Centro Federal de Educação Tecnológica de Minas Gerais
Engenharia de Computação - Inteligência Artificial

Trabalho 2 de IA: Problema das 8 Rainhas

Pedro Auguto Gontijo Moura
Samuel Silva Gomes

Prof. Thiago Alves de Oliveira

Divinópolis/MG
Outubro de 2025

Introdução

O problema das 8 Rainhas consiste em posicionar oito rainhas em um tabuleiro de xadrez 8x8, de forma que nenhuma delas possa atacar outra. Isso significa que é necessário garantir que não haja duas rainhas na mesma linha, coluna ou diagonal. Apesar de sua formulação simples, o problema é um exemplo clássico de otimização combinatória, pois envolve um espaço de estados amplo e sujeito a armadilhas como máximos locais.

Para abordar esse tipo de problema, técnicas de otimização local são frequentemente utilizadas. Entre elas, destaca-se o algoritmo de *Hill Climbing*, que parte de uma solução inicial e tenta melhorá-la passo a passo, escolhendo vizinhos que reduzam o número de conflitos entre as rainhas. Esse método, no entanto, pode ficar preso em situações onde não há melhora aparente, conhecidas como máximos locais ou platôs.

Para lidar com essas limitações, foram implementadas duas variações do Hill Climbing: a primeira permite movimentos laterais, dentro de um limite, e a segunda utiliza reinícios aleatórios (Random-Restart) para explorar novas soluções quando o algoritmo fica preso, com base em uma versão padrão do *Hill Climbing*. O desempenho dessas abordagens é avaliado por meio de métricas como tempo de execução, número de reinícios e taxa de sucesso na obtenção de soluções válidas.

Além da implementação, o estudo discute o impacto de características do espaço de busca, como a presença de platôs, e como estratégias adicionais podem contribuir para aumentar a probabilidade de encontrar soluções válidas. Dessa forma, busca-se compreender melhor o comportamento do *Hill Climbing* e suas variações no contexto de problemas de otimização.

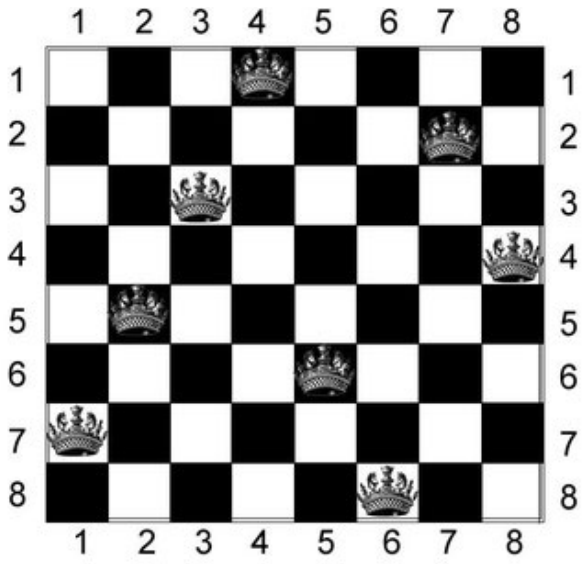

Figura 1: Uma das soluções para o problema das 8 rainhas

Objetivos

- Implementar uma solução para o problema das 8 rainhas usando *Hill Climbing*;
- Comparar os métodos de movimentos laterais (*sideaways*) e *Random-Restart* na eficiência para resolver o problema.

Metodologia

Materiais Utilizados

- Python 3.11.9 (linguagem de programação)
- Github (repositório de código) [1]
- Visual Studio Code (editor de texto)

Heurística Geral do problema:

A heurística utilizada nesse problema (e o objetivo que precisa ser atendido pelo *Hill Climbing*) é a de que, para um tabuleiro T com 8 rainhas, o número de pares conflitantes t precisa ser tal que $t = 0$ para n passos que são realizados no tabuleiro. Dessa forma, isso quer dizer que o tabuleiro precisa estar com as rainhas ordenadas de forma que nenhuma esteja em conflito com outra após n modificações.

Variações utilizadas:

Movimentos laterais (*sideaways*):

Durante a execução do *Hill Climbing* tradicional, ao invés de realizar uma "subida de melhorias", o problema pode fazer com que se chegue em um platô, que é um cenário descrito como uma "área plana", em que o algoritmo não consegue realizar melhorias e pode ficar vagando pelo problema sem conseguir fazer um progresso significativo para a solução.

Pensando nessa situação, o algoritmo *Hill Climbing* com movimentos laterais (*sideaways*) é uma versão modificada do algoritmo original em que são permitidos movimentos laterais para estados do problema com o mesmo valor na tentativa de sair do platô, com base em um limite de movimentos laterais permitidos para evitar que o algoritmo vague eternamente dentro de um platô sem saída.

Random-Restart:

Para um determinado problema, o algoritmo pode acabar iniciando em uma posição que o desfavorece, seja por ele acabar chegando em um platô que impede melhorias e o faz vagar eternamente nele ou por causa de máximos locais que, graças à tendência do *Hill Climbing* de sempre procurar um pico, podem fazer com que ele fique preso em uma solução melhor que as dos vizinhos, mas que não é a máxima global.

Portanto, para lidar com essa situação, é usado o *Hill Climbing* com *Random-Restart*. Essa é uma modificação no algoritmo original que aplica *restarts* em pontos aleatórios. Sempre que o algoritmo fica preso em uma solução, ele é reiniciado a partir de um novo ponto inicial, de forma que ele possa acabar evitando de ter que lidar com platôs e máximos locais durante a execução.

No problema das 8 rainhas, o *Random-Restart* acontece de tal forma que, quando o algoritmo chega em um máximo local (quando a posição atual é melhor que todos os seus vizinhos, mas não é o máximo global), o reinício do problema envolve gerar um novo posicionamento inicial das rainhas no tabuleiro, de forma que o algoritmo tenta novamente e reinicia esse ciclo até encontrar um arranjo inicial de rainhas em que ele não caia em platôs ou máximos locais e chegue a um cenário sem conflitos entre as rainhas.

Geração de Resultados:

Foram efetuados 100 testes (*trials*) diferentes para cada método testado. Em cada teste, um tabuleiro inicial diferente é gerado, mas todos seguem a mesma *seed*, de forma que sempre são gerados os mesmos 100 tabuleiros, a fim de se padronizar a replicação do experimento. Por fim, é realizada uma média da performance do método com base em cada métrica utilizada para que seja possível analisar um desempenho geral do método.

Limites utilizados:

- Para o método de *Hill Climbing* com movimentos laterais, foram permitidos até 100 movimentos laterais, além de um máximo de 1000 iterações sobre o problema;
- Para o método de *Hill Climbing* com *Random-Restart*, foram permitidos até 50 *restarts*, com até 1000 iterações por *restart*.

Resultados e Conclusões

Distribuição de laterais vs. Distribuição de reinícios

A seguir, as Figuras 2 e 3 apresentam, respectivamente, a distribuição de movimentos laterais (*sideways*) e do número de reinícios (*Random-Restart*), em que ambas as métricas foram registradas ao longo das mesmas 100 execuções:

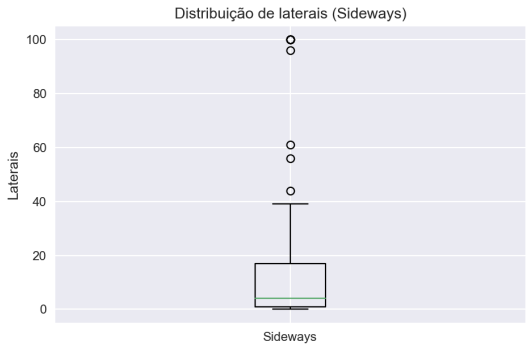

Figura 2: Boxplot da distribuição de laterais do método *Sideways*.

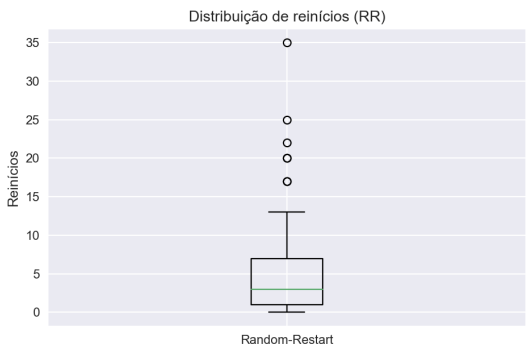

Figura 3: Boxplot da distribuição de reinícios do método *Random-Restart*.

Observa-se que, no método dos movimentos laterais, a mediana é baixa e a maior parte dos testes concentra-se em poucos movimentos, embora existam execuções com valores elevados devido à presença de platôs. Já no método de reinícios, a mediana também é baixa, indicando que a maioria das soluções foi alcançada com poucos reinícios, havendo poucos casos com valores altos.

De modo geral, o *sideways* tende a gastar mais tempo em uma única tentativa, enquanto o *Random-Restart* distribui o esforço em várias tentativas curtas. Assim, o primeiro é mais eficiente em casos simples, e o segundo é mais robusto em casos complexos.

Análise dos passos médios

A Figura 4 apresenta o número médio de passos por algoritmo, com desvio padrão, enquanto a Figura 5 mostra a distribuição completa dessas medidas:

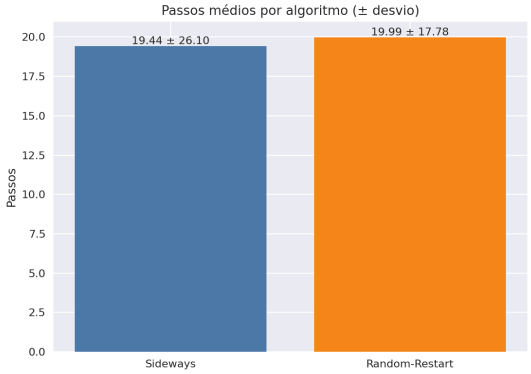

Figura 4: Gráfico de comparação dos passos médios por algoritmo.

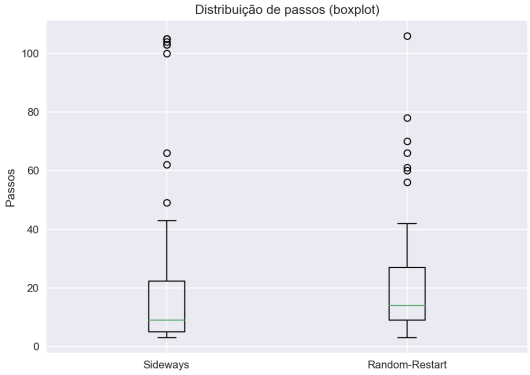

Figura 5: Boxplot comparativo da distribuição de passos dos métodos.

Observa-se que ambos os métodos possuem médias próximas — cerca de 19,44 passos para o *sideways* e 19,99 para o *Random-Restart* — indicando desempenho semelhante em termos de quantidade média de passos.

Contudo, o desvio padrão é mais elevado no *sideways* (±26,10) do que no *Random-Restart* (±17,78), refletindo maior variabilidade entre execuções. Essa diferença é confirmada pelo boxplot, que mostra maior dispersão e presença de outliers no *Sideways*, enquanto o *Random-Restart* apresenta distribuição mais concentrada.

De modo geral, ambos exigem número semelhante de passos para alcançar a solução, mas o *Random-Restart* apresenta comportamento mais estável, enquanto o *sideways* pode variar significativamente conforme o estado inicial do problema.

Análise de tempo médio

A Figura 6 mostra o tempo médio com desvio padrão cada método, enquanto a Figura 7 apresenta o boxplot da distribuição de tempo:

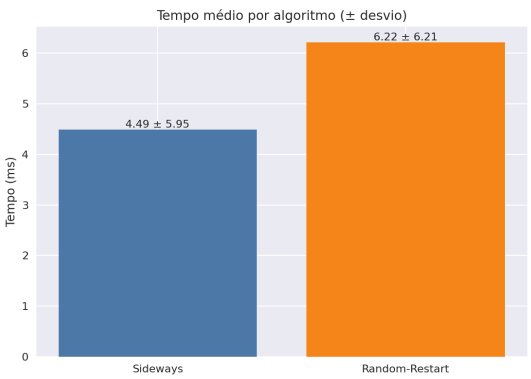

Figura 6: Gráfico comparativo do tempo médio por algoritmo.

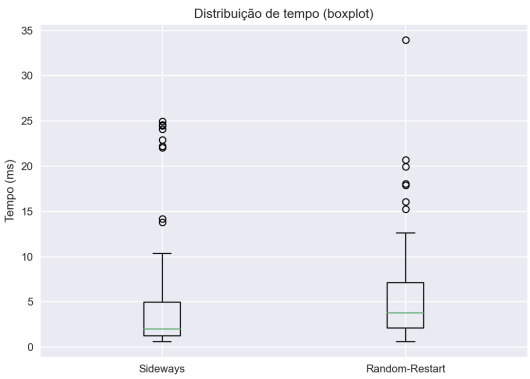

Figura 7: Boxplot comparativo da distribuição de tempo dos métodos.

Observa-se que ambos os métodos têm medianas baixas, mas o Random-Restart exibe dispersão maior e mais outliers de alto tempo do que o *sideways*. Além disso, a média também aponta diferença a favor do *sideways*.

O maior tempo observado no Random-Restart decorre principalmente de execuções em que foram necessários vários reinícios ou em que as tentativas demandaram mais passos até convergir; já o *sideways* tende a resolver rapidamente os casos favoráveis e, quando falha, tende a ficar preso (o que limita o tempo consumido por tentativa, dependendo do limite de movimentos laterais imposto).

Em resumo, o *sideways* demonstra melhor desempenho médio em tempo de execução, enquanto o *Random-Restart* sacrifica tempo em alguns casos para garantir maior taxa de sucesso, resultando em média e variância superiores.

Análise da taxa de sucesso

A Figura 8 mostra a taxa de sucesso percentual por algoritmo:

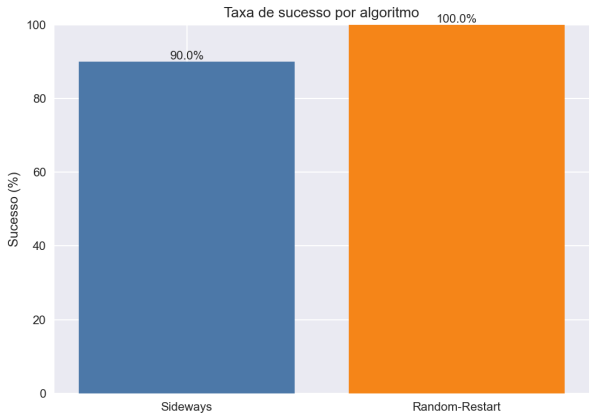

Figura 8: Gráfico de comparação da taxa de sucesso por algoritmo.

Nota-se que o *Random-Restart* alcança taxa máxima (100%) nas execuções apresentadas, enquanto o *Sideways* apresenta taxa inferior (90%).

O Random-Restart, ao gerar novas posições iniciais quando o algoritmo fica preso, garante maior probabilidade de encontrar uma solução válida em algum dos reinícios, o que explica a taxa de sucesso superior. O *sideways*, embora capaz de escapar de pequenos platôs, não reinicia e, portanto, pode permanecer preso em máximos locais, reduzindo sua taxa de sucesso global.

No geral, em termos de completude prática e taxa de sucesso, o Random-Restart se mostra superior e é a estratégia mais robusta para garantir solução em instâncias diversas, ao custo de trabalho extra.

Resultados médios do método de movimentos laterais:

Esse método teve uma média de 90% de taxa de sucesso para os tabuleiros utilizados, já que ele obtém sucesso em 57 dos 100 tabuleiros aleatórios gerados. Isso pode indicar que, quando ocorre a falha, o algoritmo fica preso em um platô local de tal forma que realizar movimentos laterais não o ajuda a sair, ou o algoritmo atinge o limite de movimentos laterais estabelecidos. Outra falha que pode acontecer é a presença de máximos locais, já que o *sideaways* não é capaz de tratar esse cenário e também não reinicia quando cai nessa situação.

Resultados médios do método *Random-Restart*:

Esse método, diferente do anterior, garantiu o sucesso em todas as tentativas para os mesmos tabuleiros. Isso acontece porque, uma vez que ele não consegue finalizar um tabuleiro, ele começa do zero reordenando as rainhas novamente e repete isso até que ele consiga atingir o objetivo. Claro, isso também acontece dentro do limite imposto de reinícios e iterações para justificar o sucessso constante. Porém, a grande diferença está no tempo de execução, que é maior em média do que o do método anterior. Isso acontece porque, em média, o *Random-Restart* realiza mais passos do que o *sideaways*, além de reiniciar o ponto inicial do labirinto quando é preciso.

Conclusão dos Resultados

Através dos resultados obtidos, é possível concluir que o método com movimentos laterais é mais eficiente em tempos de execução, mas não consegue garantir com certeza que uma solução vai ser atingida. No geral, ele performa mais rápido em cenários em que ocorrem poucos passos e poucos movimentos laterais (ou nenhum) precisam ser feitos. Quando ele fica preso em um platô grande, fica evidente que explorar esse platô pode acabar sendo mais ineficente do que buscar outro caminho para resolver o problema.

O método *Random-Restart* se mostrou capaz de resolver todos os tabuleiros propostos dentro das faixas de limite estabelecidas, de forma que se torna mais eficiente quando aleatoriamente escolhe um cenário muito favorável para se resolver o problema em poucos passos. Em média, ele faz menos reinícios por problema do que o algoritmo anterior percorre laterais por problema, o que indica que recomeçar o problema pode ser melhor sob as circunstâncias certas.

Porém, quando as circunstâncias não são favoráveis (como, por exemplo, escolher ir para um vizinho que leva a um máximo local no critério de desempate aleatório entre vizinhos), o *Random-Restart* pode ser tornar mais ineficiente do que um método *sideaways* que escolhe um vizinho no critério de desempate que leva rapidamente à solução do problema em poucos passos e com quase nenhum (ou nenhum) movimento lateral. Com isso, é possível concluir que, quando um ou outro algoritmo precisa tomar uma decisão que depende do acaso e isso leva a um conjunto de escolhas ruins, ele pode ser bastante prejudicado em performance, assim perdendo a sua eficiência teórica.

Créditos e Declaração de Autoria

Autores e Papéis

O trabalho foi desenvolvido pelos seguintes integrantes da equipe:

- **Pedro Augusto Gontijo Moura**: responsável pela implementação dos algoritmos de busca, coleta de métricas e desenvolvimento do relatório.
- **Samuel Silva Gomes**: responsável pela coleta de métricas, análise dos resultados, elaboração de gráficos e desenvolvimento do relatório.

Uso de Inteligência Artificial

Ferramentas de Inteligência Artificial, como o ChatGPT (OpenAI), foram utilizadas apenas para auxílio básico na redação do texto, revisão de formatação e comentários, e esclarecimento de dúvidas conceituais. Nenhum trecho de código foi gerado automaticamente ou copiado de fontes externas.

Declaração de Autoria

Declara-se que este trabalho foi integralmente desenvolvido pelos autores listados, respeitando as diretrizes acadêmicas da disciplina e as políticas de integridade do curso. Todos os resultados, análises e códigos aqui apresentados refletem o entendimento e o esforço dos autores.

Referências Bibliográficas

[1] GOMES, S. S; MOURA, P. A. **IA-Busca-8Rainhas**. Disponível em: https://github.com/PedroAugusto08/IA-Busca-8Rainhas.git. Acesso em out. 2025

[2] A. Thiago, **Capítulo 4: Estruturas e Estratégias de Busca em Ambientes Complexos**, Sistema Integrado de Gestão de Atividades Acadêmicas. CEFET-MG. Acesso em out. 2025.

[3] RUSSELL, Stuart J.; NORVIG, Peter. Artificial Intelligence: A Modern Approach. 4. ed. Hoboken: Pearson, 2021.