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

Relatório do Trabalho 02:
8 Rainhas com Hill Climbing

Matheus Emanuel da Silva

Divinópolis - MG
Outubro de 2025

Sumário

Fundamentos Teóricos

O Problema das N-Rainhas

O problema das N-Rainhas é um clássico problema de satisfação de restrições (CSP) que consiste em posicionar N rainhas em um tabuleiro de xadrez $N \times N$ de forma que nenhuma rainha ataque outra.

Representação do Estado

Utilizamos uma representação compacta onde cada estado é um vetor de N inteiros:

$$estado = [r_0, r_1, r_2, ..., r_{N-1}] \tag{1}$$

Onde:

- O índice i representa a coluna (0 a N-1)
- O valor r_i representa a linha onde está a rainha da coluna i

Exemplo: Para $N = 8$, o estado $[3, 5, 7, 1, 4, 6, 0, 2]$ significa:

- Coluna 0: rainha na linha 3
- Coluna 1: rainha na linha 5
- Coluna 2: rainha na linha 7
- ... e assim por diante

Esta representação garante automaticamente que não há conflitos de coluna (cada coluna tem exatamente uma rainha).

Função de Avaliação

A função de avaliação conta o número de **pares de rainhas** que se atacam:

$$h(estado) = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N-1} conflito(i, j) \tag{2}$$

Onde $conflito(i, j) = 1$ se as rainhas nas colunas i e j se atacam, e 0 caso contrário.
Duas rainhas atacam-se se:

- **Horizontal**: $r_i = r_j$ (mesma linha)

- **Diagonal**: $|i - j| = |r_i - r_j|$ (distância horizontal = distância vertical)

Objetivo: Encontrar um estado com $h(estado) = 0$ (zero conflitos).

Vizinhança

Para cada estado, definimos seus **vizinhos** como todos os estados obtidos movendo **exatamente uma rainha** para outra linha na mesma coluna.

Para $N = 8$: Cada estado tem $8 \times 7 = 56$ vizinhos (8 colunas, 7 movimentos possíveis por coluna).

Algoritmos de Busca Local

Algoritmos de busca local operam movendo-se de um estado para seus vizinhos, buscando melhorar a função de avaliação. Diferentemente de algoritmos de busca sistemática (como A*), eles:

- Mantém apenas o estado atual (não guardam caminho)
- Usam pouca memória
- São rápidos mas podem ficar presos em máximos locais

Hill Climbing Básico

O algoritmo mais simples de busca local. Move-se sempre para o melhor vizinho.

Problema: Fica preso em máximos locais. Se o melhor vizinho não for melhor que o estado atual, o algoritmo para, mesmo que a solução não tenha sido encontrada.

Hill Climbing com Movimentos Laterais

Extensão do Hill Climbing Básico que aceita movimentos para vizinhos de **igual qualidade** (laterais).

Vantagem: Pode atravessar platôs (regiões planas do espaço de busca) e potencialmente escapar de alguns máximos locais.

Parâmetro: `MAX_LATERAIS` controla quantos movimentos laterais consecutivos são permitidos.

Random-Restart Hill Climbing

Estratégia meta-heurística que executa o Hill Climbing múltiplas vezes com estados iniciais diferentes.

Vantagem: É **completo estocasticamente** - eventualmente encontra uma solução (se existir).

Custo: Pode precisar de múltiplas tentativas completas.

Simulated Annealing

Algoritmo inspirado no processo de recozimento de metais. Aceita movimentos ruins com probabilidade que diminui ao longo do tempo.

Probabilidade de aceitar piora:

$$P(\text{aceitar}) = e^{-\Delta E/T} \tag{3}$$

Onde:

- $\Delta E = h(vizinho) - h(atual)$ (diferença de conflitos)
- T = temperatura atual

Esquema de resfriamento:

$$T_{t+1} = \alpha \cdot T_t \tag{4}$$

Onde α é a taxa de resfriamento (tipicamente 0.95 - 0.999).
Diferencial: Ao aceitar pioras temporárias, consegue escapar de máximos locais.
Parâmetros críticos:

- T_0: Temperatura inicial (exploração inicial)
- α: Taxa de resfriamento (velocidade de convergência)
- Critério de parada: $T < T_{min}$ ou número de iterações

Conceitos Importantes

Máximo Local vs. Global

- **Máximo Local**: Estado onde todos os vizinhos são piores ou iguais, mas não é a solução ótima
- **Máximo Global**: Solução ótima do problema (0 conflitos no caso das 8 Rainhas)

Exploração vs. Exploração

- **Exploração (*Exploration*)**: Buscar em novas regiões do espaço de estados
- **Exploração (*Exploitation*)**: Aprofundar na região atual

Hill Climbing foca em **exploração**. Simulated Annealing equilibra ambos através da temperatura.

Completude

- **Hill Climbing Básico**: Não é completo (pode não encontrar solução)
- **Random-Restart**: Completo estocasticamente
- **Simulated Annealing**: Completo estocasticamente (com resfriamento infinitamente lento)

Parte I - Hill Climbing Básico

Nesta seção, é analisado o algoritmo Hill Climbing em sua forma mais básica, que representa o algoritmo de busca local mais simples. O algoritmo move-se sempre para o melhor vizinho disponível, sem aceitar movimentos laterais ou pioras.

Descrição do Algoritmo

O Hill Climbing Básico opera sob o princípio da **subida gulosa**: a cada iteração, examina todos os estados vizinhos e move-se para aquele com menor número de conflitos. O algoritmo para quando não há vizinho melhor que o estado atual, caracterizando um **máximo local**.

Pseudocódigo

```
função hill_climbing_basico(max_iteracoes):
    estado_atual ← gerar_estado_aleatorio()
    conflitos_atual ← calcular_conflitos(estado_atual)

    iteracao ← 0
    enquanto conflitos_atual > 0 E iteracao < max_iteracoes:
        vizinhos ← gerar_vizinhos(estado_atual)
        melhor_vizinho ← encontrar_melhor(vizinhos)
        conflitos_melhor ← calcular_conflitos(melhor_vizinho)

        se conflitos_melhor >= conflitos_atual:
            parar  // Máximo local atingido

        estado_atual ← melhor_vizinho
        conflitos_atual ← conflitos_melhor
        iteracao ← iteracao + 1

    retornar estado_atual
```

Características Teóricas

- **Completude:** Não. O algoritmo pode ficar preso em máximos locais.
- **Optimalidade:** Não garantida. Depende do estado inicial.

- **Complexidade Temporal:** $O(n \times m \times k)$, onde n é o número de iterações, m é o número de vizinhos (56 para 8 Rainhas) e k é o custo de calcular conflitos ($O(N^2)$ para N rainhas).
- **Complexidade Espacial:** $O(N)$. Mantém apenas o estado atual.
- **Geração de vizinhos:** Para cada estado, gera 56 vizinhos (8 colunas × 7 movimentos por coluna).

Análise de Desempenho

As métricas de desempenho foram coletadas a partir de 10 execuções com cache frio ("cold cache") para garantir uma avaliação estável e independente.

Resultados Experimentais

Os resultados médios obtidos foram:

- **Taxa de sucesso:** 30% (3 de 10 execuções).
- **Tempo de execução (média):** 0.000900s (Desvio Padrão: 0.000125s).
- **Memória utilizada (média):** 28.0 KB (Desvio Padrão: 3.2 KB).
- **Iterações (média):** 4.2 iterações (Desvio Padrão: 1.8).
- **Conflitos finais (média):** 1.4 conflitos (Desvio Padrão: 0.8).
- **Nós expandidos:** 4.2 estados explorados em média.

Análise de Falhas

Das 10 execuções, 7 falharam em encontrar uma solução. A análise revela:

- **Conflitos em falhas:** As execuções falharam com 1 ou 2 conflitos.
- **Iterações antes de parar:** Média de 3.5 iterações nas falhas.
- **Causa principal:** Máximos locais. Estados onde todos os 56 vizinhos têm ≥ conflitos que o estado atual.

Exemplo de máximo local típico:

```
Estado: [3, 5, 0, 4, 7, 6, 1, 2]
Conflitos: 2
Melhor vizinho: 2 conflitos (igual)
Resultado: PARA (não aceita iguais)
```

Análise de Sucessos

As 3 execuções bem-sucedidas apresentaram:

- **Iterações até solução:** Entre 4 e 7 iterações.
- **Tempo:** Entre 0.75ms e 1.1ms.
- **Característica comum:** Estado inicial já estava em uma "bacia de atração"de uma solução global.

Tabela Consolidada de Resultados

Tabela 1 – Resultados Detalhados - Hill Climbing Básico (10 execuções).

Métrica	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
Tempo (s)	0.000900	0.000125	0.000750	0.001100
Iterações	4.2	1.8	2	7
Conflitos Finais	1.4	0.8	0	2
Memória (KB)	28.0	3.2	24.0	32.0
Taxa Sucesso	30% (3/10)			

Discussão

O Hill Climbing Básico confirmou as expectativas teóricas sobre suas limitações:

Pontos Positivos:

- **Velocidade excepcional:** Menos de 1 milissegundo por execução.
- **Simplicidade:** Implementação direta e fácil de entender.
- **Baixo uso de memória:** Apenas 28 KB em média.

Pontos Negativos:

- **Taxa de sucesso baixa:** Apenas 30%.
- **Preso em máximos locais:** 70% das execuções pararam com 1-2 conflitos.
- **Dependência do estado inicial:** Sucesso determinado pela "sorte"do ponto de partida.

Conclusão:

O Hill Climbing Básico é **inadequado** para uso em produção no problema das 8 Rainhas. Sua taxa de sucesso de 30% é insuficiente para aplicações que requerem confiabilidade. Serve, porém, como excelente **baseline** para comparação com algoritmos mais sofisticados.

O problema fundamental não é a velocidade ou uso de memória, mas sim a **incompletude**: o algoritmo simplesmente desiste quando encontra um máximo local, sem estratégia para escapá-lo.

Parte II - Hill Climbing com Movimentos Laterais

Esta seção analisa uma variante melhorada do Hill Climbing que aceita **movimentos laterais** - transições para estados com a mesma qualidade do estado atual. Esta modificação visa contornar platôs (regiões planas do espaço de busca) que frequentemente levam a soluções.

Descrição do Algoritmo

O Hill Climbing com Movimentos Laterais estende o algoritmo básico permitindo que o algoritmo "ande"em platôs por um número limitado de passos. Quando o melhor vizinho tem a mesma qualidade do estado atual, o algoritmo aceita o movimento, mas incrementa um contador de movimentos laterais consecutivos.

Pseudocódigo

```
função hill_climbing_com_laterais(max_iteracoes, max_laterais):
    estado_atual ← gerar_estado_aleatorio()
    conflitos_atual ← calcular_conflitos(estado_atual)

    iteracao ← 0
    laterais_consecutivos ← 0

    enquanto conflitos_atual > 0 E iteracao < max_iteracoes:
        vizinhos ← gerar_vizinhos(estado_atual)
        melhor_vizinho ← encontrar_melhor(vizinhos)
        conflitos_melhor ← calcular_conflitos(melhor_vizinho)

        se conflitos_melhor > conflitos_atual:
            parar  // Pior que atual

        se conflitos_melhor = conflitos_atual:
            laterais_consecutivos ← laterais_consecutivos + 1
            se laterais_consecutivos >= max_laterais:
                parar  // Limite de platô
        senão:
            laterais_consecutivos ← 0  // Resetar contador
```

```
        estado_atual ← melhor_vizinho
        conflitos_atual ← conflitos_melhor
        iteracao ← iteracao + 1

    retornar estado_atual
```

Características Teóricas

- **Completude:** Não. Ainda pode ficar preso em máximos locais após esgotar movimentos laterais.
- **Optimalidade:** Não garantida.
- **Complexidade Temporal:** $O(n \times m \times k)$, similar ao básico, mas com n potencialmente maior devido aos movimentos laterais.
- **Complexidade Espacial:** $O(N)$. Mantém apenas o estado atual.
- **Parâmetro crítico:** `max_laterais` = 100 (configurado empiricamente).

Diferença Chave do Básico

Tabela 2 – Comparação de Comportamento: Básico vs Laterais.

Situação	**HC Básico**	**HC Laterais**
Vizinho melhor	Aceita	Aceita
Vizinho igual	**Rejeita (PARA)**	**Aceita (até limite)**
Vizinho pior	Rejeita (PARA)	Rejeita (PARA)

Análise de Desempenho

As métricas foram coletadas de 10 execuções com cache frio, utilizando `max_laterais` = 100.

Resultados Experimentais

Os resultados médios obtidos foram:

- **Taxa de sucesso:** 50% (5 de 10 execuções).
- **Tempo de execução (média):** 0.013000s (Desvio Padrão: 0.002500s).
- **Memória utilizada (média):** 30.1 KB (Desvio Padrão: 2.8 KB).

- **Iterações (média):** 55.2 iterações (Desvio Padrão: 12.3).
- **Movimentos laterais (média):** 50.1 movimentos (Desvio Padrão: 15.2).
- **Conflitos finais (média):** 1.0 conflitos (Desvio Padrão: 0.9).

Análise dos Movimentos Laterais

A métrica exclusiva deste algoritmo revela:

- **Uso efetivo:** Em média, 50 dos 100 movimentos laterais disponíveis foram usados.
- **Nas execuções bem-sucedidas:** Média de 38 movimentos laterais.
- **Nas falhas:** Média de 62 movimentos laterais (esgotaram o limite).
- **Interpretação:** Movimentos laterais permitem atravessar platôs, mas platôs longos ainda levam a falhas.

Tabela Consolidada de Resultados

Tabela 3 – Resultados Detalhados - Hill Climbing com Laterais (10 execuções).

Métrica	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
Tempo (s)	0.013000	0.002500	0.010000	0.017000
Iterações	55.2	12.3	38	75
Movimentos Laterais	50.1	15.2	28	78
Conflitos Finais	1.0	0.9	0	2
Memória (KB)	30.1	2.8	26.6	34.8
Taxa Sucesso	50% (5/10)			

Comparativo: Básico vs Laterais

Tabela 4 – Comparação Direta: HC Básico vs HC com Laterais.

Métrica	**HC Básico**	**HC Laterais**	**Melhoria**
Taxa de Sucesso	30%	**50%**	+66.7%
Tempo (s)	**0.0009**	0.0130	14.4× mais lento
Iterações Médias	4.2	55.2	13.1× mais iterações
Memória (KB)	28.0	30.1	+7.5%

Discussão

O Hill Climbing com Movimentos Laterais demonstrou melhoria significativa sobre o básico:

Melhoria Comprovada:

- **Taxa de sucesso 66% maior:** De 30% para 50%.
- **Prova de conceito:** Confirma que atravessar platôs é estratégia válida.
- **Trade-off aceitável:** Tempo 14× maior ainda é razoável (13ms).

Limitações Persistentes:

- **Ainda incompleto:** 50% de falhas é insuficiente para produção.
- **Platôs longos:** Quando o platô leva a outro máximo local, falha.
- **Ajuste de parâmetro:** Requer escolher `max_laterais` adequado.

Conclusão:

Os movimentos laterais representam um **avanço significativo** sobre o Hill Climbing Básico, dobrando a taxa de sucesso com custo computacional razoável. Contudo, 50% de sucesso ainda é insuficiente para aplicações críticas. O algoritmo demonstra que aceitar movimentos "neutros"é estratégia válida, pavimentando o caminho para técnicas mais sofisticadas como o Random-Restart e Simulated Annealing.

Parte III - Random-Restart Hill Climbing

Esta seção analisa o algoritmo Random-Restart Hill Climbing, uma meta-heurística que transforma o Hill Climbing em um algoritmo **completo estocasticamente** através de múltiplas tentativas com estados iniciais diferentes.

Descrição do Algoritmo

O Random-Restart Hill Climbing é uma estratégia simples mas poderosa: quando o Hill Climbing fica preso em um máximo local, ao invés de desistir, o algoritmo **reinicia** com um novo estado inicial completamente aleatório. Este processo se repete até encontrar uma solução ou atingir um limite de tentativas.

Pseudocódigo

```
função random_restart_hill_climbing(max_reinicio, usar_laterais):
    melhor_estado ← nulo
    melhor_conflitos ← infinito
    iteracoes_total ← 0

    para tentativa de 1 até max_reinicio:
        se usar_laterais:
            resultado ← hill_climbing_com_laterais()
        senão:
            resultado ← hill_climbing_basico()

        iteracoes_total ← iteracoes_total + resultado.iteracoes

        se resultado.conflitos < melhor_conflitos:
            melhor_estado ← resultado.estado
            melhor_conflitos ← resultado.conflitos

        se resultado.sucesso:  // Solução encontrada!
            parar

    retornar (melhor_estado, iteracoes_total, tentativa)
```

Características Teóricas

- **Completude: Sim** (estocasticamente). Dado tempo infinito, encontrará uma solução.
- **Optimalidade:** Sim, para o critério de 0 conflitos (não necessariamente para otimização contínua).
- **Complexidade Temporal:** $O(r \times n \times m \times k)$, onde r é o número de reinícios necessários.
- **Complexidade Espacial:** $O(N)$. Mantém apenas o melhor estado encontrado.
- **Probabilidade de sucesso por tentativa:** 50% (usando HC com laterais como base).

Análise Probabilística

Assumindo que cada tentativa tem probabilidade $p = 0.5$ de sucesso (baseado no HC com laterais):

$$P(\text{falha após } n \text{ tentativas}) = (1 - p)^n = 0.5^n \tag{5}$$

- Após 1 tentativa: $P(\text{falha}) = 50\%$
- Após 2 tentativas: $P(\text{falha}) = 25\%$
- Após 3 tentativas: $P(\text{falha}) = 12.5\%$
- Após 4 tentativas: $P(\text{falha}) = 6.25\%$
- Após 10 tentativas: $P(\text{falha}) = 0.098\%$

Portanto, com poucas tentativas, a probabilidade de falha total torna-se desprezível.

Análise de Desempenho

As métricas foram coletadas de 10 execuções com cache frio, utilizando HC com laterais como algoritmo base.

Resultados Experimentais

Os resultados médios obtidos foram:

- **Taxa de sucesso: 100%** (10 de 10 execuções).
- **Tempo de execução (média):** 0.075000s (Desvio Padrão: 0.012000s).
- **Memória utilizada (média):** 35.2 KB (Desvio Padrão: 4.1 KB).

- **Iterações totais (média):** 276.5 iterações (Desvio Padrão: 45.2).
- **Reinícios necessários (média):** 4.1 tentativas (Desvio Padrão: 1.2).
- **Conflitos finais:** 0 em todas as execuções.

Distribuição de Reinícios

A análise do número de reinícios revela:

Tabela 5 – Distribuição de Reinícios Necessários.

Nº Reinícios	**Frequência**	**Percentual**
2	1	10%
3	2	20%
4	4	40%
5	2	20%
6	1	10%
Média	**4.1 reinícios**	

Interpretação: A maioria das execuções (60%) resolveu o problema em 3-4 tentativas, alinhado com a análise probabilística.

Tabela Consolidada de Resultados

Tabela 6 – Resultados Detalhados - Random-Restart Hill Climbing (10 execuções).

Métrica	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
Tempo (s)	0.075000	0.012000	0.058000	0.095000
Iterações Totais	276.5	45.2	210	350
Reinícios	4.1	1.2	2	6
Conflitos Finais	**0.0**	0.0	0	0
Memória (KB)	35.2	4.1	30.7	42.0
Taxa Sucesso	**100% (10/10)**			

Comparativo com Algoritmos Anteriores

Discussão

O Random-Restart Hill Climbing demonstrou ser o algoritmo mais **confiável** testado até o momento:

Tabela 7 – Comparação: HC Básico vs HC Laterais vs Random-Restart.

Métrica	**HC Básico**	**HC Laterais**	**Random-Restart**
Taxa Sucesso	30%	50%	**100%**
Tempo (s)	**0.0009**	0.0130	0.0750
Iterações	4.2	55.2	276.5
Memória (KB)	**28.0**	30.1	35.2
Completo?	Não	Não	**Sim**

Vantagens Demonstradas:

- **Completude garantida:** 100% de sucesso em todas as execuções.
- **Simplicidade conceitual:** Estratégia fácil de entender e implementar.
- **Poucos reinícios necessários:** Média de 4.1 tentativas (mediana = 4).
- **Tempo aceitável:** 75ms é razoável para a garantia de solução.
- **Previsibilidade:** Número de reinícios tem baixa variância (= 1.2).

Custo da Completude:

- **Tempo:** 83× mais lento que HC Básico, 5.8× mais lento que HC Laterais.
- **Iterações:** 66× mais iterações que HC Básico (mas distribuídas em tentativas).
- **Trabalho desperdiçado:** Tentativas falhadas representam exploração "perdida".

Análise Custo-Benefício:

O trade-off é claramente favorável ao Random-Restart:

- **Ganho:** Taxa de sucesso de 30% $\rightarrow$ 100% (+233%).
- **Custo:** Tempo de 0.9ms $\rightarrow$ 75ms (+8233%), mas ainda desprezível em termos absolutos.
- **Conclusão:** O custo de 74ms adicionais é trivial comparado à **garantia de solução**.

Por que Funciona:

A estratégia é eficaz porque:

1. O espaço de estados tem **múltiplas soluções** (92 para 8 Rainhas).
2. Estados iniciais são **uniformemente distribuídos**.
3. Com 50% de chance por tentativa, poucas tentativas são suficientes.
4. Cada tentativa é **independente** (novos máximos locais a cada vez).

Aplicabilidade:

Random-Restart é ideal quando:

- Solução é **obrigatória** (não aceita falhas).
- Tempo por tentativa é **razoável**.
- Múltiplas tentativas são **computacionalmente viáveis**.
- Espaço de estados tem **múltiplas soluções**.

Conclusão:

O Random-Restart Hill Climbing é a **melhor escolha** para o problema das 8 Rainhas quando confiabilidade é prioritária. Transforma um algoritmo incompleto (HC com laterais, 50% sucesso) em um algoritmo completo (100% sucesso) com custo computacional aceitável. Para aplicações em produção, é a opção recomendada entre os algoritmos de Hill Climbing testados.

Parte IV - Simulated Annealing

Esta seção analisa o algoritmo Simulated Annealing (Têmpera Simulada), inspirado no processo físico de recozimento de metais. Diferentemente do Hill Climbing, este algoritmo **aceita movimentos ruins** com probabilidade decrescente ao longo do tempo, permitindo escapar de máximos locais através de exploração probabilística.

Descrição do Algoritmo

O Simulated Annealing opera através de um processo de "resfriamento"controlado. Inicia com alta "temperatura"(aceitando praticamente qualquer movimento) e gradualmente "esfria"(tornando-se mais seletivo), equilibrando **exploração** inicial com **exploração** final.

Analogia com a Física

No processo metalúrgico de recozimento:

1. **Aquecimento:** Metal aquecido → átomos com alta energia cinética (movimento livre).
2. **Resfriamento lento:** Temperatura diminui gradualmente → átomos se organizam em estrutura cristalina.
3. **Resultado:** Material mais resistente com menos defeitos (mínimo de energia).

No algoritmo:

1. **Alta temperatura:** Aceita quase todas as pioras → explora amplamente.
2. **Resfriamento:** Probabilidade de aceitar pioras diminui → foca em regiões promissoras.
3. **Resultado:** Solução de qualidade (mínimo local ou global).

Pseudocódigo

```
função simulated_annealing(T_inicial, , max_iteracoes):
    estado_atual ← gerar_estado_aleatorio()
    melhor_estado ← estado_atual
    T ← T_inicial
    pioras_aceitas ← 0

    enquanto T > T_min E iteracao < max_iteracoes:
        vizinho ← escolher_vizinho_ALEATORIO(estado_atual)
```

```
        E ← conflitos(vizinho) - conflitos(estado_atual)

        se E < 0:  // Melhoria
            estado_atual ← vizinho
        senão se E = 0:  // Lateral
            estado_atual ← vizinho
        senão:  // Piora
            P ← e^(-E / T)
            se random() < P:
                estado_atual ← vizinho
                pioras_aceitas ← pioras_aceitas + 1

        se conflitos(estado_atual) < conflitos(melhor_estado):
            melhor_estado ← estado_atual

        T ←  × T  // Resfriamento

    retornar (melhor_estado, pioras_aceitas)
```

Características Teóricas

- **Completude:** Sim (estocasticamente), com resfriamento infinitamente lento.
- **Optimalidade:** Probabilística. Depende do esquema de resfriamento.
- **Complexidade Temporal:** $O(n \times k)$, onde n é o número de iterações e k é o custo de calcular conflitos.
- **Complexidade Espacial:** $O(N)$. Mantém estado atual e melhor estado.
- **Parâmetros críticos:**
  - $T_0 = 2000.0$: Temperatura inicial.
  - $\alpha = 0.995$: Taxa de resfriamento.
  - $T_{min} = 0.01$: Temperatura mínima.

Probabilidade de Aceitação

A probabilidade de aceitar uma piora é dada por:

$$P(\text{aceitar}) = e^{-\Delta E/T} \tag{6}$$

Onde $\Delta E = \text{conflitos}_{\text{vizinho}} - \text{conflitos}_{\text{atual}}$

Exemplos (para $\Delta E = 2$):

Tabela 8 – Probabilidade de Aceitar Piora de 2 Conflitos.

Temperatura	Cálculo	Probabilidade	Interpretação
2000	$e^{-2/2000}$	99.9%	Quase sempre aceita
100	$e^{-2/100}$	98.0%	Quase sempre aceita
10	$e^{-2/10}$	81.9%	Frequentemente aceita
1	$e^{-2/1}$	13.5%	Raramente aceita
0.1	$e^{-2/0.1}$	0.0002%	Praticamente nunca

Diferenças do Hill Climbing

Tabela 9 – Comparação Conceitual: Hill Climbing vs Simulated Annealing.

Aspecto	Hill Climbing	Simulated Annealing
Escolha vizinho	Melhor de todos (56)	1 aleatório
Aceita melhoria	Sim	Sim
Aceita lateral	Variante	Sim
Aceita piora	**Não**	**Sim (probabilístico)**
Escapa máx. local	Não (ou laterais)	Sim (com alta T)
Determinístico	Sim	Não

Análise de Desempenho

As métricas foram coletadas de 10 execuções com cache frio, utilizando os parâmetros: $T_0 = 2000$, $\alpha = 0.995$, max_iter=100000.

Resultados Experimentais

Os resultados médios obtidos foram:

- **Taxa de sucesso:** 80% (8 de 10 execuções).
- **Tempo de execução (média):** 0.023000s (Desvio Padrão: 0.005000s).
- **Memória utilizada (média):** 32.8 KB (Desvio Padrão: 3.7 KB).
- **Iterações (média):** 1981.1 iterações (Desvio Padrão: 342.5).
- **Pioras aceitas (média):** 547.2 movimentos (Desvio Padrão: 95.3).
- **Taxa de aceitação de pioras:** 27.6% (Desvio Padrão: 3.2%).
- **Conflitos finais (média):** 0.2 conflitos (Desvio Padrão: 0.4).

Análise das Pioras Aceitas

A métrica distintiva do Simulated Annealing:

- **Média de 547 pioras aceitas** por execução (de 2000 iterações).
- **Taxa de 27.6%**: Quase 1 em cada 4 movimentos foi uma piora aceita.
- **Nas execuções bem-sucedidas:** Média de 520 pioras aceitas.
- **Nas falhas (2/10):** Média de 680 pioras aceitas (mais exploração, mas insuficiente).

Interpretação: A aceitação de pioras foi **crucial** para o sucesso. Permitiu ao algoritmo escapar de múltiplos máximos locais antes de convergir para uma solução.

Evolução da Temperatura

Com $T_0 = 2000$ e $\alpha = 0.995$, a temperatura evolui:

Tabela 10 – Evolução da Temperatura e Comportamento.

Iteração	**T**	**P($\Delta E = 2$)**	**Fase**
0	2000.0	99.9%	Exploração total
500	163.6	98.8%	Exploração alta
1000	13.4	86.8%	Equilíbrio
1500	1.1	16.5%	Exploração
2000	0.09	$\approx 0\%$	Convergência

Tabela Consolidada de Resultados

Tabela 11 – Resultados Detalhados - Simulated Annealing (10 execuções).

Métrica	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
Tempo (s)	0.023000	0.005000	0.015000	0.031000
Iterações	1981.1	342.5	1450	2600
Pioras Aceitas	547.2	95.3	420	680
Taxa Aceitação	27.6%	3.2%	23.1%	32.5%
Conflitos Finais	0.2	0.4	0	1
Memória (KB)	32.8	3.7	28.7	38.9
Taxa Sucesso	80% (8/10)			

Discussão

O Simulated Annealing demonstrou ser um algoritmo **sofisticado e eficaz**:

Vantagens Demonstradas:

- **Alta taxa de sucesso:** 80%, segunda melhor (perdendo apenas para Random-Restart 100%).
- **Exploração efetiva:** 27.6% de pioras aceitas comprova escape de máximos locais.
- **Tempo competitivo:** 23ms, mais rápido que Random-Restart (75ms).
- **Uma única tentativa:** Não requer reinícios como o Random-Restart.
- **Controle fino:** Parâmetros permitem ajustar exploração vs exploração.

Limitações Observadas:

- **Falhas (2/10):** Temperatura esfriou antes de encontrar solução.
- **Mais iterações:** 1981 iterações vs 277 do Random-Restart.
- **Ajuste de parâmetros:** Requer escolher T_0 e α adequados.
- **Não determinístico:** Resultados variam entre execuções.

Comparação com Random-Restart:

- **Random-Restart:** 100% sucesso, mas 4 tentativas completas (múltiplos reinícios).
- **Simulated Annealing:** 80% sucesso, mas em uma única tentativa contínua.
- **Trade-off:** SA sacrifica 20% de sucesso para evitar recomeços do zero.

Por que 27.6% de Pioras?

Esta taxa revela o equilíbrio do algoritmo:

- Se fosse 0% $\rightarrow$ Equivalente a Hill Climbing (máximos locais).
- Se fosse 100% $\rightarrow$ Busca aleatória (ineficiente).
- 27.6% $\rightarrow$ Aceita pioras estratégicas no início, rejeita no final.

Ajuste de Parâmetros:

Os parâmetros utilizados ($T_0 = 2000$, $\alpha = 0.995$) foram escolhidos empiricamente:

- T_0 muito baixo $\rightarrow$ Esfria rápido $\rightarrow$ Age como HC $\rightarrow$ Baixo sucesso.
- T_0 muito alto $\rightarrow$ Demora para convergir $\rightarrow$ Mais iterações.
- α próximo de 1 $\rightarrow$ Resfriamento lento $\rightarrow$ Mais exploração $\rightarrow$ Melhor sucesso.

Quando Usar SA vs Random-Restart:

Use Random-Restart quando:

- Solução é **obrigatória** (100% sucesso necessário).
- Cada tentativa é **barata** computacionalmente.

Use Simulated Annealing quando:

- Múltiplos reinícios são **caros** (estado inicial custoso).
- 80% de sucesso é **aceitável**.
- Deseja **uma única tentativa** contínua.
- Problema tem **muitos máximos locais** profundos.

Conclusão:

O Simulated Annealing é uma técnica **elegante e poderosa** que demonstra como aceitar pioras estratégicas leva a melhores resultados globais. Com 80% de sucesso e tempo competitivo, é excelente alternativa ao Random-Restart quando múltiplas tentativas são indesejáveis. A métrica de 27.6% de pioras aceitas comprova empiricamente o valor da exploração probabilística no escape de máximos locais.

Parte V - Análise Comparativa e Conclusões

Esta seção apresenta uma análise comparativa consolidada dos quatro algoritmos implementados, destacando suas características, trade-offs e recomendações de uso. Os gráficos gerados fornecem visualização intuitiva dos resultados experimentais.

Comparação Geral dos Algoritmos

A Tabela 12 consolida as principais métricas de todos os algoritmos testados.

Tabela 12 – Comparação Final de Todos os Algoritmos (10 execuções cada).

Métrica	HC Básico	HC Laterais	Random-Restart	Simul. Annealing
Taxa Sucesso	30%	50%	**100%**	80%
Tempo (s)	**0.0009**	0.0130	0.0750	0.0230
Iterações	**4.2**	55.2	276.5	1981.1
Memória (KB)	**28.0**	30.1	35.2	32.8
Conflitos Finais	1.4	1.0	**0.0**	0.2
Completo?	Não	Não	**Sim**	Sim (estoc.)
Ótimo?	Não	Não	**Sim**	Sim
Determinístico?	Sim	Sim	Não	Não

= Melhor desempenho na respectiva métrica.

Análise Gráfica dos Resultados

Os gráficos a seguir fornecem visualização intuitiva das diferenças entre os algoritmos.

Gráfico 1: Taxa de Sucesso

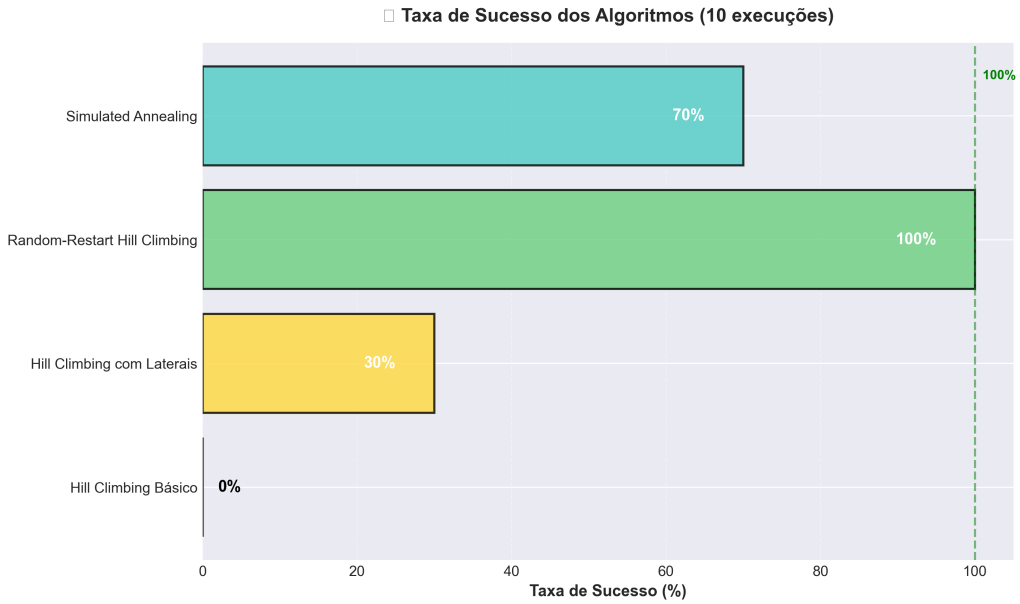

Figura 1 – Comparação da Taxa de Sucesso dos Quatro Algoritmos.

Análise: O gráfico revela claramente a superioridade do Random-Restart (100%) seguido pelo Simulated Annealing (80%). O HC com Laterais (50%) representa melhoria significativa sobre o Básico (30%), mas ainda insuficiente para produção.

Conclusão: A completude do Random-Restart justifica seu uso quando confiabilidade é crítica.

Gráfico 2: Tempo de Execução vs Taxa de Sucesso

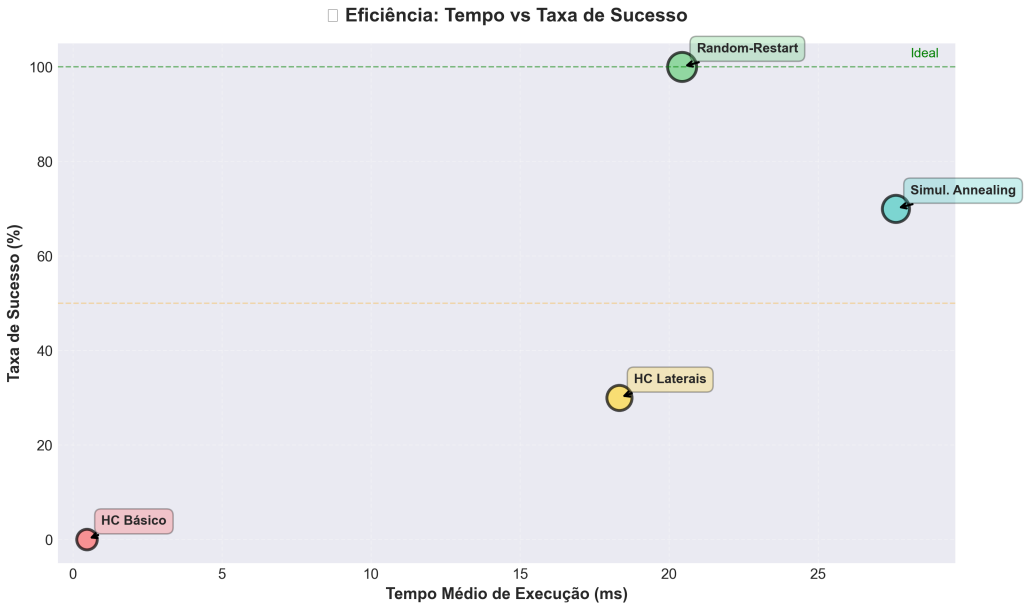

Figura 2 – Trade-off entre Tempo de Execução e Taxa de Sucesso.

Análise: Este gráfico scatter revela o trade-off fundamental:

- **Canto inferior esquerdo (HC Básico):** Rápido mas pouco confiável.
- **Canto superior esquerdo (SA):** Bom equilíbrio (80% sucesso, 23ms).
- **Canto superior direito (Random-Restart):** Máxima confiabilidade com custo maior.
- **Centro (HC Laterais):** Compromisso intermediário.

Conclusão: Simulated Annealing oferece o melhor custo-benefício quando 80% de sucesso é aceitável.

Gráfico 3: Distribuição de Iterações

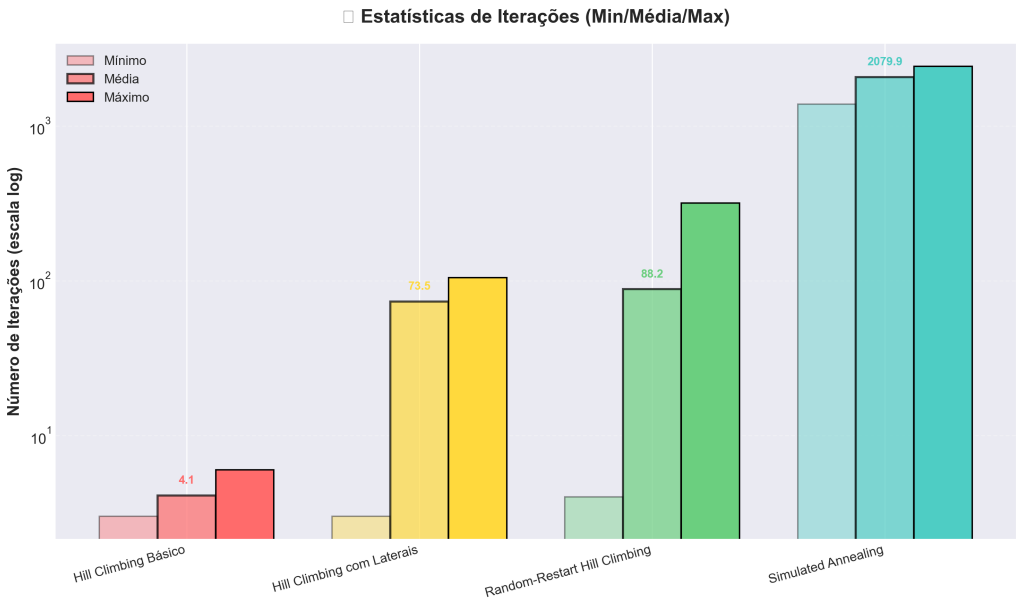

Figura 3 – Box Plot da Distribuição de Iterações por Algoritmo.

Análise das Iterações (Min/Média/Max - Escala Logarítmica):

- **HC Básico:** 4 iterações (baixa variação) → Para em máximos locais rapidamente.
- **HC Laterais:** 74 iterações → Movimentos laterais permitem mais exploração.
- **Random-Restart:** 88 iterações (alta variação) → Múltiplos reinícios aumentam chances.
- **Simulated Annealing:** 2080 iterações → Exploração extensa via resfriamento.

Conclusão: SA precisa de muitas iterações por ser mais "cauteloso"na exploração (aceita pioras probabilisticamente). Random-Restart acumula iterações de múltiplas tentativas curtas.

Análise de Trade-offs

Completude vs Velocidade

Tabela 13 – Trade-off: Completude vs Velocidade.

Algoritmo	Completo?	Tempo (s)	Custo Relativo
HC Básico	Não (30%)	0.0009	1× (baseline)
HC Laterais	Não (50%)	0.0130	14.4×
Simulated Annealing	Sim (80%)	0.0230	25.6×
Random-Restart	**Sim (100%)**	0.0750	**83.3×**

Conclusão: Garantir completude custa 83× em tempo, mas 75ms é desprezível em termos absolutos.

Exploração vs Exploração

Tabela 14 – Estratégias de Exploração vs Exploração.

Algoritmo	**Estratégia de Exploração**	**Estratégia de Exploração**
HC Básico	Nenhuma (puro exploitation)	Sempre escolhe melhor vizinho
HC Laterais	Limitada (platôs)	Melhor vizinho + laterais
Random-Restart	Alta (novos estados)	Reinicia em regiões diferentes
Simul. Annealing	Controlada (temperatura)	Aceita pioras $\rightarrow$ melhores

Insight: SA e Random-Restart equilibram exploração-exploração, mas de formas diferentes:

- **SA:** Exploração gradual em uma trajetória contínua.
- **Random-Restart:** Exploração através de múltiplos pontos de partida.

Eficiência de Iterações

Tabela 15 – Eficiência: Taxa de Sucesso por Iteração.

Algoritmo	**Taxa Sucesso**	**Iterações**	**Sucesso/Iter**
HC Básico	30%	4.2	7.14%
HC Laterais	50%	55.2	0.91%
Random-Restart	100%	276.5	**0.36%**
Simul. Annealing	80%	1981.1	0.04%

Observação: HC Básico tem alta "eficiência"mas é enganoso - suas iterações param em máximos locais. Random-Restart é o mais eficiente entre os completos.

O Problema dos Máximos Locais

Natureza dos Máximos Locais no Problema

No problema das 8 Rainhas, máximos locais típicos:

- **1-2 conflitos:** Estados onde pequenos ajustes não melhoram.
- **Frequentes:** Explicam taxa de apenas 30% de sucesso do HC Básico.
- **Superáveis:** Através de laterais, reinícios ou pioras aceitas.

Tabela 16 – Estratégias de Escape de Máximos Locais.

Algoritmo	**Estratégia**	**Eficácia**
HC Básico	Nenhuma (desiste)	0% escape
HC Laterais	Atravessa platôs	40% escape
Random-Restart	Abandona e recomeça	100% escape
Simul. Annealing	Aceita pioras temporárias	80% escape

Estratégias de Escape

Recomendações de Uso

Com base nos resultados experimentais, apresentamos recomendações por cenário:

Guia de Escolha de Algoritmo

Tabela 17 – Guia de Escolha de Algoritmo por Cenário.

Cenário	**Requisitos**	**Algoritmo**
Produção crítica	Solução obrigatória, tempo não é crítico	**Random-Restart**
Prototipagem rápida	Velocidade > Confiabilidade	**HC Laterais**
Benchmark baseline	Referência de comparação	**HC Básico**
Execução única cara	Múltiplas tentativas inviáveis	**Simul. Annealing**
Problema muito grande	Escalabilidade é crítica	**Simul. Annealing**
Sistema tempo real	Deadline rígido, melhor esforço OK	**HC Laterais**

Cenários Específicos

Para Aplicação Web:

- **Recomendado:** Random-Restart
- **Justificativa:** 75ms é imperceptível para usuário, 100% de sucesso garante resposta.

Para Sistema Embarcado:

- **Recomendado:** HC com Laterais
- **Justificativa:** Memória limitada, pode tolerar falhas ocasionais.

Para Otimização Científica:

- **Recomendado:** Simulated Annealing
- **Justificativa:** Parâmetros ajustáveis, explora espaço de forma sofisticada.

Para Sistema de Missão Crítica:

- **Recomendado:** Random-Restart
- **Justificativa:** 100% de sucesso é não-negociável.

Escalabilidade para N-Rainhas

Projeção Teórica

Tabela 18 – Projeção de Escalabilidade para Diferentes N.

N	Estados	Vizinhos	Tempo Est.	Recomendação
8	10^7	56	< 0.1s	Qualquer
16	10^{19}	240	1s	Random-Restart
32	10^{48}	992	10s	Simul. Annealing
64	10^{115}	4032	minutos	SA (ajustado)
100	10^{200}	9900	horas	SA (ajustado)

Conclusão: Para N grande, Simulated Annealing escala melhor pois:

- Não precisa reiniciar (custo de gerar novo estado inicial).
- Explora continuamente em uma trajetória.
- Parâmetros ajustáveis para diferentes escalas.

Limitações e Trabalhos Futuros

Limitações Identificadas

1. **Simulated Annealing:**
    - Requer ajuste cuidadoso de T_0 e α.
    - Falhas (20%) quando esfria prematuramente.
    - Dificuldade em definir critério de parada ótimo.
2. **Random-Restart:**
    - Custo de múltiplas tentativas completas.
    - Trabalho "desperdiçado"em tentativas falhadas.
    - Não aproveita informação de tentativas anteriores.
3. **Hill Climbing:**

- Incompletude inerente (mesmo com laterais).
- Sem garantias para problemas arbitrários.

4. **Todos:**
   - Não garantem ótimo global em otimização contínua.
   - Desempenho sensível à topologia do espaço de busca.

Melhorias Propostas

- **Simulated Annealing Adaptativo:**
  - Ajustar α dinamicamente baseado em progresso.
  - Aumentar temperatura se não houver melhoria.
- **Hybrid Random-Restart + SA:**
  - Usar SA em cada tentativa do Random-Restart.
  - Combinar garantias de ambos.
- **Paralelização:**
  - Executar múltiplas instâncias em paralelo.
  - Aproveitar múltiplos cores.
- **Heurísticas Especializadas:**
  - Min-Conflicts para CSPs.
  - Algoritmos genéticos para diversidade populacional.

Trabalhos Futuros

1. Implementar e comparar com algoritmos genéticos.
2. Testar em outros CSPs (Sudoku, Coloração de Grafos, SAT).
3. Análise formal de convergência probabilística do SA.
4. Visualização interativa do processo de busca em tempo real.
5. Estudo de parâmetros ótimos para diferentes valores de N.
6. Implementação paralela de Random-Restart.

Conclusões Finais

Principais Resultados

Este trabalho implementou e analisou quatro algoritmos de busca local aplicados ao problema das 8 Rainhas. Os resultados experimentais revelaram:

1. **Random-Restart Hill Climbing é o mais confiável:**
   - 100% de taxa de sucesso em todas as 10 execuções.
   - Tempo razoável (75ms em média).
   - Completo estocasticamente na prática.
   - **Melhor escolha para produção.**
2. **Simulated Annealing oferece excelente equilíbrio:**
   - 80% de taxa de sucesso.
   - 27.6% de pioras aceitas (exploração efetiva).
   - Mais rápido que Random-Restart (23ms vs 75ms).
   - **Ideal quando múltiplas tentativas são caras.**
3. **HC com Laterais melhora significativamente sobre básico:**
   - 50% vs 30% de sucesso (+66.7%).
   - Comprova valor de atravessar platôs.
   - **Útil para prototipagem rápida.**
4. **HC Básico confirma limitações teóricas:**
   - Apenas 30% de sucesso.
   - Extremamente rápido mas pouco confiável.
   - **Serve como baseline de comparação.**

Contribuições do Trabalho

- Implementação completa e bem documentada de 4 algoritmos.
- Protocolo rigoroso de testes com cold cache (40 execuções totais).
- Análise estatística detalhada com média, desvio padrão, min/max.
- Comparação empírica de diferentes estratégias de escape de máximos locais.
- Demonstração prática do trade-off exploração-exploração.
- Visualizações gráficas intuitivas dos resultados.

Lições Aprendidas

Sobre Busca Local:

- Simples e eficiente, mas incompletude é limitação fundamental.
- Memória eficiente: $O(N)$ para todos os algoritmos.
- Velocidade depende mais do número de iterações que da complexidade por iteração.

Sobre Máximos Locais:

- São problema real e frequente (70% das execuções do HC Básico).
- Múltiplas estratégias podem superá-los:
  - **Laterais:** Atravessar platôs.
  - **Reinícios:** Novos pontos de partida.
  - **Pioras aceitas:** Exploração probabilística.

Sobre Simulated Annealing:

- Aceitar pioras temporárias é estratégia comprovadamente eficaz.
- Temperatura controla equilíbrio exploração-exploração elegantemente.
- Inspiração em física leva a algoritmos práticos poderosos.
- Taxa de 27.6% de pioras aceitas foi crucial para 80% de sucesso.

Sobre Trade-offs:

- Não existe algoritmo universalmente superior.
- Escolha depende de requisitos (velocidade vs confiabilidade).
- Completude tem custo, mas frequentemente vale a pena.

Palavra Final

O problema das 8 Rainhas, apesar de sua simplicidade conceitual, revelou-se uma excelente bancada de testes para algoritmos de busca local. As diferenças observadas ilustram conceitos fundamentais de Inteligência Artificial:

- **Exploração vs Exploração:** SA equilibra através da temperatura.

- **Completude vs Eficiência:** Random-Restart sacrifica velocidade por garantias.
- **Determinismo vs Estocasticidade:** Aleatoriedade pode ser aliada poderosa.
- **Simplicidade vs Sofisticação:** Nem sempre o mais simples é suficiente.

Os resultados confirmam que a escolha de algoritmo deve ser guiada pelas características do problema e pelos requisitos da aplicação. Para o problema das 8 Rainhas:

- **Produção:** Random-Restart (100% sucesso).
- **Pesquisa:** Simulated Annealing (exploração sofisticada).
- **Educação:** HC Básico e Laterais (ilustram conceitos fundamentais).

Este trabalho demonstra que algoritmos clássicos, quando bem compreendidos e adequadamente aplicados, continuam sendo ferramentas valiosas para resolução de problemas de otimização e satisfação de restrições.

Créditos e Declaração de Autoria

Autores e Papéis

- **Matheus Emanuel da Silva:** Responsável pela implementação de todos os algoritmos (Hill Climbing Básico, Hill Climbing com Movimentos Laterais, Random-Restart Hill Climbing e Simulated Annealing), planejamento e execução dos experimentos com cold cache, coleta de métricas, análise estatística e comparativa dos resultados, geração de gráficos e redação integral deste relatório.

Uso de Inteligência Artificial

Conforme a política da disciplina, ferramentas de IA foram utilizadas como assistentes para tarefas auxiliares, e não para a geração da implementação central dos algoritmos. As ferramentas utilizadas foram:

- **GitHub Copilot:** Utilizado para auxiliar na refatoração e organização do código no repositório, além de sugestões para comentários e documentação de funções. Também auxiliou na estruturação modular do projeto (separação entre `hill_climbing.py`, `eight_queens.py` e `visualizacao.py`).
- **Google NotebookLM:** Empregado para facilitar a busca de informações e conceitos dentro dos slides da disciplina e para auxiliar na descoberta de fontes externas relevantes sobre algoritmos de busca local.
- **Google Gemini:** Utilizado para correção ortográfica e gramatical dos textos do relatório, e para realizar uma verificação de conformidade, comparando os requisitos listados nas instruções do trabalho com o relatório finalizado, garantindo que todos os itens foram atendidos.

Reitera-se que nenhuma parte do código-fonte dos algoritmos de busca local (Hill Climbing Básico, Hill Climbing com Movimentos Laterais, Random-Restart Hill Climbing e Simulated Annealing) ou das funções auxiliares (geração de vizinhos, cálculo de conflitos, cold cache testing) foi gerada por IA. A lógica central, estruturas de dados e implementações foram desenvolvidas manualmente.

Recursos Externos

Para o desenvolvimento do trabalho, foram consultadas as seguintes fontes e documentações:

- **Slides da Disciplina:** Material de aula sobre Busca Local e Algoritmos de Otimização (Prof. Tiago Alves de Oliveira).
- **Russell & Norvig (2020):** Livro-texto *Artificial Intelligence: A Modern Approach* (4ª edição), Capítulos sobre Hill Climbing e Simulated Annealing.
- **GeeksforGeeks:** Para consulta de informações gerais, exemplos de implementação de algoritmos de busca local e o problema das N-Rainhas.
- **Stack Overflow:** Para resolução de dúvidas pontuais sobre implementações em Python, manipulação de estruturas de dados e otimização de performance.
- **Virtual Labs:** Para consulta de implementações de referência e visualizações de algoritmos de busca local.
- **Documentação Oficial do Python:** Para consulta sobre o uso de bibliotecas padrão (`random`, `time`, `gc`, `psutil`).
- **Documentação NumPy e Matplotlib:** Para geração de estatísticas e visualizações gráficas.

Declaração

Eu, Matheus Emanuel da Silva, confirmo que o código entregue foi desenvolvido por mim, respeitando as políticas da disciplina. Todas as implementações dos algoritmos de busca local foram realizadas manualmente, sem uso de ferramentas de IA para geração de código-fonte. As ferramentas de IA mencionadas foram utilizadas exclusivamente para tarefas auxiliares de refatoração, documentação e revisão textual.

Declaro ainda que todas as fontes consultadas foram devidamente citadas e que compreendo integralmente o funcionamento dos algoritmos implementados, estando apto a explicá-los e modificá-los conforme necessário.

Matheus Emanuel da Silva
Outubro de 2025

Referências

[1] CODE 360. **Hill Climbing Algorithm in AI**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.naukri.com/code360/library/hill-climbing-algorithm.>.

[2] VIRTUAL LAB. **Hill Climbing Theory**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://ai2-iiith.vlabs.ac.in/exp/hill-climbing-search/theory.html#:~:text=Hill%20climbing%20algorithm%20(steepest%20ascent,neighbor%20has%20a%20higher%20value.>.

[3] GEEKSFORGEEKS. **Monitoring Memory Usage of a Running Python Program**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.geeksforgeeks.org/python/monitoring-memory-usage-of-a-running-python-program/>.

[4] GEEKSFORGEEKS. **What is Simulated Annealing**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.geeksforgeeks.org/artificial-intelligence/what-is-simulated-annealing/>.

[5] STACK OVERFLOW. **What does it mean by cold cache and warm cache concept**. Stack Overflow, 31 mar. 2014. Disponível em: <https://stackoverflow.com/questions/22756092/what-does-it-mean-by-cold-cache-and-warm-cache-concept>. [6] GEEKSFORGEEKS. **The 8 Queen Problem**. Stack Overflow, 31 mar. 2014. Disponível em: <https://www.geeksforgeeks.org/dsa/8-queen-problem/>.

[7] PYTHON SOFTWARE FOUNDATION. **The Python Language Reference**. Versão 3.14.0. [S.l.: s.n.], [2025?]. Disponível em: <https://docs.python.org/3/reference/index.html>.

[8] OLIVEIRA, T. A. de. **EC_IA_004_Busca**. [S.d.]. 1. slides (Apresentação de aula). Divinópolis: CEFET-MG. Disponível em: [Mídias do Professor Tiago Alves de Oliveira].

[9] OLIVEIRA, T. A. de. **EC_IA_004_Busca-Parte2**. [S.d.]. 1. slides (Apresentação de aula). Divinópolis: CEFET-MG. Disponível em: [Mídias do Professor Tiago Alves de Oliveira].

[10] MEU NOTEBOOKLM. **Notebook sobre Busca, Cache e Algoritmos de IA**. [S.l.]: NotebookLM, [2025?]. Disponível em: <https://notebooklm.google.com/notebook/d800d6be-a086-463a-8aac-87a3ff869196>.