Algoritmos de otimização com meta-heurística

Daniel Arruda Ponte - 121048284

08 de Julho de 2025

Resumo

Este trabalho compara quatro algoritmos de otimização por meta-heurísticas: Algoritmo Genético (AG), Evolução Diferencial (DE), Otimização por Enxame de Partículas (PSO) e o C-DEEPSO. Os experimentos foram realizados nas funções de benchmark Rosenbrock e Rastrigin, em dimensões 10, 30 e 50, com 30 execuções por cenário. Os parâmetros foram ajustados via **optuna** e os resultados analisados com estatísticas descritivas, teste de Friedman e teste post-hoc Nemenyi. O DE apresentou os melhores desempenhos médios, enquanto o C-DEEPSO destacou-se por sua rápida convergência inicial e maior estabilidade em altas dimensões. O PSO se mostrou competitivo em baixas dimensões, e o AG teve desempenho inferior nos cenários mais desafiadores (dimensões mais altas e função multimodal).

1 Introdução

Nesse artigo, será comparada a implementação do algoritmo de otimização com metaheurística C-DEEPSO com os algoritmos DE, PSO e Algoritmo Genético, sendo todas elas implementações autorais. Esses testes serão feitos com as funções de benchmark Rosenbrock e Rastrigin para as dimensões 10, 30 e 50; além disso, os resultados serão validados com os métodos de inferência estatística do teste de Friedman e Nemenyi.

2 Algoritmos evolutivos

Algoritmos evolutivos são métodos de otimização originalmente inspirados em comportamentos da natureza, especialmente nos princípios da evolução natural, como seleção, cruzamento e mutação. Um exemplo clássico é o comportamento coletivo de enxames ou populações, em que os indivíduos interagem entre si e com o ambiente para adaptar-se e sobreviver – um princípio que é refletido na forma como as soluções candidatas interagem em busca de melhorias ao longo das gerações. Esses algoritmos operam sobre uma população de soluções e, de forma iterativa, selecionam e modificam os indivíduos com base em seu desempenho em relação a uma função objetivo.

Por serem métodos estocásticos, incorporam elementos de aleatoriedade em seu processo de busca, o que contribui para explorar eficientemente o espaço de soluções. Uma de suas principais vantagens é a flexibilidade: não exigem conhecimento analítico ou derivadas da função a ser otimizada, sendo adequados para problemas complexos, não lineares e com múltiplos ótimos locais. No entanto, por sua natureza heurística, não garantem a obtenção do ótimo global — seu objetivo é fornecer boas aproximações da melhor solução possível. A seguir, serão apresentados individualmente os algoritmos evolutivos abordados neste artigo, com ênfase em suas características distintivas e desempenho comparativo.

2.1 Algoritmo Genético (AG)

O *Algoritmo Genético* (AG) é uma meta-heurística inspirada nos princípios da evolução natural e seleção biológica. Ele busca encontrar a solução ótima de uma função $f(x)$ por meio da evolução de uma população de soluções candidatas ao longo de várias gerações.

A cada iteração, também chamada de *geração*, o algoritmo executa as etapas de *seleção*, *crossover* e *mutação* para gerar uma nova população. A **seleção** privilegia os indivíduos mais aptos com base nos valores da função objetivo; o **crossover** combina indivíduos para gerar descendentes; e a **mutação** introduz variações aleatórias nos descendentes para manter a diversidade da população.

Na implementação feita neste trabalho, foram utilizados operadores de **seleção por torneio**, **cruzamento BLX-α** e **mutação creep**. A seguir, detalhamos os três operadores aplicados em cada geração do algoritmo.

Seleção por Torneio. A seleção por torneio é um método estocástico que consiste em amostrar aleatoriamente N indivíduos da população atual e selecionar o mais apto entre eles para reprodução. Esse processo é repetido até que a quantidade necessária de indivíduos selecionados seja atingida.

Crossover BLX-α. O operador de *crossover BLX-α* (Blend Crossover) é uma técnica utilizada para gerar um descendente N a partir de dois pais $x^{(1)}$ e $x^{(2)}$ no espaço contínuo; essa técnica recebe também um parâmetro α para o cálculo da distribuição uniforme.

$$N = x^{(1)} + \mathcal{U}(-\alpha, 1+\alpha) \times (x^{(2)} - x^{(1)})$$

Mutação Creep. A mutação *creep* consiste em adicionar uma pequena perturbação aleatória a cada gene de um indivíduo com uma certa probabilidade. Essa perturbação é usualmente amostrada de uma distribuição normal ou uniforme centrada em zero, para essa implementação foi utilizada a distribuição normal.

$$x_i \leftarrow x_i + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

2.2 Evolução Diferencial (DE)

A *Evolução Diferencial* (DE) é uma meta-heurística estocástica inspirada na recombinação e mutação de vetores direcionada por diferenças entre indivíduos da população.

Assim como o Algoritmo Genético, a cada geração, novos candidatos são criados por meio de três operações principais: **mutação**, **recombinação** e **seleção**. A **mutação** consiste em gerar um vetor perturbado a partir da diferença entre dois ou mais indivíduos da população; a **recombinação** mistura esse vetor mutante com o indivíduo atual para formar uma solução candidata; por fim, a **seleção** escolhe entre a solução atual e a nova, mantendo aquela com melhor valor de função objetivo.

Na implementação feita neste trabalho, utilizou-se a estratégia de mutação **DE/current-to-best/1** e a recombinação do tipo binária. A seguir, detalham-se esses mecanismos utilizados em cada etapa da DE.

Mutação Diferencial. A mutação é realizada pela combinação linear de vetores da população. Na implementação deste trabalho, o vetor mutante v_i é construído utilizando a estratégia **DE/current-to-best/1**, que possui o seguinte cálculo:

$$v_i = x_i + F \cdot (x_{\text{best}} - x_i) + F \cdot (x_{r1} - x_{r2})$$

Nessa expressão, os índices $r1, r2$ referem-se a vetores distintos escolhidos aleatoriamente da população (diferentes entre si e do vetor-alvo x_i), F é o fator de escala e x_{best} é o indivíduo mais apto da população atual.

Crossover Binário. O operador de *crossover* combina o vetor-alvo x_i com o vetor mutante v_i para formar o vetor trial u_i. Para cada dimensão j, define-se:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{se } rand_j \leq Cr \\ x_{i,j}, & \text{caso contrário} \end{cases}$$

onde Cr é a taxa de recombinação, $rand_j$ é uma amostra aleatória em $[0, 1]$ e j_{rand} é um índice aleatório que garante que pelo menos um gene venha de v_i.

Seleção por Substituição Direta. Após gerar o vetor trial u_i e avaliá-lo, ele substitui o vetor-alvo x_i apenas se apresentar aptidão melhor (minimização da função objetivo):

$$x_i \leftarrow \begin{cases} u_i, & \text{se } f(u_i) < f(x_i) \\ x_i, & \text{caso contrário} \end{cases}$$

Esse mecanismo garante que a nova população nunca tenha desempenho pior que a anterior, promovendo uma pressão seletiva constante.

2.3 Otimização por Enxame de Partículas (PSO)

A *Otimização por Enxame de Partículas* (PSO) é uma técnica de otimização populacional inspirada no comportamento coletivo de enxames, como bandos de aves ou cardumes de peixes, ao buscar alimento.

Cada solução candidata é representada por uma **partícula**, que se move em um espaço de busca multidimensional. As partículas possuem memória da melhor posição que encontraram individualmente (*personal best position*) e são também influenciadas pela melhor posição já encontrada por todo o enxame (*global best position*). A movimentação das partículas é controlada por duas equações principais: a atualização de **velocidade** e a atualização de **posição**.

Atualização da Velocidade. A velocidade de cada partícula é atualizada a partir de três componentes: inércia, atração pela melhor posição pessoal e atração pela melhor posição global. A equação de atualização da velocidade é dada por:

$$v_{i,j} \leftarrow w \cdot v_i + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

Nessa expressão: v_i é a velocidade da partícula i; x_i é a posição atual; p_i é a melhor posição pessoal da partícula; g é a melhor posição global do enxame; w é o coeficiente de inércia; c_1 e c_2 são os coeficientes de aceleração cognitiva e social, respectivamente; $r_1, r_2 \sim \mathcal{U}(0, 1)$ são variáveis aleatórias uniformes.

Atualização da Posição. Com a nova velocidade calculada, a posição da partícula é atualizada simplesmente somando-se a nova velocidade à posição atual:

$$x_i \leftarrow x_i + v_i$$

Seleção por Atualização de Memória. Ao final de cada iteração, o valor da função objetivo da nova posição x_i é comparado com o da melhor posição já encontrada pela partícula, p_i. Caso a nova posição seja superior (menor valor da função objetivo, no caso de minimização), a memória da partícula é atualizada:

$$p_i \leftarrow \begin{cases} x_i, & \text{se } f(x_i) < f(p_i) \\ p_i, & \text{caso contrário} \end{cases}$$

Além disso, a melhor posição global g do enxame é atualizada com base na melhor posição pessoal entre todas as partículas. Esse mecanismo de atualização assegura que a informação sobre boas soluções seja preservada ao longo das iterações, guiando o movimento coletivo do enxame para regiões promissoras do espaço de busca.

2.4 C-DEEPSO

O *Canonical Differential Evolutionary Particle Swarm Optimization* (C-DEEPSO) é uma meta-heurística híbrida que combina elementos dos algoritmos Particle Swarm Optimization (PSO) e Differential Evolution (DE). Proposto por Marcelino (2017), o C-DEEPSO estende a abordagem DEEPSO de Miranda e Alves (2013), incorporando conceitos como memória coletiva, macrogradiente direcional, mutações nos pesos de movimentação e operadores do DE para guiar a atualização das partículas.

As partículas no C-DEEPSO possuem posição e velocidade como no PSO, mas não mantêm memória individual. Em vez disso, utilizam uma **memória coletiva** que armazena as k_{mem} melhores soluções encontradas até o momento. A trajetória de cada partícula é determinada por três componentes: **inércia**, **macrogradiente direcional** e **comunicação com a melhor solução global**. Cada um desses componentes é ponderado por pesos adaptativos sujeitos à mutação gaussiana, promovendo diversidade nas partículas geradas. A comunicação entre partículas é modulada por uma matriz estocástica binária, gerada com base na probabilidade p_{com}.

Vetor de estratégias. A cada iteração, para cada partícula x_i, são gerados dois vetores guias: um vetor x_{DE} obtido por meio de operadores do DE (mutação e recombinação) e um vetor x_r amostrado aleatoriamente da população ou da memória coletiva. O vetor x_{st} é então definido como o melhor entre x_{DE} e x_r de acordo com a avaliação da função objetivo:

$$x_{st} \leftarrow \begin{cases} x_{DE}, & \text{se } f(x_{DE}) < f(x_r) \\ x_r, & \text{caso contrário} \end{cases}$$

Essa escolha permite utilizar a melhor entre uma solução gerada por variações direcionadas e uma alternativa aleatória, equilibrando exploração e intensificação.

Para este trabalho, foram escolhidas as estratégias **DE/current-to-best/1** para a geração do vetor x_{DE} e Pb para amostragem do vetor x_r da memória coletiva.

Macrogradiente. O C-DEEPSO utiliza uma espécie de gradiente local que atua na mudança de sinal da diferença entre vetores guias. O chamado **macrogradiente direcional** é definido como o **sinal da direção** do vetor mais promissor (melhor aptidão) que corrige a direção da diferença entre x_{st} e a partícula atual x_i, ou seja:

$$\hat{\nabla}(x_{st}, x_i) \leftarrow \begin{cases} 1, & \text{se } f(x_{st}) < f(x_i) \\ -1, & \text{caso contrário} \end{cases}$$

Esse vetor de sinais indica a direção componente a componente em que a partícula deve se mover para se aproximar da melhor alternativa disponível, sem depender de derivadas analíticas da função objetivo.

Mutação. Os pesos de inércia (w_I), macrogradiente (w_A) e comunicação (w_C), assim como a melhor posição global x_{gb}, são perturbados a cada iteração por ruído gaussiano, com intensidade controlada pela taxa de mutação τ_{mut}. Essa mutação adaptativa visa evitar a estagnação prematura, mantendo a diversidade do enxame.

Equação de movimento. A velocidade de cada partícula é atualizada pela seguinte equação vetorial:

$$v_i' \leftarrow \tilde{w}_I v_i + \tilde{w}_A \cdot \hat{\nabla}(x_{st}, x_i) \cdot (x_{st} - x_i) + \tilde{w}_C \cdot C \cdot (\tilde{x}_{gb} - x_i)$$

em que $\tilde{w}_I, \tilde{w}_A, \tilde{w}_C$ são os pesos mutados, C é a matriz de comunicação binária, e $\tilde{x}_{gb}$ é a versão perturbada da melhor solução global. A posição é então atualizada por:

$$x_i' \leftarrow x_i + v_i'$$

com projeção para dentro dos limites do espaço de busca, se necessário.

Seleção e Memória Coletiva. Após avaliar $f(x_i')$, a partícula adota a nova posição apenas se esta apresentar melhor desempenho (menor valor da função objetivo). As melhores soluções da população são utilizadas para atualizar a memória coletiva, que armazena as k_{mem} melhores soluções encontradas até então. A melhor solução global x_{gb} também é atualizada a cada geração, garantindo que o conhecimento coletivo seja constantemente refinado.

3 Descrição das Funções de Benchmark

3.1 Rosenbrock

A função de Rosenbrock é definida como:

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2 \right].$$

É uma função **não convexa**, **diferenciável** e **unimodal**. O mínimo global ocorre em $\mathbf{x}^* = \mathbf{1}$, com $f(\mathbf{x}^*) = 0$. Para os experimentos, considera-se $x_i \in [-2.048, 2.048]$.

3.2 Rastrigin

A função Rastrigin é definida como:

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) \right].$$

É uma função **não convexa**, **diferenciável** e altamente **multimodal**. O mínimo global ocorre em $\mathbf{x}^* = \mathbf{0}$ com $f(\mathbf{x}^*) = 0$. Para os experimentos, considera-se $x_i \in [-0.5, 0.5]$.

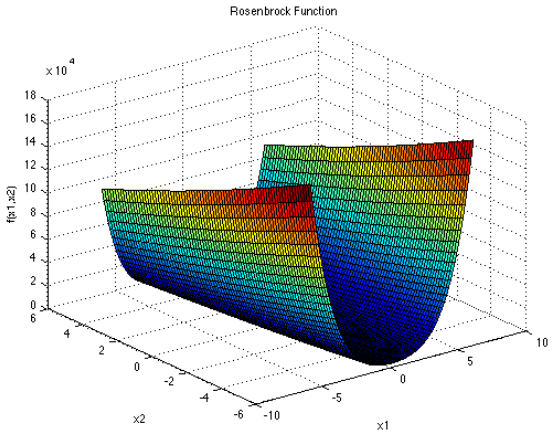

Figura 1: Função Rosenbrock

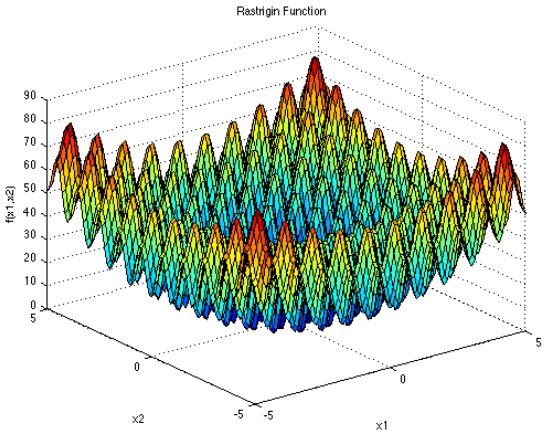

Figura 2: Função Rastrigin

4 Experimentos

Os experimentos foram conduzidos utilizando as funções Rosenbrock e Rastrigin descritas na Seção 3, sendo testadas nas **dimensões 10, 30 e 50**, com respectivas limitações de quantidade de **avaliações de função: 10.000, 30.000 e 50.000**. Além disso, o tamanho da população de todos os testes foi fixado em **100 indivíduos** e foram computadas **30 execuções** de cada cenário de teste.

Em relação aos parâmetros de execução dos algoritmos, para cada cenário de teste —tripla (Algoritmo, Função, Dimensão) —foram otimizados usando a biblioteca **optuna**; segue a tabela abaixo com os resultados:

Tabela 1: Parâmetros do AG

Função	Dimensão	tx_recomb	tx_mut	t_elite	t_torneios	var_mut
Rosenbrock	10	0.31	0.64	8	2	0.16
Rastrigin	10	0.66	0.49	7	7	0.26
Rosenbrock	30	0.79	0.75	3	10	0.12
Rastrigin	30	0.40	0.37	7	1	0.12
Rosenbrock	50	0.26	0.58	8	1	0.15
Rastrigin	50	0.37	0.58	2	5	0.14

Tabela 2: Parâmetros do CDEEPSO

Função	Dimensão	Cr	t_mut	p_com	k_mem	wi	wa	wc
Rosenbrock	10	0.61	0.38	0.21	5	0.33	0.73	0.30
Rastrigin	10	0.13	0.53	0.86	4	0.32	0.40	0.85
Rosenbrock	30	0.63	0.22	0.33	9	0.13	0.68	0.90
Rastrigin	30	0.68	0.34	0.90	2	0.12	0.28	0.74
Rosenbrock	50	0.13	0.13	0.90	7	0.13	0.48	0.52
Rastrigin	50	0.32	0.54	0.86	4	0.25	0.50	0.22

Tabela 3: Parâmetros do PSO

Função	Dimensão	w	c1	c2
Rosenbrock	10	0.61	1.61	1.74
Rastrigin	10	0.41	1.50	1.53
Rosenbrock	30	0.49	2.05	1.58
Rastrigin	30	0.46	2.37	1.50
Rosenbrock	50	0.48	1.74	1.71
Rastrigin	50	0.53	2.22	1.57

Tabela 4: Parâmetros do DE

Função	Dimensão	Cr	F
Rosenbrock	10	0.17	0.58
Rastrigin	10	0.74	0.54
Rosenbrock	30	0.19	0.62
Rastrigin	30	0.90	0.49
Rosenbrock	50	0.27	0.63
Rastrigin	50	0.90	0.47

Ao final das execuções, foram calculadas as seguintes métricas estatísticas para avaliar o desempenho dos algoritmos: Média dos valores de função obtidos, Desvio padrão, Melhor (menor) valor encontrado e Pior (maior) valor encontrado. Também foram gerados gráficos de curva de convergência e gráficos boxplot dos resultados que serão exibidos na seção abaixo.

5 Resultados

Neste trabalho, para análise de resultados, foram utilizados os testes não paramétricos de Friedman e Nemenyi para comparar o desempenho de diferentes algoritmos de otimização. O teste de Friedman avalia se há diferenças estatísticas globais entre os algoritmos, enquanto o teste de Nemenyi realiza comparações par a par para identificar quais algoritmos diferem entre si.

Teste de Friedman: Assume dados pareados e independência entre blocos, sem exigir normalidade. Hipóteses: H_0: As medianas (ou distribuições) dos algoritmos são iguais; H_1: Pelo menos um algoritmo difere significativamente dos demais. Estatística:

$$\chi_F^2 = \frac{12N}{k(k+1)} \sum_{j=1}^{k} R_j^2 - 3N(k+1)$$

com $N = \text{nº}$ de problemas, $k = \text{nº}$ de algoritmos, $R_j =$ soma dos ranks do algoritmo j.

Como mostrado na tabela a seguir, todos os testes de Friedman apresentaram p-valor $< 0{,}05$, indicando que há diferença estatística entre algum dos quatro algoritmos em todos os cenários avaliados.

Função	Dimensão	Estatística de Friedman	p-valor	Decisão sobre H_0
Rosenbrock	10	73,72	**6,814183e-16**	Rejeita
Rastrigin	10	38,12	**2,665868e-08**	Rejeita
Rosenbrock	30	66,28	**2,670068e-14**	Rejeita
Rastrigin	30	66,32	**2,617964e-14**	Rejeita
Rosenbrock	50	87,76	**6,631522e-19**	Rejeita
Rastrigin	50	72,76	**1,094217e-15**	Rejeita

Tabela 5: Resultados do teste de Friedman com $\alpha = 0{,}05$

Teste de Nemenyi (post-hoc): Hipóteses: H_0: algoritmos i e j têm o mesmo desempenho; H_1: diferem significativamente. Diferença Crítica:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

Se $|R_i - R_j| > CD$, há diferença significativa.

Para o teste de Nemenyi, temos os resultados da tabela abaixo:

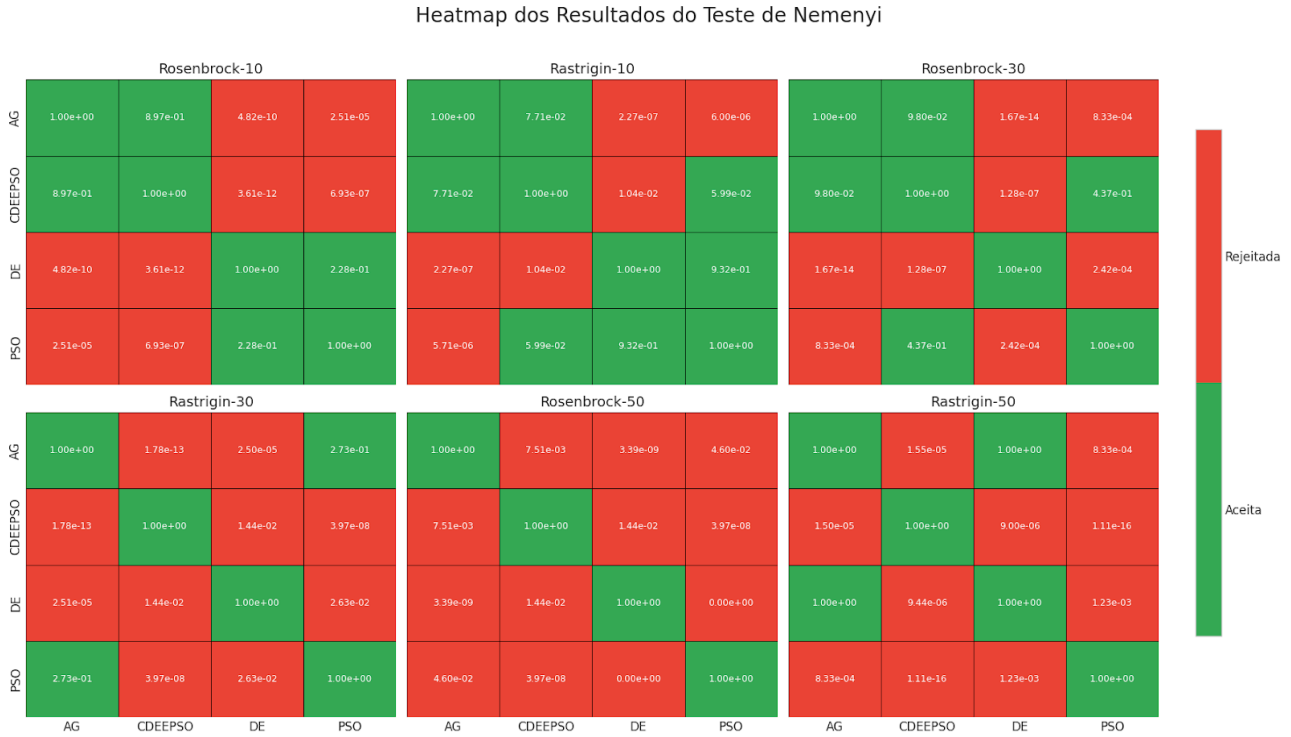

Figura 3: Heatmaps dos resultados do Teste de Nemenyi

A partir disso, temos os resultados dos experimentos realizados, destacando os casos em negrito que possuem uma diferença estatística significante em relação aos demais algoritmos; na Tabela 6 são exibidos os resultados dos experimentos feitos na função Rosenbrock e na Tabela 7 na função Rastrigin. A seguir, também são apresentados os gráficos boxplot e curvas de convergência separados pela função testada e pela dimensão do problema.

Tabela 6: Resultados dos algoritmos para a função Rosenbrock

Algoritmo	Dimensão	Média	Desvio Padrão	Mínimo	Máximo
C-DEEPSO	10	8.915419	0.986863	5.035167	9.936764
PSO	10	**5.030057**	1.532106	0.005172	8.021951
DE	10	**3.638633**	1.182451	1.184446	5.584688
AG	10	8.878965	0.971119	5.463885	10.147080
C-DEEPSO	30	29.114159	0.270088	28.488425	29.572643
PSO	30	34.001103	25.303426	20.962230	144.182200
DE	30	**20.919153**	1.409679	18.694685	24.960480
AG	30	38.998634	26.672334	28.205904	167.215489
C-DEEPSO	50	48.665503	0.065242	48.527660	48.800284
PSO	50	198.226438	202.812370	58.242394	1124.518155
DE	50	**40.030239**	1.596904	37.863213	45.663233
AG	50	68.137100	17.171135	51.499594	134.876731

Tabela 7: Resultados dos algoritmos para a função Rastrigin

Algoritmo	Dimensão	Média	Desvio Padrão	Mínimo	Máximo
C-DEEPSO	10	22.067741	15.250379	0.360426	49.586991
PSO	10	**11.215701**	6.154812	1.064827	28.853742
DE	10	**8.908251**	1.621703	5.399575	12.548462
AG	10	30.191672	17.855001	9.882670	79.747599
C-DEEPSO	30	**6.777418**	18.236059	0.192364	92.018961
PSO	30	63.164682	17.836375	32.835540	96.501044
DE	30	43.163548	4.012954	34.850333	52.665680
AG	30	89.946344	25.415220	31.655010	152.626958
C-DEEPSO	50	**4.037647**	3.004185	0.502139	14.412273
PSO	50	199.245308	39.136429	111.608895	300.547412
DE	50	121.398369	8.538013	105.299928	138.637603
AG	50	125.160492	71.516076	34.121064	349.997535

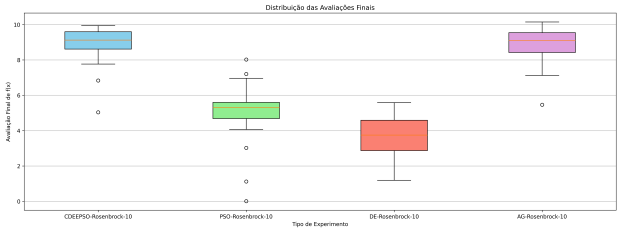

Figura 4: Boxplot para Rosenbrock (10)

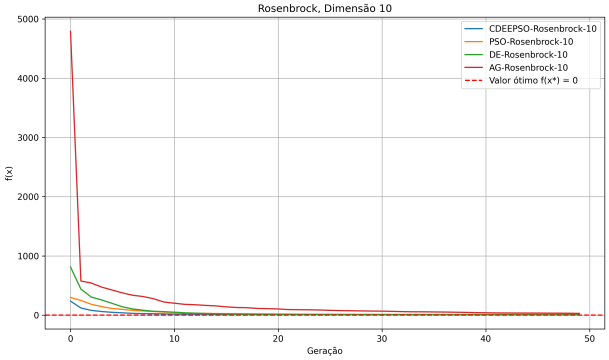

Figura 5: Curvas de convergência Rosenbrock (10)

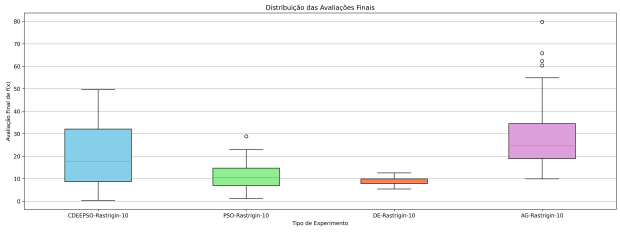

Figura 6: Boxplot para Rastrigin (10)

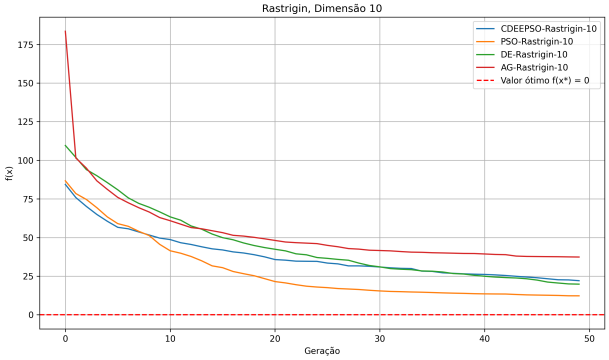

Figura 7: Curvas de convergência Rastrigin (10)

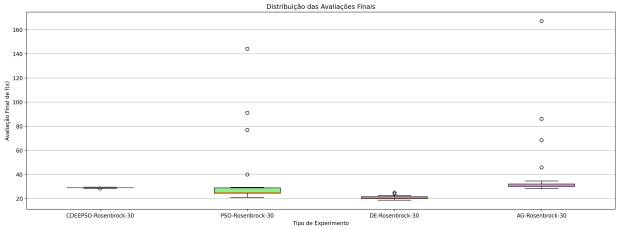

Figura 8: Boxplot para Rosenbrock (30)

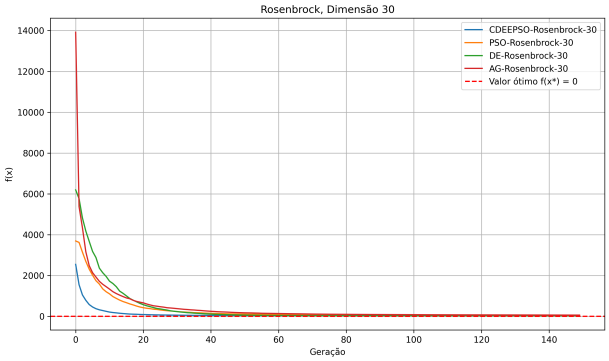

Figura 9: Curvas de convergência Rosenbrock (30)

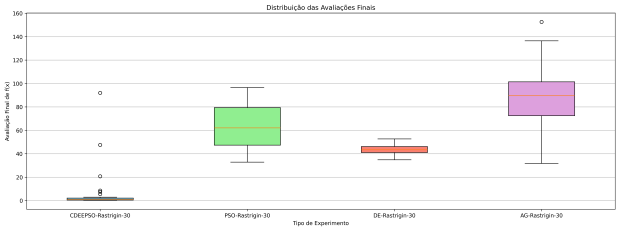

Figura 10: Boxplot para Rastrigin (30)

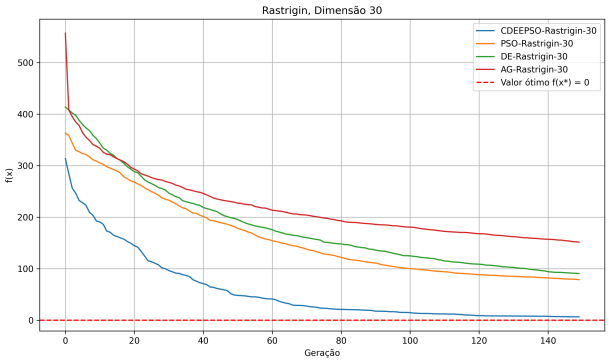

Figura 11: Curvas de convergência Rastrigin (30)

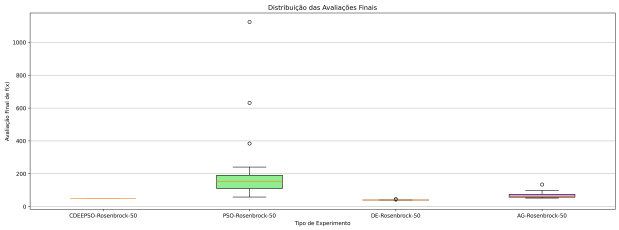

Figura 12: Boxplot para Rosenbrock (50)

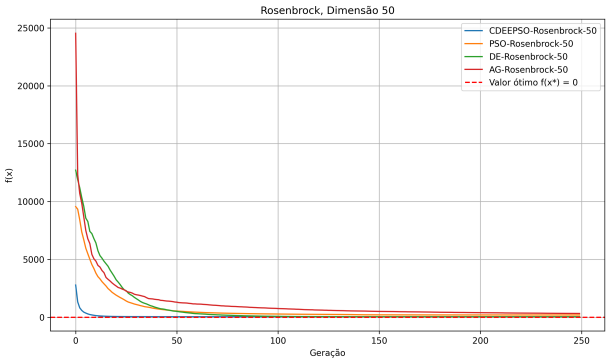

Figura 13: Curvas de convergência Rosenbrock (50)

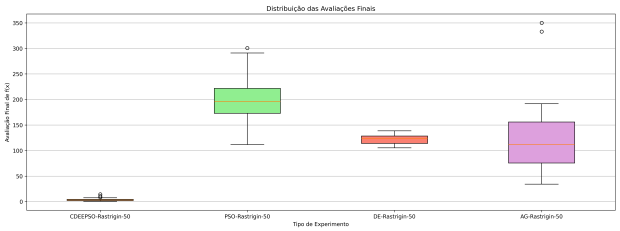

Figura 14: Boxplot para Rastrigin (50)

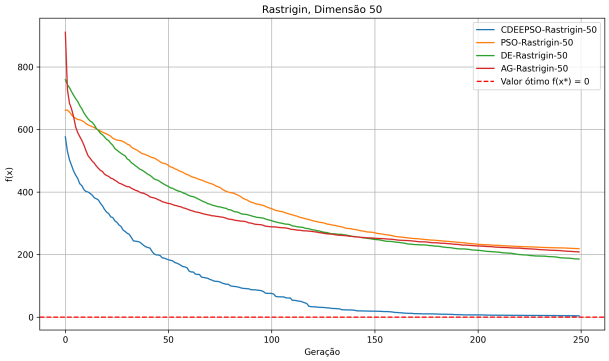

Figura 15: Curvas de convergência Rastrigin (50)

6 Conclusões e Considerações Finais

Este trabalho avaliou comparativamente o desempenho de quatro algoritmos de otimização com meta-heurística — Algoritmo Genético (AG), Evolução Diferencial (DE), Otimização por Enxame de Partículas (PSO) e C-DEEPSO — em cenários envolvendo as funções de benchmark Rosenbrock (unimodal) e Rastrigin (multimodal) em dimensões de 10, 30 e 50. Além disso, foram feitos os ajustes de hiperparâmetros com optuna, a execução de 30 rodadas independentes por cenário e a validação estatística com os testes de Friedman e Nemenyi.

A análise estatística, por meio do teste de Friedman, confirmou com alta significância ($p < 0.05$) a existência de diferenças de desempenho entre os algoritmos em todos os cenários. Para complementar, o teste post-hoc de Nemenyi e as métricas de desempenho destacaram a Evolução Diferencial (DE) como o algoritmo mais eficaz na maioria das situações, apresentando os melhores resultados médios na função Rosenbrock em todas as dimensões e na função Rastrigin em dimensão 10. O PSO demonstrou desempenho bom, especialmente em dimensões mais baixas, mas seus resultados pioraram consideravelmente em problemas de dimensões maiores, onde apresentou alto desvio padrão nos resultados.

O C-DEEPSO revelou um funcionamento bom com problemas complexos, superando com significância estatística os demais algoritmos na função multimodal Rastrigin em altas dimensões (30 e 50). Uma característica notável, observada nas curvas de convergência, foi sua velocidade superior nas iterações iniciais. Mesmo nos cenários em que não obteve a melhor média, como na função Rosenbrock de 50 dimensões, o C-DEEPSO se destacou por sua estabilidade (menor desvio padrão), superando todos os outros nesse quesito. Em contrapartida, o Algoritmo Genético (AG), foi o terceiro melhor algoritmo para todas experimentações da Rosenbrock e o pior algoritmo para todos experimentos conduzidos na Rastrigin, o que mostra uma dificuldade dessa implementação do algoritmo convergir numa função multimodal como a Rastrigin.

Em resumo, o DE se mostrou um algoritmo muito bom e robusto, o C-DEEPSO implementado se mostrou como uma ferramenta que desempenhou bem para problemas multimodais de alta dimensão. O PSO permanece como uma opção viável para problemas mais simples, e o AG requereria modificações na sua implementação para competir nestes benchmarks.

Referências

[1] S. Surjanovic and D. Bingham, "Virtual Library of Simulation Experiments: Test Functions and Datasets," Acessado em 7 de julho de 2025, em `https://www.sfu.ca/~ssurjano/optimization.html`.

[2] Y. Zhang, J. Liu, and H. Wang, "A Comparative Study of Recent Multi-objective Optimization Algorithms for Solving Engineering Problems," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, 2022, pp. 1-8.

[3] C. L. Hwang and A. S. M. Masud, "A survey of multiobjective optimization methods for engineering," *Lecture Notes in Economics and Mathematical Systems*, vol. 164, pp. 1-34, 1979.

[4] A. Gaspar-Cunha, R. Takahashi, and C. H. Antunes, Eds., *Manual de Computação Evolutiva e Metaheurísticas.* Belo Horizonte: Editora UFMG, 2013.

[5] R. H. C. Takahashi, "Otimização Escalar e Vetorial Volume 2: Otimização Escalar," *Notas de Aula, Departamento de Matemática, Universidade Federal de Minas Gerais*, 2007.

[6] Optuna Development Team, "Optuna: A hyperparameter optimization framework," Acessado em 7 de julho de 2025, em `https://optuna.org/`.

[7] DATAtab, "Friedman Test Tutorial," Acessado em 7 de julho de 2025, em `https://datatab.net/tutorial/friedman-test`.

[8] D. Wallis, "Comparing Classifiers: Friedman and Nemenyi Tests," Acessado em 7 de julho de 2025, em `https://medium.com/@diogeneswallis/comparing-classifiers-friedman-and-nemenyi-tests-32294103ee12`.

[9] Baylor.AI, "Friedman Test and Nemenyi Post-hoc Test," Acessado em 7 de julho de 2025, em `https://baylor.ai/?p=2665`.