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

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 8×8 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/minimos 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 $S = (r_1, r_2, \ldots, r_8)$ 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 $\{1, \ldots, 8\}$, 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:

$$f(S) \;=\; \sum_{1 \le i < j \le 8} \mathbf{1}\Big(|r_i - r_j| = |i - j|\Big) \tag{1}$$

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 anteriomente, adotamos um vetor $S = (r_1, \ldots, r_8)$ 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 $\{1, \ldots, 8\}$. 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 duas colunas quaisquer. 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: $N = 200$;
- 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:

Tendo como base o que foi desenvolvido durante a metodologia, nesta etapa serão expostos os resultados obtidos pela aplicação das diferentes métricas, bem como a discussão que envolve cada uma delas.

Em primeira instância, partindo de cada uma das métricas individuais, é possível construir a seguinte tabela:

Estratégia	T. de Sucesso	Tempo Médio	Desvio Padrão (Tempo)	Iter Médias	Desvio Padrão (Iter)	Reinícios Médios	Conflitos Médios
best-improvement	0,0000	0,0014	0,0007	2,867	0,618	1,000	4,367
first-improvement	0,0000	0,0008	0,0006	3,600	0,952	1,000	4,400
best-sideways	0,0000	0,0187	0,0010	52,867	0,618	1,000	4,367
best-random-restart	0,0333	0,0184	0,0012	52,567	2,704	19,933	1,600

Tabela 1: Resultados comparativos entre diferentes estratégias de resolução.

Esta tabela nos mostra que os algoritmos tem uma taxa de sucesso nula ou bem próxima disso, isto é, de 0%. Contudo, é possível notar que as diferentes abordagens apresentam mudanças significativas em relação as outras métricas, variações que impactam diretamente na performance dos algoritmos se, para essa performance, levar-se o tempo de execução como parâmetro.

Com a intenção de realizar uma análise mais clara, plotou-se alguns gráficos:

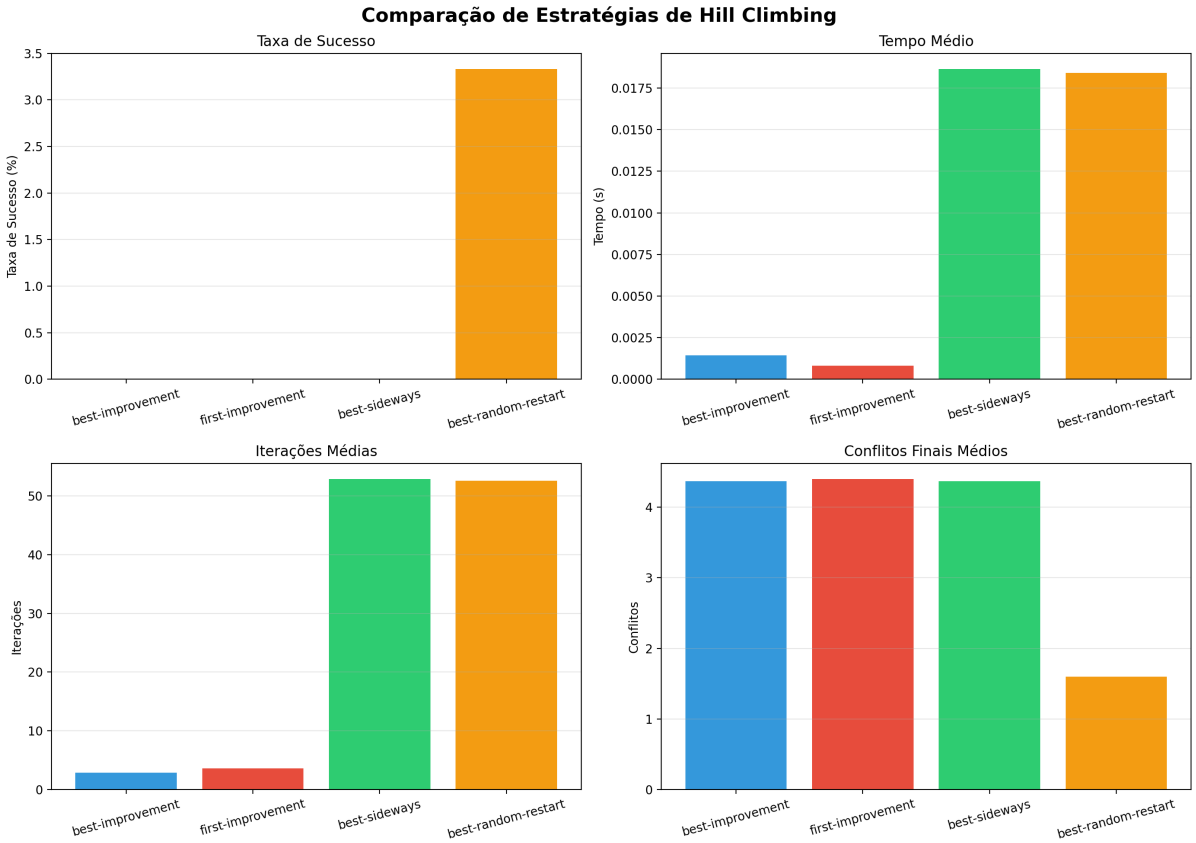

Figura 1: Gráfico comparativo contendo diferentes métricas.

Este gráfico mostra, de forma mais clara, as variações entre as métricas para os diferentes algoritmos. Por meio disso é possível reafirmar que o único algoritmo que, dadas as condições iniciais mencionadas previamente, encontrou alguma solução foi o *best random-start* que, entre os casos de teste, encontrou a solução em 3.4% das vezes.

Além disso, é possível observar a escalada do tempo médio, que para os algoritmos *best improvement* e *first improvement* é baixo, executando em tempo médio mais de 10 vezes inferior em relação aos algoritmos de *best sideways* e *best random-restart*, que levam mais tempo em execução.

A análise das iterações médias traz certa luz à mudança substancial no tempo de execução entre os algoritmos, haja vista que é possível notar que aqueles que possuem maiores interações são, também, os algoritmos acompanhados do maior tempo de execução. É possível, assim apontar o número de iterações como um dos fatores que eleva o tempo médio de execução.

Por fim, olhando para o gráfico de Conflitos Finais Médios, é possível notar uma tendência forte entre os algoritmos *best improvement*, *first improvement* e *best sideways* de apresentarem valores muito próximos para essa métrica. Isso demonstra um comportamento bastante similar, que pode ser explicado pela natureza individual destes algoritmos, que tendem a ficar presos em platôs ou mínimos locais. Nestes casos, por mais que exista uma diferença em outras métricas, seus comportamentos individuais tendem a gerar soluções semelhantes, principalmente porque, desacompanhados de soluções mais elegantes, todos tendem aos mesmos finais. Esse fator explica exatamente o que é possível ver graficamente: conflitos finais quase uniformes.

Em relação ao algoritmo *best random restart*, os conflitos finais médios apresentam valores inferiores, com uma média de casos mais que 50% inferior aos demais. Aqui, isso pode ser explicado justamente pela diferença que esse algoritmo apresenta em relação aos demais: quantidade de reinícios médios muito superior. Essa quantidade de reinícios é essencialmente a forma como essa abordagem consegue fugir dos platôs e mínimos locais, pois sempre que o algoritmo se depara com eles, realiza um reinício para um dos pontos anteriores, e isso permite que, a cada reinício, a chance de encontrar uma solução aumente. Isto é, exatamente, a justificativa para que esse algoritmo tenha sido o único capaz de determinar uma solução dada uma quantidade inicial de tentativas.

Em termos práticos, embora as variantes determinísticas sejam, mais rápidas, elas não foram capazes de garantir qualquer solução viável. Por outro lado, o *best random restart* apresenta maior custo temporal, maior valor de reinícios e quantidade de iterações maiores, mas aumenta a entrega resultados com conflitos finais bem inferiores, além da chance de encontrar uma solução de fato. Isso ilustra bem a relação clássica entre custo computacional e qualidade da solução, uma discussão presente entre os diversos algoritmos computacionais, sejam de busca, ordenação, etc.

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 à 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 restarts* 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 metódos 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 finalizamento 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*, 4ª 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*, 2ª 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`. 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.