UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Inteligência Artificial

2025/2026

Relatório do

Trabalho Prático 2

Docentes:

JOSÉ PAULO BARROSO DE MOURA OLIVEIRA
EDUARDO JOSÉ SOLTEIRO PIRES

Grupo de Trabalho (PL4):
- Liane Raquel Madureira Duarte – AL79012
- Pedro Miguel Costa Braz – AL81311

Índice

1. Introdução

A otimização baseada em métodos de pesquisa é um domínio fundamental na área da Inteligência Artificial, especialmente quando se pretende resolver problemas cuja estrutura matemática revela elevado grau de complexidade, não linearidade e presença de múltiplos mínimos ou máximos locais. Em contextos reais, como aprendizagem automática, engenharia, logística ou sistemas dinâmicos, é frequente encontrar funções cuja superfície de pesquisa apresenta irregularidades que dificultam a aplicação de métodos determinísticos clássicos. Assim, surge a necessidade de recorrer a técnicas inspiradas na natureza ou em processos estocásticos, capazes de realizar uma exploração mais abrangente do espaço de soluções e de evitar estagnação prematura em soluções subótimas.

O presente trabalho tem como objetivo a implementação, análise e comparação de quatro algoritmos de otimização amplamente utilizados em Inteligência Artificial: **Multiple Restart Hill-Climbing**, **Simulated Annealing**, **Algoritmo Genético** e **Particle Swarm Optimization**. Estes métodos diferenciam-se nas estratégias de exploração, no modo como lidam com o equilíbrio entre procurar novas regiões do espaço e refinar soluções já encontradas, e no grau de aleatoriedade que incorporam no processo de pesquisa. Cada um deles representa uma abordagem distinta: desde métodos puramente locais, como o Hill-Climbing, até métodos populacionais inspirados na evolução biológica e no comportamento de enxames.

Para avaliar o desempenho desses algoritmos, foram utilizadas duas funções de teste bidimensionais definidas no protocolo fornecido pelo docente, ambas caracterizadas por uma superfície multimodal e com um mínimo global conhecido na origem. A primeira função, frequentemente referida como função “ripples”, apresenta oscilações concêntricas que criam múltiplos mínimos locais próximos da solução ótima. A segunda função corresponde à conhecida **Rastrigin**, um dos benchmarks mais utilizados no estudo de algoritmos evolucionários e métodos meta-heurísticos, devido à sua complexidade e elevado número de mínimos locais distribuídos de forma regular ao longo do domínio. Estas características tornam as funções ideais para testar a capacidade dos algoritmos de escaparem a ótimos locais e alcançarem a solução global.

Além da implementação prática em MATLAB, este trabalho inclui a análise gráfica do comportamento dos algoritmos, acompanhada de discussão crítica sobre a sua eficácia, limitações e diferenças metodológicas. Pretende-se não apenas apresentar resultados numéricos, mas também compreender como o comportamento interno de cada método influencia a convergência, a estabilidade e a robustez da solução obtida.

Deste modo, este relatório contribui para o desenvolvimento de competências essenciais na área da otimização computacional, permitindo uma compreensão mais profunda das estratégias meta-heurísticas e da sua aplicabilidade a problemas reais. Para além disso, oferece uma visão comparativa clara entre métodos clássicos e métodos inspirados na natureza, evidenciando a importância da parametrização e das propriedades estocásticas no desempenho final das soluções encontradas.

2. Funções de Teste

Para avaliar o desempenho dos diferentes algoritmos de otimização implementados neste trabalho, foram utilizadas as duas funções bidimensionais definidas no protocolo da unidade curricular. Estas funções são amplamente utilizadas na literatura de otimização por apresentarem múltiplos mínimos locais, tornando-se adequadas para a análise da robustez e capacidade de exploração das meta-heurísticas estudadas. Ambas possuem um **mínimo global localizado na origem** $(0,0)$, mas apresentam superfícies com características distintas que desafiam os algoritmos de maneiras diferentes.

2.1. Função 1

A primeira função, designada no protocolo como $f_1(x, y)$, apresenta uma superfície ondulada, semelhante a ondas concêntricas em torno da origem, demostrado na Figura 1. É expressa por:

$$f_1(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001(x^2 + y^2))^2},$$

definida no domínio:

$$-2.048 \leq x, y \leq 2.048.$$

Esta função caracteriza-se pela presença de múltiplos mínimos locais resultantes das oscilações do termo sinusoidal, ao mesmo tempo que a componente no denominador suaviza os valores à medida que se afasta da origem. A sua natureza multimodal torna-a particularmente útil para estudar algoritmos que dependem do equilíbrio entre exploração global e pesquisa local.

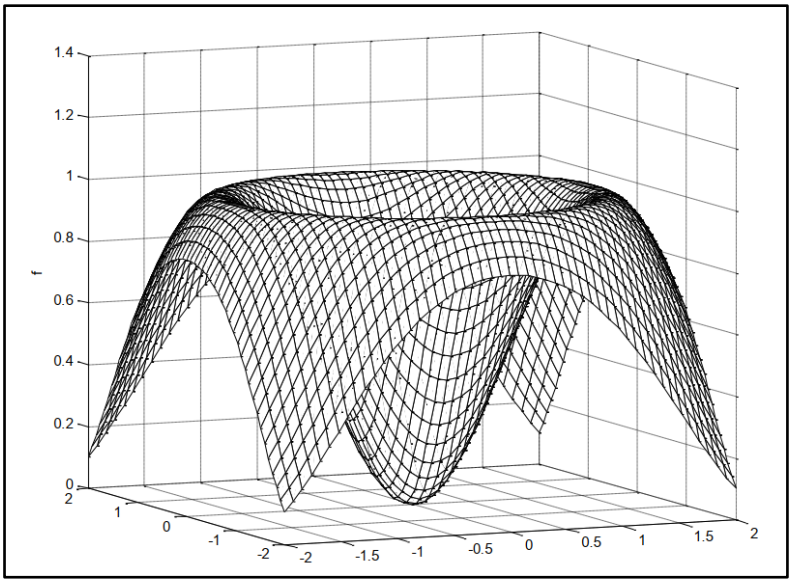

Figura 1 – Função 1.

2.2. Função 2

A segunda função corresponde à conhecida **função de Rastrigin**, uma das funções de teste mais utilizadas na área de otimização, devido à sua elevada complexidade e grande número de mínimos locais regularmente distribuídos ao longo do domínio, representado na Figura 2. É definida como:

$$f_2(x, y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y)),$$

com intervalo de pesquisa:

$$-5.12 \leq x, y \leq 5.12.$$

A função de Rastrigin combina um termo quadrático com termos cosinusoidais altamente oscilatórios, originando uma superfície extremamente multimodal. Esta característica representa um desafio significativo para algoritmos que tendem a ficar presos em mínimos locais, sendo um teste ideal para avaliar a eficácia das estratégias estocásticas implementadas neste trabalho.

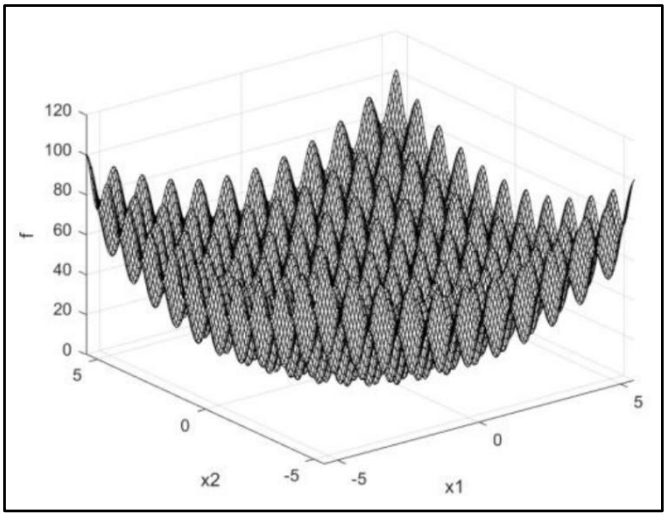

Figura 2 – Função 2.

3. Enquadramento Teórico

A otimização de funções complexas e multimodais é uma das áreas fundamentais da Inteligência Artificial, particularmente quando se pretende encontrar soluções ótimas num espaço de pesquisa com várias irregularidades, mínimos locais e superfícies altamente não lineares. Métodos determinísticos clássicos, como gradiente descendente ou Newton, apresentam limitações significativas quando a função não é diferenciável, contém ruído ou possui um grande número de mínimos locais. Por isso, surgem as **meta-heurísticas**, técnicas inspiradas na natureza ou em processos físicos, capazes de realizar uma procura mais ampla e estocástica no espaço das soluções.

Este trabalho centra-se no estudo, implementação e comparação de quatro algoritmos de otimização amplamente utilizados: **Subida da Colina com Reinicialização Múltipla (Multiple Restart Hill-Climbing)**, **Simulated Annealing (SA)**, **Algoritmo Genético (Genetic Algorithm)** e **Algoritmo de Otimização com Enxame de Partículas (Particle Swarm Optimization)**. Estes métodos fazem parte da classe de algoritmos de pesquisa inspirada na natureza, cujo enquadramento teórico é amplamente abordado no ebook *Nature Inspired Search and Optimization – Part I* disponibilizado pelo docente, o qual serve de suporte conceptual para este trabalho.

As funções utilizadas para teste são especialmente adequadas a este tipo de algoritmos, dado que apresentam múltiplos mínimos locais e apenas um mínimo global conhecido. Estas características obrigam os algoritmos a encontrar formas eficazes de explorar o espaço de procura e evitar soluções subótimas.

3.1. Subida da Colina com Reinicialização Múltipla (Multiple Restart Hill-Climbing)

O algoritmo da **Subida da Colina** é um método de pesquisa local que parte de uma solução inicial e tenta encontrar soluções vizinhas com menor valor da função (no caso de minimização). A estratégia consiste em analisar pequenas variações da solução corrente e substituí-la por uma solução vizinha que apresente melhoria. Este processo repete-se até não existirem vizinhos que conduzam a uma melhor solução, levando ao chamado **mínimo local**.

No entanto, em funções multimodais, como as utilizadas neste trabalho, este método frequentemente converge para mínimos locais em vez do mínimo global. Para contornar esta limitação, utiliza-se a versão com **Reinicialização Múltipla**, onde o algoritmo é executado várias vezes a partir de diferentes posições aleatórias no domínio da função. Esta abordagem aumenta a probabilidade de encontrar regiões promissoras do espaço e aproximar-se da solução global. Apesar disso, continua a ser um método sem mecanismos para escapar a mínimos locais de forma ativa, sendo geralmente o menos eficaz entre as meta-heurísticas estudadas.

3.2. Simulated Annealing (SA)

O **Simulated Annealing (SA)** é inspirado no processo físico de arrefecimento lento de metais, onde a temperatura é gradualmente reduzida permitindo que o sistema atinja um estado de energia mínima. A principal característica do SA, que o distingue da Subida da Colina, é a capacidade de **aceitar temporariamente soluções piores** com uma probabilidade dependente da temperatura. Esta probabilidade é dada por:

$$p = e^{-\frac{\Delta E}{T}},$$

onde ΔErepresenta a variação de energia (valor da função) e Tcorresponde à temperatura atual.

A temperatura diminui ao longo do tempo segundo um **fator de decaimento** (alfa), conduzindo a uma redução progressiva da aceitação de soluções piores. Assim, no início, o algoritmo comporta-se como uma pesquisa global, explorando amplamente o espaço; quando a temperatura é baixa, aproxima-se de uma pesquisa local semelhante ao Hill-Climbing.

Esta capacidade de "aceitar más soluções" torna o SA muito mais eficaz a escapar de mínimos locais, desempenhando geralmente melhor do que a Subida da Colina com Reinicialização Múltipla.

3.3. Algoritmo Genético (Genetic Algorithm)

O **Algoritmo Genético (GA)** é uma meta-heurística inspirada nos princípios da evolução natural, como seleção natural, reprodução e mutação. No contexto deste trabalho, as soluções são representadas como **cromossomas binários de 10 bits**, codificando as variáveis das funções.

O funcionamento do GA assenta em quatro componentes principais:

• **População inicial -** conjunto de cromossomas gerados aleatoriamente.

• **Seleção -** indivíduos com melhor aptidão (menor valor da função) têm maior probabilidade de contribuir para a geração seguinte.

• **Cruzamento -** dois cromossomas combinam partes dos seus bits para formar descendentes, explorando novas regiões do espaço.

• **Mutação -** certos bits são invertidos aleatoriamente, introduzindo diversidade genética e evitando estagnação.

Ao longo de várias gerações, o GA tende a melhorar progressivamente a qualidade das soluções da população. Por explorar várias soluções em paralelo e por incorporar variação genética, lida bem com funções multimodais, mostrando desempenho sólido e consistente.

3.4. Algoritmo de Otimização com Enxame de Partículas (Particle Swarm Optimization)

O **Algoritmo de Otimização com Enxame de Partículas (PSO)** inspira-se no comportamento coletivo de enxames de pássaros ou cardumes de peixes. Cada partícula representa uma solução e move-se pelo espaço da função influenciada por:

- **pbest** – a melhor posição individual encontrada pela própria partícula
- **gbest** – a melhor posição global encontrada pelo enxame

A dinâmica do movimento da partícula é dada pelas atualizações da velocidade e da posição, onde intervêm:

- o **fator de inércia (w)**,
- o termo cognitivo (ligado a pbest),
- o termo social (ligado a gbest).

O PSO apresenta elevada capacidade de exploração global, sendo geralmente o método com convergência mais rápida e robusta, especialmente adequado para funções com muitos mínimos locais, como a Função 2 (Rastrigin). Por não depender de codificação binária e trabalhar diretamente no espaço contínuo, as partículas conseguem ajustar as suas posições de forma mais suave e eficiente.

4. Implementação dos Algoritmos

Nesta parte, detalha-se a implementação prática de cada algoritmo em MATLAB, justificando as escolhas de parametrização e as adaptações realizadas face aos modelos teóricos.

4.1. Subida da Colina com Reinicialização Múltipla (Multiple Restart Hill-Climbing)

A implementação deste algoritmo baseou-se numa abordagem de pesquisa local estocástica, como pode ser observado no Anexo 1, desenhada para contornar a principal limitação do método clássico: a convergência prematura para mínimos locais.

Estratégia de pesquisa local e geração da vizinhança

Em cada reinicialização, o algoritmo começa por sortear uma posição inicial xi de forma uniforme dentro dos limites [x_min, x_max]. A partir deste ponto, realiza uma sequência de passos de pesquisa local, com um número máximo de iterações definido por max_iter. A cada iteração, calcula o valor atual da função val_atual = f(xi) e, de seguida, gera um vizinho estocástico xn na vizinhança de xi:

$$x_n = x_i + \text{step} \cdot (2 \cdot \text{rand}(1,2) - 1)$$

ou seja, a cada coordenada de xi é adicionada uma perturbação uniforme no intervalo [-step,step]. O parâmetro step define o raio da vizinhança local: valores mais pequenos conduzem a movimentos mais curtos e exploração fina; valores maiores permitem saltos mais largos. Para a Função 1 foi usado um passo step = 0.05, adequado à forma da superfície, enquanto para a Função 2 (Rastrigin) se aumentou para step = 0.1, de forma a facilitar saltos entre vales numa função altamente multimodal.

Depois de gerar o vizinho, este é imediatamente forçado a respeitar os limites do domínio com xn = max(xn, x_min) e xn = min(xn, x_max). Só depois deste *clamping* é avaliada a função no ponto vizinho.

Critério de aceitação e natureza gulosa do método

O critério de aceitação é estritamente guloso: o vizinho xn só é aceite se melhorar o valor da função objetivo em relação ao ponto atual. Em termos formais, se

$$f(x_n) < f(x_i),$$

então o algoritmo atualiza a solução corrente para xi = xn; caso contrário, mantém-se em xi e passa à iteração seguinte. Esta regra simples garante que, ao longo de cada descida, a sequência de valores da função nunca aumenta. No entanto, esta mesma característica faz com que o algoritmo não consiga escapar de mínimos locais: assim que não existam vizinhos que melhorem o valor atual, o processo estagna naquela região.

Para registar o comportamento ao longo do tempo, em cada iteração são guardados o valor atual da função (trace_f), as coordenadas da solução corrente (trace_x1, trace_x2) e o melhor valor global encontrado até então (trace_best). Este histórico permite estudar o "percurso" seguido pelo método, visualizar as trajetórias no espaço de pesquisa e analisar a forma como o melhor valor global melhora à medida que se acumulam reinicializações.

Reinicialização múltipla e gestão do melhor global

Para contornar a limitação de ficar preso num único mínimo local, o algoritmo foi encapsulado num ciclo de reinicializações independentes. Em cada reinicialização, o processo descrito anteriormente (ponto inicial aleatório + max_iter passos de descida gulosa) é repetido a partir de uma nova posição inicial aleatória. No final de cada reinicialização, imprime-se no terminal a posição final e o valor da função nessa posição, bem como o melhor valor global encontrado até esse momento.

Na Função 1, foram utilizadas 10 reinicializações com 100 iterações cada, uma vez que a superfície apresenta uma estrutura menos extrema e o mínimo global é relativamente acessível. Já na Função 2 (Rastrigin), aumentou-se para 50 reinicializações com 200 iterações, refletindo a necessidade de explorar melhor um espaço de pesquisa com muitos mínimos locais. Ao longo de todas as descidas, o algoritmo mantém dois registos globais: best_f_global, que guarda o menor valor da

função encontrado em qualquer reinicialização, e best_pos_global, que regista as coordenadas da solução correspondente.

Esta estratégia de reinicialização múltipla transforma um método localmente guloso numa abordagem de pesquisa global rudimentar: cada descida funciona como um otimizador local que "cai" num vale da função; ao repetir o processo a partir de pontos iniciais distintos, aumenta-se a probabilidade de, pelo menos numa das tentativas, o ponto inicial estar dentro da bacia de atração do mínimo global.

Histórico de execução e análise posterior

Durante a execução, o algoritmo acumula, de forma sequencial, o histórico de todas as reinicializações, sem reiniciar os vetores de registo entre tentativas. Isto permite representar a evolução do valor atual da função ao longo de todas as iterações (incluindo os "saltos" entre reinícios), bem como a evolução do melhor valor global encontrado (trace_best), que é uma sequência não crescente.

Com base nestes dados, foram construídos vários tipos de gráficos na secção de resultados:
– gráficos da evolução temporal do custo, onde se observa o comportamento "denteado" da solução atual e a curva suave do melhor global;
– gráficos da evolução das coordenadas x_1 e x_2, que mostram como as soluções aceites se deslocam no espaço de pesquisa;
– gráficos da trajetória 2D sobre o mapa de contornos da função, revelando os caminhos seguidos pelas descidas locais;
– e gráficos 3D onde são desenhados simultaneamente a superfície da função, os pontos visitados e o melhor ponto global encontrado.

No final de todas as reinicializações, o algoritmo devolve o melhor valor global best_f_global e a posição associada best_pos_global, que são usados posteriormente na comparação com os restantes métodos (Simulated Annealing, GA e PSO). A mesma estrutura foi utilizada para ambas as funções, ajustando apenas os parâmetros num_restarts, max_iter e step ao grau de complexidade e multimodalidade de cada uma.

Parâmetros utilizados:

Número de Reinicializações:

- Função 1: 10 reinicializações.
- Função 2: 50 reinicializações (devido à elevada multimodalidade).

Número de Iterações por Tentativa:

- Função 1: 100 iterações.
- Função 2: 200 iterações.

Tamanho do Passo (Vizinhança):

- Função 1: 0.05.
- Função 2: 0.1.

Critério de Aceitação: Estritamente guloso (apenas aceita melhorias).

4.2. Simulated Annealing (SA)

A implementação do algoritmo de Arrefecimento Simulado (*Simulated Annealing*) introduziu a capacidade de escapar a mínimos locais através da aceitação probabilística de soluções piores, como esta evidenciado no Anexo 2, um mecanismo fundamental para superfícies rugosas como a da função de Rastrigin.

Geração de vizinhos e definição da vizinhança

A exploração do espaço de pesquisa é feita através da geração de vizinhos estocásticos em torno da posição atual. Em cada iteração, o algoritmo perturba a solução corrente adicionando um desvio aleatório:

$$x_{\text{novo}} = x_{\text{atual}} + \text{step} \cdot (2 \cdot \text{rand}(1,2) - 1),$$

ou seja, é adicionada uma variação uniforme em cada coordenada no intervalo $[-\text{step}, \text{step}]$. O parâmetro step controla o "raio" da vizinhança, influenciando a amplitude dos movimentos: valores mais pequenos produzem passos curtos e exploração mais local; valores maiores permitem saltos mais largos, úteis para escapar a vales locais.

Depois de gerado o vizinho xn, este é imediatamente projetado para dentro dos limites do problema, usando max e min para garantir que nenhuma coordenada ultrapassa x_min ou x_max. Só depois deste *clamping* é que a função objetivo f(xn) é avaliada.

No caso da Função 1, utilizou-se um step mais pequeno (0.1), adequado a uma paisagem de otimização menos extrema. Para a Função 2 (Rastrigin), foi escolhido um step maior (0.2), de forma a permitir movimentos mais significativos numa função altamente ondulada, ajudando o algoritmo a ultrapassar oscilações muito locais.

Critério de aceitação – Regra de Metropolis

A principal diferença entre o SA e um método puramente guloso (como a Subida da Colina em modo de minimização) está no critério de aceitação de novas soluções. Em vez de aceitar apenas vizinhos melhores, o SA recorre à **regra de Metropolis**, que permite aceitar, com certa probabilidade, movimentos que pioram temporariamente o valor da função.

Depois de gerar um vizinho xn e calcular o seu custo fn, é computada a variação de energia (ou variação de custo):

$$\Delta E = f(x_{\text{novo}}) - f(x_{\text{atual}}).$$

Se $\Delta E \leq 0$, ou seja, se o vizinho for melhor ou igual à solução corrente, este é sempre aceite (a probabilidade de aceitação é 1). Se, pelo contrário, $\Delta E > 0$, o vizinho é pior e, num método guloso, seria descartado automaticamente. No SA, essa solução pior é aceite com probabilidade

$$P = e^{-\Delta E/T},$$

onde Té a temperatura atual. Na implementação, gera-se um número aleatório $r \in [0,1]$; se $r < P$, o vizinho pior é aceite e passa a ser o novo estado corrente. Caso contrário, o algoritmo mantém a solução anterior. Desta forma, em fases de temperatura elevada, o algoritmo está disposto a aceitar degradações com alguma frequência, explorando o espaço de forma mais livre; à medida que a temperatura desce, a probabilidade de aceitar soluções piores torna-se cada vez menor, e o método comporta-se de forma progressivamente mais gulosa.

Sempre que uma nova solução (melhor ou aceite via Metropolis) passa a ser o estado corrente, é também verificado se o seu valor fx é inferior ao melhor valor global registado até ao momento (fbest). Se for esse o caso, atualizam-se o melhor ponto global xbest e o respetivo custo.

Parâmetros de temperatura, arrefecimento e número de vizinhos

O comportamento do SA é fortemente influenciado pela escolha da temperatura inicial T, da temperatura mínima Tmin, do fator de arrefecimento alfa e do número de vizinhos testados por cada nível de temperatura (nRep).

Em ambos os casos, a temperatura inicial foi definida como T = 90, um valor suficientemente elevado para permitir, no início, a aceitação de movimentos piores com probabilidade apreciável. A temperatura mínima foi fixada em Tmin = 1e-4, que funciona como critério de paragem: quando T desce abaixo deste limiar, o algoritmo termina.

O arrefecimento é feito de forma exponencial, usando a regra

$$T_{\text{novo}} = \alpha \cdot T_{\text{atual}},$$

com alfa = 0.94. Este valor garante uma redução gradual da temperatura, permitindo uma fase inicial de exploração mais agressiva, seguida de uma fase de exploração fina. Para a Rastrigin, mantiveram-se os valores de T e alfa, mas aumentou-se o número de vizinhos por temperatura (nRep = 50 em vez de 20), de forma a permitir mais tentativas em cada patamar térmico numa função altamente multimodal.

Em cada temperatura, o algoritmo realiza nRep iterações internas: em cada uma delas gera um vizinho, calcula ΔE, aplica a regra de Metropolis e, se a solução for aceite, atualiza o estado corrente e, eventualmente, o melhor global. Ao mesmo tempo, são incrementados contadores que permitem calcular no final o número total de iterações (it_total) e a taxa de aceitação de vizinhos.

Ciclo do algoritmo e histórico de execução

O ciclo principal do SA consiste em repetir, enquanto $T > Tmin$, o seguinte processo: para cada nível de temperatura são gerados vários vizinhos, aplicando-se sempre o critério de aceitação de Metropolis. Após testar nRep vizinhos à mesma temperatura, o valor de T é atualizado multiplicando por alfa, e o ciclo repete-se com uma temperatura mais baixa.

Durante a execução são registadas várias grandezas para análise posterior: o número sequencial da iteração (hist_iter), a probabilidade de aceitação calculada na iteração (hist_prob), a temperatura corrente (hist_T), a posição atual (hist_x, hist_y) e o valor da função nessa posição (hist_fx). Este registo permite não só visualizar a trajetória do algoritmo no espaço de pesquisa (em gráficos 2D e 3D), como também estudar a evolução temporal do valor da função, da temperatura e do próprio mecanismo de aceitação (gráficos de convergência e de probabilidade).

No final da execução, o algoritmo devolve o melhor ponto encontrado xbest e o respetivo valor fbest, bem como estatísticas globais como o número total de iterações e a taxa de aceitação. A mesma estrutura foi aplicada tanto à Função 1 como à Função 2, alterando apenas os limites do espaço de pesquisa, o tamanho da vizinhança (step) e o número de vizinhos avaliados por temperatura (nRep). A Função de Rastrigin, pela sua elevada multimodalidade, beneficia particularmente da capacidade do SA em aceitar movimentos piores nas fases iniciais, o que lhe permite escapar de muitos mínimos locais antes de “arrefecer” para uma fase de exploração mais local em torno de bons candidatos.

Parâmetros de utilização:

- Temperatura Inicial (T_0): 90.
- Temperatura Final (T_{min}): 10^{-4}
- Fator de Arrefecimento (α): 0.94 (Decaimento geométrico).
- Vizinhos testados por temperatura (nRep):
  - Função 1: 20 vizinhos.
  - Função 2: 50 vizinhos.
- Tamanho do Passo (Vizinhança):
  - Função 1: 0.1.
  - Função 2: 0.2.

4.3. Algoritmo Genético (GA)

A implementação do Algoritmo Genético seguiu rigorosamente o protocolo fornecido, como pode ser observado no Anexo 3, incluindo a discretização binária obrigatória. Cada solução é representada por um cromossoma de 10 bits, correspondendo a 5 bits para cada variável. Esta discretização reduz a resolução do espaço real, impondo que cada variável seja representada através de apenas 32 níveis possíveis. Apesar desta limitação, que condiciona a precisão final das soluções, o algoritmo mantém a sua estrutura tradicional baseada nos princípios de seleção natural, recombinação e mutação.

Representação, codificação e avaliação

A população é armazenada na matriz **CHROME**, onde cada linha corresponde a um indivíduo e cada coluna a um gene binário. A descodificação do cromossoma para o espaço real envolve separar os bits de cada variável, converter a sequência binária em inteiro e escalonar esse valor para os limites reais definidos pelo problema. Este processo garante que cada cromossoma corresponde a um ponto válido no espaço de pesquisa tanto para a Função 1 como para a Função 2 (Rastrigin), variando apenas os limites numéricos aplicados no escalonamento.

A avaliação das soluções utiliza a função objetivo definida para cada função. Como o GA trabalha por maximização, a função objetivo, que pretendemos minimizar, foi convertida em aptidão através da expressão:

$$\text{aptidão} = \frac{1}{1 + \text{ObjFunc}(x, y)}.$$

Assim, quanto menor o valor da função objetivo, maior será a aptidão do indivíduo. Esta transformação permite que o algoritmo evolua naturalmente no sentido da minimização, mantendo a coerência com o mecanismo de seleção.

Os parâmetros usados foram os especificados no esqueleto fornecido: população de 50 indivíduos, cromossomas de 10 bits, 80 gerações, probabilidade de cruzamento de 0.75 e probabilidade de mutação de 0.03. Estes valores garantem diversidade genética suficiente para explorar o espaço de pesquisa, sem comprometer a convergência.

Seleção por Roda da Roleta

A seleção é realizada através do método da Roleta, um operador clássico em Algoritmos Genéticos e particularmente adequado quando os indivíduos apresentam aptidões distintas, mas relativamente próximas, como acontece neste trabalho devido à discretização binária. Neste método, a probabilidade de cada indivíduo ser escolhido é proporcional à sua aptidão, introduzindo uma pressão seletiva moderada: indivíduos com aptidão elevada têm maior probabilidade de contribuir para a geração seguinte, mas indivíduos de aptidão inferior nunca são totalmente excluídos, preservando diversidade genética.

O procedimento consiste em calcular a soma total das aptidões e gerar um número aleatório dentro desse intervalo. Percorrendo-se a população por ordem, acumulando as aptidões, seleciona-se o indivíduo cujo valor acumulado ultrapassa o limiar aleatório gerado. Este processo é repetido sempre que são necessários novos progenitores. Esta abordagem probabilística evita que a população converge demasiado rapidamente para soluções subótimas, algo particularmente importante para funções multimodais como a Rastrigin.

Cruzamento, Mutação e Elitismo

Após a seleção, os progenitores são sujeitos a crossover de ponto simples. Com probabilidade 0.75, um ponto de corte é selecionado aleatoriamente e as partes inferiores dos dois cromossomas são trocadas, originando descendentes com combinações genéticas novas. Quando o cruzamento não ocorre, os descendentes são cópias dos progenitores. O cruzamento constitui a principal força exploratória do GA, permitindo que segmentos de soluções promissoras se recombinem e deem origem a novos padrões.

A mutação é aplicada de forma independente a cada gene, invertendo o bit com probabilidade 0.03. Este operador impede que a população fique geneticamente homogénea, reintroduzindo diversidade sempre que necessário. A mutação é especialmente útil quando o crossover sozinho já não é capaz de gerar diversidade suficiente devido à discretização imposta pelos 5 bits por variável.

Para garantir que o processo evolutivo nunca perde uma solução superior, implementou-se **elitismo**, preservando o melhor indivíduo de cada geração de forma inalterada. Este mecanismo assegura estabilidade na convergência e evita retrocessos no melhor valor encontrado.

Ciclo evolutivo e histórico da execução

Cada geração do GA envolve a sequência: seleção → cruzamento → mutação → descodificação → avaliação. Após a avaliação, registam-se estatísticas relevantes, incluindo a melhor aptidão, a aptidão média e o custo associado ao melhor indivíduo. Estes valores permitem analisar a evolução da população ao longo das gerações e observar o comportamento do algoritmo face a funções com características distintas.

Ao longo do processo, o algoritmo mantém registo do melhor indivíduo encontrado e das respetivas coordenadas no espaço real. No final das 80 gerações, o melhor cromossoma é descodificado, originando a solução aproximada (x_best, y_best) e o valor mínimo estimado da função objetivo. A estrutura do GA permaneceu idêntica para ambas as funções estudadas, sendo apenas ajustados os limites do espaço de pesquisa e a função objetivo. A Função 2, devido à sua elevada multimodalidade, permitiu observar a capacidade do GA em escapar a mínimos locais graças à combinação de seleção estocástica, recombinação e mutação.

Parâmetros de utilização:

- Representação: Binária (5 bits por variável -> 10 bits totais).
- Tamanho da População: 50 indivíduos.
- Número de Gerações: 80.
- Método de Seleção: Roleta.
- Probabilidade de Cruzamento (P_c): 0.75.
- Probabilidade de Mutação (P_m): 0.03.
- Elitismo: Sim

4.4. Particle Swarm Optimization (PSO)

O algoritmo de Otimização por Enxame de Partículas foi implementado para resolver o mesmo problema de minimização considerado nos restantes métodos, tanto para a Função 1 (Schaffer N. 2) como para a Função 2 (Rastrigin). Neste trabalho, cada partícula representa uma solução candidata no espaço bidimensional (x, y), e o objetivo é deslocar o enxame de forma coordenada até à vizinhança do mínimo global.

Representação do enxame e inicialização

O enxame é constituído por um conjunto de partículas, cada uma com:

- uma posição atual $x_i = (x_{i1}, x_{i2})$,
- uma velocidade $v_i = (v_{i1}, v_{i2})$,
- um custo atual $f(x_i)$,
- a melhor posição já encontrada por essa partícula (pbest),
- e o melhor valor da função objetivo associado ao seu pbest.

Na implementação, estas grandezas são guardadas em matrizes e vetores: x (posições), v (velocidades), cost (custos), pbest (melhor posição individual) e pbest_val (melhor custo individual). O melhor ponto global encontrado por qualquer partícula é armazenado nas variáveis gbest (posição) e gbest_val (valor da função).

A inicialização é feita de forma aleatória: para cada partícula, a posição é sorteada uniformemente dentro dos limites $[VarLBounds, VarUBounds]$, a velocidade é inicialmente nula e o custo é avaliado com a função objetivo. Cada partícula começa por considerar a sua posição inicial como pbest. Em seguida, determina-se o melhor pbest de entre todas as partículas, que passa a ser o gbest inicial do enxame. Esta fase estabelece a diversidade espacial do enxame e dá um primeiro "palpite" para a melhor região do espaço de pesquisa.

Parâmetros de utilização

- Dimensão do Enxame: 50 partículas.
- Número de Iterações (Épocas): 80.
- Fator de Inércia (W):
  - Valor Inicial: 0.9.
  - Decaimento ($W_{damping}$): 0.99.
- Coeficiente Cognitivo (c_1): 2.0 (atração pelo *pbest*).
- Coeficiente Social (c_2): 2.0 (atração pelo *gbest*).

- Velocidade Máxima (v_{max}): 10% da amplitude do domínio (limita movimentos excessivos).

Dinâmica de atualização das partículas

Em cada iteração, o PSO atualiza todas as partículas do enxame. A atualização ocorre em dois passos principais: primeiro atualiza-se a velocidade, depois a posição.

A nova velocidade de cada partícula é calculada combinando três componentes: uma componente inercial, que tenta manter a direção e intensidade do movimento anterior; uma componente cognitiva, que puxa a partícula em direção à sua melhor posição passada; e uma componente social, que puxa a partícula em direção à melhor posição global encontrada pelo enxame. Em termos simplificados, para cada partícula i,

$$v_i^{(t+1)} = w\, v_i^{(t)} \;+\; c_1\, r_1 \odot (pbest_i - x_i^{(t)}) \;+\; c_2\, r_2 \odot (gbest - x_i^{(t)}),$$

onde r_1 e r_2 são vetores de números aleatórios uniformes em [0,1] e $\odot$ representa o produto elemento a elemento. Esta equação é implementada diretamente no código com as variáveis w, c1, c2, r1, r2, pbest e gbest.

Depois de calculada a nova velocidade, esta é limitada componente a componente ao intervalo $[-v_{\max}, v_{\max}]$. Esta operação de *clamping* evita que a partícula ganhe uma velocidade excessiva e atravesse o espaço de pesquisa de forma descontrolada. De seguida, a posição é atualizada simplesmente por

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}.$$

As novas posições são também forçadas a permanecer dentro dos limites [VarLBounds, VarUBounds], aplicando-se novamente um *clamping* componente a componente. Assim garante-se que o enxame não sai do domínio definido pelo problema.

Avaliação, pbest, gbest e aptidão

Depois da atualização da posição, cada partícula é reavaliada com a função objetivo, obtendo-se o novo custo cost(i). Se o novo custo for inferior ao melhor valor individual pbest_val(i), então a partícula melhora o seu pbest: a sua posição atual passa a ser a nova melhor posição individual, e o valor da função é registado

como novo pbest_val. Sempre que um pbest melhora, verifica-se se esse valor é também melhor do que o gbest atual; em caso afirmativo, atualiza-se também o melhor global gbest e gbest_val.

Tal como no Algoritmo Genético, foi também considerado um conceito de "aptidão" associada ao custo, definido por

$$\text{fitness} = \frac{1}{1 + \text{custo}},$$

embora no PSO esta métrica seja usada apenas para análise e gráficos, e não diretamente no mecanismo de atualização. Ao longo das iterações, são guardados o melhor valor global da função (histórico de gbest_val), o custo médio do enxame e as aptidões média e máxima. Estes registos permitem, mais tarde, analisar a convergência do enxame e comparar o comportamento do PSO com os restantes métodos.

O fator de inércia wé também atualizado em cada iteração, sendo multiplicado por w_damping = 0.99. Este decaimento gradual faz com que no início a componente inercial tenha mais peso (exploração mais ampla do espaço) e, à medida que as iterações avançam, o comportamento se torne mais "focado" e exploratório-local, favorecendo a convergência.

Ciclo de iterações e adaptação às duas funções

O ciclo principal do PSO repete-se até atingir o número máximo de iterações (maxit = 80). Em cada época, o enxame é atualizado de acordo com as regras descritas: atualização de velocidades, restrição pela velocidade máxima, atualização de posições, limitação aos limites do domínio, avaliação dos custos e atualização de pbest e gbest. Paralelamente, são registadas estatísticas de convergência que serão exploradas na secção de resultados.

A mesma estrutura foi utilizada tanto para a Função 1 como para a Função 2, alterando apenas:

- a função objetivo ObjFunc(x,y);
- e os limites VarLBounds e VarUBounds, que são mais amplos no caso da Rastrigin.

A Função 2, por ser altamente multimodal e conter um grande número de mínimos locais, constitui um cenário de teste exigente para o PSO. A presença das componentes cognitiva e social, combinada com o controlo de inércia, permite ao enxame escapar de mínimos locais e concentrar-se progressivamente em regiões

mais promissoras, tal como se observa nos gráficos de convergência e nos resultados finais obtidos, os quais serão explicados mais à frente.

No final da execução, o melhor ponto global gbest encontrado ao longo de todas as iterações é reportado como solução aproximada do problema, juntamente com o valor mínimo estimado da função objetivo no ponto $(x_{\text{gbest}}, y_{\text{gbest}})$.

5. Resultados

Nesta secção apresentam-se os resultados experimentais obtidos com a aplicação dos quatro métodos de otimização estudados.

Para cada método são incluídos resultados quantitativos (melhor valor, média, variabilidade) e resultados gráficos ilustrativos (evolução do custo, trajetória das soluções e visualizações 2D/3D da função), permitindo comparar o desempenho das diferentes abordagens de forma clara e estruturada.

5.1. Subida da Colina com Reinicialização Múltipla

5.1.1. Função 1

Para avaliar este algoritmo, realizaram-se 10 tentativas independentes (reinicializações) com 100 iterações cada.

Resultados Quantitativos

- **Melhor Valor Obtido:** $1.702 * 10^{-5}$ (0.00001702)
- **Melhor Posição:** (0.0030, 0.0028)
- **Erro Absoluto:** Aproximadamente 10^{-5}face ao ótimo global (0).

Analise gráfica

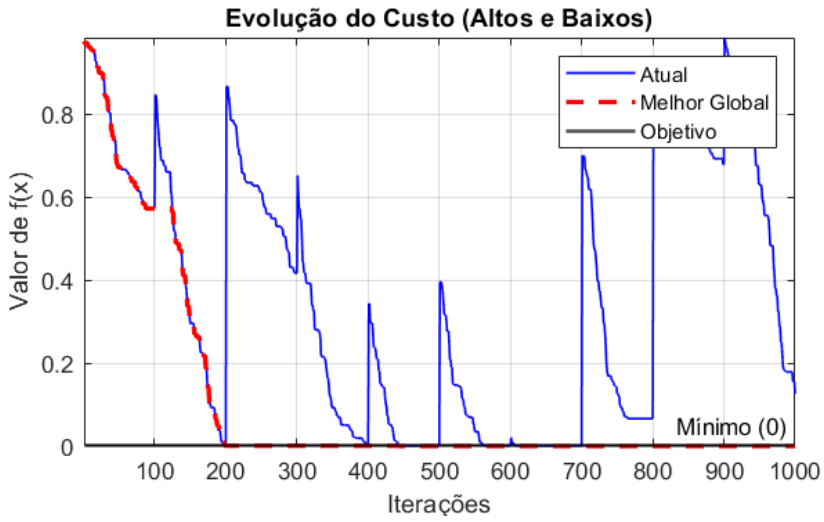

Figura 3: Evolução do Custo da Função 1.

O gráfico evidencia a dinâmica do método de Reinicialização Múltipla através de duas componentes distintas:

- **Custo Instantâneo (Linha Azul):** Apresenta um padrão oscilatório característico em "dentes de serra". Cada pico vertical corresponde ao início de uma nova tentativa (*restart*) num ponto aleatório do domínio, seguido de uma descida rápida e monótona provocada pelo algoritmo de pesquisa local (*Hill Climbing*). O facto de a descida estagnar rapidamente em valores diferentes de zero em algumas tentativas confirma a existência de mínimos locais onde o algoritmo ficou retido.
- **Melhor Global (Linha Vermelha):** Representa a memória do sistema. Ao contrário da linha azul, esta curva é não-crescente (monótona), atualizando-se apenas quando uma nova tentativa supera o melhor resultado histórico. Observa-se que, após algumas reinicializações, o algoritmo conseguiu identificar uma bacia de atração que permitiu descer até um valor na ordem de 10^{-5}, validando a eficácia da estratégia estocástica para superar as limitações da pesquisa local pura.

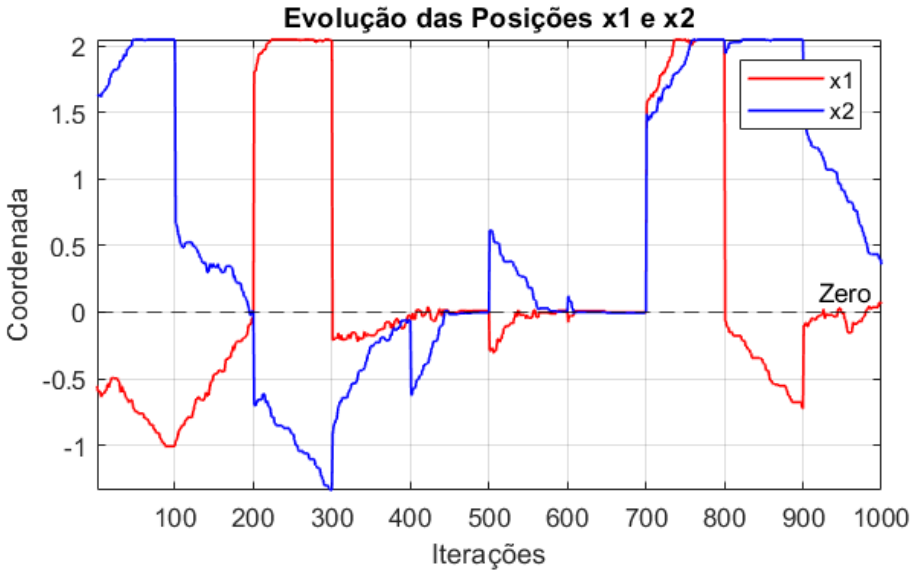

Figura 4: Trajetória das coordenadas x_1 e x_2 na Função 1.

A Figura 4 corrobora a análise anterior. As descontinuidades verticais nas linhas vermelha e azul correspondem aos momentos de reinicialização ("teletransporte" para um novo ponto aleatório). É visível que, nas tentativas bem-sucedidas, ambas as linhas convergem e estabilizam sobre a linha tracejada do zero, validando a descoberta da solução correta.

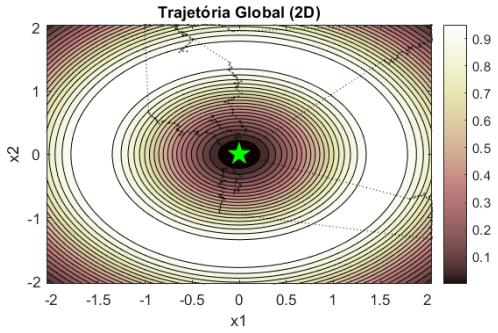

Figura 5: Trajetória 2D do Subida na Colina na função 1.

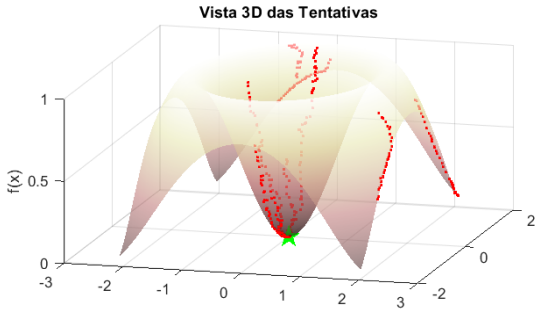

Figura 6: Trajetória 3D do Subida na Colina na função 1.

As visualizações 2D e 3D ilustram a topologia concêntrica da função Schaffer. A dispersão dos pontos mostra que o algoritmo explorou diversas regiões do espaço (bacias de atração locais) antes de conseguir "acertar" na bacia central e convergir para a solução marcada com a estrela.

Analise Função 1

O método provou ser eficaz para esta função. A limitação da precisão final (não atingir o 0.00000000 absoluto) deve-se ao tamanho fixo do passo (step = 0.05), que impede o ajuste fino final, mas o resultado de 0.000017 é mais do que satisfatório para validar a implementação

5.1.2. Função 2

Devido à elevada complexidade da superfície de Rastrigin, o algoritmo foi configurado para realizar **50 reinicializações** (5000 iterações totais no gráfico), aumentando a probabilidade de encontrar o "buraco" correto.

Resultados Quantitativos

- **Melhor Valor Obtido:** 1.0016
- **Melhor Posição:** (-0.0056, -0.9934)

A análise dos resultados mostra que, mesmo com 50 tentativas, o algoritmo nem sempre consegue atingir o mínimo global (0). No caso apresentado, a melhor solução ficou retida num mínimo local onde $x_2 \approx -1$, resultando num custo final de ≈ 1.0.

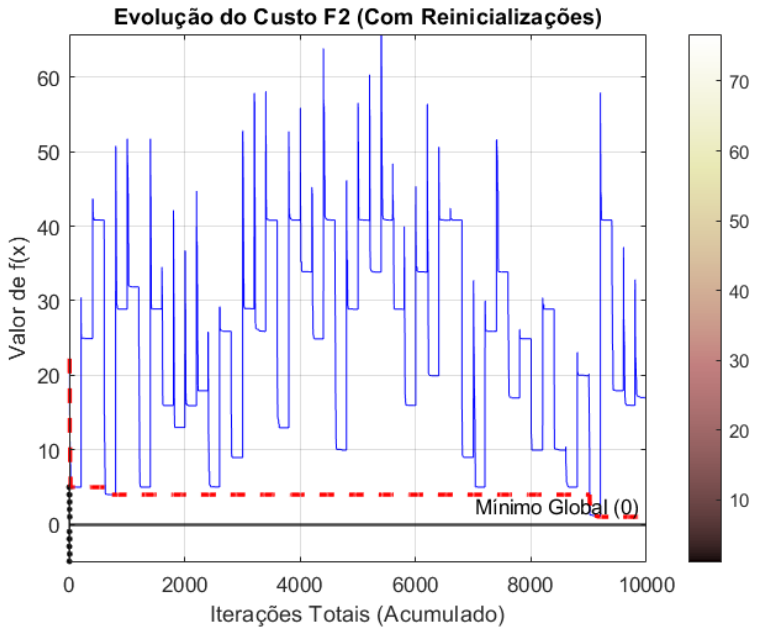

Figura 7: Evolução do Custo da Função 2.

A Figura 7 evidencia o desafio imposto pelos mínimos locais.

- **Instabilidade do Custo (Linha Azul):** Ao contrário da Função 1, onde quase todas as descidas atingiam valores baixos, aqui vemos que a maioria das tentativas estagna em patamares elevados (valores entre 10 e 60). Isto acontece quando o ponto inicial sorteado cai numa bacia de atração longe da origem.
- **Convergência por Persistência (Linha Vermelha):** A linha do "Melhor Global" desce em degraus discretos. O algoritmo passou grande parte do tempo sem melhorar o recorde, até que uma das reinicializações finais acertou numa vizinhança melhor, fazendo o erro cair significativamente, embora sem atingir o zero absoluto nesta execução.

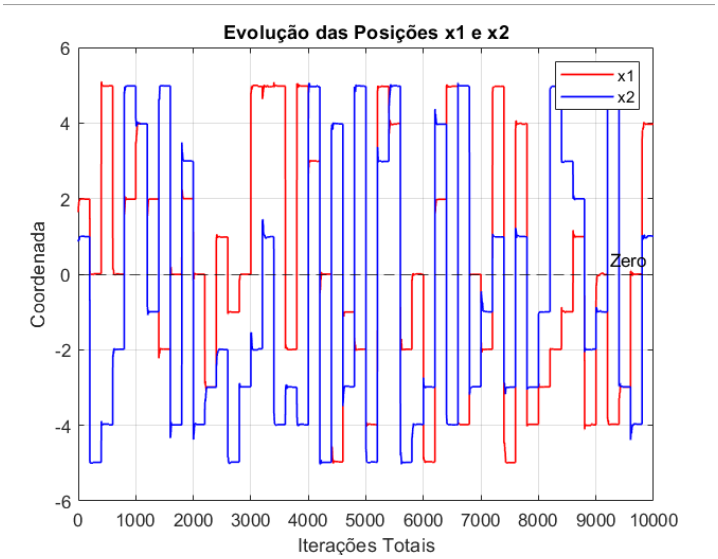

Figura 8: Trajetória das coordenadas x_1 e x_2 na Função 2.

A Figura 8 ilustra a exploração extensiva do domínio [-5.12, 5.12].

- As mudanças bruscas de nível mostram o algoritmo a "saltar" para zonas completamente diferentes do mapa a cada reinicialização.
- Observa-se que, na maioria das vezes, as linhas estabilizam em valores inteiros (ex: $x_1 \approx 2, x_1 \approx 3$) que correspondem aos mínimos locais da função Rastrigin. Apenas no final se observa a convergência de uma variável para o zero, enquanto a outra estabilizou em -1.

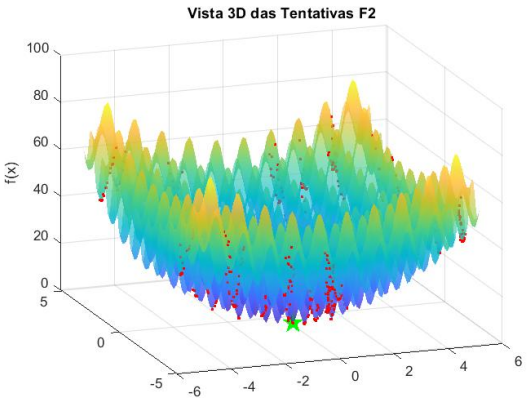

Figura 9: Trajetória 2D do Subida na Colina na função 2.

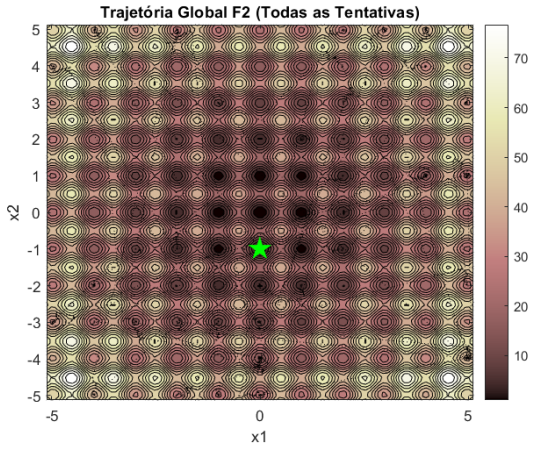

Figura 10: Trajetória 3D do Subida na Colina na função 2.

As visualizações espaciais demonstram claramente a densidade de mínimos locais. O algoritmo mapeou uma "grelha" de soluções subótimas (pontos pretos) espalhados por todo o domínio. A estrela verde destaca a melhor solução encontrada, que se situa numa das bacias centrais, mas não necessariamente na origem absoluta (0,0).

Análise da Função 2

Na execução apresentada, o algoritmo convergiu para um valor final de aproximadamente 1.0016. A análise das coordenadas finais (-0.0056, -0.9934) revela que o algoritmo conseguiu encontrar o valor correto para x_1 (aprox. 0), mas ficou retido num mínimo local para x_2(aprox. -1). Este resultado ilustra a principal fraqueza da pesquisa local em funções altamente multimodais: a forte dependência da bacia de atração inicial. Mesmo com 50 reinicializações, a garantia de encontrar o ótimo global absoluto não é de 100%, embora se consigam consistentemente soluções de boa qualidade próximas do objetivo.

5.2. Simulated Annealing (SA)

Nesta secção apresentam-se os resultados obtidos pela execução do algoritmo Simulated Annealing nas duas funções de teste consideradas. Para cada função são incluídos e analisados os principais gráficos gerados durante a execução.

5.2.1. Função 1

Nesta experiência, o algoritmo Simulated Annealing (SA) foi aplicado à Função 1 (Schaffer-like), com o objetivo de atingir o mínimo global na vizinhança de (0,0). O método recorreu a uma temperatura inicial elevada (T = 90), um arrefecimento

geométrico (α = 0.94) e 20 vizinhos testados por nível térmico, resultando num total de **4440 iterações** até a temperatura atingir o limite inferior.

Resultados quantitativos

- **Ponto Inicial:** (1.5427, –1.6897)
- **Melhor Ponto Encontrado:** (0.0027, –0.0014)
- **Melhor Valor da Função:** 0.0000091032
- **Total de Iterações:** 4440
- **Taxa de Aceitação:** 68.0%

O valor final obtido é extremamente próximo do mínimo global teórico (0). Este resultado confirma que o algoritmo conseguiu:

1. **Escapar sucessivamente aos mínimos locais periféricos**, graças à elevada temperatura inicial que permite aceitar movimentos de pior custo;
2. **Convergir para a bacia central profunda**, visível nos gráficos 2D e 3D;
3. **Refinar progressivamente a solução**, à medida que a temperatura descia e o algoritmo se tornava mais seletivo.

A taxa de aceitação de 68% é típica de um processo bem calibrado: no início, aceita quase tudo para explorar; no final, aceita apenas movimentos que melhorem a solução, assegurando precisão no ajuste fino.

Analise gráfica

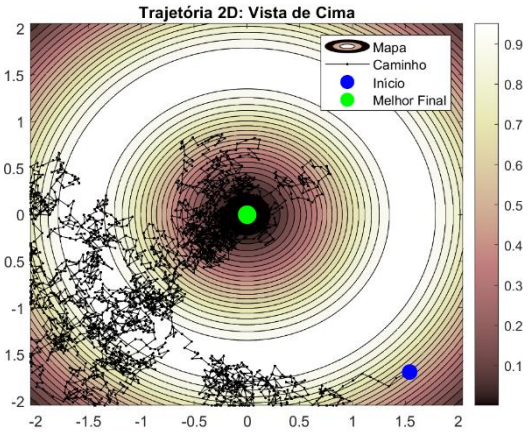

Figura 11: Trajetória 2D do SA na Função 1.

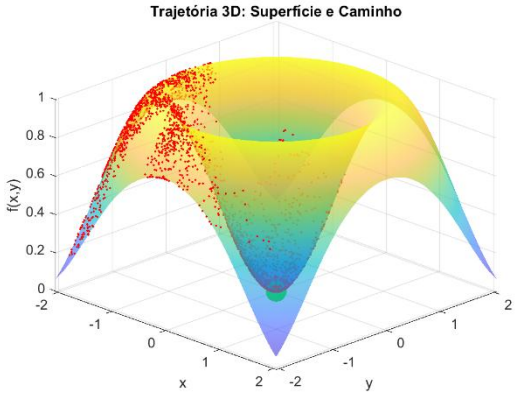

Figura 12: Trajetória 3D do SA na Função 1.

A trajetória bidimensional do algoritmo mostra claramente a natureza exploratória inicial do método (Figura 11). Nos primeiros milhares de iterações, observa-se um

percurso irregular e disperso, resultado da temperatura elevada que permite aceitar soluções piores com probabilidade significativa. Esta fase é essencial para escapar a mínimos locais, especialmente numa função como a F1, altamente multimodal e caracterizada por anéis concêntricos de máximos e mínimos. À medida que a temperatura diminui, o movimento torna-se progressivamente mais estável e orientado, aproximando-se gradualmente da bacia de atração central. A convergência para o mínimo global é também evidente na vista 3D (Figura 12), onde os pontos aceites descrevem uma descida gradual em direção ao vale central, confirmando que o algoritmo não ficou preso em regiões periféricas de menor profundidade.

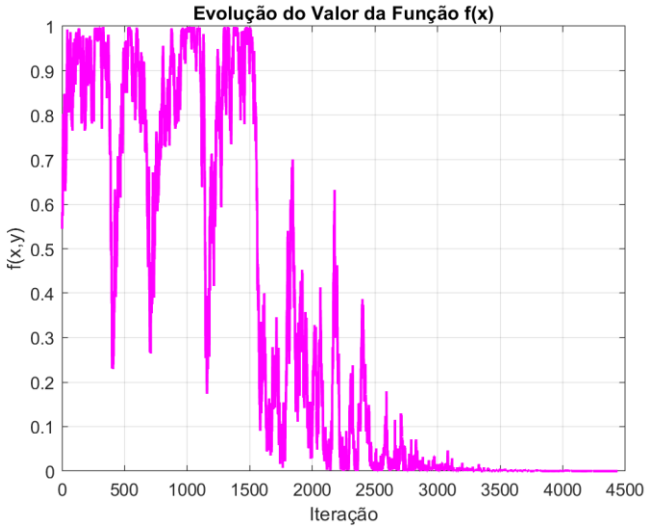

Figura 13: Evolução do Valor da Função do SA na Função 1.

A curva de f(x) ao longo das iterações evidencia três fases (Figura 13):

1. **Exploração inicial (0–1500 iterações):**
   Oscilações muito grandes, indicando aceitação frequente de soluções piores, essencial para sair de vales locais.
2. **Transição (aprox. 1500–2500):**
   Redução progressiva da variabilidade, coincidindo com o arrefecimento térmico.
3. **Convergência (2500–4400):**
   A função aproxima-se de zero sem retrocessos significativos, revelando que o algoritmo ficou mais "guloso", aceitando quase só melhorias.

Este comportamento é típico de um cronograma de arrefecimento eficaz.

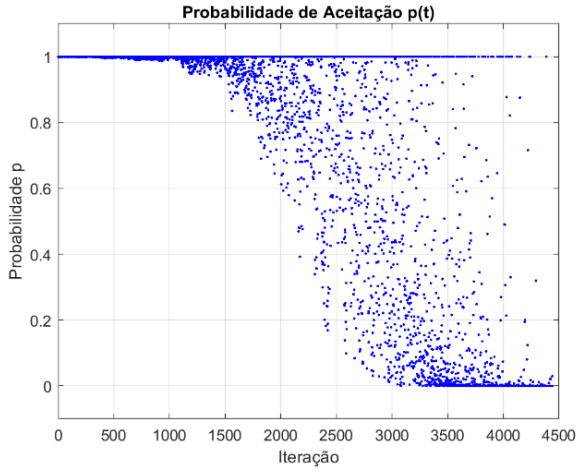

Figura 14: Probabilidade de Aceitação do SA na Função 1.

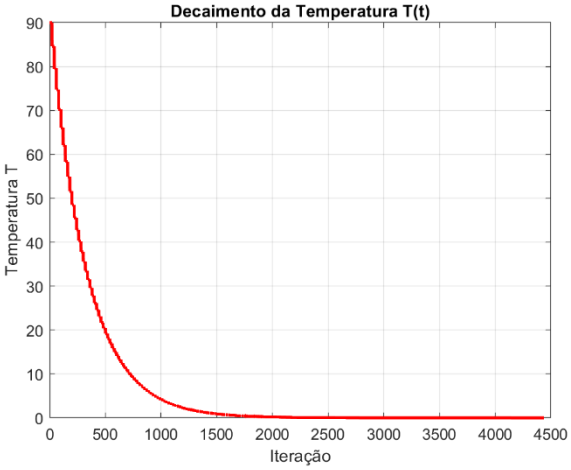

Figura 15: Decaimento da Temperatura do SA na Função 1.

Outra componente fundamental para interpretar o comportamento do SA é a probabilidade de aceitação (Figura 14). Esta variável começa muito próxima de 1 nas primeiras iterações, permitindo grande liberdade de movimento, independentemente de a solução ser melhor ou pior. No entanto, o gráfico mostra claramente um declínio progressivo da dispersão vertical: entre as iterações 2000 e 3000 a probabilidade situa-se já entre 0.8 e 0.3, e após as 3500 iterações aproxima-se consistentemente de 0. A queda da probabilidade acompanha diretamente a curva da temperatura, confirmando que o algoritmo se torna mais seletivo e "guloso" à medida que avança para a fase de exploração local fina.

O gráfico da temperatura (Figura 15) exibe um decaimento suave e monotónico, refletindo um arrefecimento controlado sem quebras bruscas. Este comportamento é fundamental para evitar convergência precoce: se a temperatura descesse demasiado rápido, o algoritmo deixaria de explorar com eficácia; se descesse demasiado devagar, gastaria mais tempo em exploração aleatória. O ritmo observado mostra um equilíbrio adequado entre ambas as fases.

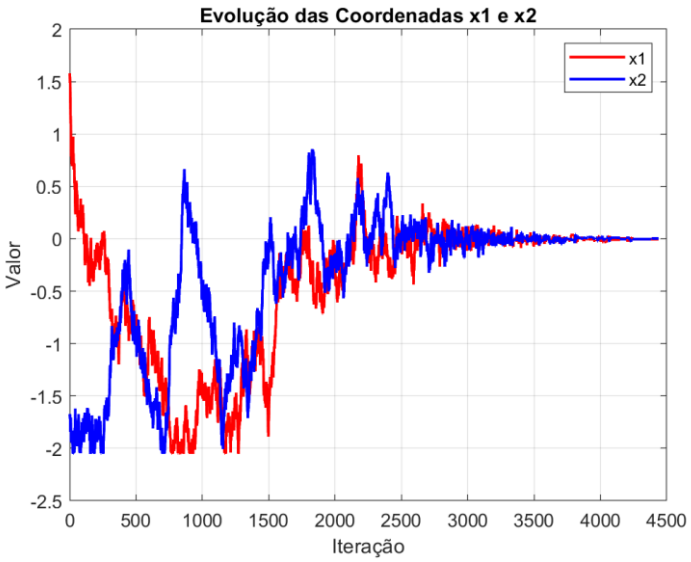

Figura 16: Evolução das Coordenadas do SA na Função 1.

Por fim, a evolução das coordenadas x_1 e x_2 (Figura 16) mostra um comportamento coerente com os restantes resultados. Inicialmente, ambos os valores oscilam de forma significativa, cobrindo regiões amplas do espaço. A partir das iterações 2000–2500, começam a apresentar oscilações cada vez menores, aproximando-se gradualmente de zero. A estabilização simultânea das duas coordenadas indica que o algoritmo entrou na bacia de atração central e que a descida local já não é perturbada por aceitação de soluções piores, pois a temperatura nessa fase é praticamente nula.

Analise Função 1

Os resultados obtidos confirmam que o Simulated Annealing foi capaz de atingir eficazmente o mínimo global da função F1. A estratégia de arrefecimento e o tamanho da vizinhança revelaram-se adequados, permitindo uma exploração inicial extensa e uma fase final estável e direcionada. A trajetória espacial, as curvas de convergência, a evolução probabilística e o comportamento das variáveis reforçam a interpretação de que o algoritmo evitou mínimos locais e convergiu consistentemente para a solução correta. O valor mínimo obtido está extremamente próximo do valor teórico, demonstrando que o método foi implementado de forma sólida e que os parâmetros utilizados constituem um equilíbrio eficiente entre exploração e refinamento local.

5.2.2. Função 2

Tal como na Função 1, foi utilizada uma temperatura inicial elevada (T = 90), um arrefecimento geométrico (α = 0.94) e vários vizinhos testados por nível térmico, conduzindo a um total de **11 100 iterações** até a temperatura atingir o limite inferior pré-definido.

Resultados quantitativos

- **Ponto Inicial:** (2.1374, –3.2791)
- **Melhor Ponto Encontrado:** (–0.9969, –0.0145)
- **Melhor Valor da Função:** 1.0375375925
- **Total de Iterações:** 11 100
- **Taxa de Aceitação:** 23.6%

O algoritmo iniciou a pesquisa num ponto afastado da origem, **(2.1374, –3.2791)**, e percorreu um total de **11 100 iterações**, terminando com uma **taxa de aceitação de 23.6%**, um valor significativamente mais baixo do que na Função 1, o que já antecipa a maior complexidade da paisagem e a necessidade de aceitar menos movimentos à medida que a temperatura diminui.
O melhor ponto encontrado ao longo da execução foi **(–0.9969, –0.0145)**, com um valor mínimo de aproximadamente **1.0375**. Embora não tenha atingido o ótimo global teórico (0,0), o algoritmo convergiu para um dos vales centrais de menor energia, o que confirma que o processo de arrefecimento foi eficaz, ainda que a complexidade da função tenha exigido um esforço exploratório muito superior ao da Função 1.

Analise Gráfica

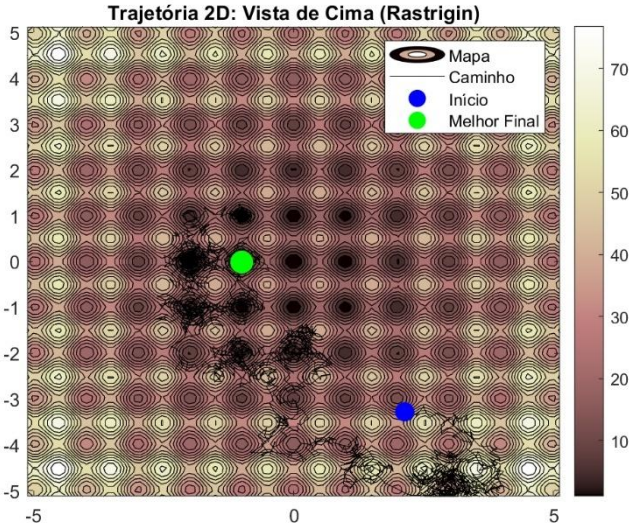

Figura 17: Trajetória 2D do SA na Função 2.

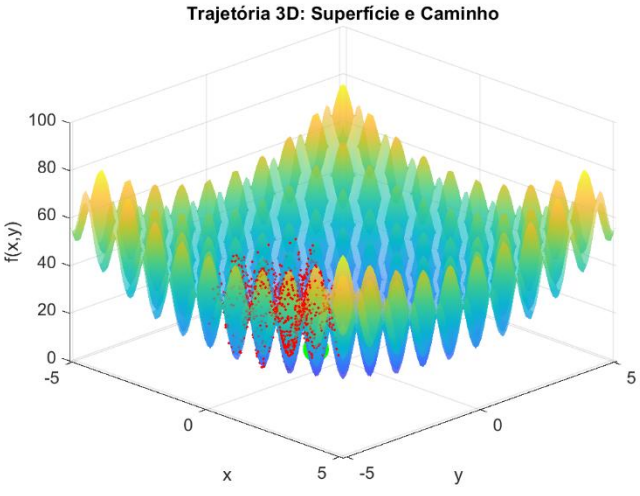

Figura 18: Trajetória 3D do SA na Função 2.

A análise dos gráficos permite compreender detalhadamente o comportamento do algoritmo ao longo da execução.

A vista 2D (Figura 17) evidencia uma grande mobilidade inicial: o caminho percorrido mostra saltos sucessivos entre regiões de energia muito distintas, algo esperado numa fase em que a temperatura é elevada e a probabilidade de aceitar soluções piores ainda é próxima de 1. O ponto final surge próximo da crista principal que envolve o mínimo global, o que demonstra que o algoritmo conseguiu escapar a vários mínimos locais intermediários antes de estabilizar numa bacia de menor profundidade. A localização do ponto ótimo encontrado é coerente com a natureza altamente fragmentada da função de Rastrigin.

Na vista 3D (Figura 18), observa-se com clareza a distribuição ondulada da função e a forma como o algoritmo navega entre vales e picos acentuados. A trajetória apresenta uma concentração mais densa na região central, refletindo o processo gradual de refinamento da busca: nos primeiros milhares de iterações, o movimento é disperso e irregular, mas à medida que a temperatura diminui, os pontos aceites aproximam-se cada vez mais de uma única região de convergência. A densidade do caminho junto do mínimo obtido confirma que o algoritmo atingiu uma zona de estabilidade térmica onde já quase não aceita movimentos de subida.

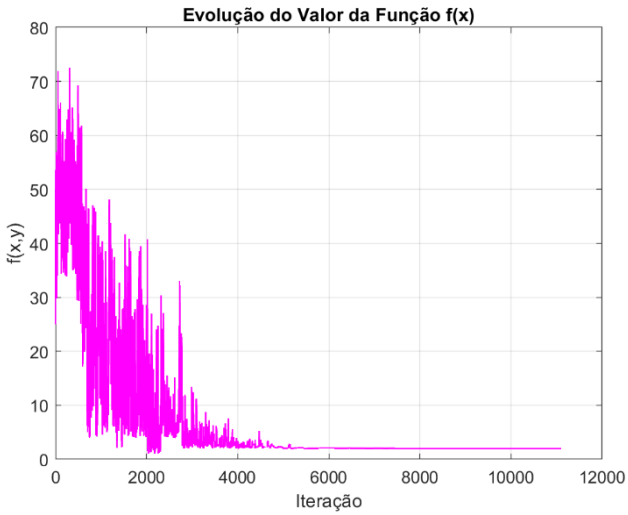

Figura 19: Evolução do Valor da Função do SA na Função 2.

O gráfico da evolução do valor da função (Figura 19) mostra uma redução acentuada durante as primeiras 2000 a 3000 iterações, período em que o algoritmo explora intensamente o espaço. Após esta fase, o decréscimo torna-se gradual e irregular, com pequenos aumentos ocasionais que refletem a aceitação de soluções piores quando a temperatura ainda o permite. A partir de cerca de 6000 iterações, observa-se uma estabilização clara: o valor oscila ligeiramente à volta do

mínimo encontrado, sinal de que a temperatura já se aproxima de zero e os movimentos permitidos são essencialmente descidas locais.

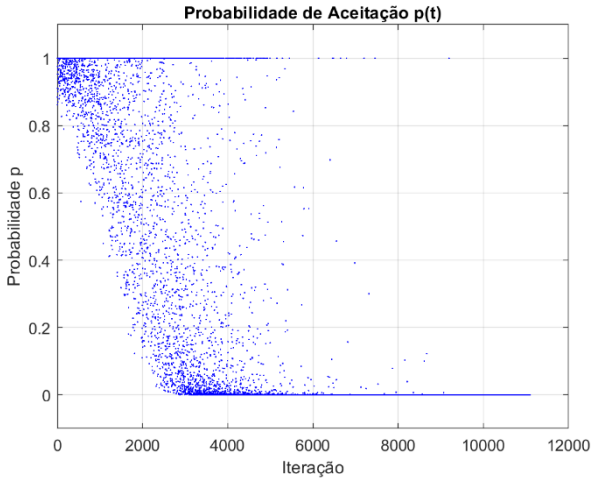

Figura 20: Probabilidade de Aceitação do SA na Função 2.

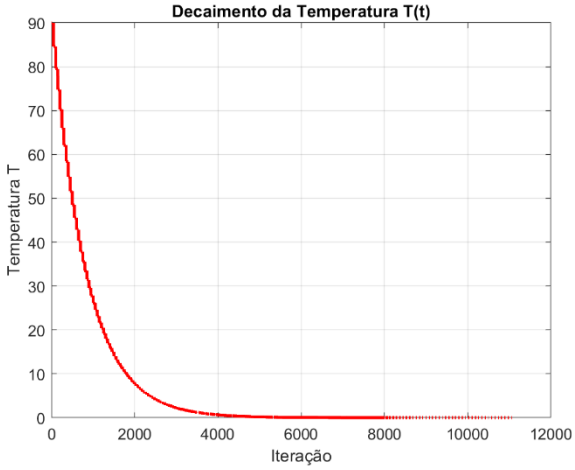

Figura 21: Decaimento da Temperatura do SA na Função 2.

O gráfico da probabilidade de aceitação (Figura 20) reforça esta leitura: no início da execução, a maioria dos movimentos é aceite ($p \approx 1$), demonstrando que o algoritmo tem liberdade para explorar quase todo o domínio. Contudo, a probabilidade decresce rapidamente à medida que o arrefecimento progride e, por volta dos 3000–3500 passos, cai praticamente para zero. A partir daí, o algoritmo passa a comportar-se de forma semelhante a um método de descida local, aceitando apenas movimentos que melhorem a solução.

O decaimento da temperatura (Figura 21) segue uma curva exponencial suave e coerente com o fator de arrefecimento escolhido ($\alpha = 0.94$). Ao fim de cerca de 5000 iterações, a temperatura já se encontra suficientemente baixa para limitar drasticamente a aceitação de novos movimentos exploratórios. O gráfico das coordenadas x1 e x2 apresenta uma evolução compatível com esta dinâmica: nos primeiros estágios, as variáveis oscilam violentamente entre diferentes regiões do domínio; porém, após a temperatura decrescer o suficiente, ambas estabilizam gradualmente até convergirem para valores próximos de (–1, 0), coincidindo com o mínimo encontrado pelo algoritmo.

Analise da função 2

Em síntese, os resultados comprovam que o Simulated Annealing foi capaz de navegar eficazmente numa função altamente multimodal como a de Rastrigin. O método explorou amplamente o espaço de pesquisa enquanto a temperatura era alta e, à medida que arrefecia, refinou a busca até convergir para uma solução de baixa energia. Embora não tenha atingido o mínimo global, conseguiu evitar a

maioria dos mínimos locais e aproximou-se significativamente da região central da função. Tendo em conta a complexidade extrema da paisagem e o facto de a função possuir centenas de mínimos locais, o desempenho observado confirma a robustez do método e a adequação dos parâmetros utilizados.

5.3. Algoritmo Genético (GA)

5.3.1. Função 1

Nesta experiência, o Algoritmo Genético foi executado com uma população de 50 indivíduos ao longo de 80 gerações, utilizando a codificação binária obrigatória de 10 bits (5 para x, 5 para y).

Resultados Quantitativos

- **Melhor Valor Obtido:** 0.00871
- **Melhor Posição:** (0.0661, -0.0661)

O valor final não é zero absoluto. Isto era **esperado** e justifica-se pela discretização do domínio. Com apenas 5 bits para representar o intervalo [-2.048, 2.048], o "passo" mínimo entre números é de aproximadamente 0.13. A coordenada (0,0) não possui representação binária nesta grelha específica. O ponto (0.0661, -0.0661) corresponde ao vizinho representável mais próximo da origem, onde a função vale exatamente 0.0087. Portanto, o GA encontrou com sucesso o **melhor ótimo possível** dada a restrição de bits.

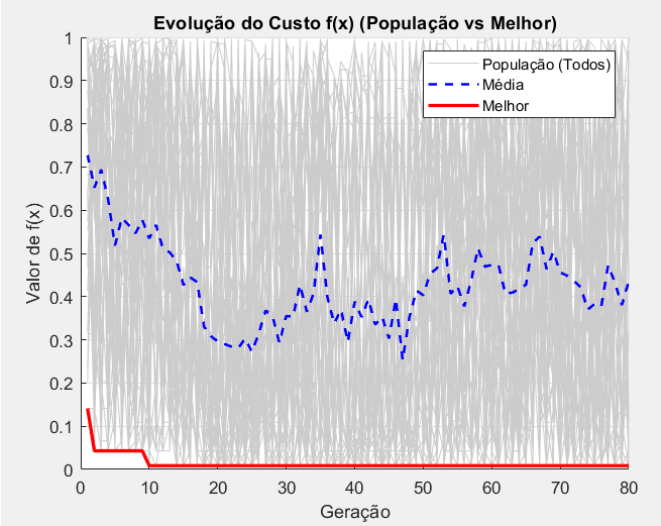

Figura 22: Convergência do Custo da Função 1.

A Figura 22 ilustra a dinâmica populacional durante a minimização:

- **Convergência Rápida (Linha Azul):** A linha do "Melhor f por geração" desce abruptamente e estabiliza num valor próximo de zero logo nas primeiras 10 gerações. A estabilidade subsequente deve-se ao mecanismo de **Elitismo**, que preserva a melhor solução encontrada.
- **Diversidade da População (Linha Laranja):** A "Média de f por geração" mantém-se significativamente acima do melhor valor e apresenta oscilações constantes. Isto é um indicador muito positivo, pois demonstra que, enquanto o indivíduo "elite" segura o ótimo, o resto da população continua a explorar o espaço de pesquisa (devido à mutação e cruzamento), mantendo a diversidade genética necessária para evitar estagnação prematura.

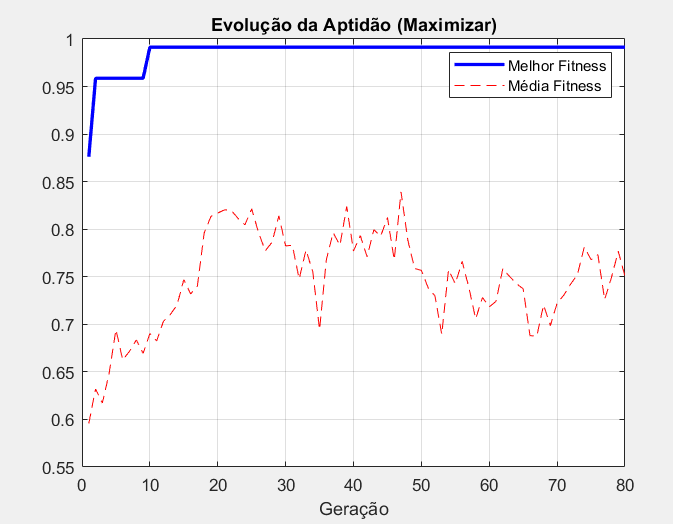

Figura 23: Evolução da Aptidão da Função 1.

A Figura 23 apresenta a comparação entre a aptidão do melhor indivíduo e a média da população: comparativa entre o melhor indivíduo e a média da população revela a "saúde" do processo evolutivo:

1. **Estabilidade do Melhor (Linha Azul):** Mantém-se maximizada, corroborando a eficácia do elitismo.
2. **Oscilação da Média (Linha Vermelha):** A média da aptidão não colapsa para o valor do melhor indivíduo, mantendo uma flutuação constante. Isto é um indicador positivo de que a **diversidade genética** foi preservada até ao final da execução.

3. **Impacto da Mutação:** A taxa de mutação de 3%, aliada ao cruzamento (75%), impediu a homogeneização da população (convergência prematura). Mesmo com o ótimo encontrado, o algoritmo continuou a explorar vizinhanças através de perturbações genéticas, o que é crucial em funções multimodais para evitar ficar preso em mínimos locais periféricos.

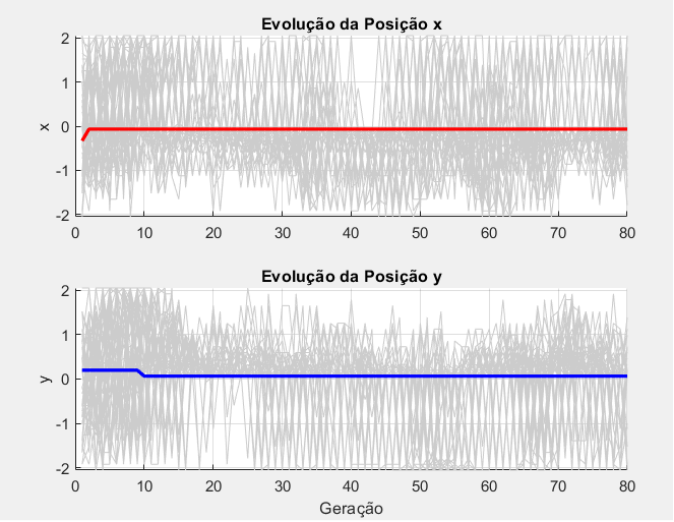

Figura 24: Evolução das coordenadas x1 e x2 ao longo das gerações.

A Figura 24 detalha o comportamento das variáveis:

- Observa-se que ambas as coordenadas estabilizam rapidamente em torno de zero (linha tracejada).
- As "ondas quadradas" ou mudanças bruscas de nível visíveis nas linhas correspondem aos momentos em que o algoritmo salta entre diferentes valores discretos da grelha de 5 bits.
- A convergência simultânea de ambas as variáveis para a vizinhança de zero confirma que o algoritmo identificou corretamente a bacia de atração central da função Schaffer.

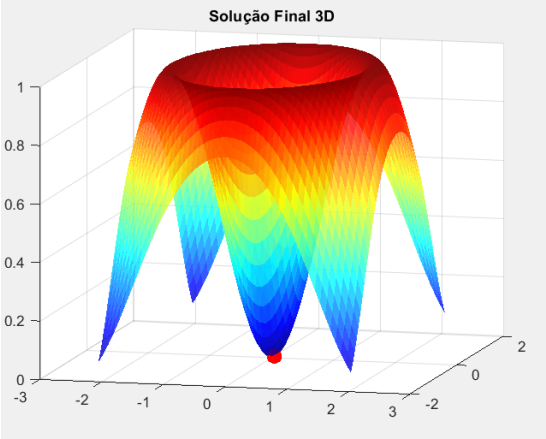

Figura 25: Trajetória 3D no AG da Função 1.

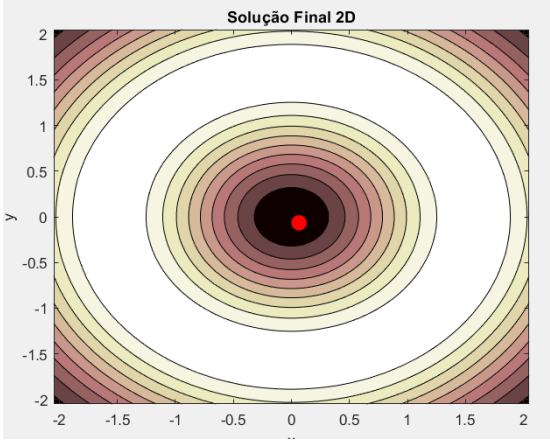

Figura 26: Trajetória 2D no AG da Função 1.

As Figuras 25 e 26 permitem uma validação visual do resultado:

- A função utilizada é multimodal, apresentando vários picos e vales locais (visíveis nas ondas concêntricas), o que dificulta a otimização.
- O ponto vermelho encontra-se claramente posicionado na "bacia" central mais profunda, confirmando que o algoritmo evitou os mínimos locais periféricos.
- A posição final (0.0661, -0.0661) está visualmente centrada, corroborando o baixo erro numérico obtido.

Análise da Função 1

Face aos resultados, conclui-se que a implementação do Algoritmo Genético foi bem-sucedida. O algoritmo evitou mínimos locais e convergiu eficazmente para a bacia de atração correta.

Como demonstrado, o erro residual deve-se exclusivamente à limitação de resolução (5 bits) e não a falha do método. Assim, confirma-se que o sistema atingiu o ótimo global representável, validando a robustez dos parâmetros de seleção e mutação utilizados para encontrar a melhor solução tecnicamente possível.

5.3.2. Função 2

Resultados Quantitativos

- Melhor Valor Obtido (Custo): 9.89118
- Melhor Posição (x, y): (0.1652, -0.1652)

O valor final de aproximadamente 9.89 é significativamente superior ao obtido na Função 1, mas não indica falha do algoritmo.

Este valor resulta da elevada sensibilidade da função Rastrigin. Como visto anteriormente, a grelha de 5 bits obriga a um desvio mínimo de ± 0.1652 da origem. Ao contrário da Função 1 (que é mais suave), na Rastrigin, devido ao termo oscilatório forte (-10*cos(2*pi *x)), esse pequeno desvio de coordenadas provoca um aumento abrupto no custo. O algoritmo encontrou, de facto, a melhor solução permitida pela resolução binária.

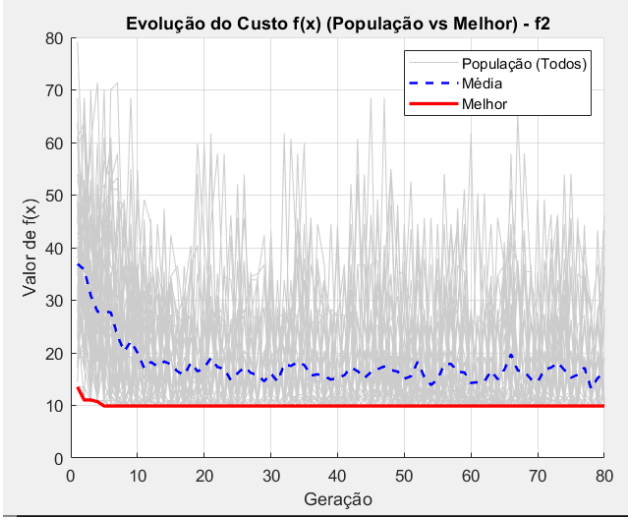

Figura 27: Convergência do Custo da Função 2.

A Figura 27 ilustra a dificuldade do problema:

- **Dispersão Elevada (Nuvem Cinzenta):** A mancha cinzenta ocupa quase todo o gráfico vertical, indicando que, durante toda a execução, existem indivíduos com custos muito elevados (entre 20 e 70). Isto acontece porque a função está cheia de mínimos locais; a mutação atira constantemente indivíduos para "fora" das boas zonas, mantendo a diversidade alta.
- **Estabilidade do Melhor (Linha Vermelha):** Apesar do caos na população geral, a linha do melhor indivíduo (vermelha) estabiliza rapidamente no valor mínimo (9.89). O **Elitismo** foi fundamental para segurar esta solução

e impedir que ela fosse destruída pelas mutações que afetaram o resto da população.

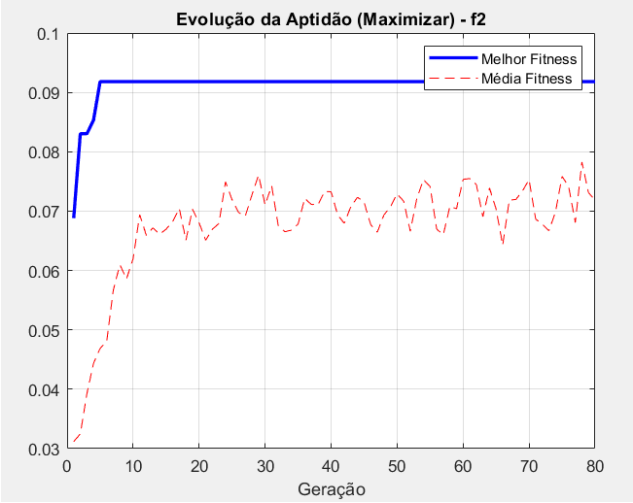

Figura 28: Evolução da Aptidão da Função 2.

A Figura 28 demonstra uma oscilação "nervosa" na linha da média (vermelha). Isto reflete a topologia acidentada da função Rastrigin: as mutações atiram frequentemente indivíduos para zonas de alto custo (mínimos locais vizinhos), baixando a média, mas garantindo que a pesquisa não estagna. A estabilidade da linha azul confirma que o melhor indivíduo sobreviveu a essa turbulência.

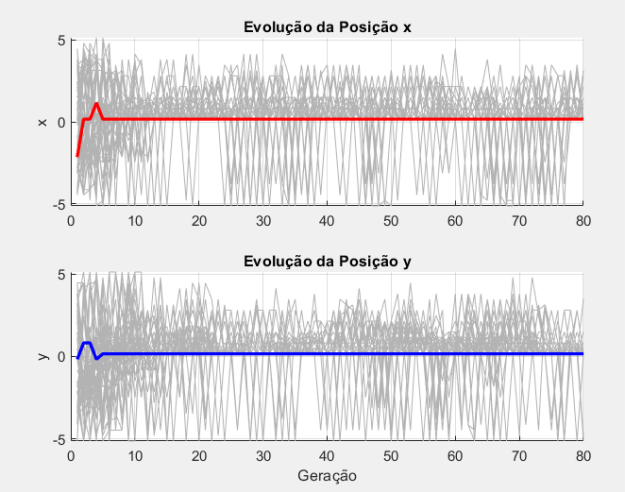

Figura 29: Evolução das coordenadas x1 e x2 na Função Rastrigin.

A análise da Figura 29 revela o comportamento de procura:

- **Convergência Tardia da População:** A nuvem cinzenta mantém-se muito larga (cobrindo o intervalo [-2, 2]) até cerca da geração 40. Isto confirma que o algoritmo não convergiu prematuramente; a população continuou a explorar várias zonas do espaço.

- **Estabilização do Melhor:** As linhas coloridas (Vermelha/Azul) mostram que o melhor indivíduo encontrou o valor estável perto de zero logo no início (aprox. geração 10), mas o resto da população continuou a "saltitar" à procura de alternativas melhores, o que valida a robustez da pesquisa.

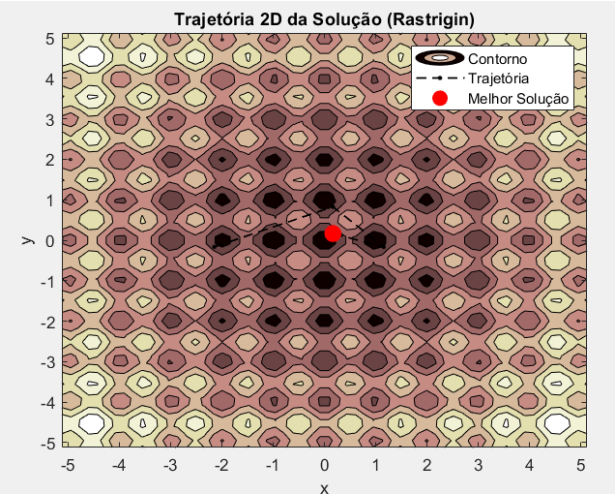

Figura 30: Trajetória 2D no AG da Função 2.

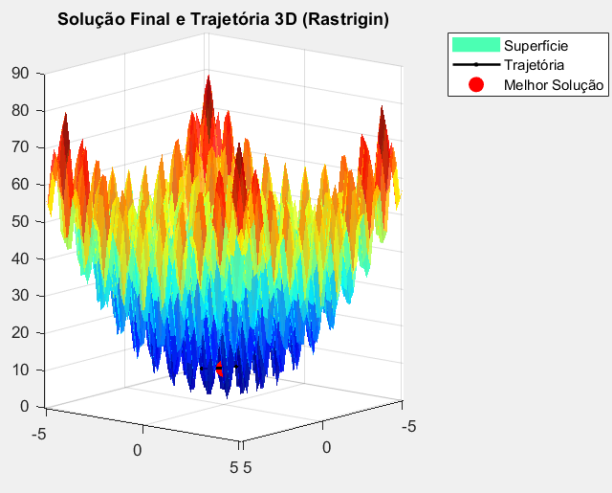

Figura 31: Trajetória 3D no AG da Função 2.

Visualmente, as Figuras 30 e 31 confirmam que o ponto vermelho está na depressão central do "tabuleiro de ovos" (Rastrigin), evitando as dezenas de mínimos locais circundantes.

Análise Função 2

A aplicação do Algoritmo Genético à função Rastrigin foi bem-sucedida, superando o desafio da elevada multimodalidade.

O principal obstáculo desta função, a existência de múltiplos mínimos locais profundos que costumam "prender" métodos de pesquisa local, foi ultrapassado eficazmente pelos operadores genéticos, que mantiveram a diversidade populacional necessária para explorar o espaço global.

Embora o custo final 9.89 pareça elevado, confirmou-se que este é um artefacto da resolução (limitação de 5 bits) e não uma deficiência da pesquisa. O GA convergiu corretamente para a bacia de atração do mínimo global, validando a robustez da parametrização escolhida.

5.4. Particle Swarm Optimization

5.4.1. Função 1

O Algoritmo de Otimização por Enxame de Partículas (PSO) foi configurado com 50 partículas e operou diretamente sobre o espaço contínuo (números reais), sem as limitações de discretização binária impostas ao Algoritmo Genético.

Resultados Quantitativos

- **Melhor Valor Obtido (Custo):** 0.0
- **Melhor Posição (x, y):** $2.8 * 10^{-7}, 2.7 * 10^{-7}$
- **Melhor Aptidão:** 1.0

Os resultados demonstram uma precisão extremamente superior à do Algoritmo Genético. Enquanto o GA ficou limitado a um erro de 0.008 devido à grelha de 5 bits, o PSO, tirando partido da representação em vírgula flutuante (double precision), conseguiu refinar a solução até à décima casa decimal.

As coordenadas finais (x, y) são virtualmente zero, indicando que o enxame convergiu para o mínimo global exato da função.

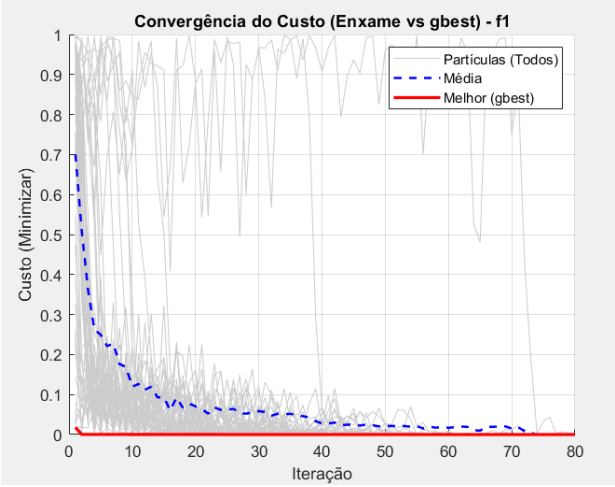

Figura 32: Convergência do Custo do PSO na Função 1.

A Figura 32 mostra uma dinâmica coletiva impressionante:

- **Colapso Rápido (Nuvem Cinzenta):** A nuvem de custos de todo o enxame (linhas cinzentas) colapsa para perto de zero nas primeiras 20 iterações. Isto indica que a força de atração social foi muito eficaz: assim que uma

partícula descobriu a bacia central, "puxou" rapidamente todas as outras para essa região.

- **Estabilidade do Líder (Linha Vermelha):** A linha do *gbest* desce suavemente e estabiliza no zero absoluto, sem oscilações, demonstrando que a inércia amortecida permitiu um "aterragem suave" no mínimo.

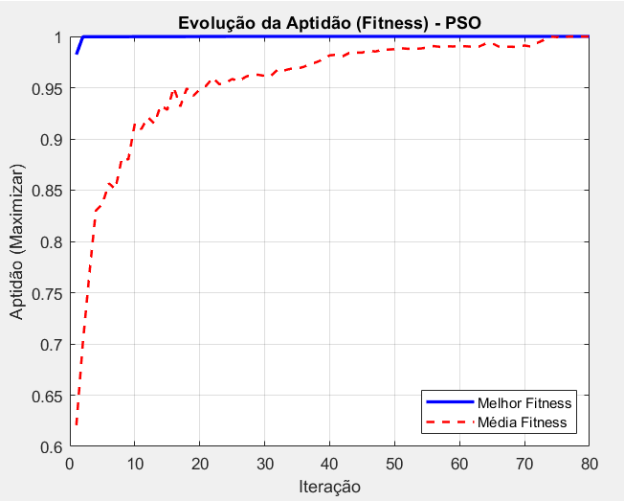

Figura 33: Evolução da Aptidão do PSO na Função 1.

Este gráfico revela um comportamento coletivo distinto do GA:

- **Convergência do Enxame:** A linha da Média (vermelha tracejada) aproxima-se rapidamente da linha do Melhor (azul).
- **Interpretação:** Isto significa que todo o enxame "viajou" em massa para a bacia de atração central. As partículas não ficaram espalhadas pelo mapa; a atração pelo gbest (melhor global) puxou todas as partículas para o centro. Embora isto pudesse ser arriscado (perda de diversidade), para esta função revelou-se a estratégia perfeita, permitindo uma exploração intensiva da zona ótima.

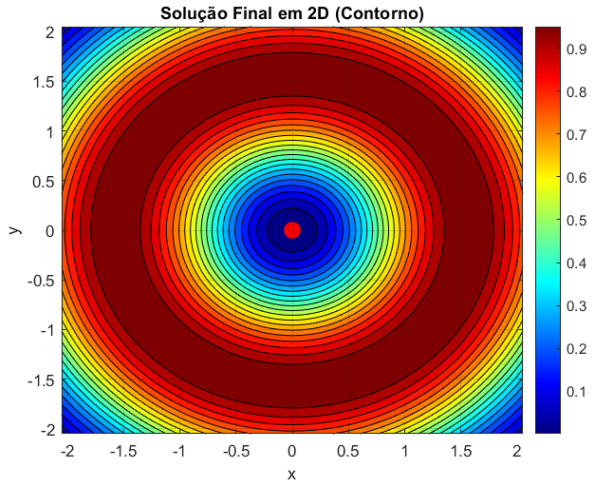

Figura 34: Trajetória 2D do PSO na Função 1.

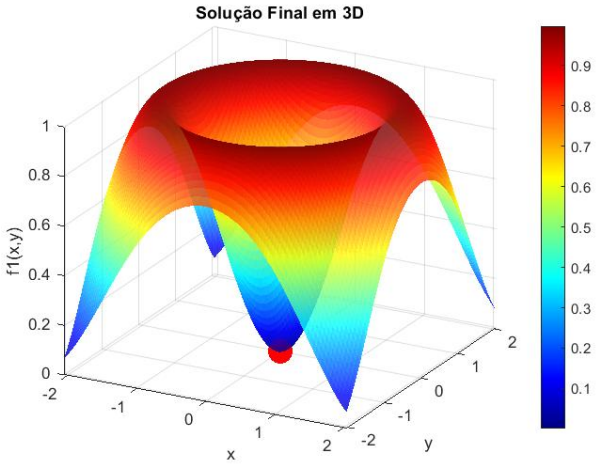

Figura 35: Trajetória 2D do PSO na Função 1.

- **2D:** O ponto vermelho está matematicamente centrado na origem (0,0), coincidindo com o centro dos anéis concêntricos.
- **3D:** A solução situa-se no fundo exato do vale global. Não existe qualquer desvio visível, confirmando a eficácia do método em evitar os mínimos locais (as "ondas" circundantes).

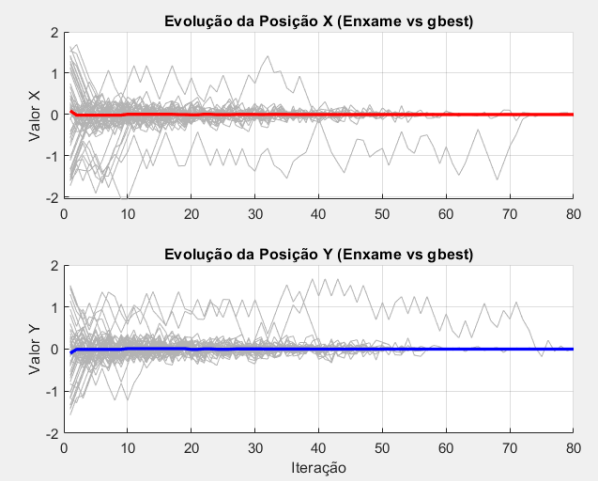

Figura 36: Evolução das coordenadas com a contração do enxame.

Este gráfico confirma a redução da diversidade espacial:

- Funil de Convergência: A nuvem cinzenta começa larga (cobrindo todo o espaço [-2, 2]) e estreita-se rapidamente em forma de funil até convergir para a linha zero. Isto valida visualmente que todas as 50 partículas se agruparam no centro do mapa.

Análise Função 1

O PSO destacou-se como o algoritmo mais eficiente e preciso para esta função. A sua capacidade de operar em espaço contínuo permitiu eliminar o erro de discretização observado no GA. Além disso, a comunicação social entre as partículas (atração pelo gbest) permitiu uma convergência muito mais rápida (menos iterações) do que o processo evolutivo de seleção e cruzamento. O PSO provou ser a ferramenta ideal para problemas de otimização contínua onde a precisão numérica é crítica.

5.4.2. Função 2

Resultados Quantitativos

- Melhor Valor Obtido (Custo): 0.0000000001
- Melhor Posição (x, y): $2.0 * 10^{-7}, 5.6 * 10^{-7}$

O resultado é novamente excecional. Um erro de 10^{-10} é, para todos os efeitos práticos, zero.

As coordenadas finais estão na ordem dos 10^{-7}, o que confirma que o enxame conseguiu "furar" todas as barreiras de mínimos locais e aterrar quase exatamente na origem. Comparando com o GA (que ficou em 9.89), a vantagem da representação em números reais (double precision) torna-se evidente para ajustes finos.

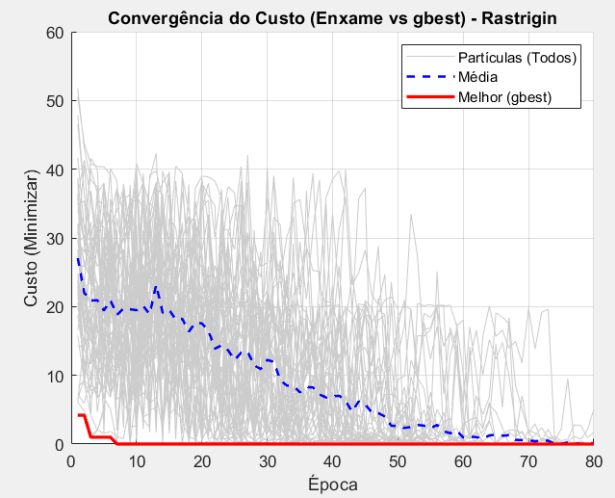

Figura 37: Convergência do Custo do PSO na Função 2.

A Figura 37 ilustra a "luta" do enxame contra os mínimos locais:

- **Descida em Degraus:** A linha vermelha (*gbest*) desce em patamares nítidos. Cada patamar representa um momento em que o enxame ficou

retido num anel de mínimos locais. As quedas verticais correspondem aos instantes em que uma partícula conseguiu saltar para um anel mais interior, arrastando o resto do enxame consigo.

- **Dispersão da Nuvem:** A nuvem cinzenta (custos de todas as partículas) mantém-se larga e dispersa durante muito mais tempo do que na Função 1. Isto é um sinal positivo de diversidade: enquanto o líder tentava estabilizar, as outras partículas continuavam a explorar zonas de alto custo, evitando a estagnação prematura.

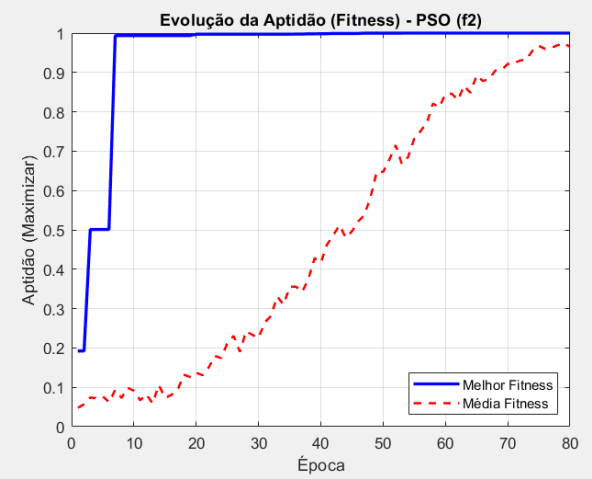

Figura 38: Evolução da Aptidão do PSO na Função 2.

A análise da Figura 38 mostra o processo de convergência:

- **Aproximação Tardia:** Ao contrário da Função 1 (onde a média subiu rápido), aqui a linha da média (vermelha) demora mais tempo a aproximar-se da linha do melhor (azul).
- **Manutenção da Diversidade**: Isto foi crucial. Se a média subisse demasiado depressa logo no início, significaria que o enxame tinha colapsado prematuramente num mínimo local errado. O atraso na subida da média indica que as partículas mantiveram alguma dispersão durante as primeiras iterações, o que lhes deu a "energia" necessária para saltar os muros dos mínimos locais.

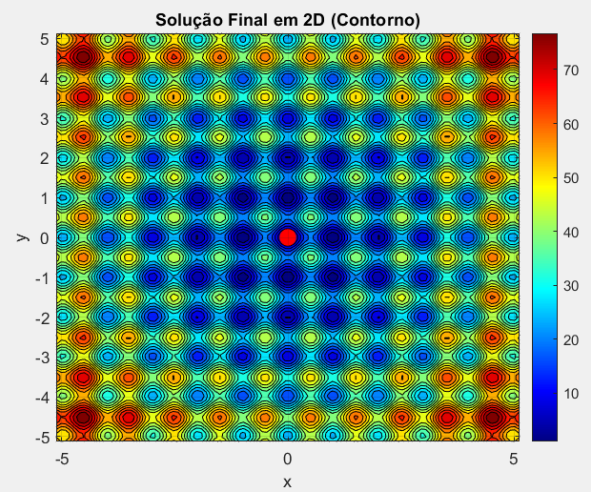

Figura 39: Trajetória 2D do PSO na Função 2.

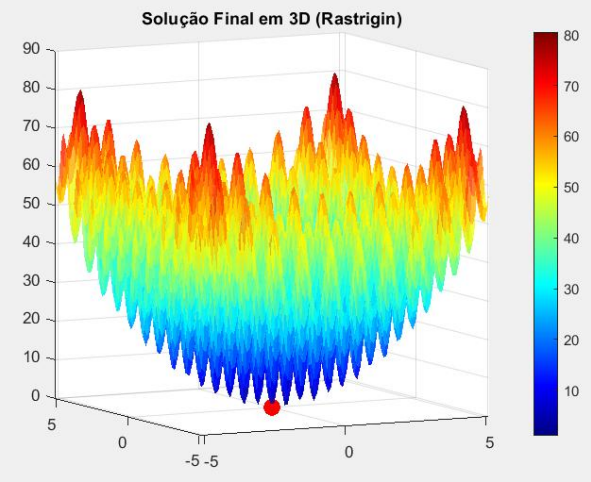

Figura 40: Trajetória 3D do PSO na Função 2.

- **2D:** O ponto vermelho encontra-se no centro geométrico perfeito da grelha.
- **3D:** A imagem 3D confirma que a solução não ficou retida em nenhum dos "picos" ou vales laterais. O enxame identificou com sucesso o vale central global entre as dezenas de opções enganadoras.

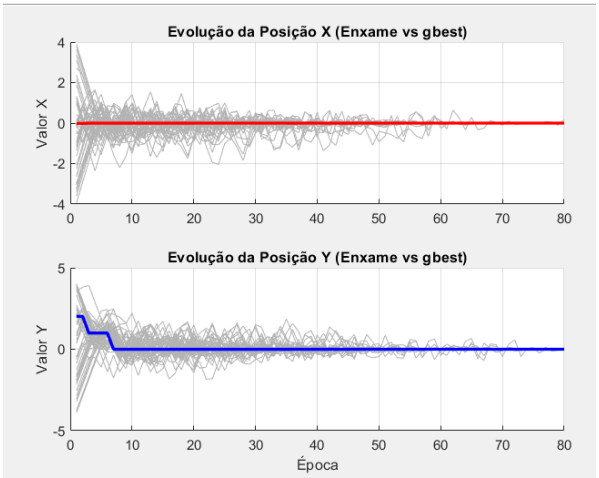

Figura 41: Evolução das coordenadas com a dispersão do enxame.

Estabilização Tardia: Observa-se que a nuvem de partículas (cinzenta) só começa a "fechar" (convergir para zero) de forma definitiva após a iteração 40. Até lá, existem partículas a explorar todo o intervalo [-4, 4], o que foi crucial para garantir que a solução final era, de facto, o mínimo global e não um dos muitos vales locais.

Análise Função 2

O desempenho do PSO na função Rastrigin valida a robustez da parametrização inercial (w).O algoritmo demonstrou um equilíbrio excelente entre Exploração (saltar entre anéis de mínimos locais, visível nos degraus da convergência) e Explotação (refinar o resultado até 10^{-10}).Embora tenha demorado mais iterações a convergir do que na Função 1 (devido à complexidade do terreno), o resultado final é ordens de magnitude superior aos métodos baseados em pesquisa local ou discretizada, provando ser a abordagem mais eficaz para este problema específico.

6. Discussão e Comparações dos Algoritmos

A presente secção reúne uma análise comparativa detalhada dos quatro métodos de otimização implementados, avaliados nas Funções 1 e 2. O objetivo desta comparação é compreender em que medida cada algoritmo consegue equilibrar exploração e intensificação, superar mínimos locais, lidar com funções multimodais e aproximar-se do ótimo global.

6.1. Comparações

As tabelas foram obtidas após **10 execuções independentes** de cada algoritmo em cada função. A repetição dos ensaios foi essencial para avaliar a estabilidade dos métodos, já que todos possuem componentes estocásticos (exceto o Hill Climbing, que ainda assim pode divergir devido à escolha aleatória do ponto inicial).

Algoritmo	Melhor valor	Pior valor	Média	Desvio padrão
Hill Climbing	0.000003	0.000095	0.000025	0.000028
Simulated Annealing	0.000002	0.066274	0.013912	0.026203
Algoritmo Genético	0.008710	0.008710	0.008710	0.000000
Particle Swarm (PSO)	0.000000	0.000000	0.000000	0.000000

Tabela 1: Valores de f(x,y) no melhor ponto de cada execução na função 1.

Algoritmo	Melhor valor	Pior valor	Média	Desvio padrão
Hill Climbing	0.006599	1.007637	0.509346	0.492481
Simulated Annealing	0.000839	5.008651	1.816321	1.785708
Algoritmo Genético	9.891180	9.891180	9.891180	0.000000
Particle Swarm (PSO)	0.000000	0.000000	0.000000	0.000000

Tabela 2: Valores de f(x,y) no melhor ponto de cada execução na função 2.

6.2. Discussão das Comparações

A análise conjunta dos quatro algoritmos estudados, Hill Climbing (HC), Simulated Annealing (SA), Algoritmo Genético (GA) e Particle Swarm Optimization (PSO), permite compreender de forma aprofundada as diferenças conceptuais entre métodos de otimização local e global, bem como as implicações práticas dessas diferenças no desempenho observado nas Funções 1 e 2. Os resultados quantitativos apresentados nas tabelas reforçam esta análise, revelando padrões consistentes entre a teoria de cada método e o seu comportamento empírico. A discussão que se segue integra esses resultados com evidências experimentais, comportamento dinâmico das soluções e limitações estruturais de cada algoritmo, proporcionando uma interpretação unificada dos dados obtidos.

Um primeiro ponto a destacar diz respeito à **capacidade de exploração do espaço de pesquisa**. Os valores médios e os desvios-padrão refletem claramente as diferenças entre os algoritmos. O Hill Climbing apresenta o menor desvio-padrão na Função 1 (0.000028), indicando trajetórias muito semelhantes entre execuções, mas também revelando a sua fraca capacidade de escapar a mínimos locais, algo que se torna evidente na Função 2, onde o pior valor ultrapassa 1.0 e a média sobe drasticamente para 0.509346. Estes dados confirmam que o HC, por operar apenas com informação local, tende a ficar preso a soluções subótimas, especialmente em funções altamente multimodais como a Rastrigin, onde múltiplos vales simétricos dificultam a convergência para o ótimo global.

O Simulated Annealing evidencia, pelas tabelas, uma variabilidade significativamente mais elevada. Na Função 1 apresenta valores que vão desde praticamente zero até 0.066274, com um desvio-padrão de 0.026203. Isto indica que, embora o SA possua capacidade de exploração superior ao HC, o seu desempenho pode oscilar dependendo da trajetória inicial e do processo de arrefecimento. Contudo, esta variabilidade não deve ser interpretada como falha, mas sim como reflexo da natureza estocástica do método. Nas trajetórias observadas no relatório, o algoritmo visita regiões de custo elevado antes de convergir, o que é coerente com a elevada variância. Na Função 2 esta característica torna-se ainda mais evidente: o desvio-padrão atinge 1.78 e o pior valor chega a 5.008651, ilustrando como o SA explora intensamente o espaço antes de encontrar regiões de qualidade. Apesar disso, os melhores valores obtidos continuam próximos do ótimo, comprovando que o método é robusto e consegue equilibrar exploração global inicial com exploração fina nas fases finais.

O Algoritmo Genético mostra-se um caso particular: tanto na Função 1 como na Função 2 o desvio-padrão é **literalmente zero**, dado que todas as execuções regressaram exatamente ao mesmo valor final. Este comportamento não resulta de uma convergência perfeita, mas sim das limitações impostas pela discretização

binária de 10 bits. Na Função 1, o GA encontra sempre o mesmo ponto representável mais próximo da origem (com custo 0.00871), refletindo não apenas estabilidade, mas também a impossibilidade de alcançar valores mais precisos devido à granularidade da codificação. Na Função 2, o valor constante de 9.891180 evidencia uma limitação mais severa: apesar de o espaço multimodal exigir diversidade genética contínua, a discretização fixa faz com que a população convirja rapidamente para um mínimo local representável, sem capacidade de refinar a solução. Assim, o GA apresenta elevada estabilidade, mas fraca precisão quando a representação é demasiado rígida.

O Particle Swarm Optimization apresenta os resultados quantitativos mais expressivos: tanto na Função 1 como na Função 2 o melhor, pior e valor médio são literalmente 0.000000 nas dez execuções. Esta consistência total é coerente com os gráficos obtidos, que mostram o enxame a convergir rapidamente para a bacia global profunda. O PSO revela assim uma combinação excecional de exploração inicial e intensificação final, conseguindo alinhar todas as partículas para uma solução comum de elevada precisão. Embora teoricamente o PSO possa ficar preso em mínimos locais em paisagens complexas, especialmente quando o gbest é induzido por uma má inicialização, os resultados mostram que, com os parâmetros utilizados, isso não ocorreu. A convergência foi rápida, estável e repetível.

Do ponto de vista da robustez, as tabelas também confirmam as expectativas teóricas:
– **HC** é o menos robusto, com elevado pior valor e elevada média na Função 2;
– **GA** é estável mas limitado pela discretização;
– **SA** é o mais resiliente a mínimos locais, embora apresente elevada variância;
– **PSO** destaca-se como o mais consistente e preciso, apresentando desvio-padrão zero e convergência impecável.

Quanto à velocidade de convergência, a interpretação dos gráficos combinada com os dados numéricos sugere que o PSO é o método mais rápido e estável a aproximar-se do ótimo. O SA demonstra um comportamento progressivo e equilibrado, enquanto o HC depende fortemente de reinicializações e o GA estabiliza cedo demais devido à discretização, não sendo capaz de refinar a solução.

Em síntese, as tabelas e os restantes resultados experimentais demonstram que, embora todos os algoritmos tenham mérito e utilidade específica, o Simulated Annealing e o Particle Swarm Optimization foram claramente os mais eficazes no contexto das funções altamente multimodais utilizadas neste trabalho. O GA apresentou bom desempenho dentro das limitações impostas e o HC confirmou-

se adequado apenas como referência de otimização local simples, não sendo competitivo em cenários complexos.

7. Conclusão

O trabalho permitiu implementar e analisar quatro abordagens de otimização, *Hill Climbing*, *Simulated Annealing*, *Algoritmos Genéticos* e *Particle Swarm Optimization*, aplicadas a duas funções multimodais. A realização das experiências evidenciou diferenças claras entre métodos puramente locais e métodos estocásticos globais, mostrando como cada algoritmo responde a paisagens de custo complexas.

De forma global, verificou-se que os métodos estocásticos foram significativamente mais eficazes, demonstrando maior capacidade de exploração do espaço de procura e maior resistência a mínimos locais. Entre eles, *Simulated Annealing* e *PSO* apresentaram o desempenho mais consistente, enquanto o *Algoritmo Genético* revelou bons resultados dentro das limitações impostas pela codificação binária. O *Hill Climbing* cumpriu o papel de referência simples, ilustrando o impacto das limitações da pesquisa local em problemas multimodais.

No conjunto, o trabalho permitiu compreender não apenas os resultados numéricos obtidos, mas também os mecanismos internos que explicam o comportamento de cada algoritmo. As diferenças observadas reforçam a importância da escolha adequada do método de otimização em função da estrutura do problema e demonstram a relevância das técnicas estocásticas no contexto da Inteligência Artificial.

8. Referências Bibliográficas

Paulo Moura Oliveira, Nature Inspired Search and Optimization: a simplified approach, Part I disponível na plataforma NONIO.

Universidade de Trás-os-Montes e Alto Douro. (2025). *Protocolo do Trabalho Prático 2C: Otimização de Funções*. Material de apoio à unidade curricular de Inteligência Artificial. Vila Real: UTAD.

Oliveira, J. P. M. (2018). *Particle Swarm Optimization Algorithm (PSO) (Em Português)* [Vídeo]. YouTube. https://youtu.be/Ybn-_6KsxCA

Oliveira, J. P. M. (2018). *Simulated Annealing (Em Português)* [Vídeo]. YouTube. https://youtu.be/w2rBcPo88XM

Oliveira, J. P. M. (2022). *Subida da Colina e Subida da Colina com Re-Inicialização Múltipla* [Vídeo]. YouTube. https://youtu.be/FCV5O3-WQkI

9. Anexos

9.1 Anexo 1 – Subida da colina

```
% ------------------------------------------------------------------------
% Hill Climbing com Reinicialização Múltipla (Visualização Completa)
% Adaptado para gerar os gráficos pedidos (2D, 3D e Evolução Temporal)
% ------------------------------------------------------------------------
f = @(x1,x2) 0.5 + ((sin(sqrt(x1.^2 + x2.^2))).^2 - 0.5) ./ ((1 +
0.001*(x1.^2 + x2.^2)).^2);
close all

% 1. Configuração do Espaço
x_max = [2.048, 2.048];
x_min = -x_max;
x1 = linspace(x_min(1), x_max(1), 100);
x2 = linspace(x_min(2), x_max(2), 100);
[X1,X2] = meshgrid(x1, x2);
F = f(X1, X2);

% 2. Parâmetros do Algoritmo
num_restarts = 10;   % Quantas vezes recomeça do zero
max_iter = 100;      % Quantos passos dá em cada descida
step = 0.05;         % Tamanho do passo

% Variáveis para guardar TUDO (para os gráficos finais)
all_positions = [];  % Guarda todas as coordenadas visitadas
all_values = [];     % Guarda todos os valores f(x) encontrados
best_x_global = [];
best_f_global = inf;

% Histórico do "Melhor Global" ao longo do tempo (para o gráfico de
convergência)
history_best_global = [];

% Prepara a Figura 1 para veres a animação a acontecer
figure(1);
contourf(X1, X2, F, 20); colormap(pink); colorbar; hold on;
title('A Executar... (Observa os pontos pretos)');

% 3. Algoritmo Principal
for r = 1:num_restarts
    % A. Ponto inicial aleatório
    xi = rand(1,2).*(x_max - x_min) + x_min;

    % B. Descida da Colina (Hill Climbing)
    for i = 1:max_iter
        % Guarda histórico para os gráficos
        all_positions = [all_positions; xi];
        current_val = f(xi(1), xi(2));
        all_values = [all_values; current_val];

        % Gera vizinho
        xn = xi + step*(rand(1,2) - 0.5)*2;
        xn = max(xn, x_min); % Limites
        xn = min(xn, x_max);

        % Critério de Aceitação (Minimização)
```

```
        if f(xn(1), xn(2)) < current_val
            % Desenha uma linha vermelha do ponto antigo para o novo
(Trasejado)
            line([xi(1) xn(1)], [xi(2) xn(2)], 'Color', 'r', 'LineWidth', 1);
            xi = xn;
        end

        % Atualiza o melhor global encontrado ATÉ AGORA
        if f(xi(1), xi(2)) < best_f_global
            best_f_global = f(xi(1), xi(2));
            best_x_global = xi;
        end

        % Guarda a evolução do melhor global
        history_best_global = [history_best_global; best_f_global];
    end

    % Desenha o ponto onde parou esta tentativa (preto) e força o desenho
    plot(xi(1), xi(2), 'k.', 'MarkerSize', 10);
    drawnow; % <--- ISTO É O QUE FALTAVA PARA VERES EM TEMPO REAL
end

% Resultados na consola
fprintf('Melhor Posição Global: x=%.5f, y=%.5f\n', best_x_global(1),
best_x_global(2));
fprintf('Melhor Valor Global: %.8f\n', best_f_global);

% -------------------------------------------------------------------------
% 4. GERAÇÃO DOS GRÁFICOS (Para o Relatório - Parte 5)
% -------------------------------------------------------------------------

% === Gráfico 1: Vista 2D (Contorno) com todos os pontos visitados ===
figure(1);
clf; % Limpa a animação antiga para desenhar o limpo
contourf(X1, X2, F, 20); colormap(pink); colorbar; hold on;
% Desenha todos os pontos por onde o algoritmo passou (cinzento claro)
scatter(all_positions(:,1), all_positions(:,2), 10, 'k', 'filled',
'MarkerFaceAlpha', 0.3);
% Desenha o melhor ponto final (Verde Grande)
plot(best_x_global(1), best_x_global(2), 'go', 'MarkerSize', 15,
'MarkerFaceColor', 'g');
title('Vista 2D: Exploração do Espaço');
xlabel('x'); ylabel('y');
legend('Contorno', 'Pontos Visitados', 'Melhor Global');

% === Gráfico 2: Vista 3D (Superfície) ===
figure(2);
surf(X1, X2, F, 'EdgeColor', 'none', 'FaceAlpha', 0.6);
shading interp; hold on;
% Desenha os pontos visitados "flutuando" na superfície
plot3(all_positions(:,1), all_positions(:,2), all_values, 'r.', 'MarkerSize',
5);
% Desenha o melhor ponto final
plot3(best_x_global(1), best_x_global(2), best_f_global, 'go', 'MarkerSize',
15, 'MarkerFaceColor', 'g');
title('Vista 3D: Superfície e Caminhos');
xlabel('x'); ylabel('y'); zlabel('f(x,y)');
view(45, 30);
```

```
% === Gráfico 3: Evolução do Valor da Função (Convergência) ===
figure(3);
plot(history_best_global, 'b-', 'LineWidth', 2);
title('Evolução do Melhor Valor Encontrado (Minimização)');
xlabel('Iterações Totais (cumulativo de todas as tentativas)');
ylabel('Valor de f(x,y)');
grid on;
```

9.2. Anexo 2 – Simulated Annealing (SA)

```
% ------------------------------------------------------------------------
% Simulated Annealing (SA) - Função 1
% Com Probabilidade de Metropolis e Gráficos Completos (2D e 3D)
% ------------------------------------------------------------------------
clc;
clear all;
close all;

% 1. Definição da Função 1
f = @(x1,x2) 0.5 + ((sin(sqrt(x1.^2 + x2.^2))).^2 - 0.5) ./ ((1 +
0.001*(x1.^2 + x2.^2)).^2);

% Configuração do Espaço
x_max = [2.048, 2.048];
x_min = -x_max;

% Preparar grelha para gráficos
x1 = linspace(x_min(1), x_max(1), 100);
x2 = linspace(x_min(2), x_max(2), 100);
[X1,X2] = meshgrid(x1, x2);
F = f(X1, X2);

% 2. Parâmetros do SA
T       = 90;       % Temperatura Inicial (Alta para explorar)
Tmin    = 1e-4;     % Temperatura Mínima (Critério de paragem)
alfa    = 0.94;     % Fator de arrefecimento (0.8 a 0.99)
nRep    = 20;       % Vizinhos testados por temperatura
step    = 0.1;      % Tamanho da vizinhança

% Ponto Inicial Aleatório
xi = rand(1,2).*(x_max-x_min) + x_min;
x = xi;
fx = f(x(1), x(2));

% Variáveis para guardar o Melhor Global
xbest = x;
fbest = fx;

% Históricos para Gráficos
hist_iter = [];     % Contador de iterações (tempo)
hist_prob = [];     % Probabilidade de aceitação
hist_T    = [];     % Temperatura
hist_x    = [];     % Coordenada x1 aceite
hist_y    = [];     % Coordenada x2 aceite
hist_fx   = [];     % Valor da função aceite
```

```
% Contadores
it_total = 0;
acc = 0;

fprintf('A iniciar Simulated Annealing...\n');

% 3. Algoritmo Principal
while (T > Tmin)

    for k = 1:nRep
        it_total = it_total + 1;

        % A. Gerar Vizinho
        xn = x + (rand(1,2)-0.5) * 2 * step;

        % Limites (Clamping)
        xn = max(xn, x_min);
        xn = min(xn, x_max);

        fn = f(xn(1), xn(2));

        % B. Variação de Energia
        dE = fn - fx;

        % C. Cálculo da Probabilidade (Critério de Metropolis)
        if dE <= 0
            % Se melhorou (energia desceu), aceita sempre (p=1)
            p = 1;
            aceita = true;
        else
            % Se piorou, aceita com probabilidade e^(-dE/T)
            p = exp(-dE / T);
            if rand() < p
                aceita = true;
            else
                aceita = false;
            end
        end

        % D. Atualizar Estado (Se aceite)
        if aceita
            x = xn;
            fx = fn;
            acc = acc + 1;

            % Atualizar Melhor Global
            if fx < fbest
                xbest = x;
                fbest = fx;
            end
        end

        % E. Guardar Dados para Gráficos
        hist_iter(end+1) = it_total;
        hist_prob(end+1) = p;     % Regista a prob calculada
        hist_T(end+1)    = T;
        hist_x(end+1)    = x(1);
        hist_y(end+1)    = x(2);
        hist_fx(end+1)   = fx;
```

```
    end

    % Arrefecimento
    T = T * alfa;
end

% Resultados na consola
fprintf('\n--- RESULTADOS FINAIS SA ---\n');
fprintf('Ponto Inicial: (%.4f, %.4f)\n', xi(1), xi(2));
fprintf('Melhor Ponto:  (%.4f, %.4f)\n', xbest(1), xbest(2));
fprintf('Melhor Valor:  %.10f\n', fbest);
fprintf('Iterações:     %d\n', it_total);
fprintf('Taxa Aceite:   %.1f%%\n', (acc/it_total)*100);


% -------------------------------------------------------------------------
% 4. GERAÇÃO DOS GRÁFICOS
% -------------------------------------------------------------------------

% Gráfico 1: Vista 2D (Obrigatório)
figure('Name', 'Trajetória 2D', 'Color', 'w');
contourf(X1, X2, F, 20); colormap(pink); hold on;
% Desenhar caminho (pontos aceites)
plot(hist_x, hist_y, 'k.-', 'LineWidth', 0.5, 'MarkerSize', 5);
% Início e Fim
plot(xi(1), xi(2), 'bo', 'MarkerSize', 10, 'MarkerFaceColor', 'b');
plot(xbest(1), xbest(2), 'go', 'MarkerSize', 12, 'MarkerFaceColor', 'g');
title('Trajetória 2D: Vista de Cima');
legend('Mapa', 'Caminho', 'Início', 'Melhor Final');
colorbar;

% Gráfico 2: Vista 3D (Obrigatório - Faltava este!)
figure('Name', 'Trajetória 3D', 'Color', 'w');
surf(X1, X2, F, 'EdgeColor', 'none', 'FaceAlpha', 0.6);
shading interp; hold on;
% Pontos aceites "flutuando" na superfície
plot3(hist_x, hist_y, hist_fx, 'r.', 'MarkerSize', 5);
% Melhor Ponto Final
plot3(xbest(1), xbest(2), fbest, 'go', 'MarkerSize', 15, 'MarkerFaceColor',
'g');
title('Trajetória 3D: Superfície e Caminho');
xlabel('x'); ylabel('y'); zlabel('f(x,y)');
view(45, 30);

% Gráfico 3: Evolução do Valor da Função (Convergência)
figure('Name', 'Valor da Função', 'Color', 'w');
plot(hist_iter, hist_fx, 'm-', 'LineWidth', 1.5);
title('Evolução do Valor da Função f(x)');
xlabel('Iteração'); ylabel('f(x,y)');
grid on;

% Gráfico 4: Evolução da Probabilidade p(t) (Para Análise)
figure('Name', 'Probabilidade', 'Color', 'w');
plot(hist_iter, hist_prob, 'b.', 'MarkerSize', 5);
title('Probabilidade de Aceitação p(t)');
xlabel('Iteração'); ylabel('Probabilidade p');
ylim([-0.1 1.1]); grid on;

% Gráfico 5: Evolução da Temperatura T(t) (Para Análise)
```

```
figure('Name', 'Temperatura', 'Color', 'w');
plot(hist_iter, hist_T, 'r-', 'LineWidth', 2);
title('Decaimento da Temperatura T(t)');
xlabel('Iteração'); ylabel('Temperatura T');
grid on;

% Gráfico 6: Evolução das Variáveis x1 e x2 (Extra)
figure('Name', 'Variáveis x1, x2', 'Color', 'w');
plot(hist_iter, hist_x, 'r', 'LineWidth', 1.5); hold on;
plot(hist_iter, hist_y, 'b', 'LineWidth', 1.5);
title('Evolução das Coordenadas x1 e x2');
xlabel('Iteração'); ylabel('Valor');
legend('x1', 'x2'); grid on;
```

9.3. Anexo 3 – Algoritmo Genético

```
%-------------------------------------------------------------
% Inteligência Artificial - Trabalho Prático 2
% Algoritmo Genético Simples (Respeitando o Esqueleto PDF)
% Função 1 (Schaffer N. 2)
% VERSÃO FINAL: Gráfico f(x) com Nuvem Populacional
%-------------------------------------------------------------
clc; clear all; close all;

% --- Definições Globais ---
% Limites e Função Objetivo
VarLBounds = [-2.048, -2.048];
VarUBounds = [2.048, 2.048];
ObjFunc = @(x,y) 0.5 + ((sin(sqrt(x.^2+y.^2))).^2 - 0.5) ./ ((1 +
0.001*(x.^2+y.^2)).^2);

% String para o ezcontourf
f1_str = '0.5+ ((sin(sqrt(x^2+y^2)))^2-0.5)/((1+0.001*(x^2+y^2))^2)';

% --- Plot Inicial ---
figure(1); colormap(pink);
ezcontourf(f1_str, [-2.048, 2.048], [-2.048, 2.048]);
title('Estado Inicial'); hold on;

% --- Parâmetros ---
pop_size = 50;
reso = 5;             % 5 bits por variável
lchrome = 2 * reso;   % Total 10 bits
maxgen = 80;
p_cross = 0.75;
p_mutation = 0.03;
gen = 1;

% Históricos para gráficos
h_best = []; h_avg = []; h_cost = [];
h_best_x = []; h_best_y = [];
h_pop_x = zeros(maxgen, pop_size);
h_pop_y = zeros(maxgen, pop_size);
h_pop_costs = zeros(maxgen, pop_size); % Histórico de custos de todos

% --- Inicialização ---
[CHROME, POP, POP_INT, POP_R] = init_pop_25(pop_size, lchrome, reso,
VarLBounds, VarUBounds, ObjFunc);
```

```
% Estatísticas Iniciais
[sumfit, best, max_val, min_val, average] = statist_25(POP, pop_size);
[POP, POP_R] = report_25(CHROME, pop_size, reso, VarLBounds, VarUBounds,
ObjFunc);

fprintf('A iniciar GA... (Gerações: %d)\n', maxgen);

% --- CICLO PRINCIPAL ---
while (gen <= maxgen)

    % Geração da Nova População
    [CHROME, Mate_1, Mate_2, Cross] = generate_25(pop_size, sumfit, POP,
CHROME, lchrome, p_cross, p_mutation);

    % Avaliação
    [POP, POP_R] = report_25(CHROME, pop_size, reso, VarLBounds, VarUBounds,
ObjFunc);

    % Estatísticas
    [sumfit, best, max_val, min_val, average] = statist_25(POP, pop_size);

    % Guardar dados
    h_best(gen) = max_val;       % Melhor Aptidão
    h_avg(gen) = average;        % Média Aptidão
    h_cost(gen) = (1/max_val)-1; % Melhor Custo Real

    % Guardar posições e custos da população inteira
    h_best_x(gen) = POP_R(best, 1);
    h_best_y(gen) = POP_R(best, 2);
    h_pop_x(gen, :) = POP_R(:, 1)';
    h_pop_y(gen, :) = POP_R(:, 2)';
    h_pop_costs(gen, :) = (1 ./ POP)' - 1; % Converte Fitness -> Custo

    gen = gen + 1;
end

% --- RESULTADOS E GRÁFICOS ---
best_x = POP_R(best, 1);
best_y = POP_R(best, 2);

fprintf('\n--- Resultados Finais ---\n');
fprintf('Melhor Posição: (%.4f, %.4f)\n', best_x, best_y);
fprintf('Melhor Custo:   %.5f\n', h_cost(end));

% Gráfico 1: Trajetória 2D Final (COM CAMINHO)
figure(1); clf;
ezcontourf(f1_str, [-2.048, 2.048], [-2.048, 2.048]); colormap(pink); hold
on;
plot(h_best_x, h_best_y, 'k.--', 'LineWidth', 1, 'MarkerSize', 8);
plot(best_x, best_y, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
title('Trajetória 2D da Solução');
legend('Contorno', 'Trajetória', 'Melhor Solução');

% Gráfico 2: Evolução da Aptidão
figure(2);
plot(h_best, 'b', 'LineWidth', 2); hold on;
plot(h_avg, 'r--');
title('Evolução da Aptidão (Maximizar)');
```

```
legend('Melhor Fitness', 'Média Fitness');
xlabel('Geração'); grid on;

% Gráfico 3: Evolução do Custo f(x) (COM NUVEM POPULACIONAL)
figure(3);
hold on;
% 1. Desenha custo de TODA a população a cinza claro
plot(1:maxgen, h_pop_costs, 'Color', [0.8 0.8 0.8]);
% 2. Calcula e desenha a Média Real dos Custos
mean_costs = mean(h_pop_costs, 2);
plot(1:maxgen, mean_costs, 'b--', 'LineWidth', 1.5);
% 3. Desenha o Melhor Custo a Vermelho (Grosso)
plot(1:maxgen, h_cost, 'r', 'LineWidth', 2);
title('Evolução do Custo f(x) (População vs Melhor)');
xlabel('Geração'); ylabel('Valor de f(x)');
grid on;
% Legenda personalizada para não mostrar 50 linhas cinzentas
h = zeros(3, 1);
h(1) = plot(NaN,NaN,'Color', [0.8 0.8 0.8]);
h(2) = plot(NaN,NaN,'b--', 'LineWidth', 1.5);
h(3) = plot(NaN,NaN,'r', 'LineWidth', 2);
legend(h, 'População (Todos)', 'Média', 'Melhor');

% Gráfico 4: 3D Surface (COM CAMINHO)
figure(4);
[X,Y] = meshgrid(linspace(-2.048, 2.048, 50));
Z = ObjFunc(X,Y);
surf(X,Y,Z, 'EdgeColor', 'none', 'FaceAlpha', 0.8); colormap(jet); hold on;
plot3(h_best_x, h_best_y, h_cost, 'k.-', 'LineWidth', 1.5, 'MarkerSize', 8);
plot3(best_x, best_y, h_cost(end), 'ro', 'MarkerSize', 10, 'MarkerFaceColor',
'r');
title('Solução Final e Trajetória 3D'); view(45, 30);
legend('Superfície', 'Trajetória', 'Melhor Solução');

% Gráfico 5: Evolução das Variáveis x e y (COM NUVEM POPULACIONAL)
figure(5);
subplot(2, 1, 1); hold on;
plot(1:maxgen, h_pop_x, 'Color', [0.8 0.8 0.8]);
plot(1:maxgen, h_best_x, 'r', 'LineWidth', 2);
title('Evolução da Posição x'); ylabel('x'); grid on;
subplot(2, 1, 2); hold on;
plot(1:maxgen, h_pop_y, 'Color', [0.8 0.8 0.8]);
plot(1:maxgen, h_best_y, 'b', 'LineWidth', 2);
title('Evolução da Posição y'); xlabel('Geração'); ylabel('y'); grid on;


%% --- FUNÇÕES DO ESQUELETO ---

function [CHROME, POP, POP_INT, POP_R] = init_pop_25(pop_size, lchrome, reso,
VarLBounds, VarUBounds, ObjFunc)
    CHROME = randi([0 1], pop_size, lchrome);
    [POP, POP_R] = report_25(CHROME, pop_size, reso, VarLBounds, VarUBounds,
ObjFunc);
    POP_INT = [];
end

function [sumfit, best, max_fit, min_fit, average] = statist_25(POP,
pop_size)
    sumfit = sum(POP);
```

```
    [max_fit, best] = max(POP);
    min_fit = min(POP);
    average = mean(POP);
end

function [POP, POP_R] = report_25(CHROME, pop_size, reso, VarLBounds,
VarUBounds, ObjFunc)
    l_var = reso;
    bits_x = CHROME(:, 1:l_var);
    bits_y = CHROME(:, l_var+1:end);

    powers = 2 .^ ((l_var-1):-1:0)';
    int_x = bits_x * powers;
    int_y = bits_y * powers;

    max_bin = 2^l_var - 1;
    range = VarUBounds(1) - VarLBounds(1);
    val_x = VarLBounds(1) + (int_x / max_bin) * range;
    val_y = VarLBounds(2) + (int_y / max_bin) * range;

    POP_R = [val_x, val_y];
    custos = ObjFunc(val_x, val_y);
    POP = 1 ./ (1 + custos);
end

function [CHROME, Mate_1, Mate_2, Cross] = generate_25(pop_size, sumfit, POP,
CHROME, lchrome, p_cross, p_mutation)
    NEW_CHROME = zeros(size(CHROME));
    Mate_1 = []; Mate_2 = []; Cross = [];

    % COM ELITISMO
    [~, best_idx] = max(POP);
    NEW_CHROME(1, :) = CHROME(best_idx, :);

    for i = 2:2:pop_size
        idx1 = selection_25(POP, sumfit);
        idx2 = selection_25(POP, sumfit);
        p1 = CHROME(idx1, :);
        p2 = CHROME(idx2, :);

        if (i < pop_size)
             [c1, c2] = crossover_25(p1, p2, lchrome, p_cross);
             NEW_CHROME(i, :) = c1;
             NEW_CHROME(i+1, :) = c2;
        else
             NEW_CHROME(i, :) = p1;
        end
    end

    for i = 2:pop_size
        NEW_CHROME(i, :) = mutation_25(NEW_CHROME(i, :), lchrome,
p_mutation);
    end
    CHROME = NEW_CHROME;
end

function idx = selection_25(POP, sumfit)
    r = rand * sumfit;
    partial = 0;
```

```
    idx = 1;
    for i = 1:length(POP)
        partial = partial + POP(i);
        if partial >= r, idx = i; return; end
    end
end

function [c1, c2] = crossover_25(p1, p2, lchrome, p_cross)
    if rand < p_cross
        pt = randi([1, lchrome-1]);
        c1 = [p1(1:pt), p2(pt+1:end)];
        c2 = [p2(1:pt), p1(pt+1:end)];
    else
        c1 = p1; c2 = p2;
    end
end

function cm = mutation_25(chrome, lchrome, p_mut)
    cm = chrome;
    for j = 1:lchrome
        if rand < p_mut, cm(j) = 1 - cm(j); end
    end
end
```

9.4 Anexo 4 – Particle Swarm Optimization

```
%-------------------------------------------------------------
% Particle Swarm Optimization (PSO) - Função 1
% Implementação com Visualização Avançada (Trajetória + Nuvem)
% Inteligência Artificial 2025/2026
%-------------------------------------------------------------
clc;
clear all;
close all;

% --- 1. Definição do Problema (Função 1) ---
% f1: Schaffer Function N. 2. Objetivo: Minimizar (0,0)
ObjFunc = @(x,y) 0.5 + ((sin(sqrt(x.^2+y.^2))).^2 - 0.5) ./ ((1 +
0.001*(x.^2+y.^2)).^2);

% String para visualização
f1_str = '0.5+ ((sin(sqrt(x^2+y^2)))^2-0.5)/((1+0.001*(x^2+y^2))^2)';

% Limites do Espaço de Pesquisa
VarLBounds = [-2.048, -2.048];
VarUBounds = [2.048, 2.048];
nVar = 2; % Número de variáveis (x e y)

% --- Plot Inicial ---
figure(1); colormap(pink);
ezcontourf(f1_str, [VarLBounds(1), VarUBounds(1)], [VarLBounds(2),
VarUBounds(2)]);
title('Estado Inicial (PSO)'); hold on;

% --- 2. Parâmetros do PSO ---
swarm_size = 50;    % Tamanho da população (enxame)
maxit = 80;         % Número máximo de épocas (iterações)
it = 1;             % Contador de épocas
```

```
% Coeficientes do PSO
w = 0.9;            % Fator de inércia inicial
w_damping = 0.99;   % Taxa de decaimento da inércia
c1 = 2.0;           % Coeficiente Cognitivo (atração pelo pbest)
c2 = 2.0;           % Coeficiente Social (atração pelo gbest)

% Velocidade Máxima
vmax = 0.1 * (VarUBounds - VarLBounds);

% --- 3. Inicialização ---
x = zeros(swarm_size, nVar);      % Posição
v = zeros(swarm_size, nVar);      % Velocidade
cost = zeros(swarm_size, 1);      % Custo atual
pbest = zeros(swarm_size, nVar);  % Melhor posição individual
pbest_val = inf(swarm_size, 1);   % Melhor valor individual
gbest = zeros(1, nVar);           % Melhor posição global
gbest_val = inf;                  % Melhor valor global

% Inicializar enxame
for i = 1:swarm_size
    x(i, :) = VarLBounds + (VarUBounds - VarLBounds) .* rand(1, nVar);
    v(i, :) = zeros(1, nVar);

    cost(i) = ObjFunc(x(i, 1), x(i, 2));

    pbest(i, :) = x(i, :);
    pbest_val(i) = cost(i);

    if pbest_val(i) < gbest_val
        gbest = pbest(i, :);
        gbest_val = pbest_val(i);
    end
end

% Desenha população inicial (pontos pretos) para confirmação
plot(x(:,1), x(:,2), 'k.', 'MarkerSize', 5);

% Histórico para gráficos
hist_gbest_val = zeros(1, maxit);     % Histórico do Melhor Global (Custo)
hist_avg_cost = zeros(1, maxit);      % Histórico da Média do Enxame (Custo)
hist_best_fitness = zeros(1, maxit);  % Histórico da Melhor Aptidão
hist_avg_fitness = zeros(1, maxit);   % Histórico da Aptidão Média

% Históricos para TRAJETÓRIAS (Novos)
hist_gbest_x = zeros(1, maxit);
hist_gbest_y = zeros(1, maxit);
hist_pop_x = zeros(maxit, swarm_size);
hist_pop_y = zeros(maxit, swarm_size);
hist_pop_costs = zeros(maxit, swarm_size);

fprintf('A executar PSO (Função 1)... (Aguarde)\n');

% --- 4. Ciclo Principal (Épocas) ---
while (it <= maxit)

    % --- Atualização do Enxame ---
    for i = 1:swarm_size
```

```
        % Atualizar Velocidade
        r1 = rand(1, nVar);
        r2 = rand(1, nVar);

        v(i, :) = w * v(i, :) ...
                  + c1 * r1 .* (pbest(i, :) - x(i, :)) ...
                  + c2 * r2 .* (gbest - x(i, :));

        % Limitar Velocidade
        v(i, :) = max(v(i, :), -vmax);
        v(i, :) = min(v(i, :), vmax);

        % Atualizar Posição
        x(i, :) = x(i, :) + v(i, :);

        % Limitar Posição (Clamping)
        x(i, :) = max(x(i, :), VarLBounds);
        x(i, :) = min(x(i, :), VarUBounds);

        % Avaliar
        cost(i) = ObjFunc(x(i, 1), x(i, 2));

        % Atualizar Pbest
        if cost(i) < pbest_val(i)
            pbest(i, :) = x(i, :);
            pbest_val(i) = cost(i);

            % Atualizar Gbest
            if pbest_val(i) < gbest_val
                gbest = pbest(i, :);
                gbest_val = pbest_val(i);
            end
        end
    end

    % --- Cálculos Estatísticos ---
    hist_gbest_val(it) = gbest_val;
    hist_avg_cost(it) = mean(cost);

    fitness_pop = 1 ./ (1 + cost);
    hist_best_fitness(it) = 1 / (1 + gbest_val);
    hist_avg_fitness(it) = mean(fitness_pop);

    % --- Guardar Trajetórias ---
    hist_gbest_x(it) = gbest(1);
    hist_gbest_y(it) = gbest(2);
    hist_pop_x(it, :) = x(:, 1)';
    hist_pop_y(it, :) = x(:, 2)';
    hist_pop_costs(it, :) = cost';

    % Decaimento da Inércia
    w = w * w_damping;

    it = it + 1;
end

%-------------------------------------------------------------------------
% GRÁFICOS FINAIS PARA O RELATÓRIO
%-------------------------------------------------------------------------
```

```
% --- FIGURA 1: Trajetória 2D Final (COM CAMINHO) ---
figure(1); clf;
ezcontourf(f1_str, [VarLBounds(1), VarUBounds(1)], [VarLBounds(2),
VarUBounds(2)]);
colormap(pink); hold on;
% Trajetória do Melhor Global (gbest)
plot(hist_gbest_x, hist_gbest_y, 'k.--', 'LineWidth', 1, 'MarkerSize', 8);
% Ponto Final
plot(gbest(1), gbest(2), 'ro', 'MarkerSize', 12, 'MarkerFaceColor', 'r');
title('Solução Final e Trajetória 2D (PSO)');
legend('Contorno', 'Trajetória gbest', 'Melhor Solução');

% --- FIGURA 2: Evolução da APTIDÃO ---
figure(2);
plot(1:maxit, hist_best_fitness, 'b-', 'LineWidth', 2); hold on;
plot(1:maxit, hist_avg_fitness, 'r--', 'LineWidth', 1.5);
title('Evolução da Aptidão (Fitness) - PSO');
xlabel('Iteração'); ylabel('Aptidão (Maximizar)');
legend('Melhor Fitness', 'Média Fitness', 'Location', 'SouthEast');
grid on;

% --- FIGURA 3: Evolução do CUSTO (COM NUVEM POPULACIONAL) ---
figure(3);
hold on;
% 1. Nuvem de custo de TODAS as partículas a cinza
plot(1:maxit, hist_pop_costs, 'Color', [0.8 0.8 0.8]);
% 2. Média a Azul Tracejado
plot(1:maxit, hist_avg_cost, 'b--', 'LineWidth', 1.5);
% 3. Melhor Global a Vermelho
plot(1:maxit, hist_gbest_val, 'r-', 'LineWidth', 2);
title('Convergência do Custo (Enxame vs gbest) - f1');
xlabel('Iteração'); ylabel('Custo (Minimizar)');
grid on;
% Legenda personalizada
h = zeros(3, 1);
h(1) = plot(NaN,NaN,'Color', [0.8 0.8 0.8]);
h(2) = plot(NaN,NaN,'b--', 'LineWidth', 1.5);
h(3) = plot(NaN,NaN,'r-', 'LineWidth', 2);
legend(h, 'Partículas (Todos)', 'Média', 'Melhor (gbest)');

% --- FIGURA 4: Visualização 3D (COM CAMINHO) ---
figure(4);
x_grid = linspace(VarLBounds(1), VarUBounds(1), 50);
y_grid = linspace(VarLBounds(2), VarUBounds(2), 50);
[X, Y] = meshgrid(x_grid, y_grid);
Z = ObjFunc(X, Y);
surf(X, Y, Z, 'EdgeColor', 'none', 'FaceAlpha', 0.8);
colormap(jet); hold on;
% Trajetória 3D (flutuante)
plot3(hist_gbest_x, hist_gbest_y, hist_gbest_val, 'k.-', 'LineWidth', 1.5,
'MarkerSize', 8);
% Ponto final
plot3(gbest(1), gbest(2), gbest_val, 'ro', 'MarkerSize', 15,
'MarkerFaceColor', 'r');
title('Solução Final e Trajetória 3D (PSO)');
xlabel('x'); ylabel('y'); zlabel('f1(x,y)');
view(45, 30); grid on;
legend('Superfície', 'Trajetória gbest', 'Melhor Solução');
```

```
% --- FIGURA 5: Evolução das Variáveis X e Y (COM NUVEM) ---
figure(5);
% Subplot X
subplot(2, 1, 1); hold on;
plot(1:maxit, hist_pop_x, 'Color', [0.7 0.7 0.7]); % Nuvem cinza
plot(1:maxit, hist_gbest_x, 'r', 'LineWidth', 2);   % gbest vermelho
title('Evolução da Posição X (Enxame vs gbest)');
ylabel('Valor X'); grid on;

% Subplot Y
subplot(2, 1, 2); hold on;
plot(1:maxit, hist_pop_y, 'Color', [0.7 0.7 0.7]); % Nuvem cinza
plot(1:maxit, hist_gbest_y, 'b', 'LineWidth', 2);   % gbest azul
title('Evolução da Posição Y (Enxame vs gbest)');
xlabel('Iteração'); ylabel('Valor Y'); grid on;

% Resultados
fprintf('\n--- Resultado Final PSO (f1) ---\n');
fprintf('Melhor posição (gbest): x = %.10f, y = %.10f\n', gbest(1),
gbest(2));
fprintf('Melhor valor (custo): %.10f\n', gbest_val);
```