



Centro Federal de Educação Tecnológica de Minas Gerais  
Curso Superior de Engenharia da Computação  
Disciplina: Inteligência Artificial

## **Relatório para Trabalho do Problema de 8 Rainhas com Hill Climbing)**

João Pedro Rodrigues Silva  
Jader Oliveira Sila  
Prof. Thiago Alves de Oliveira

Divinópolis/MG  
Outubro de 2025

# 1 Introdução:

O problema das Oito Rainhas é um clássico em Inteligência Artificial e em computação combinatória: trata-se de posicionar oito rainhas em um tabuleiro de xadrez  $8 \times 8$  de modo que nenhuma rainha ataque outra (isto é, sem compartilhamento de linha, coluna ou diagonal). Por sua simplicidade de enunciado e por seu espaço de busca relativamente pequeno, o problema é amplamente utilizado como caso-teste para técnicas de busca local e otimização.

Nesta implementação, abordamos o problema através do método de *Hill Climbing* (subida de encosta), uma estratégia de busca local que itera a partir de uma solução inicial e procura mover-se sempre para vizinhanças que melhorem a função objetivo (neste caso, minimização do número de pares de rainhas em conflito). Hill Climbing é uma técnica determinística básica de melhoria local que pode ser implementada em formas distintas. [4]

Embora simples e eficiente para muitos casos, Hill Climbing sofre de limitações importantes: pode convergir para máximos/mínimos locais, ficar preso em platôs (regiões sem melhora em fronteira) ou precisar de muitas reinicializações para alcançar uma solução global ótima. Por isso, é prática comum comparar variantes como (i) permitir movimentos laterais com limite, (ii) reinícios aleatórios e (iii) estratégias híbridas com técnicas como mitigar quedas em ótimos locais. [2, 3]

Os objetivos deste trabalho são (1) implementar a lógica básica de Hill Climbing para o problema das 8 rainhas seguindo a representação por vetor (onde o índice representa a coluna e o valor a linha da rainha), (2) comparar pelo menos duas variações do algoritmo (sendo movimentos laterais versus reinícios aleatórios), e (3) medir e discutir métricas de desempenho relevantes — número de iterações, número de reinícios, tempo até solução e taxa de sucesso estatística em múltiplas execuções.

## 2 Fundamentação Teórica

O problema das Oito Rainhas é um exemplo clássico de problema de satisfação de restrições (CSP - *Constraint Satisfaction Problem*) e de otimização combinatória: há 8 variáveis (colunas do tabuleiro) e cada variável pode assumir uma posição entre 1 e 8 (linha da rainha). A formulação e o estudo desse problema são úteis para ilustrar técnicas de busca completa (backtracking) e busca local (heurísticas de melhoria), além de fornecer um caso simples para analisar propriedades como platôs, ótimos locais e a influência da vizinhança sobre o desempenho do algoritmo [2].

Hill Climbing (subida de encosta) é uma família de métodos de busca local que parte de uma solução inicial e tenta iterativamente melhorar uma função objetivo transitando para vizinhanças melhores. Existem variações importantes: *first-improvement* (adota o primeiro vizinho que apresenta melhora), *best-improvement* (escolhe o melhor vizinho após avaliar toda a vizinhança) e formas estocásticas que introduzem aleatoriedade nas escolhas. As limitações clássicas de Hill Climbing — convergência a ótimos locais, permanência em platôs e sensibilidade à solução inicial — são bem documentadas na literatura de busca local e otimização combinatória [3, 4].

No contexto das 8 rainhas, uma representação comum para reduzir o espaço de busca é um vetor  $S = (r_1, r_2, \dots, r_8)$  tal que  $r_i$  indica a linha (1 a 8) da rainha na coluna  $i$ . Uma variante muito utilizada é restringir  $S$  a permutações de  $\{1, \dots, 8\}$ , garantindo

assim que não existam conflitos por linha (cada linha contém exatamente uma rainha) — dessa forma, a função objetivo precisa apenas contabilizar conflitos em diagonais. Essa escolha reduz o espaço de busca e simplifica a avaliação dos vizinhos, ao custo de limitar o conjunto de soluções possíveis (mas sem eliminar soluções válidas do problema original, pois toda solução válida também é uma permutação).

A função objetivo  $f$  que usaremos mede o número de pares de rainhas em conflito (quanto menor, melhor; objetivo global:  $f = 0$ ). Para uma representação por permutação, ela pode ser escrita como soma sobre pares de colunas que estejam em diagonal:

$$f(S) = \sum_{1 \leq i < j \leq 8} \mathbf{1}(|r_i - r_j| = |i - j|) \quad (1)$$

onde  $\mathbf{1}$  é a função indicadora (vale 1 se a condição é verdadeira, 0 caso contrário).

As decisões de projeto (representação, definição de vizinhança e estratégia de aceitação) impactam diretamente probabilidade de escapar de ótimos locais e a eficiência de busca. Para mitigar os problemas do Hill Climbing puro são frequentemente adotadas técnicas auxiliares: reinicializações aleatórias (random-restart), permissão limitada de movimentos laterais (sideways moves) quando não há melhora, e híbridos com métodos como *simulated annealing* ou busca tabu [3, 2].

## 3 Metodologia

Esta seção descreve a implementação experimental e os procedimentos usados para avaliar as variantes de Hill Climbing no problema das 8 rainhas.

### 3.1 Representação

Como dito anteriormente, adotamos um vetor  $S = (r_1, \dots, r_8)$  com valores inteiros 1 a 8. Adotaremos a representação por permutação (cada linha aparece exatamente uma vez), ou seja, a configuração inicial é uma permutação aleatória de  $\{1, \dots, 8\}$ . Essa representação elimina conflitos por linha automaticamente, restando apenas os conflitos diagonais a serem minimizados.

### 3.2 Vizinhança

Para a representação por permutação, definimos a vizinhança de  $S$  como o conjunto de permutações obtidas trocando as posições (swap) de uma rainha em sua coluna para outra linha. Para  $n$  rainhas, essa vizinhança tem  $\binom{n}{2}$  vizinhos e é suficiente para explorar rearranjos que podem reduzir conflitos diagonais.

### 3.3 Função objetivo

A função objetivo  $f(S)$  é o número de pares de rainhas em diagonal (definida na seção anterior). O algoritmo busca minimizar  $f$ ; uma solução com  $f(S) = 0$  é aceitável (todas as rainhas não se atacam).

### 3.4 Variante do Hill Climbing

Implementaremos e compararemos (pelo menos) as seguintes versões:

1. **Best-improvement:** avaliar todos os vizinhos e escolher aquele com menor  $f$ .
2. **First-improvement:** varrer vizinhos em ordem aleatória e aceitar o primeiro que apresentar melhora.
3. **Random-restart Hill Climbing:** aplicar Hill Climbing (best- ou first-improvement) até convergência; caso não encontre solução ótima, reiniciar com nova permutação aleatória — repetir até um número máximo de reinícios.
4. **Movimentos laterais controlados:** permitir movimentos que não alterem  $f$  (sideways moves) até um limite  $L$  consecutivo, como estratégia para atravessar platôs.

### 3.5 Parâmetros experimentais (valores padrão)

Para experimentos usaremos parâmetros como padrão que podem ser ajustados conforme necessidade:

- número de execuções independentes:  $N = 201$ ;
- máximo de iterações por execução:  $I_{\max} = 1000$ ;
- limite de reinícios no random-restart:  $R_{\max} = 50$ ;
- limite de movimentos laterais (quando habilitado):  $L = 100$ .

### 3.6 Métrica de avaliação

Para cada variante, registraremos:

- taxa de sucesso (fração das execuções que atingiram  $f = 0$ );
- tempo médio até solução (tempo de CPU ou wall-clock);
- número médio de iterações até convergência;
- número médio de reinícios (quando aplicável);
- distribuição do  $f$  final (estatísticas: média, mediana, desvio-padrão).

### 3.7 Procedimento experimental

Para cada combinação de variante e parâmetros:

1. executar  $N$  tentativas independentes, cada uma iniciada com solução gerada aleatoriamente;
2. rodar o Hill Climbing segundo a variante até atingir  $f = 0$  ou até  $I_{\max}$  iterações ou  $R_{\max}$  reinícios;
3. registrar métricas por execução e compor estatísticas agregadas.

Caso o usuário queira entender melhor as opções de execução, é possível executar o arquivo *guia\_rapido.py*. Se assim o fizer, será exibido um menu contendo opções pré-setadas de utilização.

## 4 Resultados e Discussões

Estratégia	Taxa de Sucesso (%)	Tempo médio (s)	$\sigma_{tempo}$ (s)	Iterações	$\sigma_{it}$	Conflitos finais
best-improvement	0.1	0.00424	0.00109	4.15	0.92	1.249
first-improvement	0.2	0.00279	0.00106	6.20	1.80	1.169
best-sideways	0.4	0.04956	0.03660	65.65	48.37	0.741
best-random-restart	1.0	0.01920	0.01610	27.95	23.67	0.000

Tabela 1: Médias e desvios-padrão das métricas por estratégia

Neste trabalho com o problema das 8 rainhas com quatro estratégias de *hill-climbing* (*best-improvement*, *first-improvement*, *best-sideways* e *best-random-restart*) comparamos quatro métricas principais: taxa de sucesso, tempo médio de execução, número médio de iterações e conflitos finais médios. As figuras a seguir resumem os resultados obtidos nos experimentos realizados.

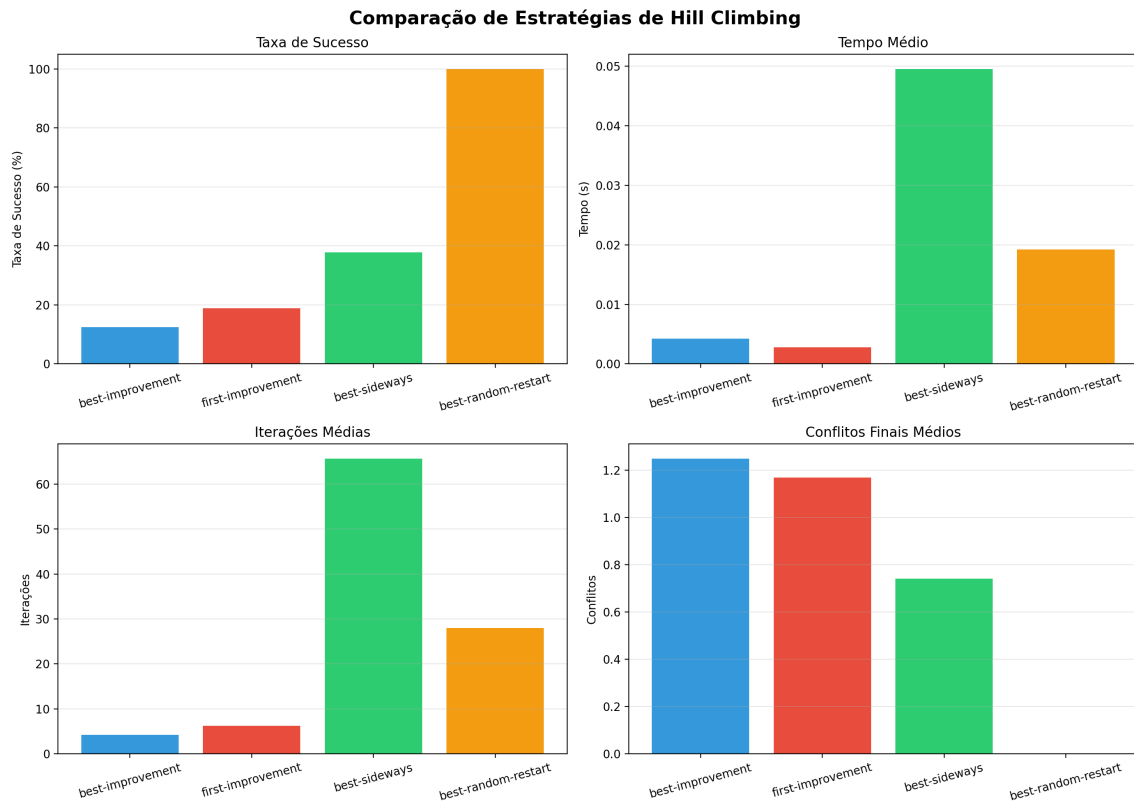


Figura 1: Resumo das quatro métricas comparadas entre as estratégias.

Abaixo discutimos as observações mais relevantes retiradas das figuras e das estatísticas:

## 4.1 Taxa de sucesso:

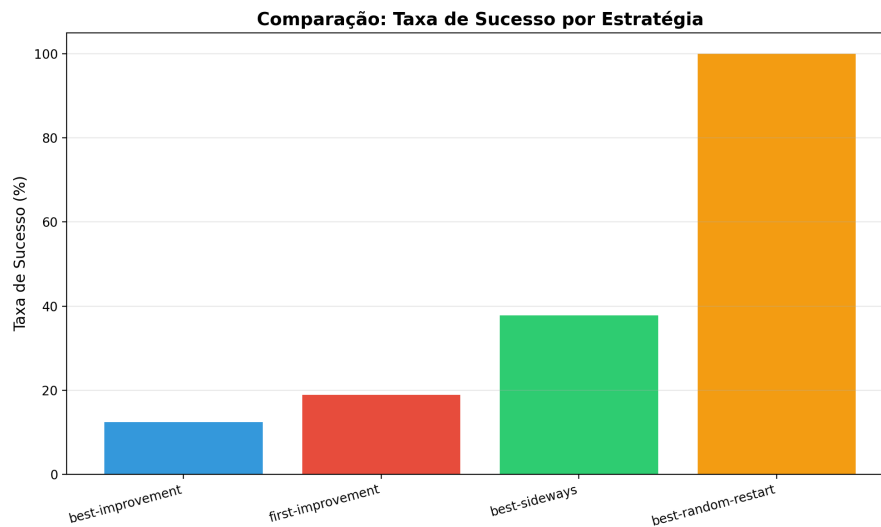


Figura 2: Taxa de sucesso por estratégia (porcentagem de execuções que encontraram solução válida).

Conforme ilustra a Figura 2, a estratégia *best-random-restart* alcançou 100% de sucesso nas execuções realizadas. Em ordem decrescente de sucesso, aparecem *best-sideways* (sucesso moderado), *first-improvement* e, por fim, *best-improvement* (a menor taxa de sucesso). Uma interpretação direta é que mecanismos que introduzem recomeços aleatórios ou que permitem movimentos laterais ajudam o algoritmo a escapar de mínimos locais e planícies do espaço de busca — justamente as armadilhas clássicas do *hill-climbing*.

## 4.2 Número de iterações e variação.

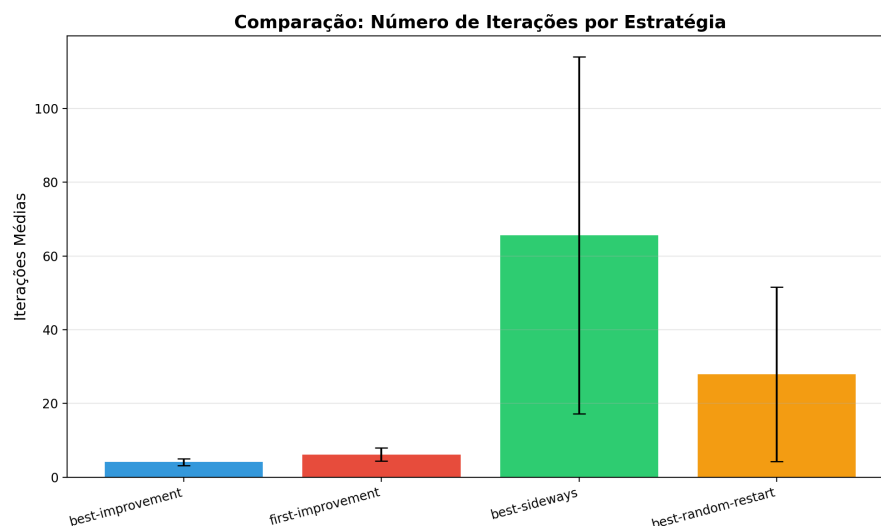


Figura 3: Iterações médias por estratégia (com barras de erro mostrando a variabilidade).

A Figura 3 mostra que *best-sideways* exige muito mais iterações em média e apresenta alta variabilidade (grande desvio padrão). Isso é coerente: permitir movimentos laterais (*sideways*) permite escapar de planícies, mas também pode alongar a busca, já que o

algoritmo percorre regiões com custo inalterado antes de encontrar uma melhoria. A estratégia *best-random-restart* tem um número médio de iterações intermediário com também grande variância (pois depende do número de reinícios e do ponto de partida sorteado), enquanto *best-improvement* e *first-improvement* têm médias de iterações muito menores e menor dispersão.

### 4.3 Tempo de execução.

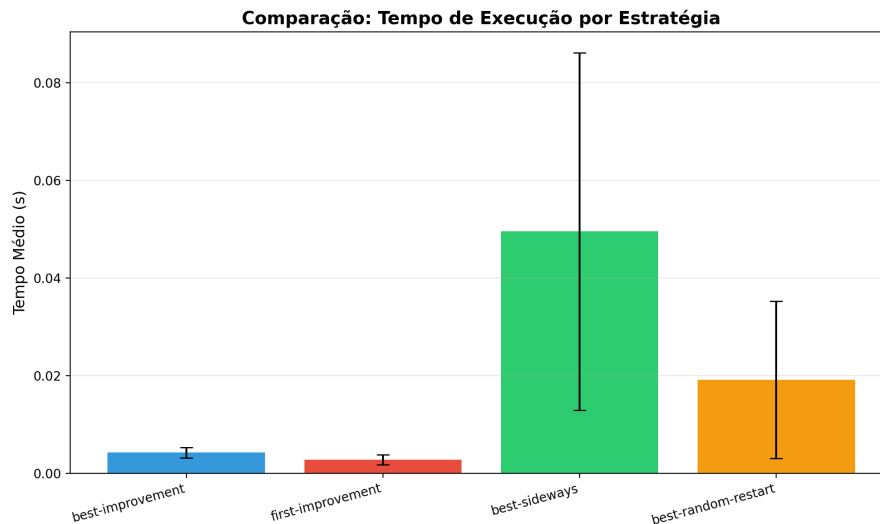


Figura 4: Tempo médio de execução por estratégia (com barras de erro).

A Figura 4 confirma que *best-sideWAYS* é a mais custosa em tempo médio por execução, seguidos por *best-random-restart*. As estratégias *best-improvement* e *first-improvement* são as mais rápidas. Aqui aparecem dois efeitos:

- (i) o tempo está correlacionado com o número de iterações;
- (ii) a implementação de *best-improvement* exige, a cada passo, avaliar todos os vizinhos e escolher o melhor — no entanto, para o caso pequeno das 8 rainhas isso ainda é barato; já *first-improvement* geralmente termina mais rápido porque aceita a primeira melhoria encontrada, reduzindo a quantidade de avaliação por passo.

### 4.4 Conflitos finais.

Nos experimentos o *best-random-restart* obteve, em média, conflitos finais zero (pois sempre encontrou solução), enquanto as estratégias sem reinício apresentam conflitos finais médios maiores — o que significa que muitas execuções terminaram presas em ótimos locais sem solução completa.

### 4.5 Interpretação por estratégia

O *best-random-restart* teve excelente taxa de sucesso (100% nos testes) porque recomençar a busca a partir de diferentes posições reduz dramaticamente a probabilidade de ficar permanentemente preso em mínimos locais. Custo: maior tempo e variância (dependendo de quantos reinícios forem necessários), mas recompensa em confiabilidade para o problema das 8 rainhas.

Já o *best-sideways* ao permitir movimentos laterais aumenta a taxa de sucesso relativamente às versões "puras" (*best/first*), pois favorece a travessia de planícies; porém, isto vem junto com grande aumento no número de iterações e no tempo médio. Alto desvio padrão indica que em algumas execuções a estratégia consegue sair rápido da planície e em outras demora muito.

Ademais, *first-improvement* tem tempo e número de iterações baixos — aceita a primeira melhoria encontrada, o que reduz custo por passo. A taxa de sucesso foi melhor que a do *best-improvement* provavelmente porque a escolha menos determinística ajuda a evitar quedas previsíveis em mínimos locais (empates e ordem de varredura introduzem uma pseudo-aleatoriedade útil).

Por fim, *best-improvement*, apesar de sempre escolher o melhor vizinho, apresentou a menor taxa de sucesso. Explicação provável: o critério sempre-ótimo a curto prazo favorece quedas em mínimos locais que não têm vizinhos melhores, tornando a fuga difícil sem mecanismos adicionais (*sideways* ou *restarts*). É uma estratégia mais "cupada" em busca localmente ótima, mas globalmente conservadora.

Os resultados mostram o clássico trade-off entre confiabilidade (taxa de sucesso) e custo computacional (tempo/iteraões): estratégias com mais aleatoriedade (*restart*) ou permissividade (*sideways*) aumentam sucesso às custas de tempo e variância. Para o problema das 8 rainhas (espaço de busca relativamente pequeno), *best-random-restart* é a escolha mais prática quando o objetivo é garantir solução; para instâncias maiores (*N* maior), recomenda-se para trabalhos futuros combinar *first-improvement* com um número controlado de reinícios ("restart budget") — boa relação custo/benefício, limitar o número máximo de movimentos *sideways* para evitar explorações excessivas e avaliar métodos estocásticos como *simulated annealing* ou algoritmos populacionais (genéticos) caso o espaço aumente muito.

Em resumo: *best-random-restart* garantiu a solução em todas as execuções; *best-sideways* aumenta consideravelmente a exploração (maiores iterações/tempo) e melhora a chance de sucesso; *first-improvement* é rápido e razoavelmente eficaz; *best-improvement* tende a ficar preso em mínimos locais e apresenta a pior taxa de sucesso entre as quatro estratégias testadas.

## 5 Conclusões

Esse trabalho tornou possível a avaliação experimental de desempenho de diferentes variantes do algoritmo de *Hill Climbing* aplicadas ao problema das 8 rainhas. A abordagem utilizada permitiu observar que os algoritmos *best improvement* e *first improvement* apresentaram tempos médios menores, mas ficaram presos em platôs e mínimos locais, levando a conflitos finais semelhantes e, de forma mais expressiva, a taxa de sucesso nula.

O algoritmo de *best sideways*, apesar de tempo médio maior e quantidade de iterações tão altas quanto o *best random restart*, não apresentou soluções viáveis e, mais ainda, teve resultados próximos dos anteriores. Isso evidenciou que esse algoritmo, apesar de ter chances reais de encontrar uma solução, não conseguiu tal feito e, pior ainda, sequer foi capaz de diminuir os conflitos médios.

Em contraste, a variante *best random restart* obteve desempenho superior, apresentando menor número médio de conflitos finais e sendo a única capaz de encontrar qualquer solução válida, ainda que com custo computacional mais elevado. Isso demonstrou, de



forma clara, a importância de uma implementação capaz de lidar com platôs e mínimos locais: enquanto abordagens puramente determinísticas ficaram presas em mínimos locais ou platôs, essa estratégia com reinicializações aleatórias aumentou a capacidade de exploração do espaço de busca, e com isso teve alguma chance real de atingir uma solução. Dessa forma, a introdução de aleatoriedade, mesmo que simples, se revelou como um fator decisivo para a eficiência de métodos de otimização baseados em *Hill Climbing*.

## 6 Créditos e Autoria:

Essa seção tem-se como foi feita a divisão de tarefa dentro do grupo, adjunto aos recursos externos utilizados (que estarão explicitados dentro das referências).

### 6.1 João Pedro Rodrigues Silva (Autor)

- Implementação inicial dos métodos de mapeamento das rainhas (baseado nos materiais disponibilizados pelo Professor). [6]
- Documentação inicial do relatório (utilizando-se de fontes bibliográficas para embasamento teórico). [2, 3, 4]
- Questionamentos e elucidações com o Professor orientador do projeto.

### 6.2 Jader Oliveira Silva (Autor)

- Revisão dos algoritmos de busca implementados e aprimorando as métricas utilizadas. [1]
- Revisão e finalização do relatório acerca do trabalho.
- Questionamentos e elucidações com o Professor orientador do projeto.

**Uso de IA:** A ferramenta utilizada foi o ChatGPT para revisão textual ortográfica e estilização de tabelas. Nenhuma parte de código de algoritmos foi gerada por IA.

**Declaração:** Confirmamos que o código entregue foi desenvolvido pela equipe, respeitando as políticas da disciplina.

## Referências

- [1] João Pedro Rodrigues e Jader Oliveira, *MAZE-SEARCH-AND-HILL-CLIMBING*, repositório no GitHub(Jottynha), 2025. Disponível: <https://github.com/Jottynha/MAZE-SEARCH-AND-HILL-CLIMBING>. Acesso em: 19 de outubro de 2025.
- [2] S. Russell e P. Norvig, *Artificial Intelligence: A Modern Approach*, 4<sup>a</sup> ed. Disponível em: <https://aima.cs.berkeley.edu/>. Acesso em: 19 de outubro de 2025.
- [3] E. H. L. Aarts & J. K. Lenstra (eds.), *Local Search in Combinatorial Optimization*, 2<sup>a</sup> ed., Princeton University Press, 2003. Disponível em: <https://doi.org/10.1515/9780691187563>. Acesso em: 19 de outubro de 2025.
- [4] *Hill climbing* — Wikipédia. Disponível em: [https://en.wikipedia.org/wiki/Hill\\_climbing](https://en.wikipedia.org/wiki/Hill_climbing). Acesso em: 19 de outubro de 2025.

- [5] A. Thiago, *Capítulo III: Conhecimento*, Sistema Integrado de Gestão de Atividades Acadêmicas. CEFET-MG. Acesso em: 19 de outubro de 2025.
- [6] A. Thiago, *Capítulo IV: Busca*, Sistema Integrado de Gestão de Atividades Acadêmicas. CEFET-MG. Acesso em: 19 de outubro de 2025.