Relatório: Resolução do Problema das 8 Rainhas com Algoritmos de Busca e Otimização

1. Introdução

O problema das 8 rainhas é um clássico problema de inteligência artificial e teoria da computação. O desafio consiste em posicionar oito rainhas em um tabuleiro de xadrez 8x8 de forma que nenhuma rainha possa atacar outra, ou seja, não podem estar na mesma linha, coluna ou diagonal.

Este relatório apresenta a resolução do problema utilizando três abordagens distintas: Subida de Encosta com Reinício Aleatório, Têmpera Simulada e Algoritmo Genético. Cada algoritmo foi implementado em Python utilizando a biblioteca Pygame para interface e visualização.

2. Abordagem Teórica dos Algoritmos

2.1 Subida de Encosta com Reinício Aleatório

A Subida de Encosta (Hill Climbing) é um algoritmo de busca heurística que tenta melhorar iterativamente uma solução movendo-se para vizinhos melhores. No entanto, ele pode ficar preso em ótimos locais (soluções que parecem boas, mas não são ótimas). Para contornar isso, usamos reinícios aleatórios, onde o algoritmo reinicia com uma nova solução inicial ao encontrar um platô.

2.2 Têmpera Simulada

A Têmpera Simulada (Simulated Annealing) é inspirada no processo de resfriamento de metais. Ela aceita, de forma probabilística, soluções piores no início da execução para escapar de ótimos locais. Com o tempo, essa aceitação diminui (resfriamento), tornando o algoritmo mais seletivo.

2.3 Algoritmo Genético

O Algoritmo Genético simula a evolução natural. Ele mantém uma população de soluções, seleciona as melhores (por aptidão), cruza pares para gerar filhos (crossover) e muta aleatoriamente algumas características. Esse processo se repete por gerações até encontrar uma solução ideal.

3. Implementação Prática

Cada algoritmo foi implementado separadamente em arquivos próprios:
- subida.py — Subida de Encosta com Reinício
- tempera.py — Têmpera Simulada
- generico.py — Algoritmo Genético

A seleção do algoritmo é feita dinamicamente via o arquivo settings.py, que define a abordagem usada ao pressionar a tecla R na interface gráfica Pygame.

A interface gráfica mostra um tabuleiro 8x8, onde as posições das rainhas são renderizadas, e o algoritmo escolhido tenta resolver o problema automaticamente.

4. Resultados Obtidos e Comparação

Algoritmo	Tempo de Execução	Iterações / Gerações	Reinícios	Conflitos antes da solução	Solução Final
Subida de Encosta com Reinício	0.0076 segundos	10 iterações	1	1	[5, 2, 6, 1, 3, 7, 0, 4]
Têmpera Simulada	0.0035 segundos	270 iterações	-	1	[5, 3, 1, 4, 7, 3, 0, 2]

Algoritmo Genético	0.0392 segundos	5 gerações	-	0	[3, 6, 4, 2, 0, 5, 7, 1]

5. Discussão

5.1 Subida de Encosta com Reinício

Vantagens:
- Simples de implementar.
- Rápido para problemas com poucos ótimos locais.

Desvantagens:
- Fica facilmente preso em ótimos locais sem reinício.
- Pouco eficaz para problemas de busca com muitos platôs.

5.2 Têmpera Simulada

Vantagens:
- Capaz de escapar de ótimos locais por aceitar soluções piores no início.
- Mais eficiente que a subida pura.

Desvantagens:
- Requer ajuste fino da temperatura inicial e taxa de resfriamento.
- Pode demorar para convergir em problemas maiores.

5.3 Algoritmo Genético

Vantagens:
- Busca em paralelo diversas soluções.
- Alta taxa de sucesso em encontrar a solução ideal.

Desvantagens:
- Mais complexo de implementar.
- Tempo de execução maior em comparação aos outros.

6. Conclusão

Com base nos testes realizados:

- O Algoritmo Genético foi o único que encontrou a solução ideal (sem conflitos) de forma consistente.
- A Têmpera Simulada teve o melhor desempenho em tempo, mas não atingiu a solução perfeita neste teste.
- A Subida de Encosta com Reinício é útil e eficiente em tempo, mas menos confiável sem múltiplos reinícios.

✅ Algoritmo mais eficaz: Genético, para garantir soluções sem conflitos.
⚡ Algoritmo mais rápido: Têmpera Simulada, com menor tempo de execução.