UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HUGO NASCIMENTO DA SILVA

APRIMORANDO ALGORITMOS DE NICHO COM BUSCA LOCAL: UMA ABORDAGEM PARA OTIMIZAÇÃO MULTIMODAL

RIO DE JANEIRO
2025

HUGO NASCIMENTO DA SILVA

APRIMORANDO ALGORITMOS DE NICHO COM BUSCA LOCAL: UMA ABORDAGEM PARA OTIMIZAÇÃO MULTIMODAL

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profa. Carolina Gil Marcelino

RIO DE JANEIRO
2025

CIP - Catalogação na Publicação

S586a

Silva, Hugo Nascimento da
Aprimorando algoritmos de nicho com busca local: uma abordagem para otimização multimodal / Hugo Nascimento da Silva. -- Rio de Janeiro, 2025.
61 f.

Orientadora: Carolina Gil Marcelino.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2025.

1. FER-PSO. 2. Otimização multimodal. 3. Busca local. 4. Hill Climbing. I. Marcelino, Carolina Gil, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

HUGO NASCIMENTO DA SILVA

APRIMORANDO ALGORITMOS DE NICHO COM BUSCA LOCAL: UMA ABORDAGEM PARA OTIMIZAÇÃO MULTIMODAL

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 24 de janeiro de 2025

BANCA EXAMINADORA:

Carolina Gil Marcelino
D.Sc. (IC/UFRJ)

Carla Amor Divino Moreira Delgado
D.Sc. (IC/UFRJ)

João Carlos Pereira da Silva
D.Sc. (IC/UFRJ)

Dedico este trabalho aos meus pais, que são minha maior inspiração. Ao meu pai, com quem aprendi o valor do trabalho duro e da dedicação nas horas que trabalhamos juntos. À minha mãe, que nunca mediu esforços para cuidar de mim e me apoiar. Obrigado por serem meu alicerce em todos os momentos.

AGRADECIMENTOS

Agradeço aos meus pais por terem me apoiado em todos os momentos da minha vida. Obrigado por serem meus maiores incentivadores, sempre me motivando a perseguir meus sonhos e a ingressar na graduação. Sem vocês, eu não estaria aqui.

À Viviane, minha namorada e melhor amiga, minha companheira de vida. Obrigado por estar ao meu lado desde antes de eu sonhar em entrar na faculdade. Sua presença iluminou cada etapa dessa jornada, desde o início incerto até a formatura. Nos momentos mais difíceis, você foi minha fortaleza, trazendo leveza quando tudo parecia pesado e sendo a pessoa com quem eu pude dividir minhas maiores inseguranças, medos e sonhos. Ter você ao meu lado transformou desafios em superações e tornou cada dia mais especial. Obrigado por tudo, por ser quem é e por acreditar em mim mesmo quando eu duvidei.

Agradeço à minha orientadora, professora Carolina, por sua paciência e dedicação ao orientar um aluno que, no início, não fazia ideia do que gostaria de estudar. Obrigado por me guiar para um caminho que nunca imaginei gostar e por toda a paciência durante a graduação e na orientação deste trabalho.

Agradeço aos meus amigos Sthefany e Guilherme, que estão comigo há muito tempo. Obrigado pelas risadas, pelas conversas, pela amizade e pelos momentos incríveis que compartilhamos. Amo muito vocês.

Aos amigos que a faculdade me trouxe, como Patrick, Pedro, Wemerson, Arthur e tantos outros, minha gratidão por terem estado ao meu lado durante esses quatro anos e meio. Com vocês, o caminho da graduação foi muito mais leve.

Agradeço ao meu irmão Vitor, pelas conversas e pela inspiração que sempre me deu ao longo desta jornada.

Sou profundamente grato à minha família. À Raylle, pelos momentos vividos juntos e pelas risadas. Ao meu tio Gilson, pelo incentivo constante e por me ensinar a importância dos estudos. Ao meu primo Léo, que foi uma grande inspiração, pelas conversas antes e durante a graduação. Ao meu padrinho José e à minha madrinha Cida, pelo carinho, pelas conversas e pelo apoio em todos os momentos que me guiaram até aqui.

Vocês são parte essencial da minha trajetória, e sem vocês este sonho não teria se realizado.

RESUMO

Este trabalho apresenta a implementação e modificação do algoritmo FER-PSO (*Fitness Euclidean-distance Ratio Particle Swarm Optimization*), um algoritmo de nicho empregado para resolver problemas de otimização multimodal, com o objetivo de aprimorar sua capacidade de resolver esses problemas. Algoritmos de nicho são métodos desenvolvidos especificamente para lidar com múltiplos ótimos globais, facilitando a identificação de soluções globais. O objetivo principal foi reduzir a necessidade de uma população grande para encontrar os ótimos globais de funções multimodais. Para isso, foram incorporadas modificações no cálculo do fator escalar e na equação de movimento, além da inclusão de estratégias de busca local utilizando o algoritmo *Hill Climbing* modificado. Os experimentos foram realizados em funções de teste multimodais presentes na literatura, comparando o desempenho das modificações com o algoritmo original. Os resultados demonstraram que as propostas reduziram satisfatoriamente o número de partículas necessárias e aumentaram a eficiência do algoritmo na identificação de mínimos globais, especialmente em funções de alta complexidade. Concluiu-se que a combinação de estratégias de busca local e ajustes no movimento das partículas melhorou a performance do FER-PSO, tornando-o um algoritmo mais robusto e consistente para problemas de otimização multimodal.

Palavras-chave: FER-PSO; Otimização multimodal; Busca local; Hill Climbing.

ABSTRACT

This study aims to implement and modify the FER-PSO algorithm (*Fitness Euclidean-distance Ratio Particle Swarm Optimization*), a niche algorithm used to solve multimodal optimization problems, with the aim of improving its ability to solve these problems. Niche algorithms are methods developed specifically to deal with multiple global optima, making it easier to identify global solutions. The main objective this study was to reduce the need for a large population to find the global optima of multimodal functions. To this end, modifications were incorporated into the calculation of the scalar factor and the equation of motion, as well as the inclusion of local search strategies using the modified *Hill Climbing* algorithm. Experiments were carried out on multimodal test functions found in the literature, comparing the performance of the modifications with the original algorithm. The results showed that the proposals satisfactorily reduced the number of particles required and increased the algorithm's efficiency in identifying global minima, especially in highly complex functions. It was concluded that the combination of local search strategies and adjustments to particle movement improved the performance of FER-PSO, making it a more robust and consistent algorithm for multimodal optimization problems.

Keywords: FER-PSO; Multimodal optimization; Local search; Hill Climbing.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DC	Deterministic Crowding (Aglomeração Determinística)
DE	Differential Evolution (Evolução Diferencial)
FDR-PSO	Fitness-Distance-Ratio Particle Swarm Optimization (Otimização por Enxame de Partículas com Razão Aptidão-Distância)
FER-PSO	Fitness Euclidean-distance Ratio Particle Swarm Optimization (Otimização por Enxame de Partículas com Razão Aptidão-Distância Euclidiana)
GA	Genetic Algorithm (Algoritmo Genético)
gbest	Global Best Position (Melhor Posição Global)
K-Means	Algoritmo de Clusterização Particional (K-Means)
pbest	Personal Best Position (Melhor Posição Pessoal)
PSO	Particle Swarm Optimization (Otimização por Enxame de Partículas)
RTS	Restrictive Tournament Selection (Seleção por Torneio Restritiva)
SPSO	Species-based Particle Swarm Optimization (Otimização por Enxame de Partículas Baseada em Espécies)
SR	Success Rate (Taxa de sucesso)

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, será introduzido o conceito de otimização como área de estudo, abordaremos problemas de otimização escalar e apresentaremos os conceitos de computação evolutiva e heurísticas.

1.1 CONTEXTUALIZAÇÃO

A otimização é uma área do conhecimento que tem como objetivo encontrar a melhor solução possível para um problema dentro de um conjunto de soluções viáveis, utilizando critérios específicos. Um problema de otimização é definido por uma função objetivo que precisa ser minimizada ou maximizada, sujeita ou não a restrições que limitam o espaço de busca. De acordo com Takahashi (TAKAHASHI, 2007), a otimização envolve a aplicação de um conjunto de métodos e ferramentas que podem ser empregados em diversos contextos, como a engenharia elétrica, controle de processos industriais, economia e até mesmo epidemiologia. Embora esses problemas tenham diferentes contextos práticos, uma vez formulados matematicamente, eles compartilham uma estrutura comum e são resolvidos por meio das mesmas técnicas de otimização. Diante da abrangência da otimização, o foco principal será nos problemas de otimização escalar, que são importantes para entender os métodos que serão discutidos neste trabalho.

1.2 OTIMIZAÇÃO ESCALAR

A otimização escalar refere-se a problemas em que se deseja minimizar ou maximizar uma única função objetivo escalar, sujeita a possíveis restrições. Matematicamente, um problema de otimização escalar pode ser formulado como (TAKAHASHI, 2007):

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{sujeito a:} \quad \begin{cases} g(\mathbf{x}) \leq 0 \\ h(\mathbf{x}) = 0 \end{cases} \qquad \text{(Eq. 1.1)}$$

Onde:

- $\mathbf{x} \in \mathbb{R}^n$ é o vetor de variáveis de decisão;
- $f : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo a ser minimizada;
- $g : \mathbb{R}^n \to \mathbb{R}$ são as restrições de desigualdade;
- $h : \mathbb{R}^n \to \mathbb{R}$ são as restrições de igualdade;
- $\mathbf{x}^*$ é a solução ótima do problema.

Nesta formulação, o objetivo é encontrar o vetor $\mathbf{x}^*$ que minimiza a função objetivo $f(\mathbf{x})$, enquanto satisfaz as restrições $g(\mathbf{x})$ e $h(\mathbf{x})$. Problemas de otimização escalar estão presentes em diversas áreas, como engenharia, economia e ciência de dados, e são o foco principal das heurísticas que serão discutidas neste trabalho (TAKAHASHI, 2007).

1.3 HEURÍSTICAS E COMPUTAÇÃO EVOLUTIVA

Alguns problemas de otimização apresentam características que dificultam a aplicação de métodos determinísticos tradicionais. Entre esses métodos, destacam-se o método de Newton (POLYAK, 2007) e o gradiente descendente (WANG; YAN; ZHANG, 2021), que utilizam informações das derivadas da função objetivo para encontrar o mínimo ou máximo. O método do gradiente descendente, por exemplo, busca iterativamente a direção oposta ao gradiente da função objetivo, movendo-se em direção ao ponto de menor valor. Já o método de Newton utiliza tanto a primeira quanto a segunda derivada (Hessiana) da função, permitindo uma convergência mais rápida em funções convexas. No entanto, esses métodos requerem que a função seja continuamente diferenciável em todo o domínio e podem enfrentar dificuldades na presença de muitos ótimos locais, que podem confundir o processo de busca.

Sendo assim, para problemas mais complexos, são utilizadas heurísticas, que são procedimentos que, embora não garantam a obtenção da solução ótima exata, buscam soluções de qualidade dentro de um espaço de busca complexo (CUNHA; TAKAHASHI; ANTUNES, 2012). Essas heurísticas são utilizadas por não dependerem de informações prévias da função, podendo ser melhor aplicadas a problemas em que a função não seja diferenciável ou possua um espaço de busca mais complexo.

A principal inspiração para a criação das heurísticas que serão citadas nesse trabalho vem de processos biológicos observados na natureza, como a busca coordenada por alimentos realizada por formigas ou o movimento coletivo de um enxame de pássaros mudando de *habitat*. Esses processos envolvem a execução de ações simples por múltiplos agentes que interagem entre si e com o ambiente, resultando em soluções adaptativas coletivas. Por isso, tais algoritmos são frequentemente denominados bioinspirados (JONG, 2006; GOLDBERG, 1989; CUNHA; TAKAHASHI; ANTUNES, 2012).

Entre os algoritmos evolutivos mais conhecidos estão os Algoritmos Genéticos (GA, sigla em inglês) (HOLLAND, 1992; GOLDBERG, 1989) e a Evolução Diferencial (DE, sigla em inglês) (STORN; PRICE, 1997). Além disso, o algoritmo de Otimização por Enxame de Partículas (PSO, sigla em inglês) (KENNEDY; EBERHART, 1995) é um método populacional que, assim como os algoritmos evolutivos, possui uma capacidade maior de realizar buscas globais. Esses e outros métodos possuem uma capacidade maior de realizar buscas globais, sendo mais eficientes em problemas com muitas variáveis e múltiplos mínimos, onde os métodos determinísticos podem falhar.

1.4 APLICAÇÕES DA OTIMIZAÇÃO

Problemas de otimização estão presentes em diversas áreas do conhecimento e têm um papel fundamental na resolução de desafios complexos. Na engenharia, por exemplo, a otimização é aplicada no desenvolvimento de designs eficientes de estruturas, na melhoria de processos produtivos e na minimização de custos de materiais (RAO, 2009). Na área de logística, técnicas de otimização são utilizadas em problemas como roteamento de veículos, planejamento de rotas aéreas e gerenciamento de cadeias de suprimentos, visando reduzir tempos e custos operacionais (CORDEAU et al., 2007; DANTZIG; RAMSER, 1959).

1.5 OTIMIZAÇÃO MULTIMODAL E ALGORITMOS MULTIMODAIS

Problemas de otimização multimodal envolvem funções objetivo que possuem múltiplos pontos de mínimos locais e/ou globais, chamadas de funções multimodais. Assim como os problemas de otimização mono-objetivo, a otimização multimodal possui diversas aplicações no mundo real, como a resolução de sistemas de equações lineares (BRITS; ENGELBRECHT; BERGH, 2002b) e o reconhecimento de padrões em imagens (DELIBASIS; ASVESTAS; MATSOPOULOS, 2010).

Nesse contexto de otimização multimodal, surgem os algoritmos de nicho, que são modificações das heurísticas bioinspiradas anteriormente citadas como o Algoritmo Genético (GA sigla em inglês), a Evolução Diferencial (DE sigla em inglês) e a Otimização por Enxame de Partículas (PSO sigla em inglês) — baseadas no conceito de nichos ecológicos, ou seja, espaços onde grupos de indivíduos com características semelhantes coexistem. Essa ideia é incorporada às heurísticas, tornando-as capazes de manter a diversidade e preservar populações em pontos ótimos (MAHFOUD, 1996). Dessa forma, ao formar grupos de indivíduos com características similares, a população consegue explorar melhor o espaço de busca e encontrar mais pontos de mínimo da função, fazendo com que os algoritmos de nicho se destaquem como soluções interessantes para problemas multimodais, especialmente em espaços de busca mais complexos.

Na década de 1970, surgiram os primeiros métodos de nicho, como o *Fitness Sharing*, proposto por Holland em 1975, no qual a aptidão de cada indivíduo é reduzida com base em sua proximidade a outros indivíduos, visando preservar a diversidade da população (HOLLAND, 1992). Outro método dessa época foi o *Crowding*, introduzido por De Jong também em 1975 (JONG, 1975), em que descendentes são comparados a uma amostra da população e substituem indivíduos mais semelhantes, mantendo soluções distintas em diferentes nichos.

Nos anos 1990, surgiram avanços como o *Deterministic Crowding* (DC), proposto por Mahfoud em 1995, que elimina erros presentes no *Crowding* original, permitindo que múltiplos picos sejam mantidos sem a necessidade de parâmetros predefinidos para os nichos (MAHFOUD, 1996). Nesse período, também foi introduzido o *Restrictive Tournament*

Selection (RTS), um método que compara candidatos próximos para manter a diversidade, enquanto penaliza soluções com base na proximidade de outras (HARIK, 1995).

Para o algoritmo PSO, foram introduzidos métodos como o *NichePSO* e o *Species-based PSO* (SPSO) em 2005, que preservavam subpopulações de partículas em nichos distintos, permitindo a solução de problemas multimodais (BRITS; ENGELBRECHT; BERGH, 2002a; LI, 2004). Outro trabalho relevante é o artigo (LI, 2010), que introduz uma variante do PSO baseada em uma topologia de anel. Essa abordagem permite a formação de subpopulações estáveis em diferentes nichos, promovendo a diversidade e a identificação de múltiplos ótimos sem a necessidade de parâmetros utilizados em outros algoritmos de nicho.

Mais um artigo que implementa um PSO para funções multimodais é o FER-PSO (*Fitness Euclidean-distance Ratio Particle Swarm Optimization*) (LI, 2007). O FER-PSO introduz uma métrica que combina o valor de *fitness* (uma medida da qualidade para uma solução) com a distância Euclidiana, direcionando as partículas para os ótimos locais mais próximos e aptos. O algoritmo utiliza duas subpopulações: o *explorer-swarm*, responsável por explorar novas regiões do espaço de busca, e o *memory-swarm*, que preserva as melhores soluções já encontradas.

O artigo *Finding multiple global optima exploiting differential evolution's niching capability* (EPITROPAKIS; PLAGIANAKOS; VRAHATIS, 2011) propõe novas estratégias de mutação no algoritmo da Evolução Diferencial (DE sigla em inglês), explorando a capacidade de nicho sem a necessidade de parâmetros adicionais. Essas estratégias utilizam informações da vizinhança das soluções para manter múltiplos ótimos globais, demonstrando desempenho competitivo em funções multimodais complexas.

1.6 PERGUNTAS NORTEADORAS

Durante o levantamento dos trabalhos relacionados, foi observado que, na maioria das vezes, algoritmos de nicho requerem um número muito elevado de partículas na população para alcançar um desempenho satisfatório em funções mais complexas. Essa necessidade de grandes populações pode aumentar significativamente o custo computacional e o tempo de execução, limitando a aplicabilidade dos métodos em problemas com muitas dimensões. Diante disso, surgem as seguintes perguntas que nortearam a construção deste estudo: é possível aprimorar um algoritmo de nicho de modo a reduzir a necessidade de uma grande população? A busca local diminui a demanda por indivíduos na população? O foco do presente trabalho será no algoritmo de nicho FER-PSO (LI, 2007).

1.7 OBJETIVOS

a) Implementar um tipo de busca local no algoritmo FER-PSO a fim de diminuir o número de partículas necessárias para encontrar os mínimos das funções;

b) Medir o impacto que a busca local causa no desempenho do FER-PSO.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo desta seção é introduzir conceitos que serão utilizados e mencionados ao longo do trabalho, como a explicação do algoritmo PSO original, o conceito de busca local e a abordagem empregada para possibilitar o retorno de todos os mínimos encontrados pelo algoritmo de nicho.

2.1 PARTICLE SWARM OPTIMIZATION (PSO)

O algoritmo PSO é um dos métodos heurísticos existentes e foi desenvolvido pela primeira vez em 1995 por James Kennedy e Russell Eberhart (KENNEDY; EBERHART, 1995). Ele foi inspirado no comportamento social de revoadas de pássaros e cardumes de peixes em busca de alimento, conforme proposto por Frank Heppner. Como o PSO baseia-se na ideia de um comportamento social, temos um conjunto de agentes que interagem entre si "compartilhando" informações sobre o espaço de busca a fim de encontrar o melhor ponto para a função. Primeiramente, o PSO foi desenvolvido para problemas de otimização com variáveis contínuas; hoje em dia pode ser aplicado em diversos domínios, incluindo saúde (diagnóstico inteligente), meio ambiente (monitoramento de vegetação selvagem) e aplicações industriais (otimização de sistemas de energia) (GAD, 2022).

No PSO, cada partícula é uma solução candidata ao problema que queremos resolver que possui uma velocidade e uma posição. A partícula se move em um espaço de busca com o objetivo de encontrar o ótimo global de uma função objetivo. As partículas interagem entre si e se ajustam de acordo com informações obtidas de suas explorações do espaço de busca. A atualização das posições de cada partícula e suas velocidades são feitas utilizando as duas equações abaixo:

$$v_i = w \cdot v_i + c_1 \cdot \text{rand}() \cdot (pbest_i - x_i) + c_2 \cdot \text{rand}() \cdot (gbest - x_i) \quad \text{(Eq. 2.1)}$$

$$x_i = x_i + v_i \quad \text{(Eq. 2.2)}$$

Onde:

- v_i é o vetor que representa a velocidade da partícula i;
- w é o fator de inércia, que define o quanto a velocidade atual da partícula influencia o cálculo da sua nova velocidade;
- c_1 é o coeficiente cognitivo, que determina o quanto a melhor posição já encontrada pela própria partícula influencia o cálculo da nova velocidade;
- c_2 é o coeficiente social, que controla a influência da melhor posição alcançada pelo enxame no cálculo da nova velocidade da partícula;

- `rand()` é um número aleatório entre 0 e 1;
- $pbest_i$ é a melhor posição encontrada pela partícula i;
- $gbest$ é a melhor posição global conhecida pelo grupo;
- x_i é o vetor que representa a posição atual da partícula i.

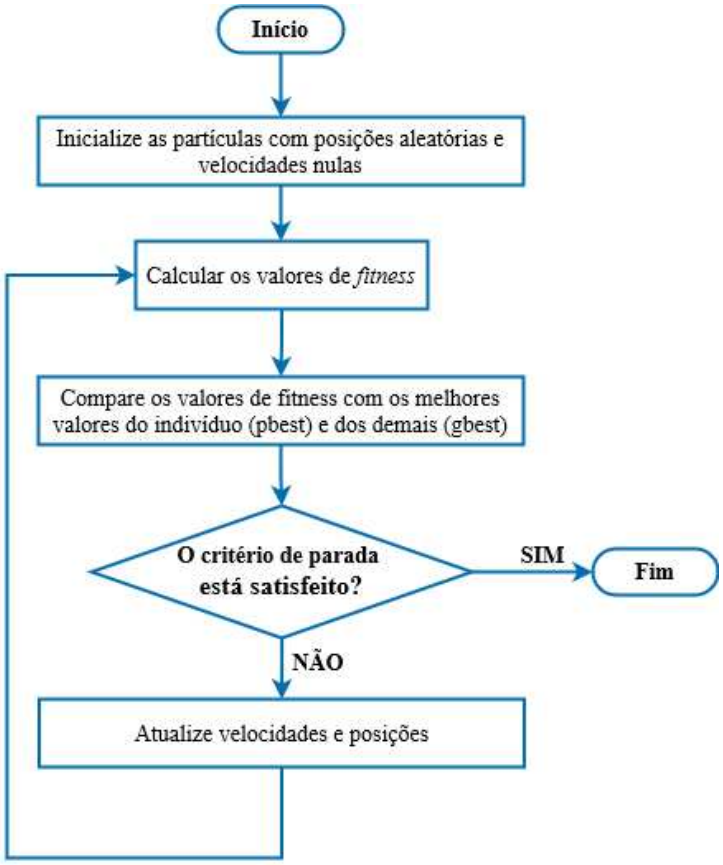
2.1.1 Funcionamento do PSO

O PSO é um algoritmo iterativo, no qual um conjunto de partículas se movimenta pelo espaço de busca, ajustando suas posições e velocidades com base em uma condição de parada. Essa condição de parada pode ser definida por um número máximo de iterações ou por uma quantidade máxima de avaliações de função permitida para o algoritmo. Antes de iniciar a execução do algoritmo, é necessário definir alguns parâmetros que serão utilizados durante a execução.

Os parâmetros incluem:

- Número de partículas;
- w: fator de inércia;
- c_1 e c_2: coeficientes cognitivo e social, respectivamente;
- Dimensão do problema;
- Limite superior do espaço de busca;
- Limite inferior do espaço de busca;
- Número máximo de iterações ou avaliações de função.

Figura 1 – Fluxo de execução do algoritmo PSO



Fonte: Adaptado de (SISSINVOU et al., 2022)

1. **Inicialização do Enxame**
   A execução do algoritmo começa com a inicialização do enxame, que consiste em definir as posições iniciais e as velocidades de cada partícula dentro do espaço de busca. Essas posições iniciais são geradas de forma aleatória dentro dos limites do problema e as velocidades são inicializadas nulas.

2. **Avaliação da Função Objetivo**
   A cada iteração, a posição atual de cada partícula é avaliada por meio da função objetivo. Esta função retorna um valor de *fitness* que determina quão boa é a solução representada por aquela partícula. A função objetivo pode variar dependendo do problema sendo resolvido, no caso de minimização, o objetivo é encontrar a partícula com o menor valor de *fitness*. No caso da primeira iteração, as posições iniciais das partículas são automaticamente consideradas as melhores conhecidas até o momento, uma comparação é feita em toda a população para gerar o *global best*.

3. **Atualização da Melhor Posição**
   Após a avaliação, cada partícula compara o valor da função objetivo com sua melhor posição conhecida (*pbest*). Se a nova posição for melhor que a *pbest*, a partícula

atualiza sua *pbest*. Além disso, a melhor posição global (*gbest*), que é compartilhada por todo o enxame, também é atualizada se uma partícula encontra uma solução melhor que a *gbest* atual.

4. **Atualização das Velocidades e Posições**
   Com base nas informações coletadas até aquele momento (*pbest* e *gbest*), cada partícula terá o seu vetor de velocidade v_i atualizado utilizando a Eq. 2.1 mostrada anteriormente.

   Após isso, a partícula terá a sua posição x_i atualizada utilizando a Eq. 2.2, somando a nova velocidade calculada à posição atual da partícula.

   Neste passo, é importante verificar a nova posição gerada para a partícula e garantir que ela permaneça dentro do espaço de busca. O tratamento mais comum para isso é restringir a posição da partícula aos limites do espaço de busca, evitando que ela ultrapasse esses limites.

5. **Critério de Parada**
   O PSO continua executando até que um critério de parada seja atingido, como um número pré definido de iterações ou avaliações de função, ou uma condição de convergência da população (por exemplo, quando as mudanças nas posições das partículas entre as iterações se tornam insignificantes).

Algorithm 1 Pseudocógido do PSO utilizado como base

Require: TamPop, Dim, LSup, LInf , w, c_1, c_2
Ensure: Melhor posição global (gbest)
Inicialize o enxame de partículas:
for cada partícula i no enxame **do**
    Inicialize o vetor de posição $\mathbf{x}_i$ aleatoriamente no espaço de busca
    Inicialize o vetor de velocidade $\mathbf{v}_i$ aleatoriamente
    Defina a posição pessoal ótima $\text{pbest}_i \leftarrow \mathbf{x}_i$
    Calcule o valor da função objetivo $f(\mathbf{x}_i)$
    if $f(\mathbf{x}_i) < f(\text{pbest}_i)$ **then**
        $\text{pbest}_i \leftarrow \mathbf{x}_i$
    end if
end for
Defina a melhor posição global gbest $\leftarrow \arg\min_i f(\text{pbest}_i)$
while Critério de parada não satisfeito **do**
    for cada partícula i no enxame **do**
        Atualize a velocidade da partícula i:

$$\mathbf{v}_i \leftarrow v_i = w \cdot v_i + c_1 \cdot \text{rand}() \cdot (pbest_i - x_i) + c_2 \cdot \text{rand}() \cdot (gbest - x_i)$$

        Atualize a posição da partícula i:

$$\mathbf{x}_i \leftarrow x_i = x_i + v_i$$

        Calcule o valor da função objetivo $f(\mathbf{x}_i)$
        if $f(\mathbf{x}_i) < f(\text{pbest}_i)$ **then**
            $\text{pbest}_i \leftarrow \mathbf{x}_i$
        end if
        if $f(\text{pbest}_i) < f(\text{gbest})$ **then**
            $\text{gbest} \leftarrow \text{pbest}_i$
        end if
    end for
end while
return gbest como a melhor solução encontrada

2.2 BUSCA LOCAL

A busca local é uma estratégia utilizada em problemas de otimização onde se busca uma solução ótima ou muito próxima de ótima para o problema sendo resolvido. Métodos de busca local buscam refinar soluções encontradas ou, então, atuar no auxílio para a melhor exploração do espaço de busca e uma convergência mais rápida em algoritmos de otimização. A busca local pode ser bastante efetiva em heurísticas, como as citadas anteriormente, AG e PSO que costumam explorar o espaço de busca globalmente e usar a busca local para intensificar a exploração em áreas promissoras.

A ideia central da Busca Local é aplicar pequenas perturbações em uma solução atual, gerando soluções vizinhas e selecionando a melhor entre elas. Em cada iteração, a solução

é substituída por uma de suas vizinhas, desde que esta apresente um valor de função objetivo melhor ao da solução atual. Esse processo é repetido até que uma condição de parada seja atingida, como o número máximo de iterações ou a estabilização da solução (HOOS; TSANG, 2006).

2.2.1 Hill Climbing

O algoritmo de *hill climbing* é uma heurística que pode ser aplicado na resolução de problemas de otimização escalar como uma busca local. Seu funcionamento é bem simples e baseia-se em gerar perturbações na solução atual em direções aleatórias, definidas por um tamanho do passo (δ), que determina a amplitude das perturbações para gerar novas soluções vizinhas. Após gerar uma nova solução, o algoritmo avalia se esta é melhor que a anterior. Em problemas de maximização, a melhor posição é aquela com o maior valor de *fitness*; em problemas de minimização, é a que possui o menor valor (HOOS; TSANG, 2006).

Inicialmente, é escolhida uma solução no espaço de busca, que geralmente corresponde à partícula atual da iteração do algoritmo (x_i). Em seguida, é calculado o valor da função objetivo $f(x_i)$ para essa solução.

Para gerar novas soluções vizinhas, cria-se um vetor de direção aleatório $\mathbf{d}$, onde cada componente de $\mathbf{d}$ é um número dentro de um intervalo definido, por exemplo, entre -1 e 1. A nova solução x_i' é calculada pela equação:

$$x_i' = x_i + \delta \cdot \mathbf{d} \qquad \text{(Eq. 2.3)}$$

Após obter x_i', o algoritmo verifica se esta nova solução está dentro dos limites do espaço de busca. Se necessário, a posição x_i' é ajustada para garantir que respeite as restrições do espaço de busca do problema. Em seguida, é calculado o valor da função objetivo $f(x_i')$ para a nova solução.

A etapa seguinte consiste em comparar as soluções. Para problemas de minimização, se $f(x_i') < f(x_i)$, a nova solução x_i' é considerada melhor, e x_i é atualizado com x_i'. Em problemas de maximização, se $f(x_i') > f(x_i)$, a nova solução é melhor, e x_i é atualizado com x_i'. Caso a nova solução não seja melhor que a atual, x_i é mantido inalterado, e o algoritmo avança para a próxima iteração (HOOS; TSANG, 2006).

Esse processo é repetido até que um critério de parada seja atingido, similar ao utilizado no algoritmo PSO. O critério de parada pode ser definido por um número máximo de iterações, uma quantidade máxima de avaliações destinadas à busca local ou qualquer outra condição. De maneira geral, o algoritmo continuará refinando as soluções até que o critério seja satisfeito.

No presente trabalho, o *hill climbing* será aplicado individualmente a cada partícula x_i da população, durante um número definido de avaliações da função objetivo. Inicialmente,

uma perturbação será gerada para criar um único vizinho x_i' como uma possível melhor solução. No Capítulo 3, será apresentada uma modificação nessa implementação do *hill climbing*. O algoritmo 2 a seguir ilustra o pseudocódigo dessa implementação do *hill climbing* padrão.

Algorithm 2 Pseudocódigo Hill climbing

Require: Posição inicial x_i no espaço de busca, Tamanho do passo (δ)
Ensure: Melhor solução encontrada x_i

while critério de parada não satisfeito **do**
    Obtenha uma nova posição x_i' a partir da posição atual x_i
    $x_i' \leftarrow x_i' = x_i + \delta \cdot \mathbf{d}$
    if x_i' está fora dos limites do espaço de busca **then**
        Ajuste x_i' para que respeite as restrições do espaço de busca
    end if
    Calcule o valor da função objetivo $f(x_i')$ para a nova solução
    if problema de minimização **then**
        if $f(x_i') < f(x_i)$ **then**
            $x_i \leftarrow x_i'$
        else
            Mantenha x_i inalterado
        end if
    else if problema de maximização **then**
        if $f(x_i') > f(x_i)$ **then**
            $x_i \leftarrow x_i'$
        else
            Mantenha x_i inalterado
        end if
    end if
end while
return x_i como a melhor solução encontrada

2.3 CLUSTERIZAÇÃO

A clusterização é uma técnica de análise de dados cujo objetivo é dividir um conjunto de dados em subconjuntos ou grupos denominados *clusters*, de forma que os elementos dentro de cada cluster apresentem maior similaridade entre si do que com os de outros clusters. Conforme abordado em *Survey of clustering algorithms* (XU; WUNSCH, 2005), a clusterização pode ser utilizada em diversas áreas, incluindo aprendizado de máquina, reconhecimento de padrões e mineração de dados. O processo de clusterização é não supervisionado, o que significa que não há rótulos predefinidos nos dados; o objetivo é descobrir estruturas naturais ou padrões implícitos nos dados.

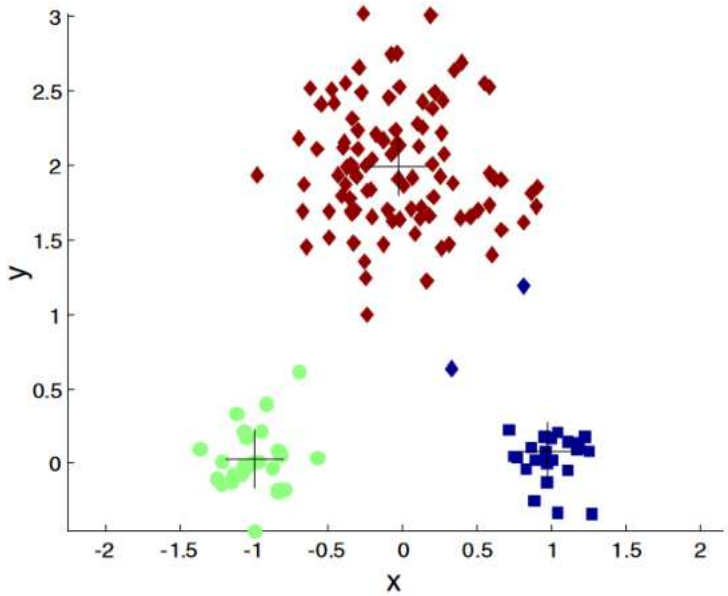
Dentre os tipos de algoritmos de clusterização, destacam-se os algoritmos hierárquicos e os particionais. Os primeiros geram uma estrutura hierárquica em forma de árvore, enquanto os últimos dividem os dados diretamente em um número predefinido de clusters.

No contexto de algoritmos particionais, o *K-Means* é um dos métodos mais conhecidos e amplamente utilizados (XU; WUNSCH, 2005). Neste trabalho, o algoritmo *K-Means* será incorporado com o objetivo de auxiliar o FER-PSO na identificação de diferentes subgrupos dentro do espaço de busca. Após essa identificação, o menor valor presente em cada um dos *clusters* encontrados será considerado como um dos melhores pontos identificado pelo enxame. Além disso, os centroides dos *clusters* gerados pelo *K-Means* também serão utilizados como base para uma métrica de desempenho do algoritmo, apresentada no Capítulo 4, que emprega os centroides para avaliar a qualidade das soluções encontradas.

2.3.1 K-Means

O *K-Means* é um algoritmo de clusterização particional que busca agrupar dados em k clusters distintos, onde k é um parâmetro que pode ser definido pelo usuário ou com a utilização de métodos como o método do cotovelo (*elbow method*) ou método *silhouette* para a definição do número ótimo de forma dinâmica. Após essa atribuição inicial, o *K-Means* recalcula os centroides com base na média dos pontos atribuídos a cada *cluster*, ajustando as fronteiras dos *clusters* de maneira iterativa. Esse processo é repetido até que os centroides não apresentem mudanças significativas ou até que um critério de parada seja atingido. Uma das principais vantagens do *K-Means* é a simplicidade de implementação e eficiência em termos computacionais, o que o torna aplicável a grandes conjuntos de dados (XU; WUNSCH, 2005).

Figura 2 – Exemplo de clusterização utilizando *K-Means*



Fonte: (TAN et al., 2021)

O *K-Means* opera de maneira iterativa e envolve duas etapas principais: atribuição e atualização. Na etapa de atribuição, cada ponto de dado x_i do conjunto $\mathcal{X}$ é atribuído ao

cluster C_j cujo centroide μ_j é o mais próximo, de acordo com uma métrica de distância, tipicamente a distância Euclidiana. Esta atribuição é feita da seguinte forma:

$$j = \arg\min_{1 \leq l \leq k} \|x_i - \mu_l\| \quad \text{(Eq. 2.4)}$$

Na etapa de atualização, os centroides de todos os *clusters* são recalculados para refletir a nova composição dos *clusters*. O centroide μ_j de um *cluster* C_j é atualizado pela média das posições dos pontos atribuídos a ele:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \quad \text{(Eq. 2.5)}$$

Essas duas etapas são repetidas de forma alternada até que os centroides não apresentem mudanças significativas entre iterações consecutivas ou até que um critério de parada, como um número máximo de iterações, seja atingido (TAN et al., 2021).

2.3.2 Método da Silhueta

O *Método da Silhueta* é uma técnica utilizada para avaliar a qualidade de uma clusterização e auxiliar na determinação do número ótimo de *clusters* em um conjunto de dados. Introduzido por Peter J. Rousseeuw em 1987, o método fornece uma medida de quão bem cada ponto de dado se encaixa no seu próprio *cluster* em comparação com outros *clusters* (ROUSSEEUW, 1987).

A silhueta de um ponto de dado combina medidas de coesão e separação, permitindo avaliar se o número de *clusters* k é adequado. O coeficiente de silhueta $s(i)$ para um ponto de dado i é definido como (TAN et al., 2021):

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad \text{(Eq. 2.6)}$$

Onde:

- $a(i)$ é a distância média entre o ponto i e todos os outros pontos do mesmo *cluster*. Essa medida de coesão indica o quão bem o ponto está associado ao seu próprio *cluster*.
- $b(i)$ é a menor distância média entre o ponto i e todos os pontos de qualquer outro *cluster* ao qual i não pertence. Essa medida de separação indica o quão distante o ponto está dos demais *clusters*.

O coeficiente de silhueta varia entre -1 e 1:

- Valores próximos de 1 indicam que o ponto está bem ajustado ao seu próprio *cluster* e mal ajustado aos *clusters* vizinhos.

- Valores próximos de 0 sugerem que o ponto está na fronteira entre dois *clusters*.
- Valores negativos indicam que o ponto pode ter sido atribuído ao *cluster* errado.

Cálculo do Coeficiente de Silhueta Médio:

Para avaliar a qualidade global da clusterização com k *clusters*, calcula-se o coeficiente de silhueta médio $\bar{s}$:

$$\bar{s} = \frac{1}{n} \sum_{i=1}^{n} s(i) \qquad \text{(Eq. 2.7)}$$

Onde n é o número total de pontos de dados.

Determinação do Número Ótimo de Clusters:

O *Método da Silhueta* pode ser usado para determinar o número ótimo de *clusters* k ao executar o algoritmo de clusterização para diferentes valores de k e calcular o coeficiente de silhueta médio para cada um. O valor de k que maximiza $\bar{s}$ é considerado o mais apropriado, pois indica que a clusterização resultante possui coesão interna e separação externa satisfatórias.

Procedimento para aplicar o Método da Silhueta:

1. **Executar o algoritmo de clusterização** (por exemplo, *K-Means*) para uma faixa de valores de k (por exemplo, de 2 a $k_{\max}$).
2. **Calcular o coeficiente de silhueta** $s(i)$ para cada ponto de dado em cada clusterização resultante.
3. **Calcular o coeficiente de silhueta médio** $\bar{s}$ para cada valor de k usando a Equação Eq. 2.7.
4. **Selecionar o número ótimo de *clusters*** k^* que maximiza $\bar{s}$:

$$k^* = \arg\max_k \ \bar{s} \qquad \text{(Eq. 2.8)}$$

No presente trabalho, o *Método da Silhueta* será utilizado em conjunto com o algoritmo *K-Means* para determinar de forma dinâmica o número ótimo de *clusters* k. Em vez de definir k previamente, o algoritmo explorará diferentes valores de k dentro de um intervalo máximo, calculando o coeficiente de silhueta médio. Dessa forma, o valor de k que maximiza $\bar{s}$ será selecionado, garantindo uma clusterização que melhor reflete as estruturas dos dados gerados pelo FER-PSO.

2.4 FITNESS EUCLIDEAN-DISTANCE RATIO (FER)

Como mencionado no Capítulo 1, o foco deste trabalho é avaliar se a inclusão de busca local (no caso, o algoritmo *Hill Climbing*) em um algoritmo de nicho pode reduzir a quantidade de partículas necessárias para encontrar todos os mínimos globais conhecidos em problemas de otimização multimodal. O algoritmo escolhido para este estudo é o FER-PSO (*Fitness Euclidean-distance Ratio Particle Swarm Optimization*) (LI, 2007), o qual é uma variante do PSO adaptada para problemas multimodais.

No FER-PSO, o espaço de busca é explorado por dois sub-enxames: o *memory-swarm* e o *explorer-swarm*. O *memory-swarm* é composto pelos melhores pontos já encontrados (*pbest*), enquanto o *explorer-swarm* inclui as posições atuais das partículas, responsáveis pela exploração de novas regiões do espaço de busca. Diferentemente do PSO tradicional, que utiliza um único melhor global (*gbest*) para orientar todas as partículas, o FER-PSO determina, para cada partícula, um vizinho mais próximo e apto ($\mathbf{p}_n$), definido pelo valor do *Fitness Euclidean-distance Ratio* (FER).

O cálculo do ($\mathbf{p}_n$) foi proposto no artigo *Optimization using particle swarms with near neighbor interactions* (VEERAMACHANENI et al., 2003), no qual é introduzido o conceito de interação entre vizinhos mais próximos. Nesse modelo, cada partícula é influenciada não apenas pela melhor posição global e pessoal, mas também pelas posições de vizinhos próximos que apresentem um valor de *fitness* superior. Esse método de cálculo do ($\mathbf{p}_n$) é denominado FDR (*Fitness-Distance-Ratio*), e o algoritmo desenvolvido com base nesse cálculo é chamado FDR-PSO (*Fitness-Distance-Ratio Particle Swarm Optimization*). Essa abordagem permite que as partículas explorem regiões mais amplas do espaço de busca, evitando a convergência prematura a mínimos locais. O FDR é calculado utilizando a seguinte equação:

$$\text{FDR}_{(j,i,d)} = \frac{f(\mathbf{p}_j) - f(\mathbf{x}_i)}{|p_{jd} - x_{id}|} \qquad \text{(Eq. 2.9)}$$

Onde:

- $f(\mathbf{p}_j)$ é o *fitness* da melhor posição conhecida pela partícula j,
- $f(\mathbf{x}_i)$ é o *fitness* da partícula i,
- p_{jd} e x_{id} representam a coordenada d-ésima da melhor posição da partícula j e da posição atual da partícula i, respectivamente, onde d representa a dimensão da coordenada no espaço do problema.

No FER-PSO, os autores do artigo *A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio* (LI, 2007) identificaram limitações no cálculo do FDR (VEERAMACHANENI et al., 2003), que, em certos casos, não selecionava o vizinho mais

adequado. Para contornar esse problema, eles modificaram o cálculo original, resultando no FER (*Fitness Euclidean-distance Ratio*), calculado pela Equação Eq. 2.10:

$$\mathrm{FER}(j,i) = \alpha \cdot \frac{f(\mathbf{p}_j) - f(\mathbf{p}_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|} \qquad \text{(Eq. 2.10)}$$

Onde:

- $f(\mathbf{p}_j)$ é o *fitness* da melhor posição conhecida pela partícula j,
- $f(\mathbf{p}_i)$ é o *fitness* da melhor posição conhecida pela partícula i,
- $\|\mathbf{p}_j - \mathbf{p}_i\|$ é a distância Euclidiana entre as melhores posições das partículas j e i,
- $\alpha = \frac{\|\mathbf{s}\|}{f(\mathbf{p}_g) - f(\mathbf{p}_w)}$ é um fator escalar que equilibra a influência do valor de *fitness* e da distância. Aqui, p_g e p_w são a melhor e a pior posições encontradas pelo enxame, e $\|\mathbf{s}\|$ é o tamanho do espaço de busca, que pode ser estimado pela diagonal do espaço de busca $\|\mathbf{s}\| = \sqrt{\sum_{k=1}^{\mathrm{Dim}} \left(x_k^{\mathrm{u}} - x_k^{\mathrm{l}}\right)^2}$, onde x_k^{u} e x_k^{l} representam os limites superior e inferior do espaço de busca, respectivamente, e k é a dimensionalidade do problema.

Além disso, ao invés de utilizar a posição atual $\mathbf{x}_i$, o FER-PSO considera a melhor posição conhecida pela partícula $\mathbf{p}_i$, tornando o cálculo mais estável. Ao utilizar a distância Euclidiana, o método evita problemas associados ao uso de diferenças dimensionais individuais. Dessa forma, cada partícula passa a ter seu próprio vizinho mais apto e próximo ($\mathbf{p}_n$), determinado pelo FER, no lugar de um único *gbest*.

A seleção do melhor vizinho com base no FER é descrita pelo Algoritmo 3. Como o cálculo do FER é feito para todas as partículas a complexidade é $O(N^2)$, onde N é o número de partículas, pois cada partícula é comparada com todas as demais.

Algorithm 3 Seleção do melhor vizinho com base no FER (LI, 2007)

Require: Lista de todas as partículas na população
Ensure: Melhor vizinho $\mathbf{p}_n$ para a partícula i

```
FER ← 0, temp ← 0, euDist ← 0
for j = 1 até Tamanho da População do
    Calcular a distância Euclidiana euDist entre p_i e o pbest da partícula j, p_j
    if euDist ≠ 0 then
        Calcular FER de acordo com a Equação Eq. 2.10
        if j = 1 then
            temp ← FER
        end if
        if FER > temp then
            temp ← FER
            p_n ← p_j
        end if
    end if
end for
return p_n
```

Por fim, com a adoção do *pn* calculado utilizando o Algoritmo 3 em vez do *gbest*, a equação de atualização da velocidade original do PSO mostrada anteriormente (Eq. 2.1) passa a assumir a forma da equação de velocidade do FER-PSO (LI, 2007):

$$v_i = w \cdot v_i + c_1 \cdot \text{rand}() \cdot (pbest_i - x_i) + c_2 \cdot \text{rand}() \cdot (p_n - x_i) \qquad \text{(Eq. 2.11)}$$

Com essa mudança, o termo p_n substitui o *gbest*, essa alteração permite que cada partícula seja direcionada ao melhor vizinho próximo, em vez de um ponto ótimo compartilhado globalmente, aumentando a diversidade da busca e a capacidade de exploração em funções multimodais (LI, 2007).

3 METODOLOGIA

O objetivo deste capítulo é detalhar as modificações e inclusões propostas no algoritmo FER-PSO (LI, 2007), com o intuito de atingir os objetivos definidos anteriormente no Capítulo 1.

3.1 MODIFICAÇÕES PROPOSTAS NO FER-PSO

Duas alterações principais foram propostas no algoritmo FER-PSO: uma modificação no cálculo do FER e outra na equação de movimento.

3.1.1 Primeira Modificação

A primeira modificação foi feita no cálculo do fator escalar α, que originalmente era definido como:

$$\alpha = \frac{\|\mathbf{s}\|}{f(\mathbf{p}_\mathrm{g}) - f(\mathbf{p}_\mathrm{w})} \qquad \text{(Eq. 3.1)}$$

Onde:

- $\|\mathbf{s}\| = \sqrt{\sum_{k=1}^{\mathrm{Dim}} \left(x_k^\mathrm{u} - x_k^\mathrm{l}\right)^2}$

Um novo termo foi proposto nesse trabalho, denominado *diversidade*, foi inserido no cálculo do fator escalar. Assim, a nova expressão para α é dada por:

$$\alpha = (1 + \mathrm{Diversidade}) \cdot \frac{\|\mathbf{s}\|}{f(\mathbf{p}_\mathrm{g}) - f(\mathbf{p}_\mathrm{w})} \qquad \text{(Eq. 3.2)}$$

O objetivo desta modificação é ajustar o impacto do movimento das partículas de acordo com a diversidade da população. A diversidade da população é calculada como a média das posições de todas as partículas. Espera-se que esse ajuste promova uma maior exploração do espaço de busca quando as partículas estiverem mais dispersas, e uma exploração mais focada quando elas convergirem. Além disso, esse fator de diversidade atua como uma ferramenta para evitar que as partículas fiquem presas em mínimos locais.

3.1.2 Segunda Modificação

A segunda modificação foi realizada na equação de movimento do FER-PSO (Eq. 2.11), com a introdução de um novo termo chamado matriz de comunicação $\mathbf{C}$ na parte social do cálculo da velocidade. Com essa inclusão, a equação de movimento é reescrita como:

$$\mathbf{v}_i = w \cdot \mathbf{v}_i + c_1 \cdot \text{rand}() \cdot (\mathbf{pbest}_i - \mathbf{x}_i) + \mathbf{C} \cdot (c_2 \cdot \text{rand}() \cdot (\mathbf{p}_n - \mathbf{x}_i)) \qquad \text{(Eq. 3.3)}$$

Onde:

- $\mathbf{C}$ é a matriz de comunicação.

A matriz de comunicação $\mathbf{C}$ é uma matriz diagonal de dimensão $d \times d$, sendo d a dimensionalidade do problema. Ela é construída em cada iteração do algoritmo de acordo com a seguinte regra:

$$C_{ii} = \begin{cases} 1, & \text{se } r_i \leq \tau_{\text{com}} \\ 0, & \text{caso contrário} \end{cases}$$

para $i = 1, \ldots, d$, onde:

- r_i é uma variável aleatória amostrada da distribuição normal para cada dimensão do vetor $\mathbf{x_i}$
- τ_{com} é a taxa de comunicação, um parâmetro que define a probabilidade de comunicação em cada dimensão.

A função da matriz $\mathbf{C}$ é controlar a influência social sobre cada partícula, selecionando aleatoriamente quais componentes da atualização social serão consideradas. Para cada dimensão, há uma probabilidade τ_{com} de que a influência do vizinho mais apto $\mathbf{p}_n$ seja aplicada naquela dimensão, e uma probabilidade $1 - \tau_{\text{com}}$ de que seja ignorada. Isso significa que a matriz $\mathbf{C}$ atua como um filtro estocástico, permitindo que as partículas mantenham certa independência em algumas dimensões, o que pode favorecer a exploração do espaço de busca.

Essa abordagem foi inspirada no trabalho de Miranda *et al.* (MIRANDA; ALVES, 2013), no qual é introduzida uma topologia de comunicação probabilística. Nesse modelo, a interação entre partículas ocorre com uma certa probabilidade, permitindo que, em algumas iterações, as partículas ignorem a informação do *gbest*. Isso promove maior diversidade na população e auxilia na prevenção da convergência prematura para ótimos locais. No contexto deste trabalho, a matriz de comunicação $\mathbf{C}$ é utilizada para controlar a influência do vizinho mais apto $\mathbf{p}_n$, análogo ao papel do *gbest* no PSO tradicional.

A escolha da taxa de comunicação τ_{com} influencia o equilíbrio entre a exploração global e local do algoritmo. Valores moderados de τ_{com}, como 0,6, 0,7 e 0,8, mostraram-se eficientes em melhorar o desempenho do algoritmo no estudo de Miranda *et al.* (MIRANDA; ALVES, 2013). Esses valores permitem uma boa combinação entre a influência social e a independência das partículas. Com base nesses resultados, a incorporação da

matriz de comunicação no presente trabalho foi motivada pela observação de melhorias no desempenho do algoritmo durante os testes realizados com essa estratégia.

Ao utilizar a matriz de comunicação $\mathbf{C}$, espera-se que as partículas tenham maior capacidade de explorar novas regiões do espaço de busca, pois a influência do vizinho mais apto é aplicada apenas em algumas dimensões. Isso pode facilitar a evasão de mínimos locais e contribuir para uma melhor cobertura do espaço de busca, aumentando a probabilidade de encontrar os ótimos globais da função objetivo.

3.2 MODIFICAÇÕES NA BUSCA LOCAL

Conforme citado na Seção 2.2, a heurística *Hill Climbing* será incorporada ao FER-PSO neste trabalho como um tipo de busca local. Foram feitas duas modificações neste algoritmo para adaptá-lo a diferentes tipos de espaços de busca.

3.2.1 Primeira Modificação

A primeira mudança refere-se ao tamanho do passo (δ). O tamanho do passo é um parâmetro necessário para o *Hill Climbing*, pois determina a amplitude das perturbações para gerar novas possíveis posições ótimas. O valor de δ pode influenciar diretamente a performance do algoritmo, dependendo do tamanho do espaço de busca. Em espaços de busca menores, como com limites entre 0 e 1, um passo grande pode afastar a partícula de uma posição melhor. Em contrapartida, em espaços de busca maiores, um passo grande pode ser benéfico para aproximar a partícula de regiões promissoras.

Para evitar uma seleção inadequada de δ, foi implementada uma abordagem adaptativa. Utilizamos o tamanho do espaço de busca $\|\mathbf{s}\|$, conforme definido anteriormente, como base para o cálculo do passo. No início da execução do PSO, o tamanho do passo é definido como 10% do espaço de busca. Ao final da execução, o tamanho do passo é reduzido para 1% do espaço de busca. Esses valores demonstraram resultados satisfatórios para os tamanhos do espaço de busca diversos e foram utilizados nesse trabalho.

Essa estratégia permite que, no início da execução do PSO com busca local, o enxame apresente um comportamento mais exploratório. À medida que a execução progride, o comportamento torna-se mais concentrado nas bacias de atração, visando refinar as boas soluções já encontradas.

3.2.2 Segunda Modificação

A segunda adição ao algoritmo *Hill Climbing* visa melhorar sua capacidade de convergência para um ponto ótimo. Foi introduzida uma variável denominada *Multi Start*, que define a quantidade de tentativas que uma partícula fará para encontrar uma posição melhor que a atual. Por exemplo, durante a execução do PSO com busca local, se a quantidade de tentativas for 5, e a partícula atual for $\mathbf{x}_i = [0, 1]$ com $f(\mathbf{x}_i) = 3$, enquanto

a posição ótima é $\mathbf{x}^* = [-1, -1]$ com $f(\mathbf{x}^*) = 0$, o algoritmo *Hill Climbing* tentará 5 vezes melhorar a posição da partícula. Ao final dessas tentativas, a melhor posição encontrada será mantida. Ou seja, com essa modificação, o algoritmo que antes gerava apenas 1 vizinho agora poderá gerar M vizinhos, onde M é o número de tentativas definido pela variável *Multi Start*. O Algoritmo 4 apresenta o pseudocódigo com essa modificação.

No presente trabalho, quando aplicado no FER-PSO, o *hill climbing* modificado terá duas variáveis de *Multi Start*, sendo uma delas para ser utilizada no início do algoritmo e outra ao fim. Para o início, a ideia é que o enxame explore mais o espaço de busca, tendo um valor de M maior somado a um passo grande, possibilitando uma convergência mais rápida ao gerar mais vizinhos. Ao fim, a ideia é que o enxame busque menos, com um valor de M menor em conjunto com um passo menor.

Algorithm 4 Pseudocódigo *Hill Climbing* com Modificações

Require: Posição inicial x_i no espaço de busca, Tamanho inicial do passo δ, Número de tentativas Multi Start M
Ensure: Melhor solução encontrada x_i

while critério de parada não satisfeito **do**
    $\delta \leftarrow$ Atualize o tamanho do passo adaptativamente com base no progresso da execução ▷ ver Seção 2.2
    $x'_i \leftarrow x_i$
    for $t = 0$ até $M - 1$ **do**
        Atualize $x_j \leftarrow x'_i = x_i + \delta \cdot \mathbf{d}$
        if x_j está fora dos limites do espaço de busca **then**
            Ajuste x_j para que respeite as restrições do espaço de busca
        end if
        Calcule o valor da função objetivo $f(x_j)$ para a nova solução
        if $f(x_j) < f(x'_i)$ **then**
            $x'_i \leftarrow x_j$
        end if
    end for
    if $f(x'_i) < f(x_i)$ **then**
        $x_i \leftarrow x'_i$
    end if
end while
return x_i como a melhor solução encontrada

Para este trabalho, o critério de parada adotado para o algoritmo foi o número de avaliações da função objetivo. Essa informação é essencial para compreender o pseudocódigo apresentado a seguir. Com a incorporação das modificações propostas no capítulo 3, o algoritmo 1 atualizado pode ser reescrito conforme descrito no algoritmo 5.

Algorithm 5 Pseudocódigo do FER-PSO modificado

Require: TamPop, Dim, LSup, LInf, w, c_1, c_2, $Eval_{max}$,$ProbComunicacao$, $BuscaLocal$
Ensure: Melhores soluções encontradas

$Eval \leftarrow 0$, $EspacoBusca \leftarrow \|\mathbf{s}\|$, $\delta_{\text{inicio}} \leftarrow EspacoBusca \times 0.1$, $\delta_{\text{fim}} \leftarrow EspacoBusca \times 0.01$, $M_{\text{inicio}} \leftarrow 10$, $M_{\text{fim}} \leftarrow 5$
Inicializar o enxame:
for cada partícula i no enxame **do**
    Inicializar posição $\mathbf{x}_i$ aleatoriamente no espaço de busca
    Inicializar velocidade $\mathbf{v}_i$ aleatoriamente
    Definir $\text{pbest}_i \leftarrow \mathbf{x}_i$
    Calcular $f(\mathbf{x}_i)$
    $Eval \leftarrow Eval + 1$
end for
while $Eval < Eval_{max}$ **do**
    for cada partícula i no enxame **do**
        Calcular o melhor vizinho $\mathbf{p}_n$ com base no FER:
        $\mathbf{p}_n \leftarrow$ **Seleção do Melhor Vizinho com base no valor FER** (Algoritmo 3 com modificações, para a partícula i)
        Calcular matriz de comunicação C utilizando a $ProbComunicacao$
        Atualizar velocidade $\mathbf{v}_i$:

$$\mathbf{v}_i \leftarrow \mathbf{v}_i = w \cdot \mathbf{v}_i + c_1 \cdot \text{rand}() \cdot (\mathbf{pbest}_i - \mathbf{x}_i) + \mathbf{C} \cdot (c_2 \cdot \text{rand}() \cdot (\mathbf{p}_n - \mathbf{x}_i))$$

        Atualizar posição $\mathbf{x}_i$:

$$\mathbf{x}_i \leftarrow x_i = x_i + v_i$$

        Aplicar limites do espaço de busca em $\mathbf{x}_i$ se necessário
        Calcular $f(\mathbf{x}_i)$
        $Eval \leftarrow Eval + 1$
        if $f(\mathbf{x}_i) < f(\text{pbest}_i)$ **then**
            $\text{pbest}_i \leftarrow \mathbf{x}_i$
        end if
        if $((\text{Eval} \leq \text{Eval}_{\text{max}})$ || $(\text{Eval} \geq 0.8 \times \text{Eval}_{\text{max}})$ & $(\text{BuscaLocal} == 1))$ **then**
            Aplicar Busca Local (Hill Climbing) na partícula i:
            Definir tamanho do passo $\delta \leftarrow \delta_{\text{fim}}$ se $\text{Eval} \geq 0.8 \times \text{Eval}_{\text{max}}$, caso contrário $\delta \leftarrow \delta_{\text{inicio}}$
            Definir tamanho de $M \leftarrow M_{\text{fim}}$ se $\text{Eval} \geq 0.8 \times \text{Eval}_{\text{max}}$, caso contrário $M \leftarrow M_{\text{inicio}}$
            $\mathbf{x}_i \leftarrow$ **Hill Climbing**$(\mathbf{x}_i, \delta, M)$
            Atualizar $f(\mathbf{x}_i)$
            Incrementar $Eval$ com o número de avaliações feitas na busca local
            if $f(\mathbf{x}_i) < f(\text{pbest}_i)$ **then**
                $\text{pbest}_i \leftarrow \mathbf{x}_i$
            end if
        end if
    end for
    if $Eval \bmod 10{,}000 = 0$ **then**
        Aplicar Clusterização *K-Means* nas partículas
        Determinar número ótimo de *clusters* usando o método *silhouette*
        Atualizar a lista de mínimos encontrados
    end if
end while
return Melhores soluções encontradas (a partir dos *clusters* identificados)

A seguir, será apresentada uma explicação detalhada do algoritmo modificado para facilitar a compreensão do algoritmo 5.

1. **Inicialização dos Parâmetros e Variáveis**
   O algoritmo inicia definindo os parâmetros necessários para sua execução. Estes incluem o tamanho da população ($TamPop$), a dimensionalidade do problema (Dim), os limites superiores ($LSup$) e inferiores ($LInf$) do espaço de busca, o fator de inércia (w), os coeficientes cognitivo (c_1) e social (c_2), o número máximo de avaliações da função objetivo ($Eval_{max}$), a probabilidade de comunicação ($ProbComunicacao$) e uma variável de controle para a busca local ($BuscaLocal$). Além disso, são inicializadas variáveis auxiliares como o contador de avaliações ($Eval$), o tamanho do espaço de busca ($EspacoBusca$), os tamanhos de passo inicial e final para a busca local (δ_{inicio} e δ_{fim}), e os valores iniciais e finais para o parâmetro *Multi Start* do *hill climbing* (M_{inicio} e M_{fim}).

2. **Inicialização do Enxame**
   Cada partícula i do enxame é inicializada atribuindo-se uma posição $\mathbf{x}_i$ aleatória dentro do espaço de busca definido pelos limites $LInf$ e $LSup$. As velocidades iniciais $\mathbf{v}_i$ também são geradas aleatoriamente. Cada partícula define sua melhor posição pessoal inicial (pbest_i) como sendo a posição inicial $\mathbf{x}_i$. Em seguida, a função objetivo $f(\mathbf{x}_i)$ é avaliada para cada partícula, e o contador de avaliações $Eval$ é incrementado.

3. **Loop Principal de Otimização**
   O algoritmo repete os seguintes passos até que o número máximo de avaliações da função objetivo ($Eval_{max}$) seja alcançado.

   a) **Cálculo do Melhor Vizinho com Base no FER**
   Para cada partícula i, é determinado o melhor vizinho $\mathbf{p}_n$ utilizando a métrica FER (*Fitness Euclidean-distance Ratio*), como foi introduzido anteriormente no capítulo 2. O algoritmo de seleção do melhor vizinho é detalhado no Algoritmo 3, com as modificações necessárias para este contexto.

   b) **Cálculo da Matriz de Comunicação**
   Uma matriz de comunicação C é calculada utilizando a probabilidade de comunicação ($ProbComunicacao$). Esta matriz controla a influência social das partículas em determinadas dimensões, permitindo ou bloqueando a comunicação com base na probabilidade definida.

   c) **Atualização da Velocidade**
   A velocidade de cada partícula $\mathbf{v}_i$ é atualizada utilizando a equação modificada do FER-PSO, que incorpora a matriz de comunicação e o melhor vizinho $\mathbf{p}_n$. A equação utilizada é a Equação Eq. 3.3.

d) **Atualização da Posição**
A posição de cada partícula $\mathbf{x}_i$ é atualizada somando-se a nova velocidade calculada à posição atual, conforme a Equação Eq. 2.2. Após a atualização, é verificado se a nova posição está dentro dos limites do espaço de busca; caso contrário, ela é ajustada.

e) **Avaliação e Atualização da Melhor Posição Pessoal**
A função objetivo é avaliada na nova posição $\mathbf{x}_i$, e o contador de avaliações *Eval* é incrementado. Se o valor de $f(\mathbf{x}_i)$ for melhor que o valor de $f(\text{pbest}_i)$, a melhor posição pessoal pbest_i é atualizada com a nova posição.

f) **Aplicação da Busca Local (*Hill Climbing*)**
Se a variável de controle *BuscaLocal* estiver ativada e o número de avaliações *Eval* estiver dentro dos critérios definidos (no início do algoritmo ou após 80% das avaliações máximas), a busca local é aplicada à partícula i. O tamanho do passo δ e o parâmetro *Multi Start* M são definidos com base no estágio da otimização:

- No início ($Eval \leq 0.8 \times Eval_{max}$): utiliza-se $\delta = \delta_{\text{inicio}}$ e $M = M_{\text{inicio}}$ para promover maior exploração.
- No final ($Eval \geq 0.8 \times Eval_{max}$): utiliza-se $\delta = \delta_{\text{fim}}$ e $M = M_{\text{fim}}$ para promover das soluções refinamento.

O algoritmo de *hill climbing* é então aplicado, gerando novas soluções vizinhas e buscando melhorar a posição atual da partícula. Após a busca local, a função objetivo é reavaliada, o contador *Eval* é incrementado com o número de avaliações realizadas durante a busca, e pbest_i é atualizado se uma melhor posição for encontrada.

4. **Clusterização com *K-Means***
A cada 10.000 avaliações da função objetivo ($Eval \bmod 10.000 = 0$), é aplicada a clusterização das partículas utilizando o algoritmo *K-Means*. O número ótimo de clusters é determinado pelo método do coeficiente *silhouette*, que avalia a qualidade da separação entre os clusters. A lista de mínimos encontrados é atualizada com base nos clusters identificados, permitindo ao algoritmo manter um histórico das melhores soluções encontradas em diferentes regiões do espaço de busca.

5. **Critério de Parada**
O loop principal continua até que o número máximo de avaliações da função objetivo ($Eval_{max}$) seja atingido. Ao final, o algoritmo retorna as melhores soluções encontradas durante o processo de otimização.

As modificações implementadas resultaram em benefícios para o desempenho do algoritmo, que foram avaliados por meio de testes. Esses testes e os resultados serão apresenta-

dos no próximo capítulo, onde serão discutidos os impactos das alterações no desempenho do algoritmo.

4 EXPERIMENTOS E RESULTADOS

O objetivo desse capítulo é realizar experimentos e analizar os resultados das modificações que foram propostas no algoritmo, buscando responder as perguntas norteadoras que foram levantadas no Capítulo 1.

4.1 CONFIGURAÇÃO EXPERIMENTAL

4.1.1 Funções de Teste

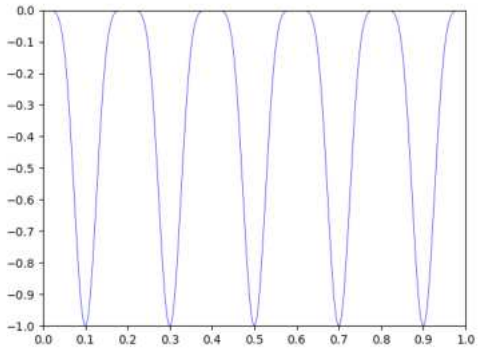
Para realizar os experimentos desse capítulo, foram selecionadas funções multimodais que foram encontradas nos artigos utilizados como referência, essas funções são utilizadas em outros trabalhos para medir a eficiência dos algoritmos de nicho que já foram anteriormente propostos/implementados, sendo assim, a escolha dessas funções são justificáveis.

Tabela 1 – Funções multimodais para testes

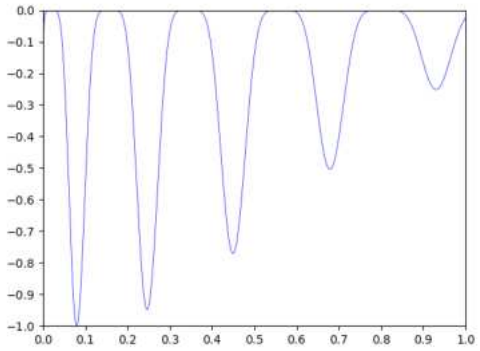
Função	**Dimensão**	**Domínio**	**N° de mínimos**	$F(x^*)$	**Observações**
Equal Minima (LI; ENGELBRECHT; EPITROPAKIS, 2013): $f_1(x) = -\sin^6(5\pi x)$	1	$x \in [0,1]^1$	5 Globais	-1	Função adaptada de (LI; ENGELBRECHT; EPITROPAKIS, 2013). Essa função era originalmente de maximização, mas neste estudo será utilizada a inversa, pois estamos lidando com minimização.
Uneven minima (LI; ENGELBRECHT; EPITROPAKIS, 2013): $f_2(x) = -\exp\left(-2\ln(2)\left(\frac{x-0.08}{0.854}\right)^2\right) \cdot \sin^6\left(5\pi\left(x^{3/4} - 0.05\right)\right)$	1	$x \in [0,1]^1$	1 Global, 4 Locais	-1	Função adaptada de (LI; ENGELBRECHT; EPITROPAKIS, 2013). Essa função era originalmente de maximização, mas neste estudo será utilizada a inversa, pois estamos lidando com minimização.
Himmelblau (LI, 2007): $f_3(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	2	$x_1, x_2 \in [-6,6]^2$	4 Globais	0	-
Six hump camel back (LI, 2007): $f_4(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	2	$x_1, x_2 \in [-2,2]^2$	2 Globais, 4 Locais	-1.031628	-
Shubert (LI, 2007): $f_5(x_1, x_2) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right) \cdot \left(\sum_{j=1}^{5} j\cos\left((j+1)x_2 + j\right)\right)$	2	$x_1, x_2 \in [-10,10]^2$	18 Globais, múltiplos locais	-186.730909	-
Branin (LI, 2007): $f_6(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1-f)\cos(x_1) + e$	2	$x_1, x_2 \in [-5,15]^2$	3 Globais	0.397887	$a = 1, \quad b = \frac{5.1}{4\pi^2}, \quad c = \frac{5}{\pi}, \quad d = 6, \quad e = 10, \quad f = \frac{1}{8\pi}$
Rastrigin $f_7(x_i) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	2	$x_i \in [-5.12, 5.12]^2$	1 Global, múltiplos locais	0	-
Hartmann 6 (GHASEMI; VARAEE, 2017): $f_8(x_i) = -\sum_{i=1}^{4}\alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(X_j - P_{ij})^2\right)$	6	$x_i \in [0,1]^6$	1 Global, 1 Local	-3.322368	$\alpha = [1.0, 1.2, 3.0, 3.2]$, $A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $P = 10^{-4} \times \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$
Holder Table (KIDANE; NAGY; TURáNYI, 2022): $f_9(x_1, x_2) = -\left\lvert\sin(x_1)\cos(x_2)\exp\left(1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right)\right\rvert$	2	$x_i \in [-10,10]^2$	4 Globais, múltiplos locais	-19.208502	-

Fonte: O autor, 2024

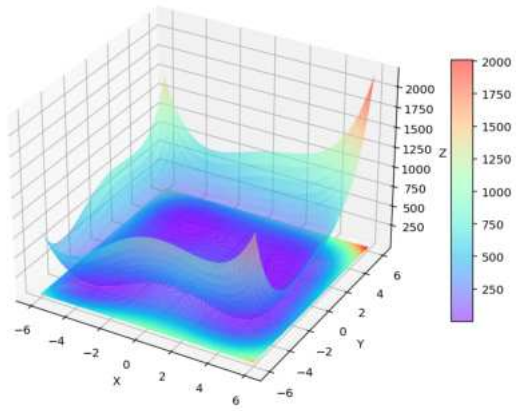
Figura 3 – Superfícies das funções de teste



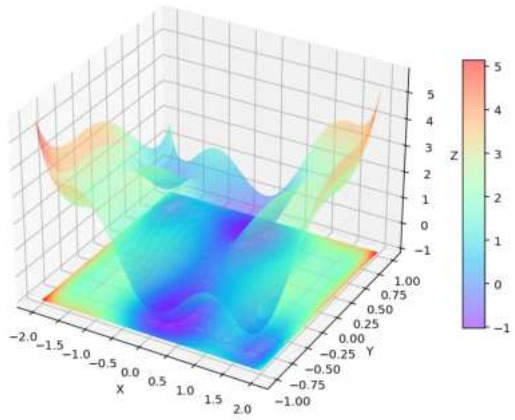
(a) *Equal Minima*
(LI; ENGELBRECHT; EPITROPAKIS, 2013)



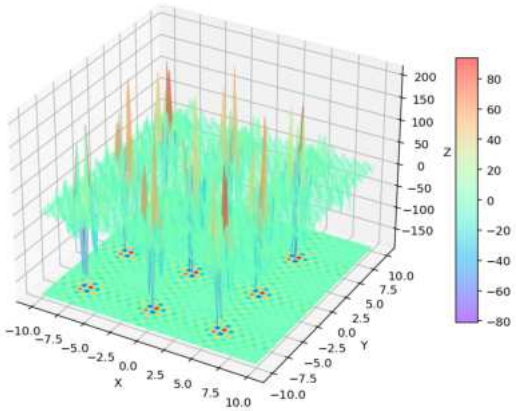
(b) *Uneven Minima*
(LI; ENGELBRECHT; EPITROPAKIS, 2013)



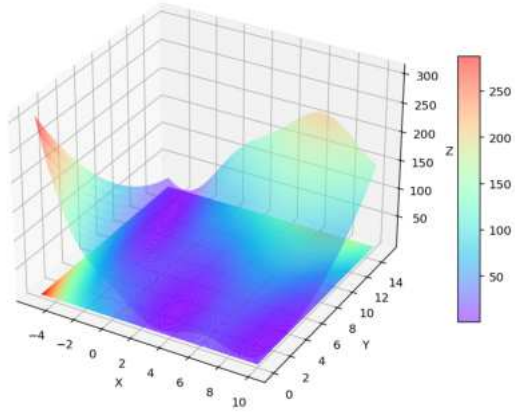
(c) *Himmelblau*
(LI, 2007)



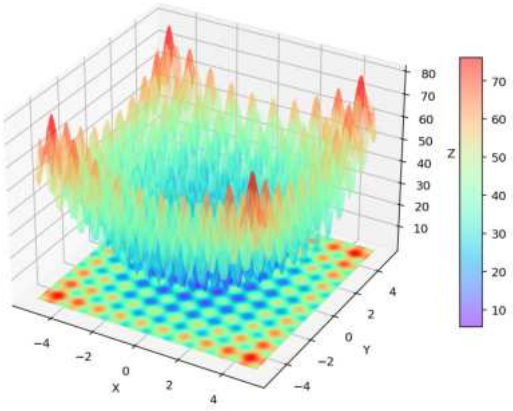
(d) *Six Hump Camel Back*
(LI, 2007)



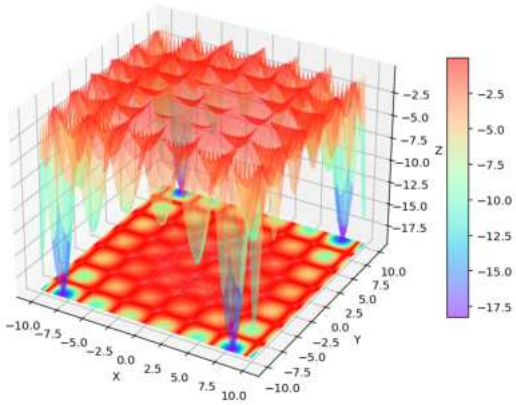
(e) *Shubert*
(LI, 2007)



(f) *Branin*
(LI, 2007)



(g) *Rastrigin*
(GHASEMI; VARAEE, 2017)



(h) *Holder Table*
(KIDANE; NAGY; TURáNYI, 2022)

Fonte: O autor, 2024

4.1.2 Métricas de Desempenho

Para avaliar o desempenho do algoritmo proposto, serão utilizadas cinco métricas. Duas dessas métricas foram identificadas durante a revisão bibliográfica realizada para este estudo, sendo amplamente utilizadas em trabalhos correlatos. As outras três métricas foram desenvolvidas pelo autor com o intuito de complementar a avaliação de desempenho.

As duas métricas encontradas na literatura são:

- **Taxa de Sucesso (*Success Rate* - SR)**: Representa a proporção de execuções em que o algoritmo conseguiu encontrar todos os ótimos globais conhecidos da função testada (LI, 2007).
- **Número Médio de Ótimos Encontrados**: Indica quantos ótimos globais, em média, são identificados pelo algoritmo em relação ao número total de execuções, considerando a quantidade de ótimos globais existentes na função (EPITROPAKIS; PLAGIANAKOS; VRAHATIS, 2011).

As três métricas adicionais propostas são:

- **Média dos Valores dos Mínimos Globais Encontrados**: Calcula a média dos valores da função objetivo nos pontos identificados como mínimos globais pelo algoritmo.
- **Desvio Padrão dos Valores dos Mínimos Globais Encontrados**: Mede a dispersão dos valores dos mínimos globais encontrados em relação à média, indicando a consistência dos resultados.

- **Distância Média dos Ótimos aos Centroides dos *Clusters***: Avalia a distância média entre os pontos ótimos encontrados e os centroides dos *clusters* identificados pelo algoritmo *K-Means*, permitindo analisar o grau de convergência das partículas em torno dos mínimos.

Cálculo das Métricas e Relevância para o Estudo

1. **Taxa de Sucesso (SR)**:

   A Taxa de Sucesso é calculada como a razão entre o número de execuções em que o algoritmo encontrou todos os mínimos globais da função e o número total de execuções realizadas. Matematicamente:

$$\text{SR} = \frac{\text{Número de execuções com sucesso}}{\text{Número total de execuções}} \times 100\% \qquad \text{(Eq. 4.1)}$$

   Esta métrica é importante para avaliar a capacidade do algoritmo em identificar consistentemente todos os ótimos globais de uma função multimodal em diferentes execuções.

2. **Número Médio de Ótimos Encontrados**:

   O Número Médio de Ótimos Encontrados é calculado pela média do número de mínimos globais identificados em todas as execuções realizadas. Essa métrica avalia o desempenho médio do algoritmo em encontrar os mínimos globais existentes na função. A fórmula é dada por:

$$\text{Média}_{\text{ótimos}} = \frac{\sum_{i=1}^{N_{\text{execuções}}} \text{Mínimos globais encontrados na execução } i}{N_{\text{execuções}}} \qquad \text{(Eq. 4.2)}$$

   Esta métrica permite avaliar a eficiência do algoritmo em encontrar os mínimos globais ao longo das execuções, mesmo que não encontre todos em uma única execução. Ela oferece uma perspectiva mais detalhada do desempenho em relação à detecção de ótimos ao longo das execuções.

3. **Média dos Valores dos Mínimos Globais Encontrados**:

   Calcula-se a média dos valores da função objetivo nos pontos identificados como mínimos globais:

$$\text{Média}_{\text{mínimos}} = \frac{1}{N_{\text{mínimos}}} \sum_{i=1}^{N_{\text{mínimos}}} f(x_i) \qquad \text{(Eq. 4.3)}$$

   Onde $N_{\text{mínimos}}$ é o número de mínimos globais encontrados, e $f(x_i)$ é o valor da função objetivo no mínimo global x_i.

Esta métrica ajuda a verificar a precisão dos mínimos globais encontrados pelo algoritmo, permitindo identificar se os valores convergem para o valor ótimo conhecido da função.

4. **Desvio Padrão dos Valores dos Mínimos Globais Encontrados**:

   O desvio padrão é calculado sobre os valores da função objetivo nos mínimos globais encontrados:

$$\sigma_{\text{mínimos}} = \sqrt{\frac{1}{N_{\text{mínimos}} - 1} \sum_{i=1}^{N_{\text{mínimos}}} \left(f(x_i) - \text{Média}_{\text{mínimos}}\right)^2} \qquad \text{(Eq. 4.4)}$$

   Um desvio padrão baixo indica que os valores dos mínimos globais encontrados são consistentes, ou seja, estão mais concentradas nos mínimos, sugerindo que o algoritmo é confiável em suas estimativas dos ótimos. Um desvio padrão mais alto indica que as partículas estão mais espalhadas pelo espaço de busca e mostra que o algoritmo falhou em convegir de forma satisfatória aos minímos.

5. **Distância Média dos Ótimos aos Centroides dos Clusters**:

   Para cada mínimo global encontrado, calcula-se a distância Euclidiana entre o ponto ótimo x_i e o centroide μ_j do *cluster* correspondente:

$$\text{Distância Média} = \frac{1}{N_{\text{mínimos}}} \sum_{i=1}^{N_{\text{mínimos}}} \|x_i - \mu_j\| \qquad \text{(Eq. 4.5)}$$

   Esta métrica permite avaliar o grau de convergência das partículas em torno dos mínimos globais. Distâncias menores indicam que as partículas estão bem agrupadas em torno dos ótimos, sugerindo uma boa capacidade do algoritmo em explorar intensivamente as regiões de interesse. Distâncias maiores indicam que os *clusters* estão sendo formados com partículas que estão muito distantes entre si portanto geram um centroide mais afastado do mínimo.

A utilização dessas métricas é relevante para o estudo pois permite uma avaliação consistente do desempenho do algoritmo. As métricas *SR* e *PR* fornecem uma visão geral da capacidade do algoritmo em encontrar os mínimos globais, enquanto as métricas adicionais (média, desvio padrão e distância média) oferecem uma visão sobre a precisão, consistência e qualidade da convergência das soluções encontradas. Além disso, também será analisado o número médio de avaliações de função necessárias para que o algoritmo encontre todos os mínimos globais da função.

4.1.3 Configuração para os Testes

Como mencionado ao longo deste trabalho, o algoritmo PSO possui vários parâmetros importantes para sua execução, tais como os coeficientes w, c_1, c_2, além do tamanho da população, dimensionalidade do problema e limites (superior e inferior) do espaço de busca. No final do Capítulo 3, foi apresentado o pseudocódigo do algoritmo PSO desenvolvido para este estudo (Algoritmo 5), o qual introduz três parâmetros adicionais necessários para sua execução: a taxa de comunicação, necessária para a criação da matriz de comunicação utilizada na atualização da velocidade através da Equação Eq. 3.3; o número máximo de avaliações de função, que define a quantidade máxima de vezes que o algoritmo avaliará a função em um ponto x_i; e um parâmetro referente à busca local, que determina se o algoritmo deve ou não realizar a busca local. A tabela 2, a seguir, resume os valores adotados para cada parâmetro do PSO:

Tabela 2 – Parâmetros utilizados nos experimentos

Parâmetro	**Valor**
Fator de inércia (w)	0,6
Coeficiente cognitivo (c_1)	1,8
Coeficiente social (c_2)	1,6
Taxa de comunicação (τ_{com})	0,6
Número máximo de avaliações de função	200.000
Tamanhos da população	20, 50, 100, 200, 300
Número de execuções por função	30

O autor, 2024

O tamanho da população variou entre 20, 50, 100, 200 e 300 partículas, uma vez que o objetivo do trabalho é medir o impacto do tamanho da população somado a busca local no desempenho do algoritmo. A dimensionalidade e o domínio foram definidos conforme especificado em cada função (ver tabela 1).

Quanto à execução do algoritmo, este foi rodado 30 vezes para cada função de teste com e sem busca local, e os resultados médios dessas 30 execuções serão apresentados nas seções seguintes deste capítulo. O ambiente de implementação utilizado foi o *Python 3.13*, e os testes foram realizados em um sistema com 16 GB de memória RAM e processador *Ryzen 5 5600x*.

4.2 EXPERIMENTOS

Nesta seção, serão apresentados os resultados dos testes realizados para avaliar o impacto das modificações propostas e o desempenho final do algoritmo. A tabela consolidada

com os resultados, que abrange todas as modificações realizadas para todas as funções apresentadas na Tabela 1, encontra-se no Apêndice A.

4.2.1 Resultados das Experimentações com o FER-PSO

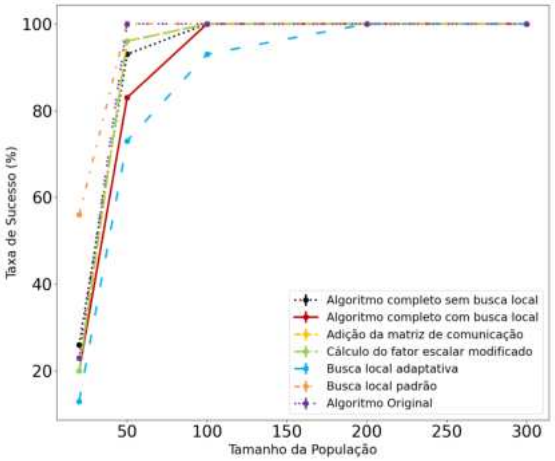
Para testar a efetividade e medir os resultados de todas as modificações e inclusões feitas no algoritmo, foram realizadas sete experimentações com o FER-PSO. A Tabela 4 apresenta todos os resultados numéricos obtidos. Nessa tabela, são exibidas as métricas definidas anteriormente: taxa de sucesso, número médio de ótimos encontrados, média dos nichos globais, desvio padrão, distância média aos ótimos e, por fim, a média de avaliações necessárias para encontrar todos os mínimos.

Para cada uma dessas métricas, são apresentados os valores obtidos em cada uma das experimentações. Portanto, para cada combinação de função e número de partículas, temos sete conjuntos de resultados correspondentes às diferentes experimentações.

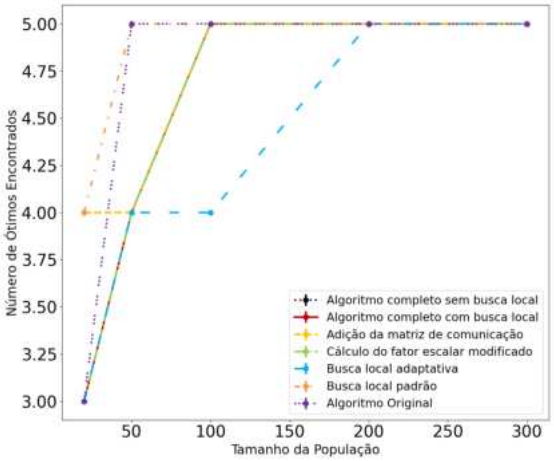
As colunas estão organizadas da seguinte forma:

- A coluna *Exp. 1* apresenta os valores encontrados ao rodar o algoritmo sem nenhuma modificação.
- A coluna *Exp. 2* apresenta os valores referentes à primeira modificação feita, a alteração no cálculo do fator escalar.
- A coluna *Exp. 3* apresenta os resultados da experimentação com a inclusão da matriz de comunicação.
- As colunas *Exp. 4* e *Exp. 5* apresentam os resultados do algoritmo com a inclusão somente das buscas locais, sendo:
  - *Exp. 4*: busca local padrão.
  - *Exp. 5*: busca local adaptativa.
- Por fim, as colunas *Exp. 6* e *Exp. 7* apresentam os resultados do algoritmo FER-PSO completo com todas as modificações, sendo:
  - *Exp. 6*: sem busca local.
  - *Exp. 7*: com busca local.

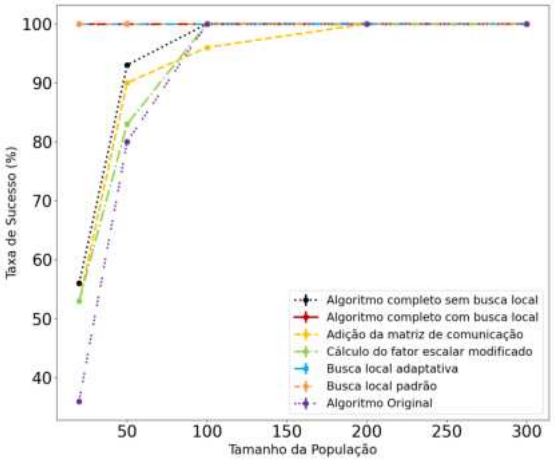
Figura 4 – Comparação das modificações feitas no algoritmo para todas as funções de teste



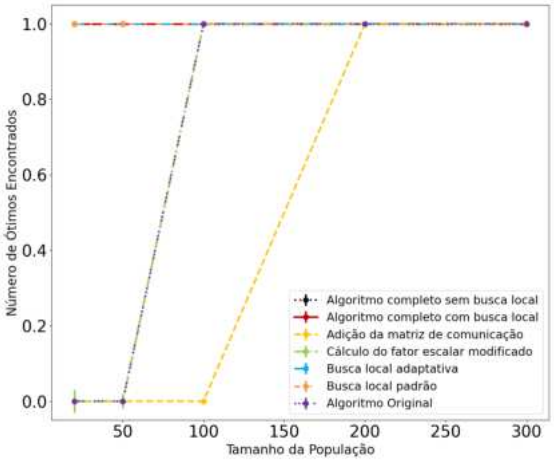
(a) Taxa de sucesso da função *Equal Minima*



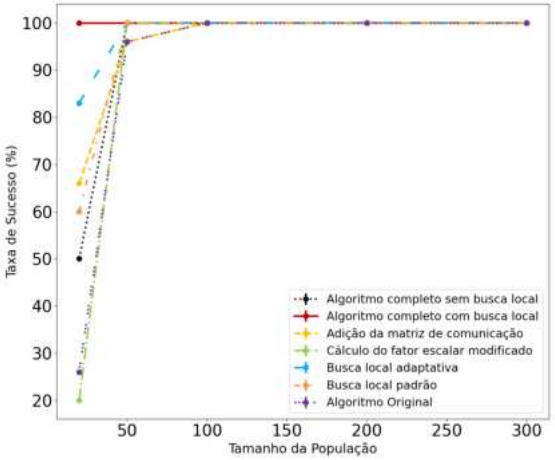
(b) Média de ótimos encontrados para a função *Equal Minima*



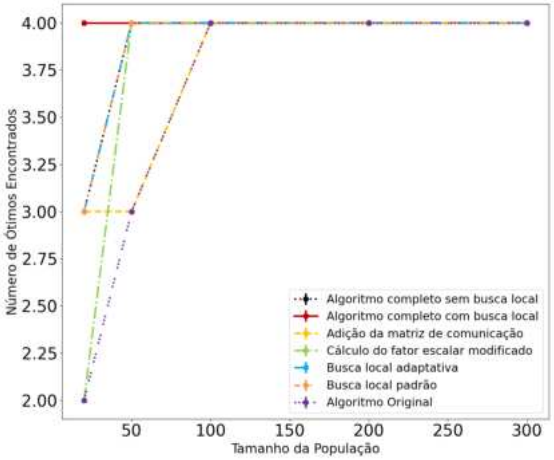
(c) Taxa de sucesso da função *Uneven Minima*



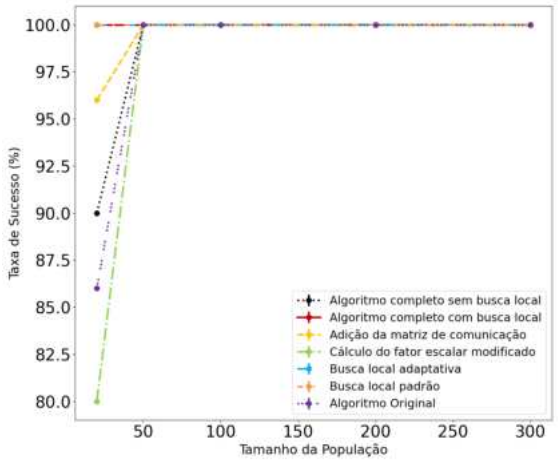
(d) Média de ótimos encontrados para a função *Uneven Minima*



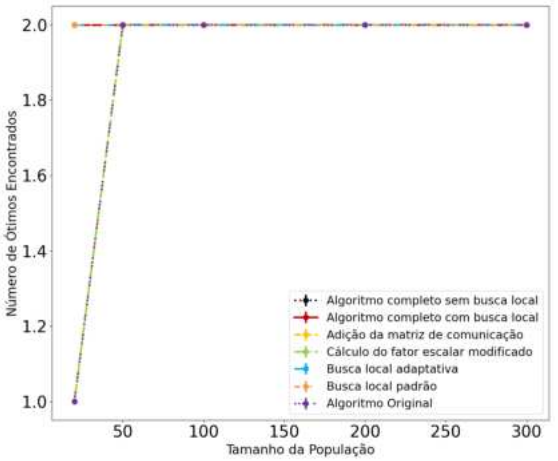
(e) Taxa de sucesso da função *Himmelblau*



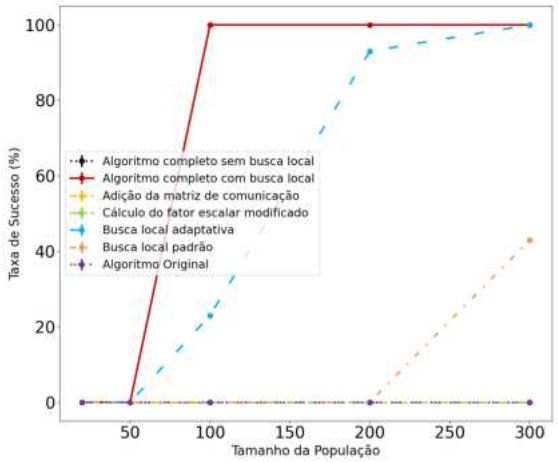
(f) Média de ótimos encontrados para a função *Himmelblau*



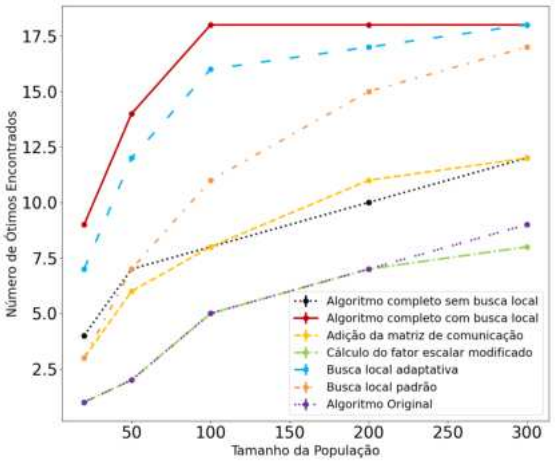
(g) Taxa de sucesso da função *Six Hump Camel Back*



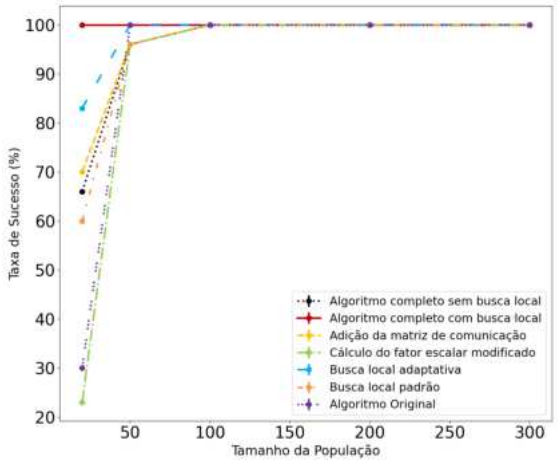
(h) Média de ótimos encontrados para a função *Six Hump Camel Back*



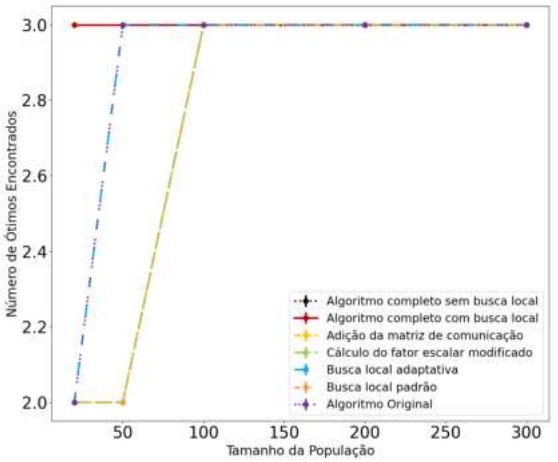
(i) Taxa de sucesso da função *Shubert*



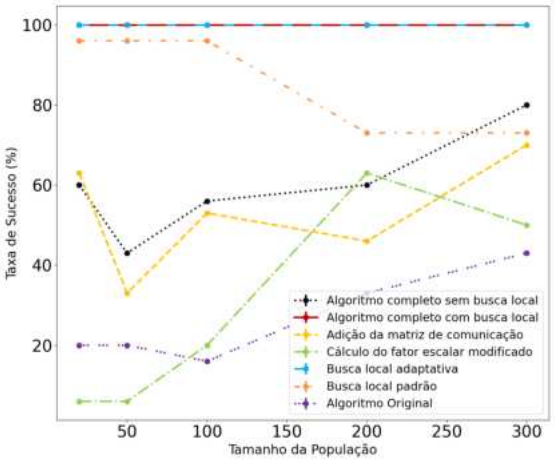
(j) Média de ótimos encontrados para a função *Shubert*



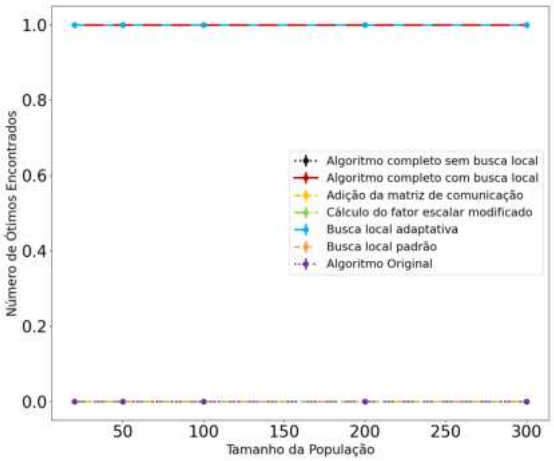
(k) Taxa de sucesso da função *Branin*



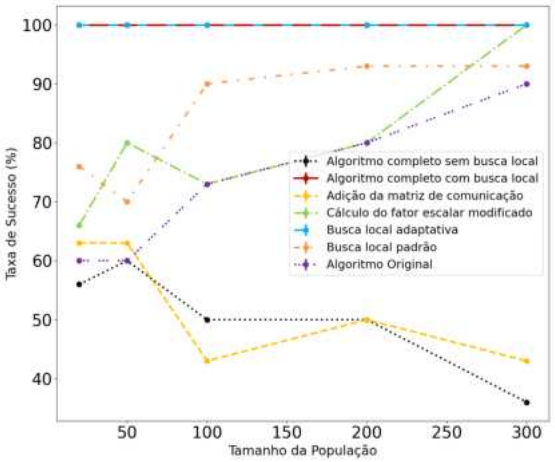
(l) Média de ótimos encontrados para a função *Branin*



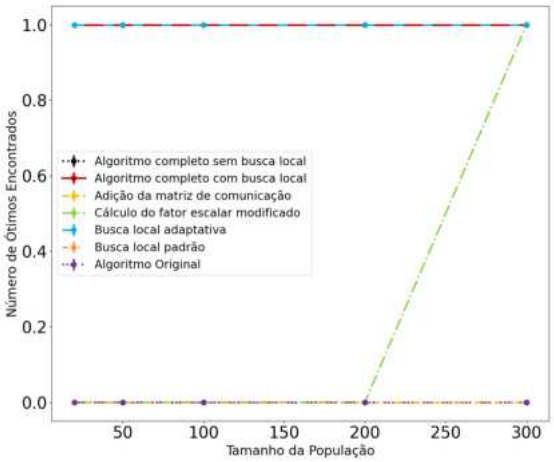
(m) Taxa de sucesso da função *Rastrigin*



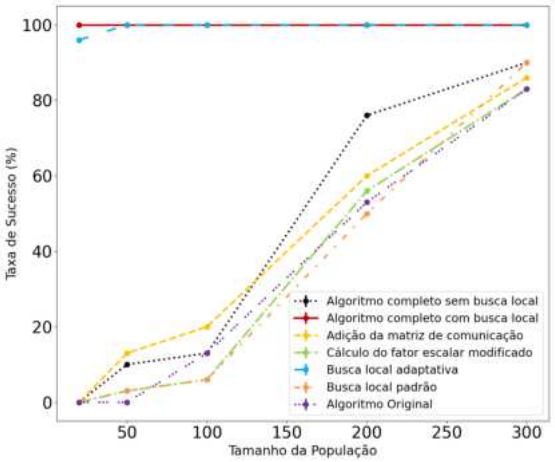
(n) Média de ótimos encontrados para a função *Rastrigin*



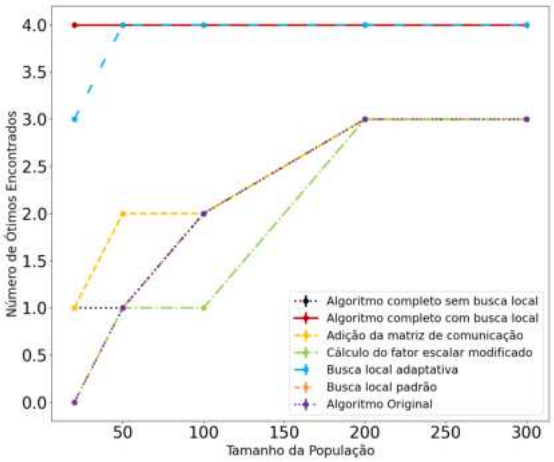
(o) Taxa de sucesso da função *Hartmann 6*



(p) Média de ótimos encontrados para a função *Hartmann 6*



(q) Taxa de sucesso da função *Holder Table*



(r) Média de ótimos encontrados para a função *Holder Table*

4.3.1 Impacto das Modificações no Fator Escalar e Inclusão da Matriz de Comunicação

As alterações no fator escalar e a inclusão da matriz de comunicação tiveram impactos limitados nas funções testadas. De modo geral, essas modificações não resultaram em melhorias significativas nas métricas de desempenho.

Para a função *Equal Minima*, as modificações não aumentaram a taxa de sucesso de forma relevante. A taxa permaneceu baixa para 20 partículas e alta para populações maiores, tanto no algoritmo original quanto nas versões modificadas, indicando que tais mudanças não influenciaram significativamente a capacidade do algoritmo em encontrar os mínimos globais nessa função.

Na função *Uneven Minima*, os resultados foram similares: as modificações não melhoraram a taxa de sucesso, que já era alta para populações maiores no algoritmo original. As funções *Six-Hump Camel Back* e *Branin* também não apresentaram melhorias significativas. O algoritmo original já tinha altas taxas de sucesso nessas funções, e as alterações não impactaram o desempenho.

Para a função *Rastrigin*, conhecida por sua alta multimodalidade, as modificações não aumentaram a taxa de sucesso de forma significativa. Mesmo com populações maiores, as melhorias foram mínimas, sugerindo que essas alterações não são suficientes para lidar com a complexidade dessa função sem estratégias adicionais.

Na função *Hartmann 6*, de alta dimensionalidade, as modificações não resultaram em avanços notáveis. As métricas permaneceram similares ao algoritmo original, indicando que essas mudanças não melhoram o desempenho em funções de alta dimensão.

A função *Himmelblau* apresentou um aumento na taxa de sucesso para populações menores com a inclusão da matriz de comunicação, passando de 26% para 66% com 20 partículas. Contudo, para populações maiores, as taxas de sucesso já eram altas no algoritmo original, e as modificações não tiveram impacto significativo.

Nas funções *holder table* e *Shubert*, que possuem espaços de busca mais complexos com múltiplos mínimos globais, as modificações não melhoraram a taxa de sucesso ou o número médio de ótimos encontrados. A taxa de sucesso permaneceu baixa ou nula, indicando que tais alterações não são eficazes sem estratégias adicionais.

4.3.2 Impacto da Modificação na Busca Local

A inclusão da busca local foi uma das principais contribuições deste trabalho, avaliada em duas variantes: o *hill climbing* (Algoritmo 2) padrão e o modificado (Algoritmo 4). Na função *Himmelblau*, a busca local aumentou a taxa de sucesso para populações menores. Com 20 partículas, a taxa de sucesso subiu de 26% no algoritmo original para 60% com

o *hill climbing* padrão e para 83% com o modificado. Para populações maiores, ambas as variantes alcançaram 100% de sucesso, indicando que a busca local melhora a consistência na identificação dos mínimos globais.

Para a função *holder table*, a busca local padrão não melhorou significativamente a taxa de sucesso em populações pequenas; com 20 partículas, a taxa permaneceu em 0%. Em contrapartida, a busca local modificada elevou a taxa de sucesso para 96% com 20 partículas e atingiu 100% em populações maiores. Esses resultados sugerem que a versão modificada da busca local pode oferecer melhorias em funções com superfícies complexas e múltiplos mínimos globais, embora essa tendência precise ser confirmada em mais estudos.

Na função *Shubert*, que possui 18 mínimos globais, a busca local padrão não aumentou a taxa de sucesso, mantendo-se em 0% até 100 partículas. Por outro lado, a busca local modificada elevou a taxa de sucesso para 23% com 100 partículas e atingiu 100% com 300 partículas. Esses achados indicam que a estratégia modificada pode ser particularmente útil para funções com um grande número de mínimos, embora seja importante analisar esses resultados em conjunto com outras métricas para validar sua eficácia.

Em outras funções, como *Equal Minima* e *Uneven Minima*, ambas as variantes de busca local melhoraram a taxa de sucesso, especialmente com populações maiores. Na função *Equal Minima*, a busca local padrão aumentou a taxa de sucesso para 56% com 20 partículas, enquanto a modificada alcançou 13%. Embora a busca local modificada tenha apresentado menor taxa inicial, mostrou maior precisão na convergência aos mínimos globais, conforme indicado pelos menores desvios padrão e distâncias médias aos ótimos.

Na função *Rastrigin*, a busca local padrão elevou a taxa de sucesso para 96% com 20 partículas, mas não atingiu 100% em populações maiores. A busca local modificada alcançou 100% já com 20 partículas, evidenciando sua eficiência em funções com múltiplos mínimos locais.

Para a função *Hartmann 6*, a busca local padrão aumentou a taxa de sucesso, enquanto a modificada atingiu 100% em todas as configurações. Isso destaca que a busca local modificada em funções de alta dimensionalidade pode ser eficaz se for implementada.

De maneira geral, a inclusão da busca local, especialmente na versão modificada, melhorou o desempenho do algoritmo em funções complexas. A busca local modificada superou a padrão, elevando a taxa de sucesso e a precisão dos resultados, mesmo em populações menores. Essas melhorias são evidentes nas reduções do desvio padrão e da distância média aos ótimos, indicando convergência mais precisa aos mínimos globais.

4.3.3 Análise do Número Médio de Avaliações de Função e Tempo Médio de Execução

A inclusão da busca local aumentou o número médio de avaliações de função necessárias para encontrar todos os mínimos, especialmente na função de maior dimensionalidade e com populações maiores, devido às avaliações adicionais durante a busca local. Con-

tudo, esse aumento é compensado pelas melhorias no desempenho, tornando o algoritmo mais eficiente na localização dos mínimos globais.

O tempo médio para a execução de cada experimentação foi de aproximadamente 1 hora, considerando todas as configurações de tamanho de população. Para as experimentações que incorporaram a busca local, o tempo médio de execução foi um pouco menor, possivelmente devido à realização de mais avaliações por iteração, o que contribuiu para uma convergência mais rápida. Esses achados sugerem que, apesar de a busca local aumentar o número de avaliações realizadas, ela pode acelerar o processo de otimização, compensando o custo computacional adicional.

4.3.4 Comparação entre o Algoritmo Completo com e sem Busca Local

Para avaliar o impacto da busca local no desempenho do algoritmo completo, comparamos os resultados obtidos com e sem a utilização da busca local, considerando todas as modificações propostas. Foi observado que a inclusão da busca local foi importante para melhorar a taxa de sucesso em várias funções testadas.

Na função *Equal Minima*, sem a busca local, a taxa de sucesso foi de 26% com 20 partículas e atingiu 100% apenas com 100 partículas. Com a inclusão da busca local, a taxa de sucesso manteve-se semelhante para 20 partículas (20%), mas aumentou para 83% com 50 partículas e atingiu 100% com 100 partículas. Além disso, a busca local reduziu o desvio padrão e a distância média aos ótimos, indicando maior precisão e consistência nos resultados.

Para a função *Branin*, sem a busca local, a taxa de sucesso foi de 66% com 20 partículas, alcançando 100% com 100 partículas. Com a busca local, a taxa de sucesso foi de 100% em todas as configurações, incluindo populações menores. A distância média aos ótimos também diminuiu, mostrando a eficácia da busca local em melhorar a precisão dos resultados.

A função *Hartmann 6*, sem a busca local, apresentou taxas de sucesso baixas, variando de 36% a 60%, mesmo com populações maiores. Com a inclusão da busca local, a taxa de sucesso atingiu 100% em todas as configurações, demonstrando que a busca local pode ser uma ferramenta importante para funções de dimensões mais altas. Houve também redução no desvio padrão e na distância média aos ótimos, indicando maior precisão na convergência.

Na função *Himmelblau*, a taxa de sucesso sem busca local foi de 50% com 20 partículas, enquanto com a busca local atingiu 100% já nessa população. A busca local melhorou significativamente a precisão dos resultados, com redução do desvio padrão e da distância média aos ótimos.

Para a função *holder table*, sem a busca local, a taxa de sucesso foi de 0% com 20 partículas e atingiu apenas 90% com 300 partículas. Com a busca local, a taxa de sucesso foi de 100% em todas as populações testadas, indicando novamente que a busca local é im-

portante para funções com múltiplos mínimos globais e superfícies complexas. Além disso, a distância média aos ótimos diminuiu, e o desvio padrão reduziu-se significativamente.

Na função *Rastrigin*, sem a busca local, a taxa de sucesso variou de 43% a 80%, não alcançando 100% em nenhuma configuração. Com a busca local, a taxa de sucesso foi de 100% em todas as populações, evidenciando a eficácia da busca local em funções altamente multimodais. A precisão dos resultados também melhorou, com redução na distância média aos ótimos.

A função *Shubert*, que possui 18 mínimos globais, não obteve sucesso sem a busca local, com taxa de sucesso de 0% em todas as populações. Com a inclusão da busca local, a taxa de sucesso alcançou 100% a partir de 100 partículas, encontrando todos os 18 mínimos globais nas populações maiores. Mais uma vez a busca local foi responsável por explorar de forma eficiente o espaço de busca em funções com muitos ótimos globais.

Nas funções Six-Hump Camel Back e *Uneven Minima*, o algoritmo sem busca local já apresentava altas taxas de sucesso. No entanto, a busca local contribuiu para reduzir a distância média aos ótimos e o desvio padrão, melhorando a precisão dos resultados.

Em geral, a inclusão da busca local no algoritmo completo aumentou a taxa de sucesso, especialmente em funções de alta complexidade e dimensionalidade. Além disso, a busca local melhorou a precisão dos resultados, reduzindo o desvio padrão e a distância média aos ótimos, sem aumentar excessivamente o número médio de avaliações de função.

5 CONCLUSÃO

Neste trabalho, foi introduzido o conceito de otimização como área de estudo, assim como as heurísticas utilizadas para resolver problemas de otimização. Além disso, apresentamos a otimização multimodal, que trata de problemas de otimização cujas funções possuem múltiplos mínimos locais e globais, para os quais foram desenvolvidas heurísticas específicas.

Este estudo surgiu, portanto, da possibilidade de implementar uma estratégia de busca local que auxiliasse algoritmos de nicho a identificar os mínimos globais conhecidos das funções de teste. Além da busca local, modificações adicionais foram implementadas no algoritmo original, como a inclusão da matriz de comunicação na equação de velocidade do PSO e a alteração no cálculo do fator escalar, com o objetivo de investigar quais tipos de ajustes poderiam melhorar o desempenho do algoritmo nas funções apresentadas no Capítulo 4.

Conforme demonstrado no Capítulo 4, as modificações isoladas não apresentaram, de maneira consistente, melhorias significativas nas métricas de desempenho. Embora, em alguns casos, essas mudanças tenham contribuído para o aumento do número de ótimos encontrados e para a elevação da taxa de sucesso, os resultados indicam que tais ajustes, por si sós, não se mostraram decisivos.

Posteriormente, os resultados relativos à inclusão da busca local no algoritmo sem modificações adicionais foram apresentados. Os dados exibidos na Tabela 4 sugerem que a busca local apresenta resultados competitivos, especialmente para funções como *Shubert* e *Holder Table*, que anteriormente apresentavam 0% de taxa de sucesso. A versão modificada da busca local mostrou, em certos cenários, um desempenho ainda mais aprimorado, evidenciando a potencial contribuição dessa estratégia para a identificação de mínimos globais.

Finalmente, o Capítulo 4 também apresentou os resultados para o algoritmo completo, com todas as modificações, comparando as versões com e sem a busca local ativada. Observou-se que a combinação da busca local com as mudanças na matriz de comunicação e no fator escalar contribuiu para uma melhoria substancial no desempenho do algoritmo. Além de aumentar a taxa de sucesso e o número médio de ótimos encontrados, o desvio padrão diminuiu quando o algoritmo foi executado com a busca local, indicando que as soluções encontradas estavam mais concentradas nos picos. A média de distância para os centróides dos *clusters* também diminuiu, o que sugere uma melhor avaliação dos nichos pelo *K-Means*.

Este trabalho evidencia que, para os cenários descritos, a busca local, quando aplicada a um algoritmo de otimização de nicho, apresentou resultados competitivos e contribuiu para reduzir a quantidade de partículas necessárias para encontrar os mínimos em diversas

funções de teste. Além disso, o uso da busca local resultou em uma diminuição do desvio padrão, tornando as soluções mais consistentes e precisas.

Apesar dos resultados positivos, é importante reconhecer as limitações deste estudo. As funções de teste utilizadas possuem baixa dimensionalidade e são bem conhecidas na literatura, o que pode não refletir completamente os desafios presentes em problemas reais de alta complexidade. Ademais, o aumento no número médio de avaliações da função, decorrente da aplicação da busca local, pode impactar o desempenho em termos de tempo computacional, especialmente em problemas nos quais cada avaliação é custosa.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, existem algumas possibilidades. Uma delas é aprimorar a busca local, tornando-a mais robusta, para que seja possível desenvolver um algoritmo de nicho estável e confiável em funções com dimensões maiores. Embora as funções utilizadas neste trabalho sejam majoritariamente de duas dimensões, é importante investigar melhorias para funções de maior dimensionalidade.

Além disso, seria interessante incluir uma forma de o algoritmo autoadaptar o tamanho da população com base no desempenho durante a execução. Atualmente, o tamanho da população é definido previamente, e este trabalho focou em uma implementação que reduzia a necessidade de uma população grande por meio da aplicação de uma busca local. Um trabalho futuro possível seria tentar diminuir ainda mais esse tamanho, incorporando uma estratégia em que o próprio enxame decida se irá aumentar ou diminuir a população conforme a exploração do espaço de busca.

Outra possibilidade de trabalho futuro é explorar a clusterização no contexto do algoritmo estudado. Um aspecto relevante a ser investigado é o comportamento da clusterização no cenário com poucas particulas, avaliando como a formação dos clusters influencia a convergência e a eficiência da busca pelos ótimos globais. Além disso, seria interessante apresentar resultados sobre o impacto da clusterização em diferentes funções de teste, levando em consideração a quantidade de mínimos presentes em cada função, permitindo entender melhor a relação entre a quantidade de nichos e o efeito da clusterização na identificação e preservação de múltiplos ótimos.

REFERÊNCIAS

BRITS, R.; ENGELBRECHT, A.; BERGH, F. A niching particle swarm optimizer. **SEAL**, 01 2002.

BRITS, R.; ENGELBRECHT, A.; BERGH, F. Solving systems of unconstrained equations using particle swarm optimization. v. 3, p. 6 pp. vol.3, 11 2002.

CORDEAU, J.-F. et al. Vehicle routing. In: _____. [S.l.]: Elsevier, 2007. v. 14, p. 195–224.

CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. **Manual de computação evolutiva e metaheurística**. Imprensa da Universidade de Coimbra, 2012. ISBN 978-989-26-0583-8. Disponível em: https://ucdigitalis.uc.pt/pombalina/item/58522.

DANTZIG, G. B.; RAMSER, J. H. The Truck Dispatching Problem. **Management Science**, v. 6, n. 1, p. 80–91, out. 1959. ISSN 0025-1909, 1526-5501. Disponível em: https://pubsonline.informs.org/doi/10.1287/mnsc.6.1.80.

DELIBASIS, K.; ASVESTAS, P. A.; MATSOPOULOS, G. K. Multimodal genetic algorithms-based algorithm for automatic point correspondence. **Pattern Recognition**, v. 43, n. 12, p. 4011–4027, dez. 2010. ISSN 00313203. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0031320310003055.

EPITROPAKIS, M.; PLAGIANAKOS, V.; VRAHATIS, M. Finding multiple global optima exploiting differential evolution's niching capability. p. 80textendash87, 04 2011.

GAD, A. G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. **Archives of Computational Methods in Engineering**, v. 29, n. 5, p. 2531–2561, ago. 2022. ISSN 1134-3060, 1886-1784. Disponível em: https://link.springer.com/10.1007/s11831-021-09694-4.

GHASEMI, M. R.; VARAEE, H. Damping vibration-based igmm optimization algorithm: fast and significant. **Soft Computing**, Springer Science and Business Media LLC, v. 23, n. 2, p. 451–481, set. 2017. ISSN 1433-7479. Disponível em: http://dx.doi.org/10.1007/s00500-017-2804-3.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. Reading, Mass: Addison-Wesley Pub. Co, 1989. ISBN 978-0-201-15767-3.

HARIK, G. Finding multimodal solutions using restricted tournament selection. 05 1995.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. The MIT Press, 1992. ISBN 978-0-262-27555-2. Disponível em: https://direct.mit.edu/books/book/2574/Adaptation-in-Natural-and-Artificial-SystemsAn.

HOOS, H. H.; TSANG, E. Local search methods. In: _____. **Handbook of Constraint Programming**. Elsevier, 2006. cap. 5, p. 135–167. Disponível em: http://dx.doi.org/10.1016/S1574-6526(06)80009-X.

JONG, K. A. D. **An analysis of the behavior of a class of genetic adaptive systems.** Tese (Doutorado) — University of Michigan, USA, 1975. AAI7609381.

JONG, K. D. **Evolutionary Computation – A Unified Approach**. [S.l.: s.n.], 2006. ISBN 978-0-262-04194-2.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **Proceedings of ICNN'95 - International Conference on Neural Networks**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4.

KIDANE, S.; NAGY, T.; TURáNYI, T. Testing various numerical optimization methods on a series of artificial test functions. **Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae, Sectio Geologica**, v. 53, p. 175–199, 10 2022.

LI, X. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. **Proceedings of the Genetic and Evolutionary Computation Conference**, v. 3102, 08 2004.

LI, X. A multimodal particle swarm optimizer based on fitness euclidean-distance ratio. p. 78–85, 07 2007.

LI, X. Niching without niching parameters: particle swarm optimization using a ring topology. **Trans. Evol. Comp**, IEEE Press, v. 14, n. 1, p. 150–169, fev. 2010. ISSN 1089-778X. Disponível em: https://doi.org/10.1109/TEVC.2009.2026270.

LI, X.; ENGELBRECHT, A. P.; EPITROPAKIS, M. G. Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization'. In: . [s.n.], 2013. Disponível em: https://api.semanticscholar.org/CorpusID:4456803.

MAHFOUD, S. W. **Niching methods for genetic algorithms**. Tese (Doutorado) — University of Illinois at Urbana-Champaign, USA, 1996. UMI Order No. GAX95-43663.

MIRANDA, V.; ALVES, R. Differential evolutionary particle swarm optimization (deepso): A successful hybrid. In: **2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence**. IEEE, 2013. p. 368–374. Disponível em: http://dx.doi.org/10.1109/BRICS-CCI-CBIC.2013.68.

POLYAK, B. Newton's method and its use in optimization. **European Journal of Operational Research**, v. 181, p. 1086–1096, 09 2007.

RAO, S. S. **Engineering optimization: theory and practice**. 4th ed. ed. Hoboken, N.J: John Wiley & Sons, 2009. OCLC: ocn320352991, Capítulo 1. ISBN 978-0-470-18352-6.

ROUSSEEUW, P. Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. **Journal of Computational and Applied Mathematics**, v. 20, p. 53–65, 11 1987.

SISSINVOU, M. et al. Investigation on torque ripple minimization in switched reluctance motor by using different techniques associated with tsf taking into account of magnetic saturation effects. **Technium: Romanian Journal of Applied Sciences and Technology**, v. 4, p. 23–46, 10 2022.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. **Journal of Global Optimization**, v. 11, p. 341–359, 01 1997.

TAKAHASHI, R. **Otimização Escalar e Vetorial**. Belo Horizonte: [s.n.], 2007. v. 2.

TAN, P.-N. et al. Cluster analysis: Basic concepts and algorithms. In: **Introduction to Data Mining**. 2. ed. [S.l.]: Pearson, 2021. cap. 7. Lecture Notes.

VEERAMACHANENI, K. et al. Optimization using particle swarms with near neighbor interactions. In: . [S.l.: s.n.], 2003. v. 2723.

WANG, X.; YAN, L.; ZHANG, Q. Research on the application of gradient descent algorithm in machine learning. In: **2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)**. [S.l.: s.n.], 2021. p. 11–15.

XU, R.; WUNSCH, D. Survey of clustering algorithms. **Neural Networks, IEEE Transactions on**, v. 16, p. 645 – 678, 06 2005.

GLOSSÁRIO

cluster Grupo de pontos em um espaço de busca que apresentam maior similaridade entre si do que com outros grupos.

heurísticas Procedimentos aproximados de busca por uma solução ótima utilizados para resolver problemas complexos de forma eficiente.

nichos Conceito utilizado em otimização para representar agrupamentos naturais de soluções semelhantes em uma população.

otimização Processo de encontrar a melhor solução para um problema.

APÊNDICES

APÊNDICE A – RESULTADOS DAS EXPERIMENTAÇÕES REALIZADAS UTILIZANDO O FER-PSO.

Tabela 3 – Resultados da Experimentação com o FER-PSO – Taxa de Sucesso, Número Médio de Ótimos Encontrados e Média de Nichos Globais

Função	N° de Partículas	Taxa de Sucesso (%)							N° Médio de Ótimos Encontrados							Média de Nichos Globais						
		Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7
Equal Minima	20	23	20	20	56	13	26	20	3	3	4	4	3	4	3	-0.998	-0.998	-1.000	-1.000	-1.000	-1.000	-1.000
Equal Minima	50	100	96	96	100	73	93	83	5	4	4	5	4	4	4	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Equal Minima	100	100	100	100	100	93	100	100	5	5	5	5	4	5	5	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Equal Minima	200	100	100	100	100	100	100	100	5	5	5	5	5	5	5	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Equal Minima	300	100	100	100	100	100	100	100	5	5	5	5	5	5	5	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Uneven Minima	20	36	53	53	100	100	56	100	0	0	0	1	1	0	1	-0.998	-0.981	-1.000	-1.000	-1.000	-1.000	-1.000
Uneven Minima	50	80	83	90	100	100	93	100	0	0	0	1	1	0	1	-0.999	-0.994	-1.000	-1.000	-1.000	-1.000	-1.000
Uneven Minima	100	100	100	96	100	100	100	100	1	1	0	1	1	1	1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Uneven Minima	200	100	100	100	100	100	100	100	1	1	1	1	1	1	1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Uneven Minima	300	100	100	100	100	100	100	100	1	1	1	1	1	1	1	-1.000	-1.000	-1.000	-0.998	-1.000	-1.000	-1.000
Himmelblau	20	26	20	66	60	83	50	100	2	2	3	3	3	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Himmelblau	50	96	100	96	100	100	100	100	3	4	3	4	4	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Himmelblau	100	100	100	100	100	100	100	100	4	4	4	4	4	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Himmelblau	200	100	100	100	100	100	100	100	4	4	4	4	4	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Himmelblau	300	100	100	100	100	100	100	100	4	4	4	4	4	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Six hump camel back	20	86	80	96	100	100	90	100	1	1	1	2	2	1	2	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
Six hump camel back	50	100	100	100	100	100	100	100	2	2	2	2	2	2	2	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
Six hump camel back	100	100	100	100	100	100	100	100	2	2	2	2	2	2	2	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
Six hump camel back	200	100	100	100	100	100	100	100	2	2	2	2	2	2	2	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
Six hump camel back	300	100	100	100	100	100	100	100	2	2	2	2	2	2	2	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
Shubert	20	0	0	0	0	0	0	0	1	1	3	3	7	4	9	-186.730	-186.731	-186.711	-186.721	-186.703	-186.712	-186.715
Shubert	50	0	0	0	0	0	0	0	2	2	6	7	12	7	14	-186.726	-186.730	-186.702	-186.721	-186.723	-186.698	-186.715
Shubert	100	0	0	0	0	23	0	100	5	5	8	11	16	8	18	-186.730	-186.729	-186.680	-186.722	-186.725	-186.694	-186.709
Shubert	200	0	0	0	0	93	0	100	7	7	11	15	17	10	18	-186.728	-186.730	-186.668	-186.727	-186.730	-186.668	-186.704
Shubert	300	0	0	0	43	100	0	100	9	8	12	17	18	12	18	-186.729	-186.726	-186.658	-186.727	-186.731	-186.660	-186.700
Branin	20	30	23	70	60	83	66	100	2	2	2	2	2	2	3	0.398	0.398	0.398	0.398	0.398	0.398	0.398
Branin	50	100	96	96	96	100	96	100	3	2	2	2	3	2	3	0.398	0.398	0.398	0.398	0.398	0.398	0.398
Branin	100	100	100	100	100	100	100	100	3	3	3	3	3	3	3	0.398	0.398	0.398	0.398	0.398	0.398	0.398
Branin	200	100	100	100	100	100	100	100	3	3	3	3	3	3	3	0.398	0.398	0.398	0.398	0.398	0.398	0.398
Branin	300	100	100	100	100	100	100	100	3	3	3	3	3	3	3	0.398	0.398	0.398	0.398	0.398	0.398	0.398
Rastrigin	20	20	6	63	96	100	60	100	0	0	0	0	1	0	1	0.001	0.000	0.002	0.001	0.000	0.002	0.000
Rastrigin	50	20	6	33	96	100	43	100	0	0	0	0	1	0	1	0.003	0.000	0.003	0.001	0.000	0.003	0.000
Rastrigin	100	16	20	53	96	100	56	100	0	0	0	0	1	0	1	0.002	0.000	0.005	0.004	0.000	0.007	0.000
Rastrigin	200	33	63	46	73	100	60	100	0	0	0	0	1	0	1	0.001	0.002	0.003	0.002	0.000	0.005	0.000
Rastrigin	300	43	50	70	73	100	80	100	0	0	0	0	1	0	1	0.003	0.004	0.008	0.004	0.000	0.006	0.000
Hartmann 6	20	60	66	63	76	100	56	100	0	0	0	0	1	0	1	-3.321	-3.321	-3.321	-3.321	-3.322	-3.321	-3.321
Hartmann 6	50	60	80	63	70	100	60	100	0	0	0	0	1	0	1	-3.321	-3.321	-3.321	-3.321	-3.321	-3.320	-3.321
Hartmann 6	100	73	73	43	90	100	50	100	0	0	0	0	1	0	1	-3.321	-3.321	-3.321	-3.321	-3.321	-3.321	-3.321
Hartmann 6	200	80	80	50	93	100	50	100	0	0	0	0	1	0	1	-3.321	-3.321	-3.320	-3.321	-3.321	-3.321	-3.321
Hartmann 6	300	90	100	43	93	100	36	100	0	1	0	0	1	0	1	-3.321	-3.321	-3.320	-3.320	-3.321	-3.321	-3.321
Holder Table	20	0	0	0	0	96	0	100	0	0	1	0	3	1	4	-19.209	-19.208	-19.208	-19.208	-19.208	-19.208	-19.208
Holder Table	50	0	3	13	3	100	10	100	1	1	2	1	4	1	4	-19.208	-19.209	-19.208	-19.208	-19.209	-19.208	-19.208
Holder Table	100	13	6	20	6	100	13	100	2	1	2	2	4	2	4	-19.208	-19.208	-19.208	-19.208	-19.209	-19.208	-19.208
Holder Table	200	53	56	60	50	100	76	100	3	3	3	3	4	3	4	-19.208	-19.208	-19.208	-19.208	-19.209	-19.208	-19.208
Holder Table	300	83	83	86	90	100	90	100	3	3	3	3	4	3	4	-19.208	-19.208	-19.208	-19.208	-19.209	-19.208	-19.208

Tabela 4 – Resultados da Experimentação com o FER-PSO – Desvio Padrão dos Nichos Globais, Distância Média aos Ótimos e Média de Avaliações

Função	N° de Partículas	Desvio padrão nichos globais							Distância média aos ótimos							Média de avaliações para todos os mínimos						
		Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7
Equal Minima	20	0.0088	0.0080	0.0000	0.0000	0.0000	0.0000	0.0000	0.0005	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	218.0	257.0	214.0	255.0	260.0	235.0	227.0
Equal Minima	50	0.0024	0.0010	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	692.0	595.0	499.0	669.0	1021.0	538.0	878.0
Equal Minima	100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1014.0	1250.0	1030.0	1297.0	2022.0	1300.0	1547.0
Equal Minima	200	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2267.0	2394.0	2367.0	2374.0	3180.0	2474.0	3107.0
Equal Minima	300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3650.0	3270.0	3620.0	3730.0	3970.0	3630.0	4590.0
Uneven Minima	20	0.0067	0.0303	0.0000	0.0000	0.0000	0.0000	0.0000	0.0004	0.0022	0.0000	0.0000	0.0000	0.0000	0.0000	62.0	99.0	55.0	20.0	20.0	66.0	20.0
Uneven Minima	50	0.0025	0.0189	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0006	0.0000	0.0000	0.0000	0.0000	0.0000	538.0	612.0	221.0	50.0	50.0	372.0	50.0
Uneven Minima	100	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	1357.0	1384.0	973.0	100.0	100.0	947.0	100.0
Uneven Minima	200	0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	1980.0	1880.0	1800.0	214.0	200.0	1620.0	200.0
Uneven Minima	300	0.0000	0.0000	0.0000	0.0085	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0000	0.0000	2900.0	2880.0	2740.0	360.0	300.0	3450.0	300.0
Himmelblau	20	0.0000	0.0000	0.0002	0.0002	0.0001	0.0002	0.0001	0.0000	0.0000	0.0018	0.0003	0.0002	0.0019	0.0010	153.0	184.0	162.0	170.0	168.0	183.0	168.0
Himmelblau	50	0.0000	0.0000	0.0002	0.0003	0.0000	0.0002	0.0001	0.0000	0.0000	0.0019	0.0001	0.0000	0.0019	0.0010	445.0	474.0	445.0	524.0	447.0	514.0	427.0
Himmelblau	100	0.0000	0.0000	0.0001	0.0000	0.0000	0.0002	0.0001	0.0000	0.0000	0.0016	0.0000	0.0000	0.0017	0.0012	794.0	747.0	780.0	840.0	770.0	804.0	800.0
Himmelblau	200	0.0000	0.0000	0.0001	0.0000	0.0000	0.0002	0.0001	0.0000	0.0000	0.0015	0.0000	0.0000	0.0016	0.0013	1547.0	1707.0	1727.0	1667.0	1640.0	1734.0	1760.0
Himmelblau	300	0.0000	0.0000	0.0002	0.0000	0.0000	0.0002	0.0001	0.0000	0.0000	0.0019	0.0000	0.0000	0.0021	0.0015	2740.0	2560.0	2410.0	2010.0	2830.0	2470.0	2670.0
Six hump camel back	20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0010	0.0009	0.0012	0.0010	0.0005	0.0014	0.0004	71.0	70.0	60.0	56.0	66.0	78.0	76.0
Six hump camel back	50	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0006	0.0006	0.0013	0.0007	0.0003	0.0013	0.0004	170.0	204.0	165.0	182.0	175.0	194.0	170.0
Six hump camel back	100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0003	0.0003	0.0012	0.0003	0.0002	0.0012	0.0005	410.0	320.0	424.0	427.0	280.0	344.0	320.0
Six hump camel back	200	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0014	0.0001	0.0000	0.0014	0.0005	840.0	847.0	760.0	827.0	627.0	834.0	640.0
Six hump camel back	300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0016	0.0000	0.0000	0.0014	0.0005	1230.0	1390.0	1330.0	1130.0	1000.0	1450.0	900.0
Shubert	20	0.0028	0.0003	0.0309	0.0270	0.0505	0.0352	0.0345	0.0002	0.0001	0.0024	0.0008	0.0023	0.0022	0.0021	inf	inf	inf	inf	inf	inf	inf
Shubert	50	0.0297	0.0075	0.0362	0.0348	0.0216	0.0382	0.0182	0.0003	0.0001	0.0030	0.0007	0.0009	0.0032	0.0023	inf	inf	inf	inf	inf	inf	inf
Shubert	100	0.0059	0.0168	0.0537	0.0369	0.0181	0.0384	0.0253	0.0001	0.0002	0.0041	0.0006	0.0006	0.0035	0.0027	inf	inf	inf	inf	6186.0	inf	5850.0
Shubert	200	0.0165	0.0039	0.0593	0.0197	0.0067	0.0583	0.0293	0.0004	0.0002	0.0046	0.0004	0.0002	0.0046	0.0030	inf	inf	inf	inf	15065.0	inf	13947.0
Shubert	300	0.0084	0.0196	0.0617	0.0226	0.0011	0.0614	0.0317	0.0005	0.0006	0.0051	0.0004	0.0002	0.0050	0.0033	inf	inf	inf	21900.0	22110.0	inf	19690.0
Branin	20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0006	0.0007	0.0021	0.0007	0.0008	0.0020	0.0013	123.0	123.0	120.0	138.0	155.0	128.0	136.0
Branin	50	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0004	0.0004	0.0022	0.0005	0.0004	0.0018	0.0018	400.0	432.0	337.0	349.0	380.0	316.0	325.0
Branin	100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0002	0.0017	0.0003	0.0002	0.0023	0.0021	784.0	684.0	540.0	717.0	804.0	674.0	757.0
Branin	200	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0018	0.0001	0.0001	0.0021	0.0019	1387.0	1627.0	1280.0	1567.0	1534.0	1320.0	1580.0
Branin	300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0022	0.0001	0.0001	0.0024	0.0020	1800.0	2070.0	2280.0	1930.0	2180.0	1930.0	1830.0
Rastrigin	20	0.0023	0.0000	0.0038	0.0021	0.0000	0.0031	0.0000	0.0010	0.0002	0.0020	0.0009	0.0000	0.0024	0.0000	84.0	60.0	142.0	28.0	20.0	98.0	20.0
Rastrigin	50	0.0062	0.0001	0.0037	0.0015	0.0000	0.0045	0.0000	0.0017	0.0007	0.0029	0.0014	0.0000	0.0032	0.0001	442.0	250.0	715.0	233.0	50.0	697.0	50.0
Rastrigin	100	0.0029	0.0001	0.0043	0.0052	0.0000	0.0050	0.0000	0.0020	0.0008	0.0046	0.0033	0.0000	0.0055	0.0002	1320.0	2434.0	3032.0	1752.0	100.0	3330.0	100.0
Rastrigin	200	0.0008	0.0031	0.0031	0.0043	0.0000	0.0041	0.0002	0.0012	0.0020	0.0032	0.0023	0.0000	0.0044	0.0005	7100.0	8200.0	7772.0	4973.0	200.0	12245.0	200.0
Rastrigin	300	0.0039	0.0057	0.0062	0.0044	0.0000	0.0051	0.0002	0.0030	0.0034	0.0056	0.0035	0.0000	0.0052	0.0006	21531.0	12660.0	15372.0	12137.0	300.0	15225.0	310.0
Hartmann 6	20	0.0004	0.0005	0.0006	0.0004	0.0003	0.0005	0.0003	0.0080	0.0081	0.0082	0.0075	0.0069	0.0088	0.0075	45.0	40.0	116.0	38.0	50.0	89.0	40.0
Hartmann 6	50	0.0005	0.0007	0.0005	0.0006	0.0004	0.0006	0.0005	0.0081	0.0090	0.0082	0.0085	0.0070	0.0090	0.0075	556.0	428.0	937.0	627.0	110.0	867.0	107.0
Hartmann 6	100	0.0004	0.0008	0.0004	0.0005	0.0005	0.0007	0.0005	0.0087	0.0079	0.0084	0.0088	0.0082	0.0084	0.0078	1573.0	2660.0	6054.0	2556.0	404.0	3727.0	244.0
Hartmann 6	200	0.0007	0.0005	0.0006	0.0006	0.0005	0.0007	0.0005	0.0086	0.0090	0.0088	0.0086	0.0080	0.0091	0.0079	9617.0	9309.0	14334.0	7922.0	814.0	20614.0	734.0
Hartmann 6	300	0.0006	0.0008	0.0007	0.0006	0.0007	0.0005	0.0006	0.0084	0.0083	0.0089	0.0090	0.0083	0.0085	0.0084	23745.0	21940.0	35170.0	21804.0	1880.0	33382.0	1390.0
Holder Table	20	0.0000	0.0000	0.0002	0.0000	0.0001	0.0002	0.0000	0.0017	0.0018	0.0031	0.0022	0.0018	0.0037	0.0021	inf	inf	inf	inf	168.0	inf	176.0
Holder Table	50	0.0001	0.0000	0.0002	0.0001	0.0000	0.0002	0.0000	0.0021	0.0018	0.0032	0.0019	0.0017	0.0030	0.0021	inf	600.0	1750.0	550.0	439.0	1567.0	440.0
Holder Table	100	0.0000	0.0000	0.0001	0.0001	0.0000	0.0002	0.0000	0.0019	0.0018	0.0036	0.0021	0.0017	0.0037	0.0021	2875.0	3550.0	5217.0	4550.0	784.0	3000.0	707.0
Holder Table	200	0.0000	0.0001	0.0002	0.0001	0.0000	0.0002	0.0000	0.0020	0.0020	0.0044	0.0024	0.0017	0.0042	0.0022	10450.0	9212.0	13989.0	10867.0	1560.0	14514.0	1927.0
Holder Table	300	0.0001	0.0001	0.0003	0.0001	0.0000	0.0002	0.0000	0.0023	0.0020	0.0046	0.0025	0.0017	0.0047	0.0023	19572.0	22080.0	23608.0	24623.0	2630.0	23423.0	2580.0