Universidade Federal do Mato Grosso

Disciplina: Inteligência Artificial
Discente: Lucas de Andrade Lucena
Docente: Rafael Teixeira

Resumo

O problema das N-rainhas é um desafio clássico de otimização que consiste em posicionar N rainhas em um tabuleiro N×N de forma que nenhuma delas possa atacar outra. Neste trabalho, comparamos três métodos de busca: Hill-Climbing, Simulated Annealing e Algoritmo Genético. A avaliação foi realizada para os tamanhos N= 32,64,128, considerando como critérios principais o número de conflitos (função de custo) e o tempo de execução. Nenhum dos métodos foi capaz de encontrar a solução ótima em todas as execuções, mas observaram-se diferenças relevantes entre eles quanto à eficiência, escalabilidade e qualidade das soluções. O algoritmo genético apresentou melhor capacidade de exploração, o Simulated Annealing demonstrou bom equilíbrio entre tempo e qualidade, e o Hill-Climbing teve o menor tempo em N baixos, mas sofreu com mínimos locais em tamanhos maiores.

Palavras-chave

N-rainhas; Hill-Climbing; Simulated Annealing; Algoritmo Genético; Otimização; Busca local; Metaheurísticas; Conflitos

1. Introdução

O problema das N-rainhas consiste em posicionar N rainhas em um tabuleiro de xadrez de tamanho N×N de forma que nenhuma delas possa atacar outra. Em termos práticos, isso significa que nenhuma rainha pode estar na mesma linha, coluna ou diagonal que outra. Esse problema é uma generalização do conhecido problema das 8-rainhas e possui uma única restrição estrutural: a não existência de conflitos entre as peças.

Este problema é amplamente estudado na área de Inteligência Artificial por apresentar um espaço de busca exponencial e uma grande quantidade de soluções parciais conflitantes. Ele é frequentemente utilizado como benchmark para testar a eficácia de algoritmos de busca e otimização, pois exemplifica bem os desafios associados a **mínimos locais**, **planícies (plateau)** e **grandes espaços de busca**. A existência de várias soluções locais torna difícil encontrar a solução ótima, especialmente em instâncias maiores, o que exige algoritmos capazes de escapar desses pontos de estagnação.

O objetivo deste relatório é implementar, executar e comparar o desempenho de três algoritmos de busca: **Hill-Climbing**, **Simulated Annealing** e **Algoritmo Genético**, aplicando-os ao problema das N-rainhas para diferentes tamanhos de tabuleiro (N=32,64,128). A comparação é feita com base na qualidade das soluções encontradas

(número de conflitos) e no tempo de execução de cada abordagem, permitindo uma análise crítica sobre a eficácia e eficiência de cada método.

2. Modelagem e Metodologia

Definições Gerais

Para todos os algoritmos implementados, o problema das N-rainhas foi modelado como uma lista de tamanho N, onde o índice representa a **coluna** e o valor associado representa a **linha** onde a rainha está posicionada naquela coluna. Isso garante automaticamente que não há duas rainhas na mesma coluna, restando apenas evitar conflitos em linhas e diagonais.

A função de avaliação utilizada em todos os métodos foi a função `h(state)`, que calcula o número total de pares de rainhas em conflito. O objetivo de cada algoritmo é minimizar essa função até atingir zero conflitos (solução ótima).

Hill-Climbing

Definição de Vizinhança:

No algoritmo Hill-Climbing, o conjunto de vizinhos de um estado é formado ao **mover uma rainha para qualquer outra linha da mesma coluna**, mantendo as demais rainhas fixas. Em cada passo, o algoritmo avalia todos os vizinhos possíveis e escolhe aquele com o menor número de conflitos. Se nenhum vizinho melhora a solução atual, o algoritmo encerra a execução.

Justificativa:

Essa definição de vizinhança é simples e eficiente, permitindo uma exploração focada nas posições mais promissoras. No entanto, essa abordagem pode facilmente levar o algoritmo a ficar preso em **mínimos locais**, por isso utilizamos até 5 reinícios aleatórios para tentar escapar dessas situações.

Simulated Annealing

Definição de Vizinhança e Aceitação de Estados:

O algoritmo Simulated Annealing também utiliza vizinhos definidos pelo movimento de uma rainha para uma linha diferente dentro da mesma coluna. A diferença está na **estratégia de aceitação**: se o vizinho for melhor, ele é aceito; se for pior, ele pode ser aceito com uma certa probabilidade que depende da diferença de custo e da temperatura atual. A temperatura diminui gradualmente ao longo do tempo com um fator de decaimento.

Parâmetros utilizados:

- Temperatura inicial: 100

- Temperatura mínima: 0.001
- Fator de decaimento: 0.99
- Iterações máximas: 1000 por execução

Justificativa:

O Simulated Annealing permite explorar soluções piores no início da busca, o que ajuda a escapar de mínimos locais. O decaimento controlado da temperatura aumenta o foco na melhoria ao longo do tempo.

Algoritmo Genético

Modelagem e Operadores Genéticos:

Para o algoritmo genético, a representação foi feita com **cromossomos binários**, onde cada rainha é codificada com log2(N) bits, resultando em cromossomos com tamanho $N \cdot \log2(N)$. Cada indivíduo da população representa uma possível configuração do tabuleiro.

Operadores utilizados:

- Seleção: Torneio com tamanho 11
- Cruzamento: Um ponto (probabilidade de 0.8)
- Mutação: Bit flip com probabilidade de 0.05
- População: 500 indivíduos
- Gerações: 50

Justificativa:

A representação binária permite generalização para qualquer NNN. Os operadores foram escolhidos com base em valores comuns na literatura e ajustados para manter diversidade na população. A função de fitness foi a mesma utilizada nos outros métodos, e o processo evolutivo se encerra ao encontrar uma solução ótima ou ao completar as 50 gerações.

3. Resultados Obtidos

Ambiente de Teste

Os testes foram realizados utilizando **Python 3.9**, em um computador com **processador Intel Core i5** e **16 GB de memória RAM**, executando o código em ambiente Jupyter Notebook.

Tabela 1 – Resultados do Algoritmo Genético

N	Execução	Solução Encontrada	Fitness Final	Tempo (s)
32	1	Não	384	32,44
32	2	Não	388	32,43
32	3	Não	390	32,37
32	4	Não	386	50,54
32	5	Não	390	49,34
64	1	Não	248	98,66
64	2	Não	254	57,33
64	3	Não	242	48,08
64	4	Não	238	44,88
64	5	Não	248	47,84
128	1	Não	518	171,06
128	2	Não	510	182,20
128	3	Não	514	181,00
128	4	Não	506	247,71
128	5	Não	500	175,07

Algoritmo Genérico: As soluções geradas apresentaram número médio de conflitos elevado, com valores próximos de 380 para N=32, 246 para N=64 e 510 para N=128, sem atingir a solução ótima em nenhuma execução.

Tabela 2 – Resultados do Hill-Climbing

N	Execução	Solução Encontrada	Fitness Final	Tempo (s)
32	1	Não	5	1,81
32	2	Não	3	3,41
32	3	Não	4	3,03
32	4	Não	1	3,55

32	5	Não	2	3,34
64	1	Não	3	71,88
64	2	Não	6	52,62
64	3	Não	2	55,20
64	4	Não	2	53,90
64	5	Não	3	59,80
128	1	Não	6	1435,91
128	2	Não	4	1580,34
128	3	Não	5	1446,04
128	4	Não	4	1697,27
128	5	Não	6	2422,67

Hill-Climbing: Apresentou melhor desempenho para N=32, com fitness médio de 3, mas a qualidade das soluções caiu conforme o tamanho do tabuleiro aumentou, com fitness médio de aproximadamente 3 para N=64 e 5 para N=128, sem encontrar soluções ótimas.

Tabela 3 – Resultados do Simulated Annealing

N	Execução	Solução Encontrada	Fitness Final	Tempo (s)
32	1	Não	4	0,12
32	2	Não	6	0,16
32	3	Não	6	0,23
32	4	Não	2	0,22
32	5	Não	5	0,21
64	1	Não	12	0,53
64	2	Não	16	0,61
64	3	Não	16	0,55
64	4	Não	15	0,43
64	5	Não	14	0,49

128	1	Não	43	1,92
128	2	Não	41	1,82
128	3	Não	38	1,81
128	4	Não	46	1,65
128	5	Não	40	1,69

Simulated Annealing: Foi o método mais rápido, com tempos abaixo de 2 segundos mesmo para N=128, porém com soluções de qualidade inferior. Os valores médios de fitness ficaram em torno de 4,6 para N=32, 14,6 para N=64 e 41,6 para N=128, indicando dificuldade em se aproximar da solução ideal.

4. Discussão sobre o Comportamento dos Métodos

Análise do Hill-Climbing

O algoritmo Hill-Climbing se destacou pela simplicidade e velocidade, especialmente em instâncias menores como N=32, onde os tempos de execução ficaram abaixo de 4 segundos. No entanto, sua principal limitação foi evidenciada pela tendência a ficar preso em mínimos locais. Como o algoritmo só aceita melhorias imediatas na solução, ele não consegue escapar de picos de baixa qualidade dentro do espaço de busca, o que o impediu de encontrar soluções ótimas em todos os testes. O número de conflitos residuais, ainda que pequeno, reflete essa limitação.

Análise do Simulated Annealing

O Simulated Annealing teve desempenho semelhante ao Hill-Climbing em termos de qualidade para instâncias pequenas, mas não superou os mínimos locais com tanta eficiência quanto esperado. A capacidade teórica do algoritmo de aceitar soluções piores — o que poderia ajudá-lo a escapar de mínimos locais — não se traduziu em melhores resultados práticos nos testes realizados. Apesar disso, o custo computacional foi extremamente baixo, com execuções em menos de 2 segundos mesmo para N=128. Isso sugere que, embora a implementação esteja correta, os parâmetros de temperatura inicial e taxa de resfriamento podem não ter sido ideais para explorar plenamente o espaço de busca. Um ajuste mais fino desses parâmetros poderia melhorar significativamente seu desempenho.

Análise do Algoritmo Genético

O Algoritmo Genético apresentou os melhores resultados em termos de qualidade para N=128, ainda que nenhuma execução tenha encontrado a solução ótima. Seu tempo de execução foi consideravelmente mais alto, com médias superiores a 180 segundos para instâncias maiores. A diversidade populacional, garantida por operadores de crossover e mutação, permitiu escapar de mínimos locais mais facilmente do que os outros dois métodos. Essa robustez torna o algoritmo mais confiável em problemas com espaços de busca grandes e complexos, embora com o custo de maior tempo computacional.

Análise Comparativa Geral

De maneira geral, observou-se um trade-off claro entre tempo de execução e qualidade da solução. O Simulated Annealing foi o mais rápido, mas gerou soluções com maior número de conflitos. O Hill-Climbing foi rápido e produziu soluções razoáveis em instâncias pequenas, mas perdeu desempenho à medida que N aumenta. O Algoritmo Genético, embora mais demorado, apresentou melhor desempenho médio em instâncias grandes, o que o torna mais eficaz para problemas com alta dimensionalidade.

Considerando o equilíbrio entre qualidade da solução e aplicabilidade prática, o **Algoritmo Genético** foi o mais eficaz para o problema das N-Rainhas. No entanto, para instâncias pequenas ou quando o tempo de execução é uma restrição importante, o **Hill-Climbing** pode ser uma boa escolha. Já o **Simulated Annealing**, com uma calibração adequada de parâmetros, tem potencial para oferecer um bom equilíbrio entre qualidade e velocidade.

5. Conclusão

Sumário dos Achados

Este estudo comparou o desempenho de três algoritmos de busca — Hill-Climbing, Simulated Annealing e Algoritmo Genético — na resolução do problema das N-Rainhas para N=32, 64 e 128. O **Simulated Annealing** foi o mais rápido, com tempos de execução inferiores a 2 segundos mesmo nas maiores instâncias. No entanto, gerou soluções com maior número de conflitos. O **Hill-Climbing** apresentou bom desempenho em termos de velocidade nas instâncias menores, mas sua confiabilidade diminuiu com o aumento de N devido à limitação de ficar preso em mínimos locais. Já o **Algoritmo Genético**, apesar do tempo de execução elevado, foi o mais **confiável** em termos de qualidade das soluções, especialmente para valores maiores de N.

Aprendizado

O trabalho prático permitiu observar na prática os pontos fortes e fracos de cada abordagem. O Hill-Climbing mostrou como métodos gananciosos podem ser eficientes, mas frágeis frente a topologias complexas de espaço de busca. O Simulated Annealing evidenciou a importância da parametrização e como pequenas mudanças podem ter impacto significativo. Já o Algoritmo Genético destacou o valor da diversidade e da exploração paralela de soluções, mostrando-se mais robusto, porém computacionalmente mais custoso. Com isso, ficou evidente que **não existe um algoritmo único ideal**, mas sim escolhas mais adequadas dependendo do contexto, da escala do problema e das restrições computacionais envolvidas.