PSO-Scout e outras variantes, como HPSOWM, WQPSO, QPSO e HPSOM. Após os experimentos, foi observado que a SBPSO superou a PSO, a PSO-Scout e as outras quatro variantes citadas. O segundo conjunto de experimentos teve doze funções de referência, comparando a nova variante com a PSO e a PSO-Scout. Mais uma vez a SBPSO foi superior.

Como observado nos resultados apresentados na Seção 6.2, a velocidade de convergência da SBPSO superou a das outras duas. Quanto aos resultados das soluções encontradas, a PSO e a PSO-Scout não encontraram o ótimo global de nenhuma das dezesseis funções de referência avaliadas. Por outro lado, a SBPSO

não o encontrou em somente duas (f_1 e f_2), porém os ótimos locais foram melhores que os dos outros dois algoritmos e, além disso, eles se mostraram bastante satisfatórios.

Em todos os experimentos, SBPSO apresentou um bom comportamento médio de convergência, superando a PSO e algumas variantes no que diz respeito à qualidade das soluções, à capacidade de encontrar ótimos globais, à estabilidade das soluções e à capacidade para reiniciar o movimento do enxame após a estagnação ter sido detectada.

Após os experimentos, foi observado que há bastante espaço para novas aplicações do conceito de serendipidade no domínio dos algoritmos de inteligência de enxames. Também se observou que, de fato, a convergência prematura foi retardada nas simulações realizadas em cada uma das dezesseis funções de referência avaliadas. Isso mostra que a utilização do conceito de serendipidade pode viabilizar o desenvolvimento de mecanismos capazes de diminuir a aleatoriedade e melhorar a performance dos algoritmos baseados em inteligência de enxames.

A implementação da dimensão sagacidade considerou um conjunto de padrões para representar o comportamento dos valores de *fitness* da função objetivo avaliada. Esses padrões foram definidos com base em apenas dois pontos de inspeção: IP_1 e IP_2. Entretanto, é interessante estudar o uso de um número maior de pontos de inspeção e, consequentemente, a definição de novos padrões de comportamento. Além disso, é importante avaliar o desempenho e o tempo de execução do algoritmo quando novos pontos de inspeção forem definidos.

Ainda em relação à utilização dos padrões de comportamento, as ações são tomadas com base em regras predefinidas para o padrão que foi identificado. Essas ações são disparadas quando

determinada regra for atendida. No entanto, a rotina pode ser melhorada a fim de aumentar a parametrização do algoritmo, uma vez que, para cada novo padrão apresentado, não haverá necessidade de mudanças no código-fonte. Uma alternativa para isso seria a utilização de matrizes para definição de regras a serem disparadas quando um novo padrão de comportamento for apresentado.

Embora o conceito de serendipidade possa ser estendido a vários algoritmos da inteligência de enxames, a adaptação do conceito foi implementada apenas no contexto da PSO. Portanto, é interessante implementar o conceito de serendipidade em outros algoritmos, como o do vaga-lume, o do morcego e outros.

É interessante também estudar a viabilidade da adaptação e da implementação de outras estratégias para geração de serendipidade. Uma delas é uma abordagem baseada em *anomalias e exceções*, uma vez que essa estratégia considera uma característica importante chamada de dissimilaridade. Ela pode ser usada para nortear o movimento das partículas escoteiras na direção oposta do enxame de partículas. Essa estratégia pode ser combinada com a *blind luck*, para melhorar o comportamento exploratório das escoteiras.

A variante SBPSO foi validada em funções de referência que podem ser usadas para representar diversos problemas contínuos do mundo real, embora elas não sejam adequadas na representação do comportamento de sistemas que possuem natureza dinâmica. Quando as mudanças no valor de *fitness* da função objetivo são moderadas, adicionar novas partículas escoteiras para aumentar a diversidade do enxame ou simplesmente reiniciá-lo pode não produzir resultados significativos. Em casos como esses, é importante que o algoritmo seja capaz de se adequar a tais mudanças ao longo do tempo.

Em um próximo estudo, além de examinar mecanismos que possam ser adicionados à nova variante para tratar problemas dinâmicos, pretende-se também aplicá-la a problemas discretos, como o do caixeiro-viajante. A validação da SBPSO nesse problema é importante por ele ser bastante utilizado em cenários reais, como roteamento de veículos, perfuração de placas de circuitos impressos, problemas de sequenciamento e outros.

Pretende-se ainda validá-la em domínios específicos nos quais exista conhecimento passível de ser utilizado para explorar a dimensão sagacidade. Um desses domínios é a otimização de problemas relacionados a telecomunicações como projetos de antenas de microfitas e *arrays* de antenas inteligentes. Eles são exemplos clássicos nos quais os algoritmos bioinspirados encontram-se frequentemente aplicados e, portanto, na literatura há várias abordagens que podem ser utilizadas para comparar o desempenho dessa nova variante em relação às meta-heurísticas tradicionais.

Referências

ANDEL, P. V. Anatomy of the Unsought Finding. Serendipity: Origin, History, Domains, Traditions, Appearances, Patterns and Programmability. **The British Journal for the Philosophy of Science**, Oxford, v. 45, n. 2, p. 631-648, 1994.

AVISE, J. C. **Flock in breeding plumage**, Texas. 2006. Disponivel em: <https://bit.ly/2MX7XGU>. Acesso em: 19 abr. 2016.

BENI, G.; WANG, J. Swarm Intelligence in Cellular Robotic Systems. **NATO Advanced Workshop on Robots and Biological Systems**. Toscana, [s.n.], p. 26-30, 1989.

CAMPOS, J.; FIGUEIREDO, A. D. **Programming for Serendipity**. Proceedings of Fall Symposium on Chance Discovery – The Discovery and Management of Chance Events. [S.l.], [s.n.], p. 48-60, 2002.

CATELLIN, S. **Sérendipité:** Du conte au concept. Paris: Seuil, 2014.

DIETERICH, J. M.; HARTKE, B. Empirical review of standard benchmark functions using evolutionary global optimization. **Applied Mathematics**, Wuhan, v. 3, n. 10A, p. 1552-1564, out. 2012.

ESMIN, A. A. A.; MATWIN, S. HPSOM: A Hybrid Particle Swarm Optimization Algorithm with Genetic Mutation. **International Journal of Innovative Computing, Information and Control**, v. 9, n. 5, p. 1919-1934, maio 2013. ISSN 1349-4198.

GANDOMI, A. H.; ALAVI, A. H. Krill herd: a new bio-inspired optimization algorithm. **Communications in Nonlinear Science and Numerical Simulation**, Amsterdam, v. 17, n. 12, p. 4831-4845, 2012.

GLOVER, F. Future Paths for Integer Programming and Links to Artificial Intelligence. **Computers & Operations Research**, Amsterdam, v. 13, n. 5, p. 533-549, maio 1986. ISSN 0305-0548.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. **Conference on Neural Networks**. Perth, [s.n.], p. 1942-1948, 1995.

LAWLEY, J.; TOMPKINS, P. Maximising serendipity: the art of recognising and fostering potential. **The Clean Collection**, [S.l.], [s.n.], 7 jun. 2008. Disponivel em: <https://bit.ly/1qXjGFz>.
Acesso em: 13 Abril 2016.

LING, S.H. et al. Hybrid particle swarm optimization with wavelet mutation and its industrial applications. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 38, n. 3, p. 743-763, 2008.

PAIVA, F. A. P.; COSTA, J. A. F.; SILVA, C. R. M. A Serendipity-Based PSO Approach to Delay Premature Convergence Using Scout Particle. **International Journal of Innovative Computing, Information and Control**, [S.l.], v. 12, n. 4, p. 1141-1163, ago. 2016. ISSN 1349-4198.

SUN, J.; FENG, B.; XU, W. Particle swarm optimization with particles having quantum behavior. **Congress on Evolutionary Computation**. Portland, [s.n.], p. 325-331, 2004.

TOMS, E. **Serendipitous Information Retrieval**. First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries. Zurich, [s.n.], p. 11-12, 2000.

VAN DEN BERGH, F. **An analysis of particle swarm optimizers**. Tese (Doutorado). Universidade de Pretoria, Pretoria. 2006.

WU, Y. et al. A. Discrete particle swarm optimization with scout particles for library materials acquisition. **The Scientific World Journal**, Cairo, v. 2013, p. 1-11, 2013.

XI, M.; SUN, J.; XU, W. An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position. **Applied Mathematics and Computation**, [S.l.], v. 205, n. 2, p. 751-759, 2008.

ZANG, H.; ZHANG, S.; HAPESHI, K. A Review of Nature-Inspired Algorithms. **Journal of Bionic Engineering**, Amsterdam v. 7, p. 232-237, 2010. Suplemento.


editoraifrn

A Editora do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN) já publicou livros em todas as áreas do conhecimento, ultrapassando a marca de 150 títulos. Atualmente, a edição de suas obras está direcionada a cinco linhas editoriais, quais sejam: acadêmica, técnico-científica, de apoio didático-pedagógico, artístico-literária ou cultural potiguar.

Ao articular-se à função social do IFRN, a Editora destaca seu compromisso com a formação humana integral, o exercício da cidadania, a produção e a socialização do conhecimento.

Nesse sentido, a EDITORA IFRN visa promover a publicação da produção de servidores e estudantes deste Instituto, bem como da comunidade externa, nas várias áreas do saber, abrangendo edição, difusão e distribuição dos seus produtos editoriais, buscando, sempre, consolidar a sua política editorial, que prioriza a qualidade.

Fábio Augusto Procópio de Paiva

Doutor (2016) e mestre (2007) em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte (UFRN) e bacharel (1999) em Sistemas de Informação pela Universidade Potiguar (UnP). É Professor do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), atuando nos cursos técnicos e superiores da área de Sistemas de Informação. As áreas de interesse incluem: Meta-heurísticas Bioinspiradas, Sistemas de Recomendação, Bancos de Dados Relacionais, Bancos de Dados NoSQL, Bancos de Dados NewSQL e Internet das Coisas.

Os problemas de otimização são comuns em diversas aplicações de engenharia, como telecomunicações, processamento paralelo, roteamento de veículos, controle de tráfego, eletromagnetismo, construção civil e muitas outras. A otimização é um processo que visa encontrar a melhor solução para determinado problema.

Existem vários algoritmos que podem ser utilizados para resolver problemas de otimização e muitos são classificados como métodos meta-heurísticos. No entanto, muitas meta-heurísticas enfrentam um problema chamado convergência prematura. Para lidar com esse problema, várias abordagens já foram apresentadas.

Este livro apresenta uma nova abordagem baseada em um conceito conhecido como serendipidade, que pode ser usada na área das meta-heurísticas. Para validar a viabilidade da adequação do conceito ao contexto meta-heurístico, uma variante chamada *Serendipity-Based Particle Swarm Optimization* (SBPSO) foi implementada considerando duas dimensões da serendipidade: acaso e sagacidade. Para avaliar a proposta apresentada, dois conjuntos de experimentos computacionais foram realizados. No primeiro, quatro funções de referência foram usadas para comparar SBPSO com a Otimização por Enxame de Partículas (PSO – *Particle Swarm Optimization*) e algumas variantes da literatura. No segundo conjunto, outras doze funções foram utilizadas, porém em alta dimensionalidade. Em todos os experimentos, os resultados da SBPSO se mostraram promissores e apresentaram um bom comportamento de convergência.

ISBN 978-85-94137-54-8
9 788594 137548

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

ABEU Associação Brasileira das Editoras Universitárias