UNIVERSIDADE
FEDERAL DO CEARÁ

Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Departamento de Matemática Aplicada
Ciência de Dados

Projeto 01 – Inteligência Artificial

Algoritmos de Busca e Otimização Local

Relatório elaborado por:
Lucas Lopes Silva – 568381
Artur Saraiva Paschoal – 555158
Artur Garcia Sales Barroso – 554042

Fortaleza – CE
20 de maio de 2025

1. Algoritmos de Busca em Grafos Ponderados

1.1 Objetivo

A etapa 1 do presente relatório visa comparar quatro algoritmos de busca clássicos em grafos ponderados: Busca em Largura (BFS), Custo Uniforme (UCS), Busca Gulosa (Greedy) e Busca A* (A-Star). O objetivo é encontrar o caminho de menor custo entre os vértices A (origem) e G (destino).

1.2 Metodologia

A construção do grafo ponderado foi realizada utilizando a biblioteca `NetworkX`, onde cada vértice recebeu uma posição bidimensional para possibilitar a visualização espacial, e as arestas foram atribuídas com pesos individuais. A visualização do grafo foi feita com `Matplotlib`.

Quatro algoritmos clássicos de busca foram implementados em Python, com um Jupyter Notebook que documenta os detalhes iterativos desde a construção das funções até os resultados, com scripts independentes para BFS, UCS, Greedy e A*. A execução de cada algoritmo retornou, além do caminho encontrado, o custo total do percurso, o número de nós expandidos e o tempo de execução computado com `time()`. Esses valores foram armazenados para comparação padronizada posterior.

Para os algoritmos informados (Greedy e A*), foi definida uma heurística $h(n)$ baseada na distância euclidiana entre cada nó e o destino G, utilizando as coordenadas espaciais dos nós. A heurística foi verificada quanto ao critério de admissibilidade, isto é, $h(n) \leq h^*(n)$ para todo nó n, onde $h^*(n)$ representa o custo real até o destino. Para essa verificação, comparou-se a heurística com os custos reais extraídos a partir da execução da UCS desde cada nó até o destino.

Para essa verificação, comparou-se a heurística com os custos reais extraídos a partir da execução da UCS desde cada nó até o destino. A Tabela 1 resume os valores comparativos.

Tabela 1: Cálculo de admissibilidade da heurística $h(n)$ em relação ao custo real UCS

Nó	$h(n)$	**Custo real UCS até G**	$h(n) \leq$ **custo?**
A	2.88	9	Sim
B	1.58	6	Sim
C	1.04	7	Sim
D	3.00	10	Sim
E	0.71	6	Sim
F	1.17	10	Sim
G	0.00	0	Sim
H	3.00	3	Sim

Todas as execuções seguiram uma estrutura uniforme de logging, permitindo rastreamento, visualização e análise integrada dos resultados.

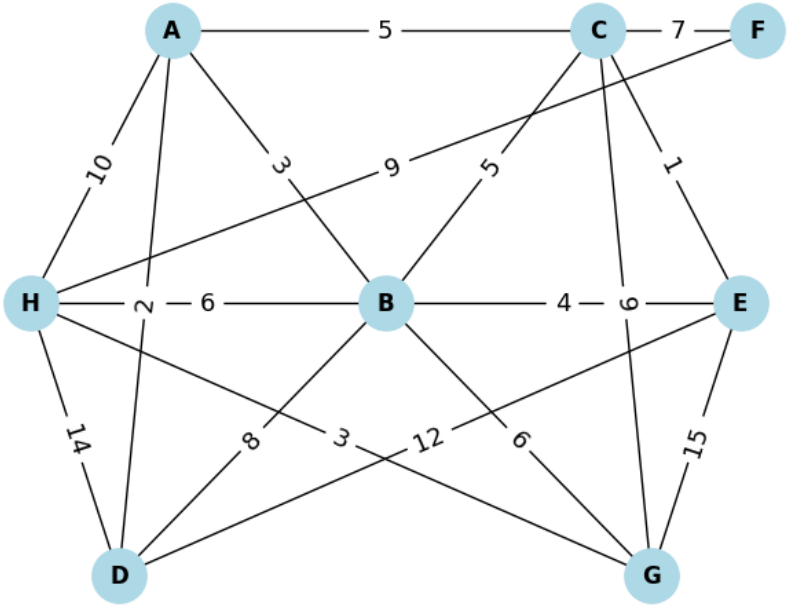

Figura 1: Grafo ponderado utilizado nos algoritmos de busca

A heurística usada nos algoritmos Greedy e A* foi baseada na distância euclidiana até o nó destino, ajustada para garantir admissibilidade.

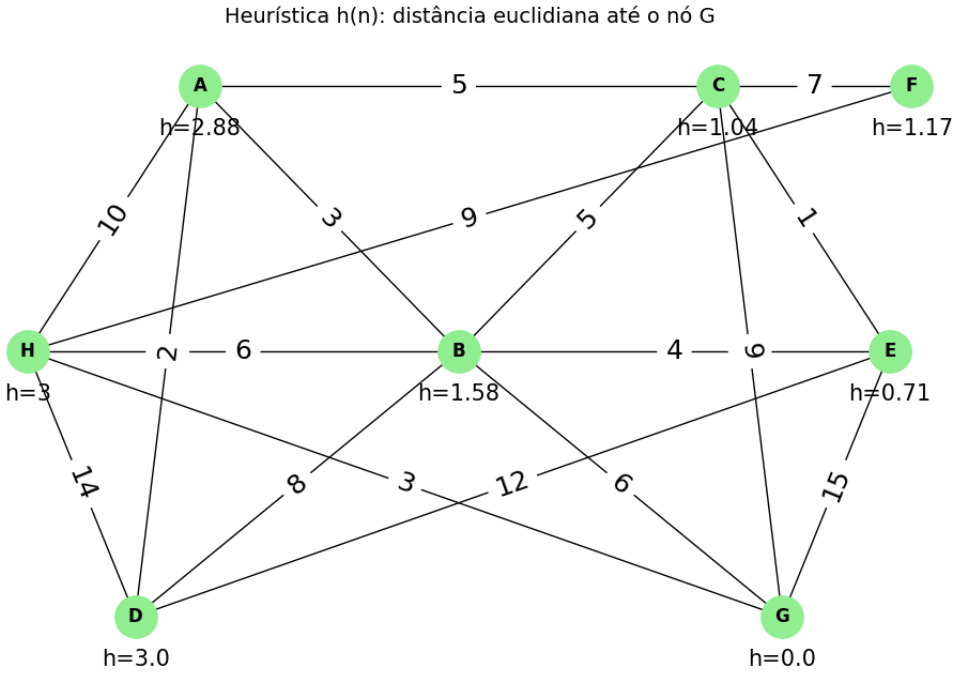

Figura 2: Visualização da heurística admissível $h(n)$ baseada na distância euclidiana ao destino

1.3 Resultados Obtidos

Tabela 2: Comparação entre algoritmos de busca

Algoritmo	Caminho	Custo Total	Nós Expandidos	Tempo (s)
BFS	A–B–G	9	7	0.000046
UCS	A–B–G	**9**	6	0.000085
Greedy	A–C–G	14	3	0.000045
A*	A–B–G	**9**	6	0.000058

1.4 Análise Comparativa

A tabela apresentada evidencia diferenças significativas entre os algoritmos testados quanto à eficiência computacional e à qualidade da solução encontrada.

O algoritmo UCS, por priorizar sempre o caminho de menor custo acumulado, encontrou o caminho ótimo com custo total igual a 9. O A* obteve o mesmo caminho e custo, porém com desempenho ligeiramente superior, tanto em tempo quanto em clareza da trajetória explorada, evidenciando o impacto positivo da heurística admissível. Ambos expanderam 6 nós.

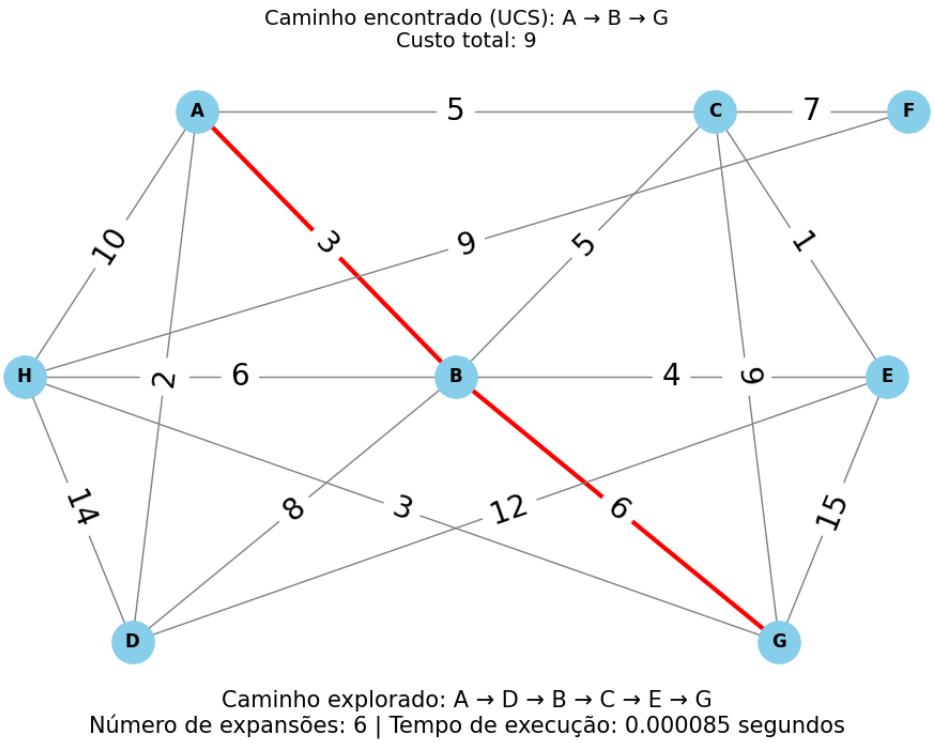

Figura 3: Caminho encontrado pela Busca de Custo Uniforme (UCS)

A Busca Gulosa (Greedy), embora mais rápida e com o menor número de expansões, retornou um caminho com custo total maior (14). Isso reforça a característica inerente desse algoritmo: decisões baseadas unicamente na estimativa até o destino podem ser precipitadas, ignorando o custo acumulado e desviando da solução ótima.

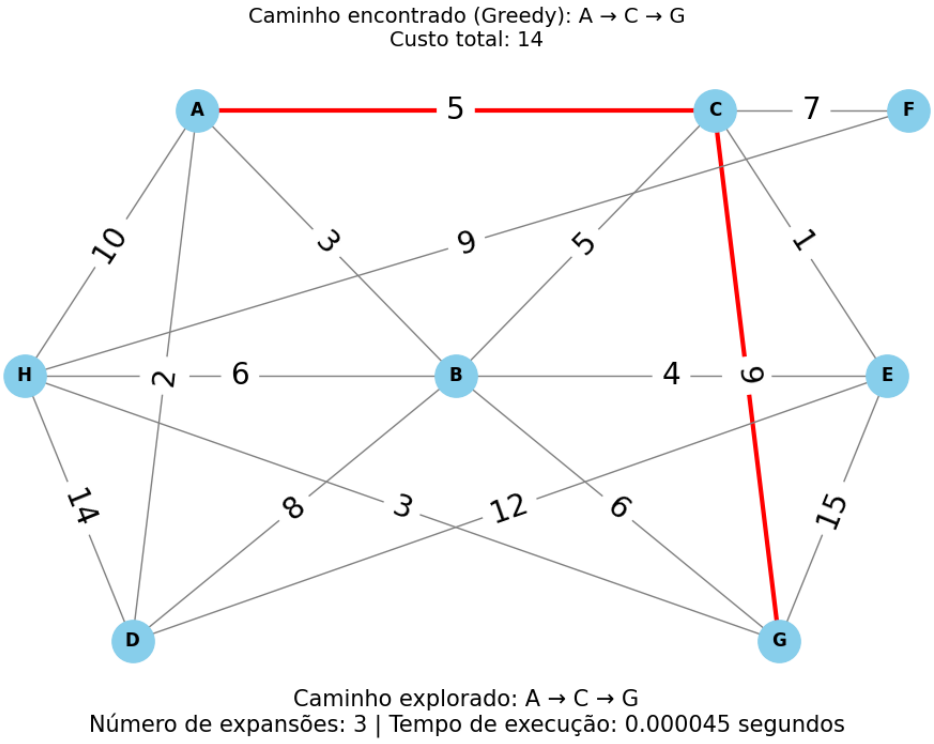

Figura 4: Caminho encontrado pela Busca Gulosa (Greedy)

A BFS apresentou a maior quantidade de expansões, reflexo direto do fato de que ela não leva em conta os pesos do grafo, tratando todos os caminhos como equivalentes em custo de transição.

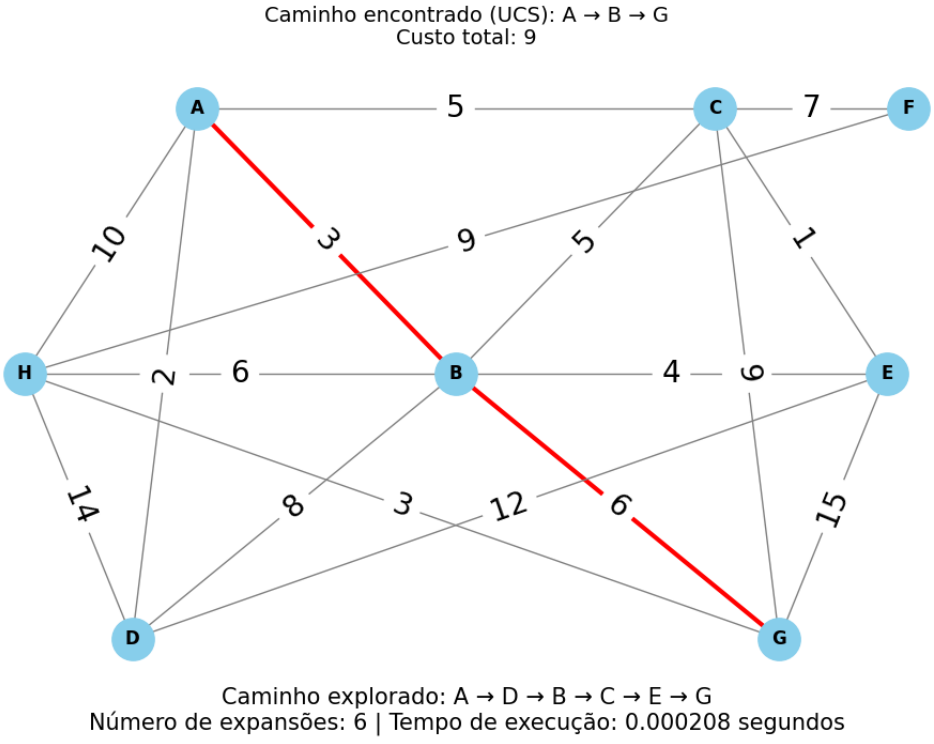

Figura 5: Caminho encontrado pelo algoritmo de Busca em Largura (BFS)

Em termos de desempenho global, o A* demonstrou o melhor equilíbrio entre custo da solução, número de expansões e tempo de execução.

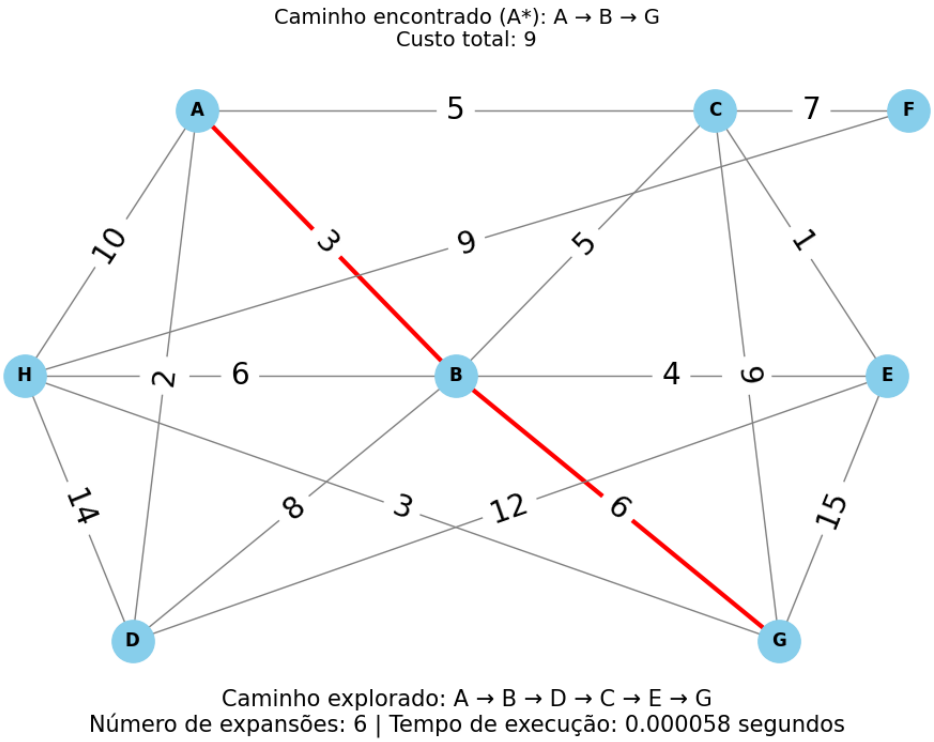

Figura 6: Caminho encontrado pela Busca A* com heurística admissível

1.5 Conclusão

A análise empírica confirmou os comportamentos teóricos esperados dos algoritmos testados. A UCS e o A* apresentaram soluções ótimas, sendo o A* mais eficiente ao empregar a heurística euclidiana admissível, o que o tornou o algoritmo mais vantajoso neste cenário.

A Greedy, embora computacionalmente econômica, apresentou limitação de qualidade da solução por considerar apenas a estimativa até o destino, negligenciando o custo do percurso acumulado. A BFS mostrou-se ineficaz em grafos ponderados, justamente por ignorar os pesos das arestas.

Dessa forma, conclui-se que a escolha do algoritmo deve considerar tanto o objetivo da busca (ótima vs. rápida) quanto o contexto do problema. A experimentação prática demonstrou, sobretudo, a importância da construção cuidadosa de heurísticas e da análise do equilíbrio entre custo computacional e precisão do resultado.

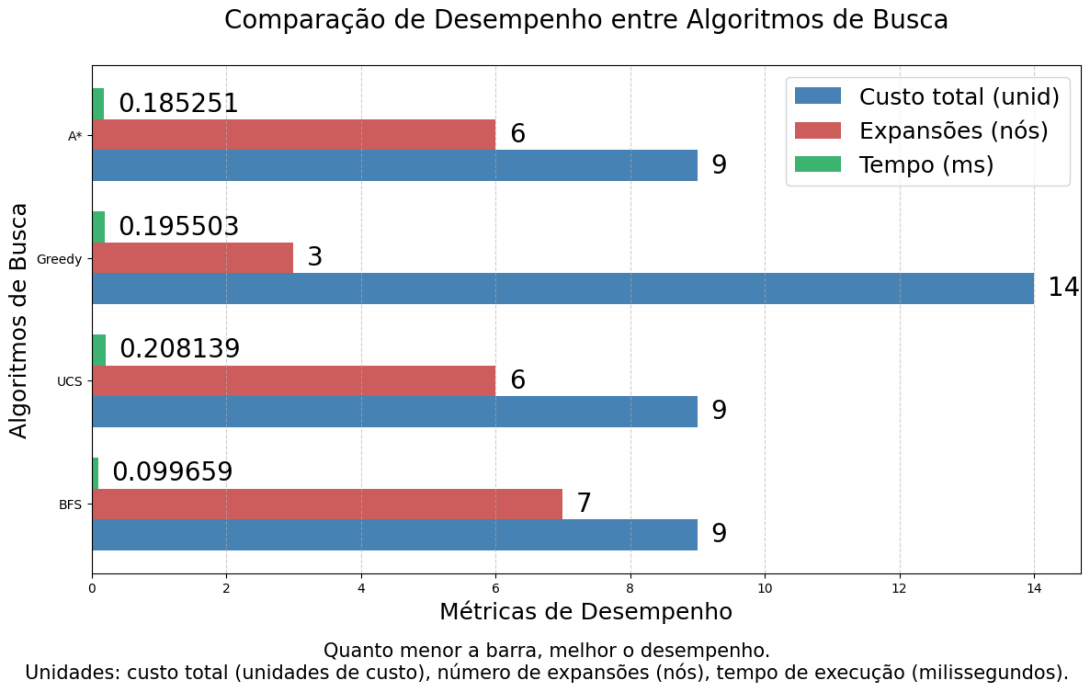

Figura 7: Comparação entre algoritmos de busca: custo total, expansões e tempo

2. Otimização Local com Hill Climbing

2.1 Objetivo

Aplicar Hill Climbing, com e sem reinício aleatório, para minimizar a função:

$$f(x, y) = x^2 + y^2 + 25(\sin^2 x + \sin^2 y)$$

2.2 Metodologia

A função $f(x, y)$ foi implementada em Python e visualizada em três dimensões com a biblioteca `Plotly`, permitindo observar sua natureza ondulatória e presença de múltiplos mínimos locais.

Inicialmente, implementou-se a versão básica do algoritmo Hill Climbing. A partir de um ponto inicial aleatório $(x_0, y_0) \in [-10, 10]$, a função foi minimizada iterativamente explorando os quatro vizinhos diretos (acréscimos e decréscimos em x e y), utilizando uma etapa de passo constante de 0.1. O processo é interrompido ao atingir um ponto onde nenhum vizinho melhora a função.

Para garantir reprodutibilidade, a seed do gerador aleatório foi fixada em 568381. Posteriormente, foi desenvolvida a versão com reinício aleatório, na qual o algoritmo básico foi executado 10 vezes, cada uma com seed diferente (definida como $568381 + i$, com i representando o índice da execução). Ao final, a melhor solução dentre todas as tentativas foi mantida como resultado global.

Durante todas as execuções, as trajetórias de pontos visitados foram armazenadas e plotadas sobre a superfície 3D da função, permitindo a análise visual da eficiência de cada tentativa.

2.3 Resultados

Versão sem reinício (seed fixa): Com a seed fixada em 568381, o algoritmo Hill Climbing iniciou em $(x, y) = (3.0113, 6.0412)$ e convergiu para um mínimo local com valor de $f(x, y) = 47.4216$ após 17 iterações. A trajetória percorrida mostrou descida estável, porém limitada a uma depressão local longe do mínimo global.

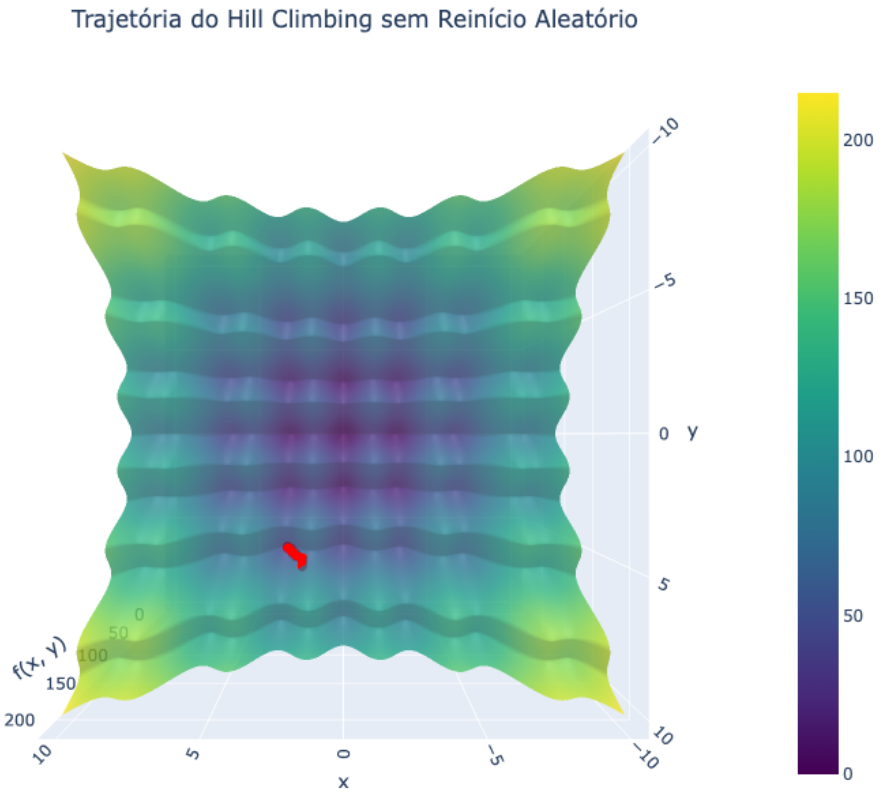

Figura 8: Trajetória do Hill Climbing sem reinício aleatório

Versão com reinício aleatório (10 execuções): O algoritmo foi executado 10 vezes, cada vez com ponto inicial distinto, utilizando seeds consecutivas definidas como $568381 + i$. O melhor resultado obtido ocorreu na nona execução (Reinício 9), com $x = -2.9905$, $y = 0.0382$ e valor mínimo aproximado $f(x, y) = 9.5474$, após 24 iterações.

As demais execuções retornaram valores significativamente superiores, com pontos de parada concentrados em regiões da função caracterizadas por mínimos locais. A diversidade dos pontos de convergência confirmou o efeito da superfície oscilatória da função e evidenciou a eficácia da estratégia de reinicialização.

A presença do mínimo global teórico em $(0, 0)$ com $f(0, 0) = 0$ não foi atingida, mas a execução com reinício aproximou-se consideravelmente.

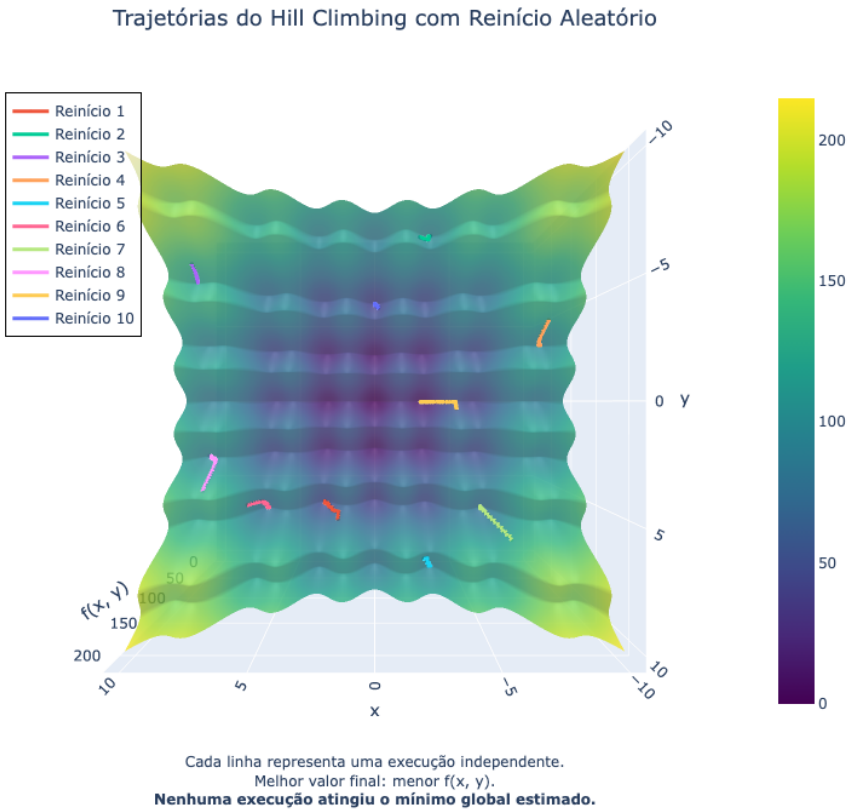

Figura 9: Trajetórias do Hill Climbing com reinício aleatório (10 execuções)

2.4 Conclusão

A análise da função $f(x, y)$ revelou um cenário clássico de otimização não convexa, caracterizado pela presença de diversos mínimos locais introduzidos pelos termos oscilatórios $\sin^2 x$ e $\sin^2 y$. O uso do Hill Climbing básico mostrou-se limitado, convergindo rapidamente a mínimos locais distantes da solução ótima, como evidenciado na primeira execução.

A estratégia de reinício aleatório, implementada com 10 execuções independentes, demonstrou ganhos expressivos na capacidade de escapar de mínimos locais e explorar regiões mais promissoras do espaço de busca. Embora nenhuma execução tenha atingido o mínimo global $(0, 0)$, os resultados mostraram aproximação significativa, com melhoria progressiva da melhor solução a cada reinício.

Essa experiência ilustra, na prática, a importância de estratégias de diversificação em algoritmos de otimização local. Além disso, evidencia o potencial de visualizações tridimensionais como ferramenta analítica para compreensão do comportamento da função e validação da trajetória das soluções.

A combinação entre heurísticas, reinício sistemático e análise gráfica se mostra uma abordagem eficiente e de fácil implementação para enfrentar desafios típicos de superfícies de erro complexas.

Referências

- Silva, L. L., Paschoal, A. S., Barroso, A. G. S. (2025). *AI Search Algorithms – Capstone Project 01 (Busca em Grafos e Hill Climbing)*. Universidade Federal do Ceará. Disponível em: `https://github.com/lucas-l-silva/AI_Search_Algorithms_CapstoneProject01.git`