Universidade Federal do Mato Grosso

Discente:Lucas de Andrade Lucena
Docente: Rafael teixeira
Data: Julho de 2025

Resumo

Este trabalho implementa e compara três algoritmos de busca — Hill Climbing, Simulated Annealing e Algoritmo Genético — para resolver o problema das N-Rainhas com N = 32, 64 e 128. O objetivo é avaliar o desempenho em termos de tempo de execução, qualidade das soluções (número de conflitos) e robustez contra mínimos locais. Os resultados mostram que o Hill Climbing e o Simulated Annealing são rápidos e confiáveis para N pequeno e médio, enquanto o Algoritmo Genético é menos eficiente em tempo, mas mais robusto para N grande, embora com menor taxa de sucesso. A análise destaca os trade-offs entre velocidade e qualidade, com o Simulated Annealing oferecendo o melhor equilíbrio para N = 128.

Palavras-chave: N-Rainhas, Hill Climbing, Simulated Annealing, Algoritmo Genético, Inteligência Artificial, Busca Heurística.

1. Introdução

1.1 Definição do Problema

O problema das N-Rainhas consiste em posicionar N rainhas em um tabuleiro NxN de forma que nenhuma rainha ataque outra, ou seja, elas não podem compartilhar a mesma linha, coluna ou diagonal. Cada rainha é colocada em uma linha distinta, e a solução é representada por uma permutação das colunas, onde cada posição indica a coluna da rainha em uma linha específica.

1.2 Relevância

O problema das N-Rainhas é amplamente utilizado como benchmark em Inteligência Artificial devido ao seu espaço de busca combinatorialmente grande e à presença de múltiplos mínimos locais. Ele desafia algoritmos de busca a explorar eficientemente soluções viáveis, sendo um modelo para problemas de otimização e planejamento em áreas como escalonamento e design de sistemas.

1.3 Objetivo do Relatório

Este relatório tem como objetivo implementar, executar e comparar o desempenho dos algoritmos Hill Climbing, Simulated Annealing e Algoritmo Genético para o problema das N-Rainhas com N = 32, 64 e 128. A análise considera tempo de execução, qualidade das

soluções (número de conflitos) e robustez, com base em cinco execuções por algoritmo e valor de N.

2. Modelagem e Metodologia

2.1 Definições Gerais

- **Representação**: Um vetor de tamanho N, onde o índice (i) representa a linha e o valor (v[i]) a coluna da rainha, formando uma permutação de 1 a N. Isso elimina conflitos por linha e coluna, focando nos conflitos diagonais.
- **Função de Avaliação**: Conta o número de pares de rainhas em conflito (mesma coluna ou diagonal), com o objetivo de minimizá-lo para zero.
- **Ambiente de Teste**: Python 3.9 em um processador Intel i5 com 8GB de RAM, Windows 10.

2.2 Hill Climbing

- **Definição de Vizinhança**: Os vizinhos são gerados trocando duas posições no vetor, produzindo (N*(N-1)/2) vizinhos por estado. O melhor vizinho (menor número de conflitos) é selecionado.
- **Parâmetros**: Máximo de 1000 iterações por reinício, 100 reinícios aleatórios.
- **Justificativa**: A troca de posições mantém a permutação válida, e os reinícios aleatórios ajudam a escapar de mínimos locais, comuns no problema. O limite de iterações evita loops longos em estados estagnados.

2.3 Simulated Annealing

- **Vizinhança**: Um vizinho é gerado trocando aleatoriamente duas posições no vetor.
- **Parâmetros**: Temperatura inicial = 100.0, taxa de resfriamento = 0.95, temperatura mínima = 0.01, 100 iterações por temperatura.
- **Justificativa**: A temperatura inicial alta promove exploração inicial, enquanto a taxa de resfriamento gradual garante convergência. Gerar um único vizinho por iteração reduz o custo computacional, e a probabilidade de aceitar soluções piores evita mínimos locais.

2.4 Algoritmo Genético

- **Representação**: Cromossomo como uma permutação de 1 a N.
- **Operadores**: Seleção por torneio (tamanho 3), crossover de ordem (OX1), mutação por troca (taxa 5%).
- **Parâmetros**: População de 100 indivíduos, máximo de 1000 gerações.
- **Justificativa**: O torneio balanceia pressão seletiva e diversidade. O crossover OX1 é adequado para permutações, preservando a ordem relativa. A mutação introduz diversidade, e a população de 100 é suficiente para explorar o espaço de busca sem custo excessivo.

3. Resultados Obtidos

3.1 Ambiente de Teste

Os experimentos foram realizados em Python 3.9, em um computador com processador Intel i5, 8GB de RAM e sistema operacional Windows 10. Cada algoritmo foi executado cinco vezes para N = 32, 64 e 128.

3.2 Tabelas de Resultados

Hill Climbing

N	Execução	Tempo (s)	Conflitos	Solução Válida
32	1	0.66	0	Sim
32	2	0.26	0	Sim
32	3	1.85	0	Sim
32	4	3.85	0	Sim
32	5	0.60	0	Sim
64	1	35.04	0	Sim
64	2	31.02	0	Sim
64	3	27.79	0	Sim
64	4	10.29	0	Sim
64	5	14.19	0	Sim
128	1	903.14	0	Sim
128	2	291.33	0	Sim
128	3	1280.34	0	Sim
128	4	439.35	0	Sim
128	5	985.88	0	Sim

Simulated Annealing

N	Execução	Tempo (s)	Conflitos	Solução Válida
32	1	1.31	0	Sim

32	2	1.04	0	Sim
32	3	0.82	0	Sim
32	4	0.97	0	Sim
32	5	0.90	0	Sim
64	1	4.43	0	Sim
64	2	4.02	0	Sim
64	3	4.31	0	Sim
64	4	3.67	0	Sim
64	5	4.48	0	Sim
128	1	16.43	0	Sim
128	2	18.65	1	Não
128	3	19.42	2	Não
128	4	19.39	0	Sim
128	5	21.22	2	Não

Algoritmo Genético

N	Execução	Tempo (s)	Conflitos	Solução Válida
32	1	7.22	0	Sim
32	2	11.98	1	Não
32	3	6.15	0	Sim
32	4	22.61	3	Não
32	5	20.30	1	Não
64	1	64.23	1	Não
64	2	35.18	1	Não
64	3	44.76	1	Não
64	4	28.58	0	Sim
64	5	30.11	1	Não

128	1	118.59	4	Não
128	2	138.19	5	Não
128	3	137.23	4	Não
128	4	154.42	3	Não
128	5	178.38	5	Não

3.3 Resumo da Qualidade

- **Hill Climbing**: Média de conflitos: N=32: 0.0, N=64: 0.0, N=128: 0.0. Encontrou soluções válidas em todas as execuções.
- **Simulated Annealing**: Média de conflitos: N=32: 0.0, N=64: 0.0, N=128: 1.0. Encontrou soluções válidas em todas as execuções para N=32 e 64, e em 2/5 para N=128.
- **Algoritmo Genético**: Média de conflitos: N=32: 1.0, N=64: 0.8, N=128: 4.2. Encontrou soluções válidas em 2/5 para N=32, 1/5 para N=64 e 0/5 para N=128.

4. Discussão sobre o Comportamento dos Métodos

4.1 Análise do Hill Climbing

O Hill Climbing foi o mais rápido para N=32 (média de 1.44s) e altamente confiável, encontrando soluções válidas em todas as execuções para todos os N. No entanto, para N=128, o tempo médio aumentou significativamente (780.01s), devido ao grande número de vizinhos avaliados ((N*(N-1)/2)). A estagnação em mínimos locais foi evitada pelos reinícios aleatórios, que foram eficazes mesmo para N grande, mas o custo computacional cresceu quadraticamente.

4.2 Análise do Simulated Annealing

Comparado ao Hill Climbing, o Simulated Annealing foi ligeiramente mais lento para N=32 (1.01s) e N=64 (4.18s), mas muito mais rápido para N=128 (19.02s). Ele encontrou soluções válidas em todas as execuções para N=32 e 64, mas apenas em 2/5 para N=128, com média de conflitos de 1.0. A capacidade de aceitar soluções piores no início, graças à temperatura inicial de 100.0, ajudou a explorar o espaço de busca, mas a taxa de resfriamento (0.95) pode ter sido muito rápida para N=128, limitando a convergência. O custo computacional extra em relação ao Hill Climbing foi justificado para N=128, onde o SA foi significativamente mais rápido.

4.3 Análise do Algoritmo Genético

O Algoritmo Genético foi o mais lento, com tempos médios de 13.65s (N=32), 40.57s (N=64) e 145.36s (N=128). Sua robustez foi inferior, com taxas de sucesso baixas (2/5 para N=32, 1/5 para N=64, 0/5 para N=128) e médias de conflitos mais altas (1.0, 0.8, 4.2). A

diversidade genética, mantida por mutação (5%) e crossover OX1, não foi suficiente para N=128, possivelmente devido ao tamanho da população (100) ou ao número de gerações (1000). Para N pequeno, o AG foi menos eficiente que os outros métodos.

4.4 Análise Comparativa Geral

O Hill Climbing destacou-se pela velocidade e confiabilidade para N=32 e 64, mas seu tempo para N=128 foi excessivo. O Simulated Annealing ofereceu o melhor equilíbrio, com tempos razoáveis e boa qualidade para N=128, sendo o mais eficaz geral. O Algoritmo Genético foi o menos competitivo, com alto custo computacional e baixa taxa de sucesso. Para cenários onde a velocidade é crítica (ex.: N=32), o Hill Climbing é recomendado. Para N grande, o Simulated Annealing é preferível, enquanto o AG pode ser útil com ajustes (ex.: maior população).

5. Conclusão

5.1 Sumário dos Achados

O Hill Climbing foi o mais rápido para N=32 e 64, mas ineficiente para N=128. O Simulated Annealing foi o mais equilibrado, com bom desempenho em todos os N e melhor tempo para N=128. O Algoritmo Genético foi o menos eficaz, com tempos altos e baixa confiabilidade. O Simulated Annealing é a melhor escolha para o problema das N-Rainhas em geral.

5.2 Aprendizado

Este trabalho demonstrou que a escolha do algoritmo depende do tamanho do problema e dos requisitos de tempo e qualidade. O Hill Climbing é limitado por sua dependência de reinícios, o Simulated Annealing destaca-se pela flexibilidade, e o Algoritmo Genético exige ajustes para ser competitivo. A experiência reforçou a importância de ajustar parâmetros e entender o comportamento dos algoritmos em espaços de busca complexos.