CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CÂMPUS V — DIVINÓPOLIS

Trabalho 2

Disciplina: Inteligência Artificial
Professor: Tiago Alves de Oliveira

Aluno:
Lucas Cerqueira Portela

Divinópolis — MG
30 de outubro de 2025

1 Introdução

O presente trabalho tem como objetivo investigar o desempenho do algoritmo Hill Climbing (Subida de Encosta) na resolução de problemas de otimização, utilizando como estudo de caso o Problema das Oito Rainhas. Foram desenvolvidas e comparadas duas variações do algoritmo: uma que emprega reinício aleatório e outra que permite movimentos laterais. Ambas têm como propósito mitigar as limitações inerentes à busca local pura, que frequentemente sofre com a estagnação em ótimos locais ou regiões planas do espaço de busca.

Diferentemente de abordagens exatas como backtracking, o Hill Climbing busca soluções de forma heurística, explorando iterativamente estados vizinhos e favorecendo aqueles que reduzem o número de conflitos. Essa estratégia é particularmente útil em problemas onde o espaço de busca é extenso e a solução exata se torna computacionalmente inviável. Assim, o presente estudo propõe não apenas a implementação do algoritmo, mas também uma avaliação empírica de seu comportamento frente a diferentes estratégias de diversificação e intensificação da busca.

Para tanto, foram realizadas múltiplas execuções independentes com parâmetros controlados, a fim de medir métricas como tempo médio de execução, taxa de sucesso, número de reinicializações e número médio de conflitos. A partir desses resultados, foram gerados gráficos comparativos que auxiliam na interpretação do desempenho relativo de cada variação.

Desse modo, o trabalho busca compreender de que forma ajustes simples no mecanismo de exploração local podem impactar a eficiência, estabilidade e robustez do algoritmo, oferecendo uma análise crítica sobre a aplicabilidade de métodos heurísticos em problemas clássicos de otimização combinatória.

2 O Problema das Oito Rainhas

2.1 Definição

O problema das *Oito Rainhas* consiste em posicionar oito rainhas num tabuleiro de xadrez 8×8 de forma que nenhuma rainha ataque outra. Em termos formais, cada rainha ocupa uma célula do tabuleiro, e duas rainhas se atacam se estiverem na mesma linha, na mesma coluna, ou na mesma diagonal. O objetivo é encontrar uma configuração em que não haja ataques entre quaisquer pares de rainhas.

2.2 Generalização

O problema é comumente generalizado para n rainhas em um tabuleiro $n \times n$. O problema *n-queens* é um problema clássico de busca combinatória e serve como benchmark para técnicas de busca local e algoritmos evolutivos.

2.3 História e relevância

O problema das 8 rainhas foi proposto em 1848 e tornou-se um problema-padrão de demonstração em inteligência artificial, algoritmos de backtracking, pesquisa local e heurísticas. Apesar de simples de enunciar, o espaço de busca cresce exponencialmente com n, tornando interessante avaliar estratégias heurísticas quando n cresce.

3 Representação e Função Objetivo

3.1 Representação padrão

Uma representação comum e compacta é usar um vetor $Q = [q_1, q_2, \ldots, q_n]$, em que q_i indica a linha ocupada pela rainha na coluna i. Assim:

- Cada coluna tem exatamente uma rainha (reduzindo simetria).
- $q_i \in \{1, \ldots, n\}$.

3.2 Função de avaliação (número de conflitos)

Uma função objetivo simples é contar o número total de pares de rainhas que se atacam. Para a representação acima:

$$\text{conflitos}(Q) = \sum_{1 \le i < j \le n} \mathbf{1}(q_i = q_j \ \vee \ |q_i - q_j| = |i - j|)$$

onde $\mathbf{1}(\cdot)$ é a função indicadora (1 se a condição é verdadeira, 0 caso contrário). Um estado ótimo tem $\text{conflitos}(Q) = 0$.

3.3 Complexidade do cálculo de conflitos

Calcular conflitos por pares tem custo $O(n^2)$ se feito ingênua e completamente. Em implementações eficientes, pode-se manter contadores por linha e por cada diagonal principal e secundária, reduzindo custo de atualização a $O(1)$ por movimento (após manutenção de estruturas auxiliares).

4 Algoritmos Baseados em Busca Local

A busca local parte de uma solução inicial e tenta melhorá-la através de movimentos locais, sem explorar toda a árvore de busca (como backtracking). Abaixo descrevemos algoritmos que foram implementados no projeto.

4.1 Hill Climbing (Subida de Encosta)

4.1.1 Ideia

Partindo de uma configuração inicial Q (geralmente aleatória, respeitando uma rainha por coluna), Hill Climbing examina a vizinhança de Q (por exemplo, todas as mudanças de posição de uma rainha dentro de sua coluna) e escolhe o movimento que produz a maior melhoria (redução do número de conflitos). O processo se repete até que não exista um vizinho melhor (pico local) ou até atingir uma solução ótima (0 conflitos).

4.1.2 Vizinhança

Um movimento básico: selecionar uma coluna i e mover a rainha para uma linha $r \neq q_i$. Assim, cada coluna tem $n - 1$ posições possíveis, resultando em uma vizinhança de tamanho $n(n - 1)$ (considerando todas as colunas).

4.1.3 Limitações

Hill Climbing pode ficar preso em picos locais (estados que não têm vizinhos melhores, mas não são soluções). Dependendo do estado inicial, pode falhar em encontrar soluções.

4.2 Movimento Lateral (Sideways Moves)

4.2.1 Motivação

Às vezes o algoritmo chega a um plateau (vizinhança com mesmos valores de avaliação). Permitir movimentos laterais — mover para um vizinho com avaliação igual — pode ajudar a atravessar plateaus e alcançar regiões onde uma melhoria é possível.

4.2.2 Implementação

Adiciona-se uma regra que permite movimentos que não aumentam a função objetivo e, em particular, que aceitam movimentos com $\Delta = 0$ (mesmo número de conflitos), até um limite de movimentos laterais consecutivos para evitar loops infinitos.

4.3 Reinício Aleatório (Random Restart)

4.3.1 Motivação

Como Hill Climbing depende do estado inicial, uma estratégia simples para evitar picos locais é executar Hill Climbing várias vezes a partir de estados iniciais aleatórios até encontrar uma solução (ou até atingir um limite de reinícios).

4.3.2 Parâmetros importantes

- `max_restarts`: número máximo de reinícios permitidos.
- `max_sideways` e `max_steps`: controlam a busca local em cada tentativa.

5 Análise Empírica e Métricas

5.1 Métricas

- **Taxa de sucesso**: proporção de tentativas que encontraram solução (0 conflitos) dentro de limites predefinidos.
- **Tempo médio de execução**: tempo médio gasto para encontrar solução (ou até o limite).
- **Número médio de iterações**: média de passos de busca local até encontrar solução ou parar.
- **Número médio de reinícios**: para Random Restart, média de reinícios até sucesso.
- **Conflitos médios finais**: quando não alcançou solução, conflito médio final.

5.2 Protocolos experimentais

- Rodar cada configuração (par de parâmetros) várias vezes (k) com sementes aleatórias distintas.
- Registrar para cada execução: sucesso/falha, tempo, iterações, reinícios, melhor e último valor de conflitos.

6 Organização do Repositório

A estrutura do projeto foi organizada conforme descrito abaixo:

```
trabalho2/
 src/
   eight_queens.py      # Funções principais do problema das 8 rainhas
   hill_climbing.py     # Implementação do algoritmo Hill Climbing
   gerar_graficos.py    # Geração de gráficos comparativos
   lateral.py           # Execução do Hill Climbing com movimento lateral
   reinicio.py          # Execução do Hill Climbing com reinício aleatório

 results/                # Pasta de saída contendo os resultados gerados
   lateral.txt          # Log completo das execuções do algoritmo com movimento latera
   reinicio.txt         # Log completo das execuções do algoritmo com reinício
   results.csv          # Resultados brutos obtidos
   conflitos_medios.png # Gráfico comparando o número médio de conflitos entre algorit
   iteracoes_medias.png # Gráfico comparando o número médio de iterações/reinícios
   taxa_sucesso.png     # Gráfico da taxa de sucesso de cada algoritmo
   tempo_medio.png      # Gráfico comparando o tempo médio de execução

 requirements.txt        # Dependências necessárias para executar o projeto
 README.md               # Descrição geral do projeto
```

7 Metodologia

A solução foi implementada de forma modular no diretório `src/`, conforme descrito abaixo:

- **eight_queens.py**: Responsável pela geração de tabuleiros iniciais e exibição visual do estado das rainhas.
- **hill_climbing.py**: Implementa o algoritmo Hill Climbing, com suporte para movimentos laterais e reinícios aleatórios.
- **lateral.py**: Executa o algoritmo Hill Climbing com movimentos laterais e armazena resultados em `lateral.txt`.
- **reinicio.py**: Executa o Hill Climbing com reinício aleatório, salvando os resultados em `reinicio.txt`.
- **gerar_graficos.py**: Lê o arquivo `results.csv` e gera gráficos comparativos de desempenho.

7.1 Execução dos Experimentos

Foram realizadas 100 execuções independentes para cada variação do algoritmo:

- **Hill Climbing com Reinício Aleatório**: permite reiniciar o tabuleiro após cair em um ótimo local, até um máximo de 100 reinícios.
- **Hill Climbing com Movimentos Laterais**: permite explorar estados vizinhos com o mesmo valor heurístico, evitando estagnações locais.

Os resultados de cada execução foram salvos no arquivo `results.csv`, contendo as seguintes métricas:

- Tempo de execução (s)
- Número de conflitos finais
- Número de iterações/reinícios
- Taxa de sucesso (em %)

8 Resultados Obtidos

Após a coleta dos dados, foram gerados gráficos comparativos para as principais métricas de desempenho dos dois algoritmos.

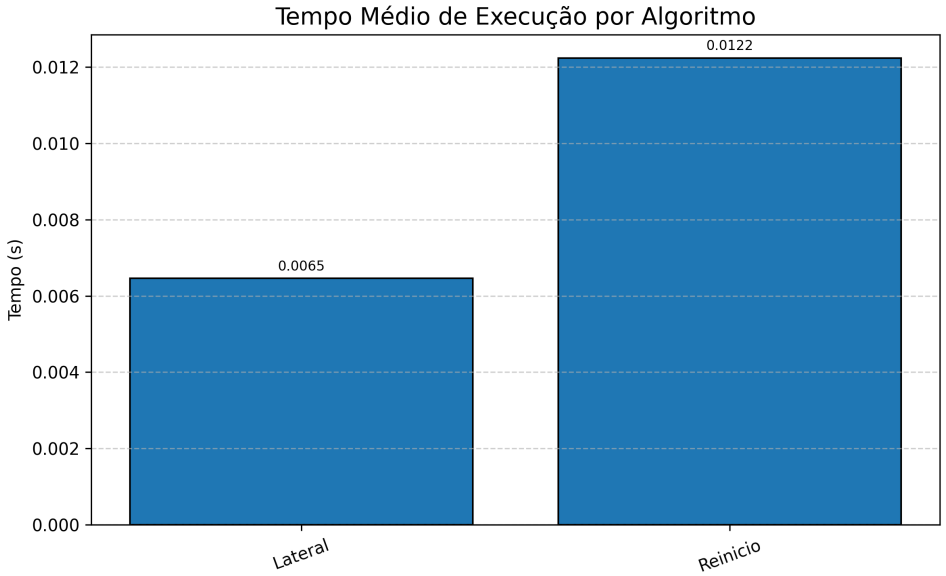

Figura 1: Comparação do tempo médio de execução entre os algoritmos.

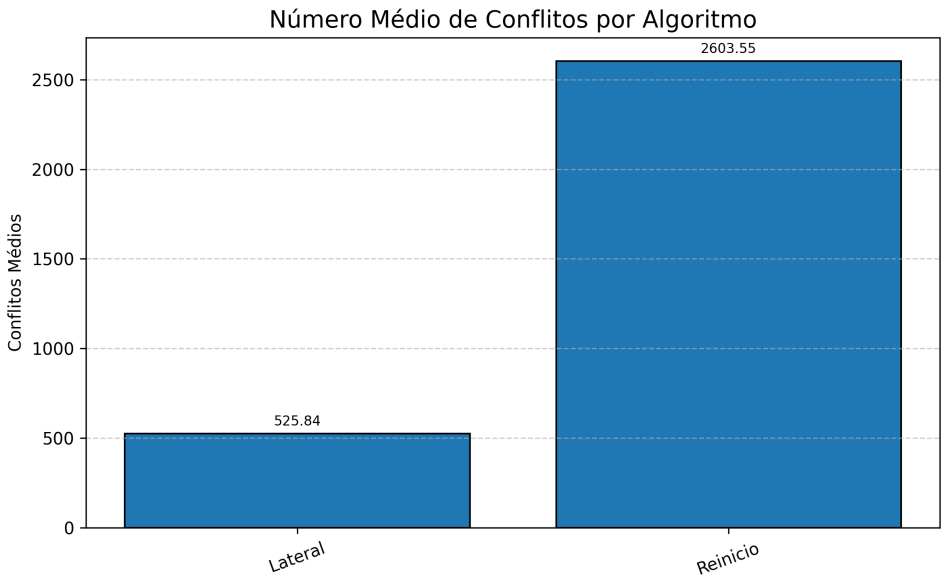

Figura 2: Comparação do número médio de conflitos finais.

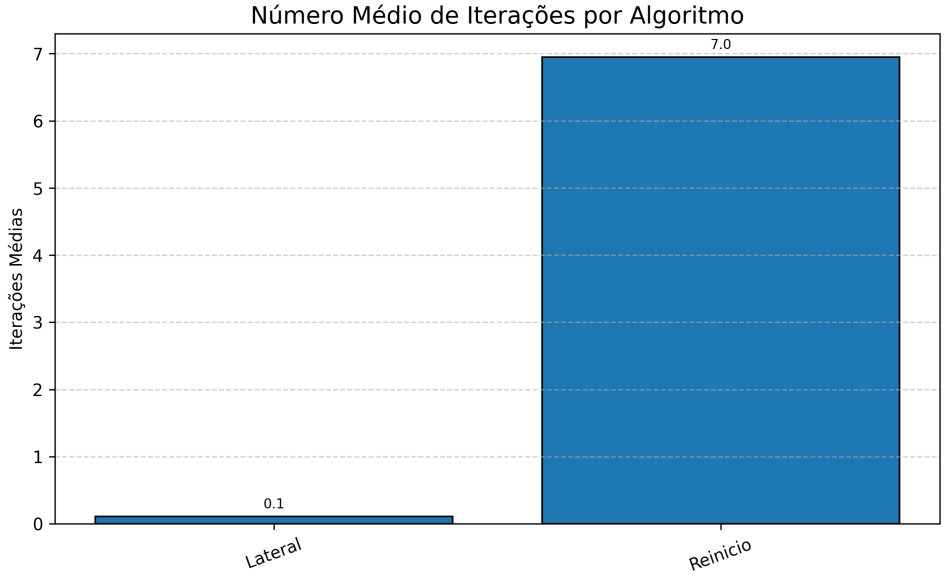

Figura 3: Comparação do número médio de iterações e reinícios.

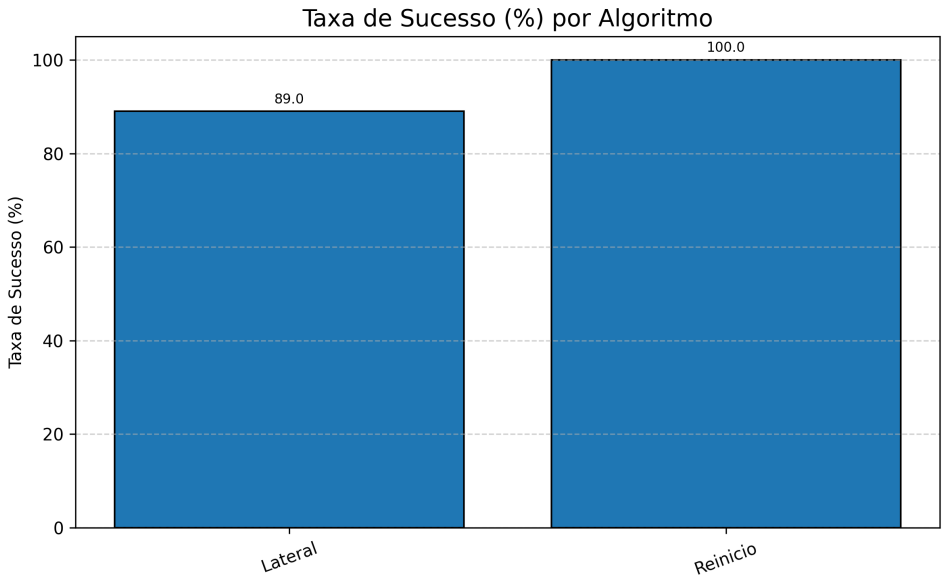

Figura 4: Taxa de sucesso de cada algoritmo.

8.1 Análise dos Resultados

Após a coleta e organização dos dados experimentais, foram gerados gráficos comparativos representando as principais métricas de desempenho entre as duas variações do algoritmo.

Os resultados demonstram que o Hill Climbing com reinício aleatório obteve uma taxa de sucesso próxima a 100%, corroborando a teoria de que múltiplos pontos de partida aleatórios aumentam significativamente a probabilidade de alcançar o ótimo global. Essa abordagem, embora apresente maior custo temporal devido às reinicializações, mostrou-se mais robusta e consistente na obtenção de soluções sem conflitos.

Por outro lado, o Hill Climbing com movimentos laterais apresentou desempenho mais eficiente em termos de tempo médio de execução, mas com menor taxa de sucesso. Esse comportamento ocorre porque os movimentos laterais permitem escapar de regiões planas, porém não asseguram uma exploração ampla do espaço de busca. Assim, o algoritmo tende a convergir rapidamente, mas com maior risco de permanecer em estados subótimos — um resultado coerente com o comportamento previsto teoricamente.

O gráfico de iterações e reinicializações indica que, apesar do custo adicional dos reinícios, essa estratégia proporciona melhor equilíbrio entre exploração global e sucesso na convergência. Por fim, o gráfico de tempo médio evidencia a eficiência do método com movimentos laterais, embora às custas da consistência dos resultados.

De modo geral, os resultados empíricos condizem com as expectativas teóricas. O reinício aleatório promove maior cobertura do espaço de busca e aumenta a confiabilidade na obtenção de soluções ótimas, enquanto os movimentos laterais priorizam a eficiência temporal, ainda que com menor taxa de acerto. Essa dualidade evidencia o clássico dilema entre eficiência e robustez em algoritmos de busca local.

9 Conclusão

O presente trabalho apresentou a implementação e análise do algoritmo, aplicado ao Problema das Oito Rainhas, abordando duas variações distintas: com reinício aleatório e

com movimentos laterais. A partir dos resultados obtidos, observou-se que a versão com reinício aleatório alcançou desempenho superior em termos de taxa de sucesso, confirmando sua maior capacidade de escapar de ótimos locais e atingir o estado ótimo global. Já a versão com movimentos laterais destacou-se pela eficiência temporal, oferecendo soluções em menor tempo, embora com menor consistência.

A análise empírica demonstrou que os resultados obtidos são compatíveis com as previsões teóricas sobre o comportamento do Hill Climbing, evidenciando a importância de se equilibrar exploração e intensificação em algoritmos de busca local.

Como perspectivas futuras, sugere-se a incorporação de outras técnicas de busca, como Simulated Annealing, podem combinar de forma mais eficiente os mecanismos de diversificação e intensificação. Essa ampliação permitiria uma análise comparativa ainda mais rica, explorando a interação entre parâmetros estocásticos, desempenho computacional e qualidade das soluções geradas.

10 Referências

- Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* 3ª edição. Pearson.
- Wikipedia: *Eight queens puzzle.* Disponível em: `https://en.wikipedia.org/wiki/Eight_queens_puzzle`

Ambiente de Execução

CPU: 13 geração Intel(R) Core(TM) i7-13620H (2.40 GHz)

Memória RAM: 32GB 5200MHz

GPU: RTX 3050 6GB

Créditos e Declaração de Autoria

Autor: Lucas Cerqueira Portela
Atividades desenvolvidas: Implementação dos algoritmos Hill Climbing com reinício e movimento lateral, criação do relatório e repositório do projeto.
Uso de IA: A ferramenta ChatGPT foi utilizada para revisão textual, aprimoramento da redação acadêmica do relatório e auxilío na criação dos códigos implementados.
.