Análise Comparativa de Técnicas de Projeto de Algoritmos Aplicadas aos Problemas das N-Rainhas e do Passeio do Cavalo

Daniel A. Dores[1], **Carlos A. Silva**[1]

[1]Departamento de Informática – Instituto Federal de Minas Gerais (IFMG)
CEP 34.590-390 – Sabará, MG – Brasil

danieldores0@gmail.com, carlos.silva@ifmg.edu.br

Abstract. *This work presents the implementation and comparative analysis of three approaches applied to the N-Queens and Knight's Tour combinatorial problems: Backtracking, Genetic Algorithms, and Greedy Search based on Warnsdorff's heuristic. All solutions were implemented in C++ and independently evaluated in terms of execution time, success rate, and computational effort. In the Knight's Tour on an* 8×8 *board, Warnsdorff's heuristic achieved the best practical performance, with a success rate above* 95% *and average execution time below* $1\,ms$*. Backtracking reached* 100% *success, although with significantly higher search cost. The Genetic Algorithm obtained a success rate of approximately* 50%*, while the simple greedy version did not produce complete solutions. In the N-Queens problem (*$N = 8$*), both Backtracking and the Genetic Algorithm achieved* 100% *success, with execution times of only a few milliseconds. These results highlight the complementarity among exact, heuristic, and metaheuristic paradigms, supporting the selection of suitable techniques in combinatorial optimization and in educational settings focused on algorithm design.*

Resumo. *Este trabalho apresenta a implementação e a análise comparativa de três abordagens aplicadas aos problemas combinatórios das N-Rainhas e do Passeio do Cavalo: Backtracking, Algoritmos Genéticos e uma Busca Gulosa que inclui um experimento adicional com a heurística de Warnsdorff. Todas as soluções foram desenvolvidas em C++ e avaliadas de forma independente quanto ao tempo de execução, taxa de sucesso e esforço computacional. No Passeio do Cavalo, a versão gulosa simples não obteve sucesso, enquanto a heurística de Warnsdorff, utilizada como experimento complementar, apresentou o melhor desempenho prático, com alta taxa de sucesso e tempo de processamento muito baixo. O Backtracking também encontrou soluções completas, porém com maior custo de busca. O Algoritmo Genético apresentou desempenho intermediário, com taxa de solução inferior à do Backtracking, refletindo a sensibilidade do método à parametrização e ao domínio do problema. No problema N-Rainhas (N = 8), o Backtracking atingiu alta taxa de desempenho com 100% de sucesso, enquanto o Algoritmo Genético obteve resultados elevados, com 100% de sucesso e convergência em poucos milissegundos. Esses resultados reforçam a complementaridade entre paradigmas exatos, heurísticos e metaheurísticos, auxiliando na escolha de técnicas adequadas em contextos de otimização combinatória e no ensino de projeto de algoritmos.*

1. Introdução

A resolução de problemas combinatórios é um tema amplamente estudado na Ciência da Computação, com aplicações em otimização, inteligência artificial, teoria dos grafos e planejamento de sistemas. Entre os desafios mais tradicionais nesse domínio destacam-se o Problema do Passeio do Cavalo e o Problema das N-Rainhas, ambos inspirados no jogo de xadrez. Esses problemas se tornaram referências clássicas para avaliação de algoritmos de busca, heurísticas e metaheurísticas, devido à combinação entre estrutura bem definida e elevada complexidade [Costa and de Sá 2015].

O Passeio do Cavalo consiste em encontrar uma sequência de movimentos válidos da peça cavalo sobre um tabuleiro de xadrez, de modo que todas as casas sejam visitadas exatamente uma vez. Já o Problema das N-Rainhas busca posicionar N rainhas em um tabuleiro $N \times N$ de forma que nenhuma peça ataque as demais, evitando conflitos em linhas, colunas e diagonais. Tais problemas apresentam alta explosão combinatória e são amplamente empregados como *benchmarks* para testar estratégias de busca e otimização [Witt 2020, Costa 2018, Besa et al. 2022, Moghimi and Amini 2024, Martinjak and Golub 2007].

Neste contexto, o presente trabalho investiga o desempenho de três abordagens aplicadas à resolução desses problemas: *Backtracking*, *Algoritmos Genéticos* e *Algoritmos Gulosos*. Adicionalmente, foi incorporada a heurística de Warnsdorff [Warnsdorff 1823] como um *plus* experimental, permitindo analisar uma variação clássica e eficiente dentro da categoria gulosa, enriquecendo a comparação entre as técnicas. O primeiro método representa uma abordagem exata, o segundo utiliza princípios evolutivos para explorar o espaço de soluções e o terceiro realiza escolhas locais visando rapidez, ainda que sem garantia de otimalidade.

Este trabalho tem como objetivo principal comparar o desempenho relativo das abordagens de Backtracking, Busca Gulosa (com a heurística de Warnsdorff) e Algoritmos Genéticos na resolução dos problemas do Passeio do Cavalo e das N-Rainhas, buscando responder como essas técnicas se comparam entre si, como o tempo de execução e o custo computacional variam conforme o tamanho das instâncias aumenta e em que medida heurísticas, como a de Warnsdorff, podem aproximar o desempenho de métodos exatos ao reduzir o custo computacional. As soluções foram implementadas em C++ e avaliadas empiricamente quanto ao tempo de execução, taxa de sucesso e esforço de busca, permitindo uma análise quantitativa e qualitativa que destaca pontos fortes, limitações e cenários de aplicabilidade de cada abordagem.

O estudo também discute o potencial pedagógico dessas técnicas, dada sua relevância na formação em projeto e análise de algoritmos, bem como em investigações sobre otimização combinatória.

Este artigo está organizado como segue. A Seção 2 apresenta a revisão bibliográfica. A Seção 3 descreve formalmente os dois problemas estudados. A Seção 4 detalha a metodologia adotada, incluindo critérios experimentais, modelagem e implementação. A Seção 5 apresenta e discute os resultados obtidos. Por fim, a Seção 6 sintetiza as conclusões e aponta direções futuras.

2. Revisão Bibliográfica

Esta seção apresenta um panorama dos principais conceitos, abordagens e desafios relacionados aos problemas do Passeio do Cavalo e das N-Rainhas, que fundamentam o desenvolvimento deste estudo.

Os dois problemas são amplamente reconhecidos na literatura como desafios combinatórios de alta complexidade e são empregados com frequência como *benchmarks* na avaliação de algoritmos de busca, otimização e inteligência artificial. Em particular, o Problema do Passeio do Cavalo, quando formulado como uma decisão sobre a existência de um percurso em tabuleiros generalizados, é classificado como NP-completo, por se relacionar diretamente ao Problema do Caminho Hamiltoniano. Já o Problema das N-Rainhas, em sua forma clássica, não pertence às classes NP-completo ou NP-difícil, pois admite soluções construtivas em tempo polinomial para $n \geq 4$; contudo, várias de suas variantes otimizadas e restritas são reconhecidas como NP-difíceis na literatura. Além de seu valor acadêmico, ambos os problemas também apresentam aplicações em áreas como criptografia, sistemas distribuídos e planejamento automatizado [Bell and Stevens 2009, Moghimi and Amini 2024].

No caso do Passeio do Cavalo, que deriva do movimento da peça cavalo no xadrez, diversas estratégias têm sido propostas ao longo das últimas décadas. Entre elas, destaca-se a Heurística de Warnsdorff (1823), que prioriza movimentos para casas com menor número de saídas futuras [Besa et al. 2022]. Embora apresente bons resultados em muitos cenários, essa heurística sofre limitações em tabuleiros maiores ou com posições iniciais desfavoráveis. Abordagens exatas baseadas em *Backtracking* continuam sendo empregadas, embora tenham dificuldade de escalabilidade devido ao crescimento exponencial do espaço de busca. Costa et al. (2013) apresentaram uma versão aprimorada da heurística AML, denominada OctAML16, com ganhos expressivos em tabuleiros maiores [Costa and de Sá 2013]. Técnicas baseadas em metaheurísticas, como *Algoritmos Genéticos*, também têm sido estudadas e podem alcançar resultados satisfatórios quando projetadas para respeitar as restrições de movimento do cavalo [Silva et al. 2003].

O Problema das N-Rainhas consiste em posicionar N rainhas em um tabuleiro $N \times N$ de forma que nenhuma ataque outra, evitando conflitos em linhas, colunas e diagonais. A solução clássica via *Backtracking* é amplamente descrita na literatura, mas outras estratégias também têm sido exploradas, incluindo *Algoritmos Genéticos*, heurísticas gulosas e abordagens híbridas que combinam benefícios de múltiplas técnicas [Mukherjee et al. 2015]. Além de seu interesse teórico, o problema apresenta aplicações práticas em alocação de recursos, escalonamento de tarefas e organização de sistemas paralelos [Moghimi and Amini 2024, Joshuva 2024].

Este estudo posiciona-se nesse contexto ao analisar comparativamente três paradigmas distintos, a saber: *Backtracking*, *Algoritmos Genéticos* e *Busca Gulosa*, aplicados aos dois problemas. Essas técnicas foram selecionadas por representarem abordagens metodológicas complementares: *Backtracking* como método exato de exploração completa do espaço de busca, a *Busca Gulosa* como heurística clássica de baixo custo computacional, e os *Algoritmos Genéticos* como uma metaheurística robusta capaz de lidar com espaços extensos e altamente combinatórios. Embora existam outras estratégias relevantes, como *Simulated Annealing* ou otimização por colônias de formigas, essas três foram escolhidas por sua ampla utilização na literatura, simplicidade de implementação e por

permitirem contrastar diferentes níveis de exatidão, custo computacional e generalização. A partir dessa avaliação integrada, busca-se identificar padrões de eficiência e limitações de cada abordagem, contribuindo para futuras pesquisas relacionadas à otimização combinatória [Nascimento and Brasil 2023].

3. Jogos Combinatórios

Problemas inspirados nas regras do xadrez são tradicionalmente empregados como estudo de caso em algoritmos de busca e otimização. Neste trabalho, são considerados dois desses desafios clássicos: o **Problema das N-Rainhas** e o **Problema do Passeio do Cavalo**. Ambos possuem formulação simples, mas apresentam crescimento combinatório elevado, tornando-se úteis para avaliar diferentes estratégias de solução.

3.1. O Problema das N-Rainhas

O Problema das N-Rainhas consiste em posicionar N rainhas em um tabuleiro $N \times N$ de modo que nenhuma peça ataque outra, evitando conflitos em linhas, colunas e diagonais. Proposto em meados do século XIX, o problema tornou-se referência em estudos de técnicas de busca e heurísticas [Bell and Stevens 2009].

Uma forma comum de representação utiliza uma permutação $Q[1], Q[2], \ldots, Q[N]$, em que $Q[i]$ indica a coluna ocupada pela rainha na linha i. Assim, conflitos em linhas e colunas são eliminados, restando apenas a verificação de diagonais, dada por:

$$|Q[i] - Q[j]| \neq |i - j|, \quad \forall i \neq j.$$

Apesar de soluções existirem para todos os $N \geq 4$, o espaço de busca cresce exponencialmente com N, o que motiva o uso de técnicas como *Backtracking*, heurísticas e *Algoritmos Genéticos* [Mukherjee et al. 2015, Moghimi and Amini 2024]. Além do valor acadêmico, o problema é relacionado a tarefas reais de alocação sob restrição, incluindo escalonamento de recursos e posicionamento de sensores.

Devido à formulação simples e ao caráter visual direto, o problema também é utilizado em contextos didáticos para introduzir busca sistemática, heurísticas e otimização.

3.2. O Problema do Passeio do Cavalo

O Problema do Passeio do Cavalo consiste em encontrar uma sequência de movimentos válidos da peça cavalo de modo que todas as casas de um tabuleiro $N \times N$ sejam visitadas exatamente uma vez, sem repetição. A movimentação em "L" do cavalo impõe restrições estruturais que tornam o espaço de estados amplo e difícil de explorar, como ilustrado na Figura 1.

Formalmente, o problema pode ser interpretado como a busca por um caminho hamiltoniano em um grafo, em que cada casa do tabuleiro corresponde a um vértice e há uma aresta entre duas casas sempre que o movimento do cavalo for possível. Um movimento válido a partir da posição (x, y) é definido por:

$$(x, y) \to (x + m, y + n), \quad m, n \in \{-2, -1, 1, 2\}, \quad |m| \neq |n|.$$

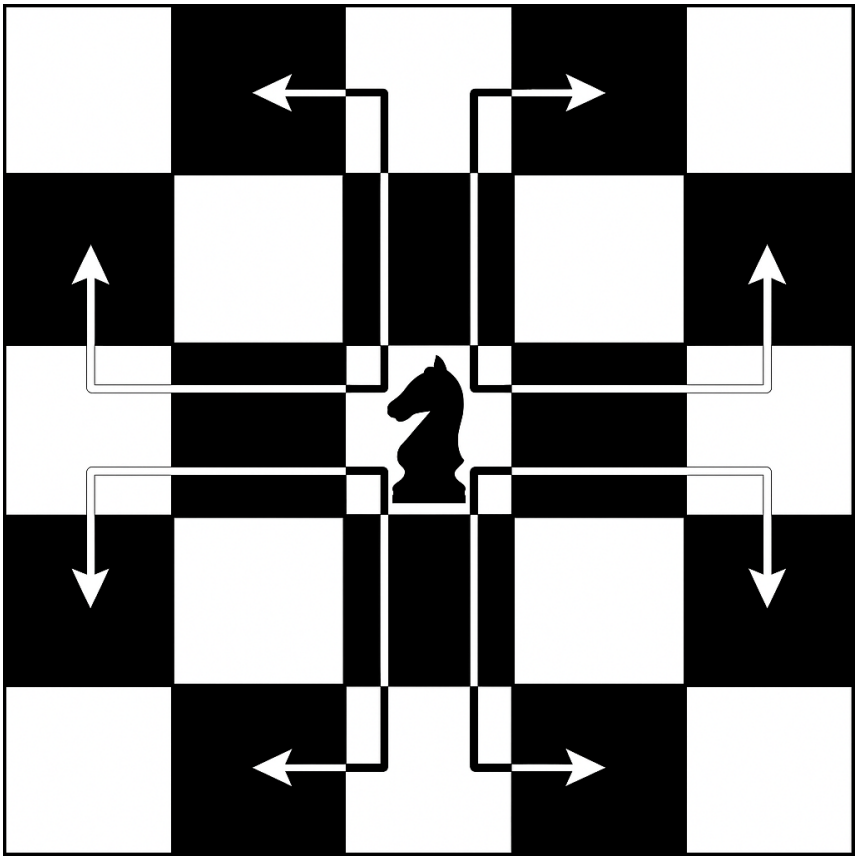

Figura 1. Representação dos movimentos possíveis da peça Cavalo no tabuleiro de xadrez. O cavalo move-se em formato de "L": duas casas em uma direção e uma casa perpendicular.

Como o número de sequências possíveis cresce rapidamente com N, métodos exaustivos tendem a se tornar custosos. Por isso, o problema é frequentemente empregado para avaliar abordagens como *Backtracking*, Heurística de Warnsdorff [Cancela and Mordecki 2006] e técnicas evolutivas [Costa and de Sá 2013].

Aplicações associadas incluem planejamento de rotas, inspeção em redes e contextos educacionais que envolvem grafos, busca e análise de complexidade.

4. Metodologia

4.1. Representação do Problema

A modelagem dos problemas do Passeio do Cavalo e das N-Rainhas utiliza uma matriz bidimensional $N \times N$ para representar o tabuleiro, em que cada célula corresponde a uma posição possível.

No **Problema das N-Rainhas**, o tabuleiro foi representado por uma matriz $N \times N$ contendo valores binários que indicam presença (1) ou ausência (0) de uma rainha. Essa forma de codificação permite verificar rapidamente se uma posição está ocupada. Um exemplo de solução para $N = 8$ é ilustrado na Figura 2.

Além disso, foi utilizada uma forma compacta de representação, em que um vetor Q armazena apenas a coluna ocupada por cada rainha. Em $Q[i] = j$, a rainha posicionada

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 2. Exemplo de representação matricial para uma solução do problema de 8-Rainhas.

na linha i encontra-se na coluna j. Essa codificação elimina conflitos em linhas e colunas, restando apenas a verificação de diagonais segundo:

$$|Q[i] - Q[j]| \neq |i - j|, \quad \forall i \neq j.$$

No **Problema do Passeio do Cavalo**, a matriz $N \times N$ registra a ordem de visitação das casas. Cada célula armazena o número do passo correspondente, e posições não visitadas são marcadas com -1. Dessa forma, é possível verificar rapidamente a validade dos movimentos e reconstruir o trajeto realizado.

A Figura 3 mostra um exemplo de passeio completo para $N = 8$, em que as casas são numeradas conforme a sequência de deslocamentos do cavalo.

$$\begin{pmatrix} 0 & 59 & 38 & 33 & 30 & 17 & 8 & 63 \\ 37 & 34 & 31 & 60 & 9 & 62 & 29 & 16 \\ 58 & 1 & 36 & 39 & 32 & 27 & 18 & 7 \\ 35 & 48 & 41 & 26 & 61 & 10 & 15 & 28 \\ 42 & 57 & 2 & 49 & 40 & 23 & 6 & 19 \\ 47 & 50 & 45 & 54 & 25 & 20 & 11 & 14 \\ 56 & 43 & 52 & 3 & 22 & 13 & 24 & 5 \\ 51 & 46 & 55 & 44 & 53 & 4 & 21 & 12 \end{pmatrix}$$

Figura 3. Representação matricial de um Passeio do Cavalo para um tabuleiro 8×8.

Essas representações (binária para N-Rainhas e sequencial para o Passeio do Cavalo) foram escolhidas por favorecer operações simples de leitura, marcação e verificação de restrições, além de facilitar o uso das abordagens consideradas neste estudo.

4.2. Abordagens Consideradas

Três técnicas foram avaliadas: *Backtracking*, heurística de Warnsdorff (abordagem gulosa) e *Algoritmos Genéticos*. Cada método foi primeiro adaptado ao Passeio do Cavalo e, posteriormente, ao N-Rainhas, considerando suas particularidades.

4.2.1. Algoritmo Genético

No *Algoritmo Genético* (AG), cada indivíduo representa uma solução candidata, cuja aptidão (*fitness*) mede quão próxima essa solução está do objetivo, seja pelo número de conflitos (no caso das N-Rainhas) ou pelo tamanho do passeio válido (no Passeio do Cavalo). A evolução populacional ocorre pela aplicação iterativa de seleção, cruzamento e mutação, seguida de avaliação dos indivíduos gerados, até que um critério de parada seja atingido.

Representação e Decodificação.

- **N-Rainhas:** cada indivíduo é representado por um vetor Q em que $Q[i] = j$ indica que a rainha da coluna i está posicionada na linha j. O *decoder* interpreta o vetor diretamente como uma configuração do tabuleiro e calcula a aptidão a partir do número de conflitos diagonais existentes. Como cada coluna contém exatamente uma rainha, não há necessidade de correções estruturais, apenas da avaliação dos conflitos.
- **Passeio do Cavalo:** cada indivíduo é representado como uma sequência de posições (x_k, y_k) que o cavalo deve visitar. O *decoder* reconstrói o percurso verificando, passo a passo, se cada movimento é válido segundo as regras do cavalo. Caso algum movimento seja inválido, o decoder aplica rotinas de reparo, como substituição por movimentos válidos mais próximos, remoção de ciclos ou extensão do caminho até atingir o tamanho desejado. A aptidão é então calculada pelo comprimento do caminho válido obtido, penalizando movimentos incorretos ou impossíveis.

Operadores. No problema N-Rainhas, foram utilizados operadores que preservam a estrutura de permutação, enquanto a mutação troca duas posições. Para o Passeio do Cavalo, cruzamento e mutação são seguidos de checagem para manter a validade da sequência.

Algorithm 1 Algoritmo Genético para o Problema das N-Rainhas

```
Inicializar população P com vetores Q de tamanho n
enquanto critério de parada não atendido faça
    para cada indivíduo Q ∈ P faça
        tab ← DecoderNRainhas(Q)
        apt(Q) ← AvaliarConflitos(tab)
    fim para
    Selecionar indivíduos
    Aplicar cruzamento e mutação gerando P'
    para cada Q' ∈ P' faça
        tab' ← DecoderNRainhas(Q')
        apt(Q') ← AvaliarConflitos(tab')
    fim para
    Atualizar população P ← P'
fim enquanto
retorne melhor solução encontrada
```

Algorithm 2 Algoritmo Genético aplicado ao Passeio do Cavalo

Inicializar população P com sequências de movimentos candidatos do cavalo
enquanto critério de parada não atendido **faça**
    para cada indivíduo $S \in P$ **faça**
        $path \leftarrow$ DECODERCAVALO(S)
        $apt(S) \leftarrow$ AVALIARCAMINHO$(path)$
    fim para
    Selecionar indivíduos de maior aptidão
    Aplicar operadores de cruzamento e mutação, gerando a nova população P'
    para cada indivíduo $S' \in P'$ **faça**
        $path' \leftarrow$ DECODERCAVALO(S')
        $apt(S') \leftarrow$ AVALIARCAMINHO$(path')$
    fim para
    $P \leftarrow P'$ ▷ Atualização da população
fim enquanto
Retorna melhor caminho encontrado

Guloso Simples

A abordagem Gulosa Simples constitui uma estratégia determinística baseada em decisões locais, em que cada passo do algoritmo é tomado considerando apenas a melhor opção imediata, sem qualquer estimativa do impacto futuro dessa escolha. No contexto dos problemas analisados, essa abordagem é utilizada como linha de base conceitualmente simples, permitindo comparar o desempenho de métodos gulosos mais estruturados, como a heurística de Warnsdorff, e técnicas evolutivas mais complexas.

Funcionamento Geral. O algoritmo segue uma lógica direta: a cada iteração, examina-se a lista de ações possíveis a partir do estado atual e escolhe-se o primeiro movimento válido encontrado em uma ordem fixa de exploração. Essa característica confere ao método baixa complexidade computacional, em troca de pouca flexibilidade e elevado risco de impasses prematuros.

Adaptação ao Passeio do Cavalo. No Passeio do Cavalo, o Guloso Simples opera da seguinte forma:

- **Representação:** o estado do tabuleiro é armazenado em uma matriz $T[n][n]$, onde cada posição não visitada é inicializada com -1 e posições visitadas recebem o índice correspondente ao passo do percurso.
- **Conjunto de movimentos:** utiliza-se o conjunto padrão dos oito deslocamentos possíveis do cavalo, dados por $(\pm 1, \pm 2)$ e $(\pm 2, \pm 1)$.
- **Exploração sequencial:** em cada etapa, os movimentos possíveis são avaliados em uma ordem predefinida. Para cada deslocamento, verifica-se se a nova posição está dentro dos limites do tabuleiro e ainda não foi visitada.
- **Escolha imediata:** o primeiro movimento válido encontrado é selecionado sem considerar estimativas de mobilidade futura (grau) ou outras heurísticas.

Essa estratégia é extremamente eficiente em termos de custo computacional, porém altamente suscetível a becos sem saída, de modo que raramente produz passeios completos em tabuleiros maiores. Serve, contudo, como uma linha de base simples para comparação com heurísticas mais informadas, como a de Warnsdorff.

Adaptação às N-Rainhas. Para o Problema das N-Rainhas, a versão Gulosa Simples pode ser definida analogamente, escolhendo movimentos de forma determinística e sem previsão de conflitos futuros:

- **Representação:** utiliza-se um vetor Q de tamanho n, em que $Q[i] = j$ indica a posição (linha j) da rainha na coluna i. Durante a construção incremental da solução, colunas ainda não atribuídas podem ser marcadas com -1.
- **Conjunto de ações:** para cada coluna i não preenchida, considera-se, em ordem fixa, as linhas $j = 1, \ldots, n$ como candidatas.
- **Exploração sequencial:** ao preencher a coluna i, o algoritmo verifica sequencialmente cada linha j e escolhe a primeira linha que não produza conflito imediato com as rainhas já posicionadas nas colunas anteriores (conflitos de linha, coluna e diagonal).
- **Escolha imediata:** assim que se encontra uma linha válida para a coluna corrente, a posição é fixada e procede-se para a próxima coluna, sem avaliar o impacto dessa escolha nas colunas futuras.

Embora esse procedimento seja muito barato computacionalmente, ele não considera a interdependência global entre as escolhas e, portanto, frequentemente leva a configurações parciais que não podem ser completadas sem retrocesso. Por esse motivo, o Guloso Simples nas N-Rainhas costuma funcionar apenas como uma heurística construtiva inicial ou como referência para medir ganhos obtidos por técnicas que incorporam lookahead ou mecanismos de reparo.

Em ambos os problemas, o Guloso Simples oferece uma implementação direta e determinística, útil como baseline experimental, mas sua eficácia é limitada pela ausência de informação sobre mobilidade futura (no caso do cavalo) ou conflito futuro (no caso das rainhas). Em comparações empíricas, espera-se que ele apresente baixo custo por tentativa e alta taxa de falha relativa frente a abordagens heurísticas ou exatas mais sofisticadas.

Algorithm 3 Guloso Simples para o Passeio do Cavalo

Criar matriz $T[N][N]$ inicializada com 0
Definir movimentos do cavalo:

$$dx = \{-2, -1, 1, 2, 2, 1, -1, -2\}, \quad dy = \{1, 2, 2, 1, -1, -2, -2, -1\}$$

Definir posição inicial (x, y)
$T[x][y] \leftarrow 1$
$passo \leftarrow 1$
enquanto verdadeiro **faça**
    $encontrou \leftarrow$ falso
    para $i = 0$ até 7 **faça**
        $nx \leftarrow x + dx[i]$
        $ny \leftarrow y + dy[i]$
        se nx, ny estão dentro do tabuleiro **e** $T[nx][ny] = 0$ **então**
            $passo \leftarrow passo + 1$
            $x \leftarrow nx$
            $y \leftarrow ny$
            $T[x][y] \leftarrow passo$
            $encontrou \leftarrow$ verdadeiro
            break
        fim se
    fim para
    se não encontrou **então**
        break ▷ nenhum movimento possível
    fim se
fim enquanto
retorne T

Algorithm 4 Guloso Simples para o Problema das N-Rainhas

```
1: Criar vetor Q[N] inicializado com −1                ▷ Q[i] = coluna da rainha na linha i
2: para linha = 0 até N − 1 faça
3:     colocou ← falso
4:     para coluna = 0 até N − 1 faça
5:         conflito ← falso
6:         para i = 0 até linha − 1 faça
7:             se Q[i] = coluna ou |Q[i] − coluna| = |i − linha| então
8:                 conflito ← verdadeiro
9:                 break
10:            fim se
11:        fim para
12:        se não conflito então
13:            Q[linha] ← coluna
14:            colocou ← verdadeiro
15:            break
16:        fim se
17:    fim para
18:    se não colocou então
19:        break                                       ▷ algoritmo guloso ficou preso
20:    fim se
21: fim para
22: retorne Q
```

Heurística de Warnsdorff (Abordagem Gulosa)

A Heurística de Warnsdorff é uma estratégia gulosa amplamente utilizada para tentar construir um Passeio do Cavalo completo. Sua ideia central consiste em selecionar, a cada passo, o movimento que leva a uma posição não visitada com o menor número de movimentos futuros possíveis, isto é, a posição de menor grau. Essa escolha busca evitar que o cavalo seja direcionado para regiões do tabuleiro com pouca mobilidade, reduzindo a probabilidade de bloqueio prematuro.

Adaptação ao Problema do Passeio do Cavalo. Para aplicar a heurística ao Passeio do Cavalo, algumas definições são necessárias:

- **Movimentos do cavalo:** considera-se o conjunto fixo de oito deslocamentos possíveis do cavalo, dados por $(\pm 1, \pm 2)$ e $(\pm 2, \pm 1)$.
- **Controle de visitas:** utiliza-se uma matriz $T[n][n]$ representando o tabuleiro, onde $T[x][y] = -1$ indica casa não visitada e $T[x][y] = k$ indica que a posição foi visitada na k-ésima jogada.
- **Cálculo do grau:** para cada movimento candidato (tx, ty), computa-se o número de movimentos válidos que o cavalo ainda poderia realizar a partir dessa nova posição. Esse número corresponde ao grau de Warnsdorff.

- **Seleção gulosa:** o próximo movimento escolhido é aquele cujo grau é mínimo. Em caso de empate, podem ser utilizados critérios simples de desempate, como a escolha da primeira posição válida encontrada ou uma escolha pseudoaleatória.

Esse processo é repetido até que todas as n^2 posições tenham sido visitadas, caracterizando um passeio completo, ou até que não haja mais movimentos válidos, o que indica falha da heurística. Embora não garanta solução para qualquer configuração, a heurística apresenta excelente desempenho prático em tabuleiros 8×8 e é amplamente utilizada pela sua eficiência computacional.

Algorithm 5 Heurística de Warnsdorff para o Passeio do Cavalo

Criar matriz $T[N][N]$ inicializada com -1.
Definir vetores de movimentos do cavalo: movX e movY.
Definir posição inicial (x, y).
$T[x][y] \leftarrow 0$.
para $passo = 1$ **até** $N \times N - 1$ **faça**
    $menorGrau \leftarrow \infty$
    $proxX \leftarrow -1$
    $proxY \leftarrow -1$
    para $i = 0$ **até** 7 **faça**
        $tx \leftarrow x + movX[i]$
        $ty \leftarrow y + movY[i]$
        se tx e ty estão dentro do tabuleiro **e** $T[tx][ty] = -1$ **então**
            $grau \leftarrow$ número de movimentos válidos a partir de (tx, ty)
            se $grau < menorGrau$ **então**
                $menorGrau \leftarrow grau$
                $proxX \leftarrow tx$
                $proxY \leftarrow ty$
            fim se
        fim se
    fim para
    se $proxX = -1$ **então**
        retorne FALHA ▷ nenhum movimento possível
    fim se
    $x \leftarrow proxX$
    $y \leftarrow proxY$
    $T[x][y] \leftarrow passo$
fim para
retorne T

Backtracking

A técnica de Backtracking constitui um método sistemático de busca em profundidade, no qual a solução é construída passo a passo e corrigida sempre que uma escolha leva a um estado inviável. Trata-se de uma abordagem completa, capaz de explorar o espaço de busca de forma estruturada, retrocedendo sempre que necessário para garantir que todas as alternativas relevantes sejam examinadas. Nos problemas analisados, o Backtracking

atua como referência exata, permitindo comparar o comportamento de heurísticas gulosas e métodos aproximados.

Funcionamento Geral

O Backtracking segue uma lógica recursiva: a cada etapa, uma decisão parcial é tomada e, caso essa decisão viole alguma restrição, ela é imediatamente desfeita. O algoritmo avança apenas quando o estado atual permanece consistente com as regras do problema. Essa estratégia garante completude, embora apresente custo computacional elevado em espaços de busca grandes.

Adaptação ao Passeio do Cavalo

No Passeio do Cavalo, o Backtracking opera da seguinte forma:

- **Representação**: utiliza-se uma matriz $T[n][n]$, em que posições não visitadas são marcadas com -1 e as casas visitadas recebem o índice do passo correspondente.
- **Exploração recursiva**: a partir da posição atual do cavalo, tentam-se todos os oito movimentos possíveis. Para cada movimento, verifica-se se a nova posição está dentro do tabuleiro e ainda não foi visitada.
- **Avanço e retrocesso**: se o movimento é válido, o algoritmo marca a posição, avança para a próxima profundidade e continua a busca. Caso nenhum dos movimentos subsequentes leve a uma solução, desfaz-se a marcação e retorna-se ao nível anterior.
- **Critério de término**: uma solução válida é encontrada quando todas as n^2 casas são visitadas.

Essa abordagem é completa e garante encontrar um passeio válido sempre que ele existir, embora o custo computacional possa crescer rapidamente para tabuleiros maiores.

Adaptação ao Problema das N-Rainhas

Para o Problema das N-Rainhas, o Backtracking é aplicado de forma igualmente natural:

- **Representação**: utiliza-se um vetor Q de tamanho n, no qual $Q[i]$ indica a coluna da rainha posicionada na linha i.
- **Posicionamento incremental**: o algoritmo tenta colocar uma rainha por linha, percorrendo as colunas possíveis em ordem fixa.
- **Verificação de restrições**: antes de inserir uma rainha na posição (i, j), verifica-se se ela conflita com alguma rainha posicionada nas linhas anteriores (conflitos de coluna e diagonais).
- **Retrocesso**: caso nenhuma coluna disponível possa receber a rainha da linha atual, o algoritmo retorna para a linha anterior e tenta uma nova posição para ela.

O método garante encontrar todas as soluções possíveis para o problema das N-Rainhas, embora o número de combinações cresça rapidamente conforme n aumenta.

Considerações Gerais

Em ambos os problemas, o Backtracking fornece uma solução exata e completa, servindo como referência para avaliar o desempenho de heurísticas. Embora seja computacionalmente mais custoso quando comparado ao Guloso Simples ou à heurística de Warnsdorff,

sua capacidade de explorar sistematicamente todo o espaço de busca permite superar impasses e produzir soluções mesmo em cenários adversos. Nas avaliações empíricas, espera-se que apresente tempos de execução maiores, porém com taxa de sucesso total.

Algorithm 6 Backtracking para o Passeio do Cavalo

```
1:  procedure CavaloBT(x, y, passo)
2:      se passo = N × N então
3:          retorne verdadeiro
4:      fim se
5:      para i = 0 até 7 faça
6:          nx ← x + dx[i]
7:          ny ← y + dy[i]
8:          se nx, ny estão no tabuleiro e T[nx][ny] = −1 então
9:              T[nx][ny] ← passo
10:             se CavaloBT(nx, ny, passo + 1) então
11:                 retorne verdadeiro
12:             fim se
13:             T[nx][ny] ← −1                          ▷ backtrack
14:         fim se
15:     fim para
16:     retorne falso
17: fim procedure
```

Algorithm 7 Backtracking para o Problema das N-Rainhas

```
1: procedure RAINHASBT(linha)
2:     se linha = N então
3:         retorne verdadeiro
4:     fim se
5:     para coluna = 0 até N − 1 faça
6:         se SEGURO(linha, coluna) então
7:             Q[linha] ← coluna
8:             se RAINHASBT(linha + 1) então
9:                 retorne verdadeiro
10:            fim se
11:            Q[linha] ← −1                                  ▷ backtrack
12:        fim se
13:    fim para
14:    retorne falso
15: fim procedure
16: procedure SEGURO(linha, coluna)
17:    para i = 0 até linha − 1 faça
18:        se Q[i] = coluna então
19:            retorne falso
20:        fim se
21:        se |Q[i] − coluna| = |i − linha| então
22:            retorne falso
23:        fim se
24:    fim para
25:    retorne verdadeiro
26: fim procedure
```

4.3. Ambiente Experimental e Verificação

Os algoritmos foram compilados com `g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0` e executados em um computador com processador AMD Ryzen 5 2500u, 8 GB de memória RAM e sistema operacional Windows 10.

A definição de realizar dez execuções para cada cenário no Ambiente Experimental e de Verificação foi adotada com o objetivo de garantir um equilíbrio entre confiabilidade estatística e viabilidade computacional. Esse número permite observar tendências consistentes no comportamento dos algoritmos, reduzindo a influência de variações pontuais sem comprometer a execução completa da bateria de experimentos.

Além disso, a escolha por dez testes facilita a reprodutibilidade dos resultados, mantendo o ambiente controlado e evitando sobrecarga computacional desnecessária. Dessa forma, o conjunto de execuções é suficiente para identificar padrões de desempenho e validar o comportamento dos algoritmos avaliados.

Os métodos avaliados apresentam limitações inerentes à sua natureza. O algoritmo de Backtracking sofre com o crescimento exponencial do custo computacional à medida que o tamanho da instância aumenta, tornando-se inviável para tabuleiros maiores. A heu-

rística de Warnsdorff, embora eficiente, não garante a obtenção de uma solução em todos os cenários, especialmente em configurações que violam suas suposições heurísticas. Já os Algoritmos Genéticos podem apresentar estagnação prematura, além de dependerem fortemente de uma parametrização adequada para alcançar resultados satisfatórios.

A seção seguinte apresenta e discute os resultados obtidos.

5. Resultados e Discussão

5.1. Potencial Pedagógico dos Problemas de Busca e Otimização

Problemas clássicos como o Passeio do Cavalo e o problema das N-Rainhas possuem um papel pedagógico amplamente reconhecido na literatura de ensino de algoritmos, estruturas de dados e raciocínio computacional. Seu potencial educativo decorre de algumas características centrais:

- **Clareza visual**: ambos os problemas podem ser facilmente representados em tabuleiros ou grades, facilitando a compreensão de estados e transições.
- **Complexidade crescente**: permitem introduzir conceitos simples (movimentos, restrições, estados válidos) e, gradualmente, discutir complexidade, explosão combinatória e técnicas de poda.
- **Comparação de abordagens**: fornecem terreno fértil para contrastar métodos exatos (como Backtracking) com heurísticas (Warnsdorff) e metaheurísticas (Algoritmos Genéticos), evidenciando trade-offs entre completude, eficiência e qualidade da solução.
- **Aplicação prática de conceitos**: possibilitam exercitar pensamento algorítmico, geração de soluções, análise de desempenho, depuração e experimentação computacional.
- **Integração com programação**: são excelentes exemplos para introduzir estruturas como vetores, listas, recursão, funções de avaliação, aleatoriedade controlada e manipulação de estados.

Como exemplos concretos, o Passeio do Cavalo permite demonstrar de forma intuitiva conceitos de busca em profundidade, heurísticas de ordenação de movimentos e princípios de poda. Já o problema das N-Rainhas é amplamente utilizado para introduzir Backtracking, estratégias de busca local e a noção de espaço de soluções. Além disso, ambos os problemas são frequentemente empregados em cursos de Inteligência Artificial para ilustrar como heurísticas e metaheurísticas podem reduzir drasticamente o esforço computacional sem garantia de completude.

Esse potencial pedagógico se manifesta também nos experimentos realizados neste trabalho, que permitiram comparar, na prática, o comportamento de diferentes abordagens, promover discussões sobre eficiência e aproximar o aluno de situações reais de projeto e avaliação de algoritmos.

5.2. Protocolo Experimental e Métricas

Os experimentos foram realizados com $n = 10$ execuções independentes por cenário. No **Passeio do Cavalo**, a posição inicial variou conforme os casos de teste. Em métodos estocásticos, utilizou-se semente fixa para assegurar reprodutibilidade. No **N-Rainhas**, cada valor de N também foi avaliado em dez execuções.

Foram coletadas as seguintes métricas: *tempo médio de execução*, *taxa de sucesso*, *movimentos testados* ou *número de gerações* e, dependendo do problema, *tamanho do passeio* (Passeio do Cavalo) ou *número de conflitos* (N-Rainhas). As tabelas apresentam as médias calculadas.

Resumo dos parâmetros experimentais. Para o *Backtracking*, cada uma das 64 posições iniciais do tabuleiro 8×8 foi avaliada individualmente, adotando-se um limite máximo de 10 minutos por execução devido ao crescimento exponencial do espaço de busca. A heurística de Warnsdorff foi aplicada também às 64 posições iniciais, utilizando um critério de desempate pseudoaleatório para mitigar empates nas escolhas de graus mínimos. A Busca Gulosa simples, por sua vez, foi igualmente executada em todas as posições do tabuleiro, encerrando o processo sempre que ocorresse um bloqueio inevitável. Por fim, para o Algoritmo Genético, foi empregada uma população inicial de 100 indivíduos, com taxa de cruzamento igual a 0,8, taxa de mutação de 0,1 e um total de 100 gerações, mantendo consistência entre execuções e viabilizando comparação direta com os demais métodos.

5.3. Resultados para o Passeio do Cavalo

***Backtracking*.**

***Backtracking*.** O método apresentou desempenho consistente, alcançando **100% de sucesso** na resolução do Passeio do Cavalo no tabuleiro 8×8. Além disso, manteve estabilidade nas métricas de tempo de execução e número de tentativas, evidenciando sua eficiência para instâncias pequenas, embora permaneça limitado pela explosão combinatória em tabuleiros maiores. A Tabela 1 sintetiza os resultados médios obtidos nas execuções.

Tabela 1. Desempenho do *Backtracking* no Passeio do Cavalo (8×8, $n = 10$).

Métrica	Média	Observação
Movimentos testados	8.250.669	busca com poda
Tempo de execução (s)	0.17–0.21	100% de sucesso

Além disso, a Figura 4 mostra o tempo médio de execução em função do tamanho do tabuleiro, destacando crescimento acentuado com o aumento de N.

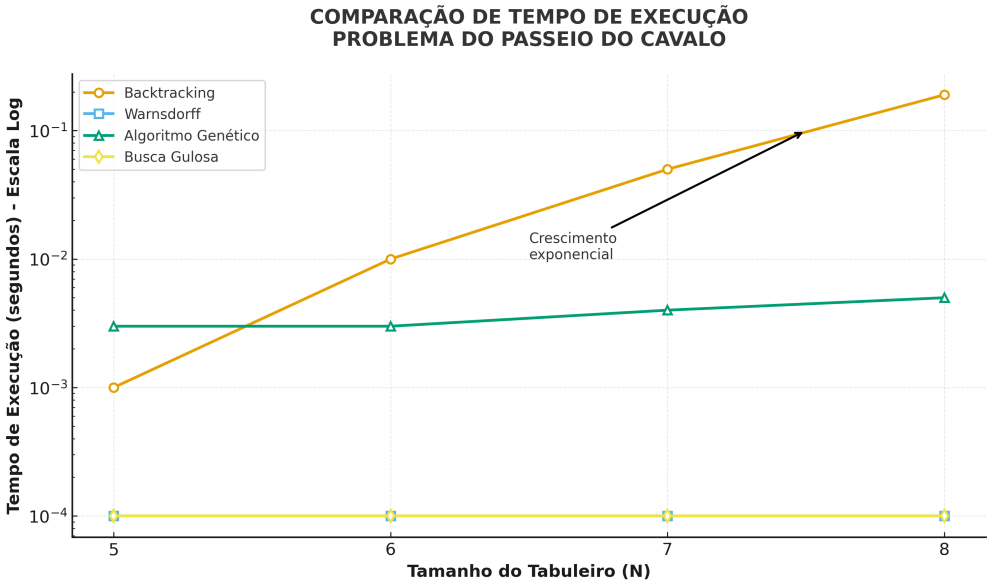

Figura 4. Tempo médio de execução no Passeio do Cavalo em função de N (escala log).

Heurística de Warnsdorff. A heurística apresentou desempenho consistente, alcançando taxa de sucesso superior a 95% no tabuleiro 8×8. Seu tempo de execução permaneceu muito baixo em todas as posições iniciais, uma vez que o método segue uma única trilha determinística (com desempates pseudoaleatórios). A Figura 5 ilustra a comparação entre as taxas de sucesso das diferentes abordagens avaliadas.

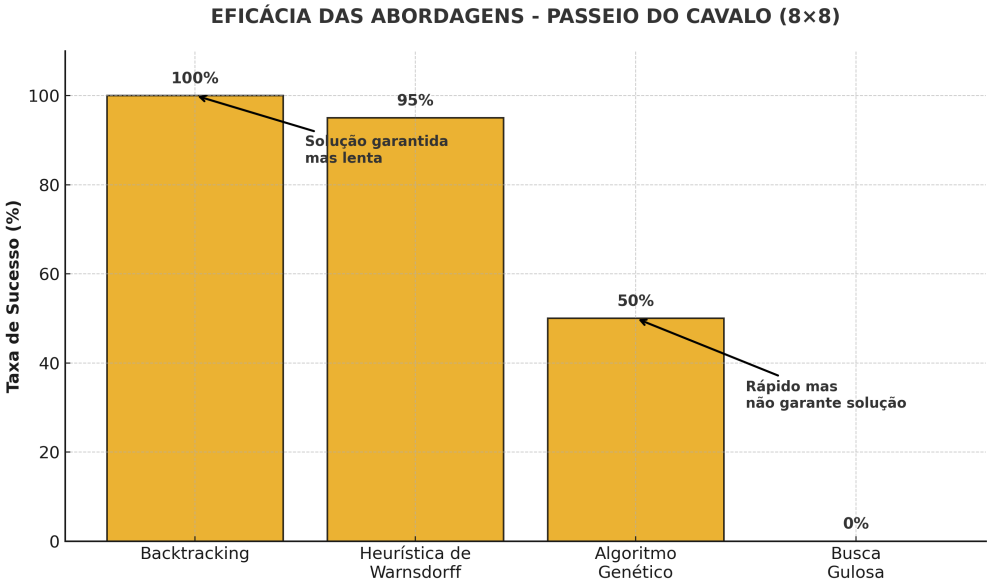

Figura 5. Taxa de sucesso das abordagens no Passeio do Cavalo (8×8).

Algoritmo Genético. O algoritmo apresentou desempenho intermediário entre as abordagens testadas. Em média, convergiu para tours completos em aproximadamente 50%

das execuções, indicando que a busca evolutiva foi capaz de encontrar soluções viáveis, mas ainda suscetível a estagnação prematura. O perfil evolutivo mostrado na Figura 6 evidencia uma rápida redução do valor de aptidão nas execuções bem-sucedidas, sugerindo que, quando a população inicial continha caminhos promissores, o processo de seleção e recombinação acelerou significativamente a convergência. Por outro lado, nas execuções fracassadas, observa-se a estabilização da aptidão ainda nas primeiras gerações, reforçando a sensibilidade do método à diversidade populacional e aos parâmetros evolutivos.

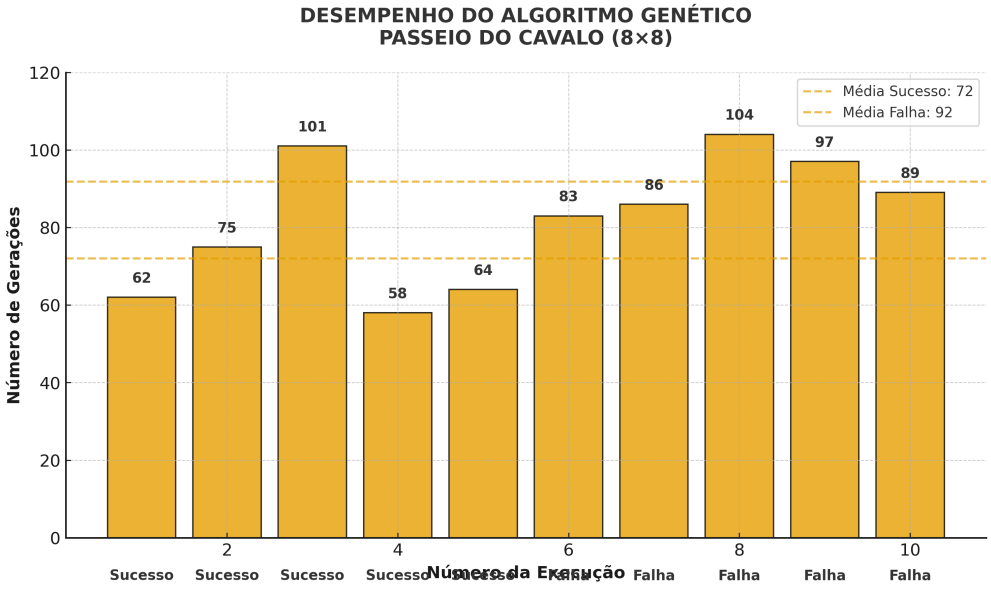

Figura 6. Perfil de convergência do *Algoritmo Genético* no Passeio do Cavalo (8×8).

A Tabela 2 sintetiza o desempenho do Algoritmo Genético (AG) na tarefa de encontrar tours completos no Passeio do Cavalo. Os valores indicam que o método apresentou comportamento consistente, porém limitado pela possibilidade de estagnação.

Em média, o AG necessitou entre 62 e 104 gerações para convergir nas execuções bem-sucedidas, mantendo tempo de processamento muito baixo (4–6 ms), o que confirma a eficiência computacional do método. Entretanto, nas execuções em que não atingiu o tour completo, o tamanho médio do caminho permaneceu entre 56 e 59 movimentos, evidenciando que a população tende a convergir prematuramente para ótimos locais.

Esses resultados reforçam que, embora o AG seja capaz de encontrar soluções completas em cerca de 50% das execuções, sua eficácia depende fortemente da parametrização e da diversidade genética ao longo das gerações. A presença de estagnação sugere a necessidade de mecanismos adicionais, como mutação adaptativa ou reinicialização parcial da população, para melhorar a robustez do método.

Tabela 2. Desempenho do *Algoritmo Genético* no Passeio do Cavalo (8×8, $n = 10$).

Métrica	Média	Observação
Gerações	62–104	$\sim$50% de tours completos
Tempo (ms)	4–6	baixo tempo
Tamanho do tour (falhas)	56–59	estagnação

Busca Gulosa simples. A versão simples não obteve tours completos, apresentando bloqueios precoces.

Tabela 3. Desempenho da *Busca Gulosa* simples no Passeio do Cavalo (8×8, $n = 10$).

Métrica	Média	Observação
Tentativas	0–29	bloqueios frequentes
Tempo (s)	0–14	sem tour completo
Taxa de sucesso	0%	—

Exemplo de solução. A Figura 7 apresenta um exemplo de passeio completo.

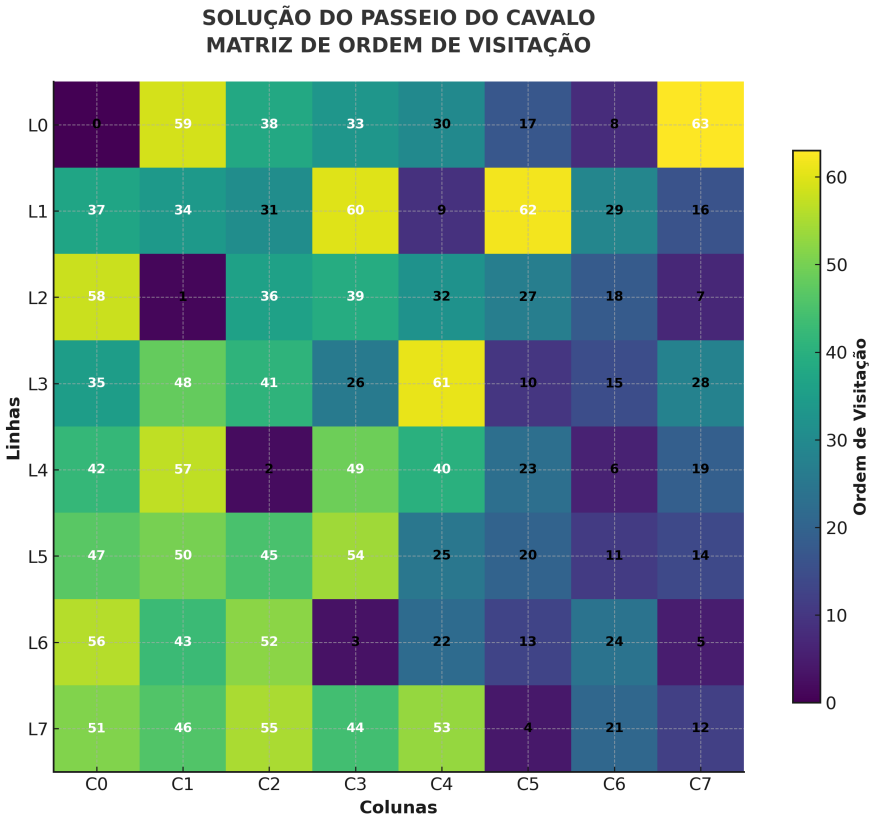

Figura 7. Exemplo de passeio completo no tabuleiro 8×8.

5.4. Resultados para o Problema das N-Rainhas

***Backtracking*.** Os resultados para o problema das 8-Rainhas mostram que o método de *Backtracking* apresentou desempenho consistente e altamente eficiente. Conforme resumido na Tabela 4, todas as dez execuções alcançaram solução válida, com tempo médio inferior a 1 ms, evidenciando a baixa complexidade prática para essa instância específica. O número médio de 105 retrocessos indica que a poda incremental implementada foi eficaz na redução do espaço de busca, eliminando precocemente configurações inviáveis e acelerando o processo de construção da solução. Esses resultados confirmam que, apesar de seu crescimento exponencial em instâncias maiores, o *Backtracking* é extremamente competitivo e confiável para dimensões reduzidas como $N = 8$.

Tabela 4. Desempenho do *Backtracking* para 8-Rainhas ($n = 10$).

Métrica	Média	Observação
Retrocessos	105	poda incremental
Tempo (ms)	< 1	100% de sucesso

A Figura 8 evidencia como o tempo de execução cresce para cada abordagem à medida que o valor de N aumenta. O *Algoritmo Genético* apresenta um crescimento moderado e relativamente estável, já que seu custo está mais associado ao número de gerações e ao

tamanho fixo da população do que ao aumento direto do tamanho do tabuleiro. Em contraste, o *Backtracking* mostra crescimento acentuado e rapidamente explosivo, refletindo sua natureza combinatória e o aumento exponencial do espaço de busca. Para valores pequenos de N, ambos os métodos apresentam tempos reduzidos; entretanto, à medida que N cresce, o Backtracking torna-se significativamente mais lento, enquanto o AG mantém desempenho mais previsível. Esses resultados reforçam a diferença de escalabilidade entre abordagens exatas e heurísticas.

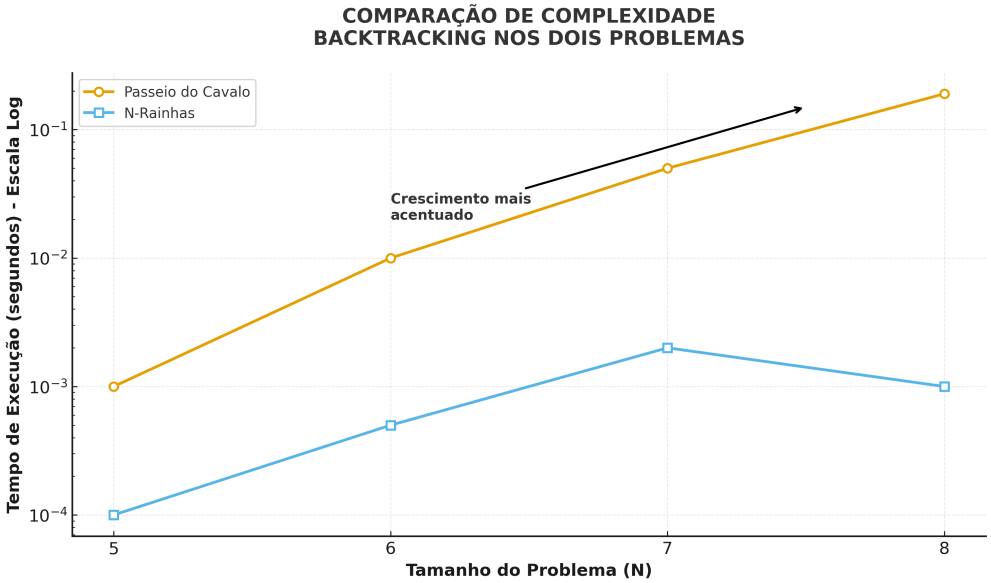

Figura 8. Tempo médio de execução no problema das N-Rainhas para diferentes valores de N (escala logarítmica).

***Algoritmo Genético*.** Os resultados para o caso de 8-Rainhas indicam que o Algoritmo Genético apresentou desempenho consistente e eficiente. Conforme mostrado na Tabela 5, todas as execuções convergiram para uma solução válida, mantendo uma taxa de sucesso de 100%. O número de gerações necessário para atingir a solução variou entre 14 e 79, evidenciando que, embora haja variabilidade inerente ao processo estocástico, o método tende a convergir rapidamente. O tempo médio de processamento permaneceu entre 4 e 6 ms, reforçando a baixa demanda computacional e a adequação do AG para lidar com instâncias pequenas do problema das N-Rainhas de maneira eficiente.

Tabela 5. Desempenho do *Algoritmo Genético* para 8-Rainhas ($n = 10$).

Métrica	Média	Observação
Gerações	14–79	100% de sucesso
Tempo (ms)	4–6	convergência rápida

Busca Gulosa. Os resultados apresentados na Tabela 6 evidenciam que a abordagem gulosa simples não é adequada para resolver o problema das N-Rainhas em sua forma geral. Embora obtenha sucesso para casos triviais, como $N = 1$, e para a configuração particular de $N = 5$, o método falha sistematicamente para quase todos os demais valores de N. Mesmo com tempos de execução inferiores a 1 ms, reflexo da baixa complexidade da estratégia, a solução tende a se estagnar após posicionar aproximadamente 70–80% das rainhas, incapaz de resolver conflitos remanescentes. Esses resultados reforçam que a Busca Gulosa carece de mecanismos de correção ou revisão de escolhas anteriores, tornando-a inadequada para instâncias maiores e confirmando sua limitação estrutural frente a métodos sistemáticos ou estocásticos.

Tabela 6. Desempenho da *Busca Gulosa* para N-Rainhas ($n = 10$ por N).

N	Tempo (ms)	Rainhas colocadas	Solução
1	< 1	1	Sim
2–4	< 1	1–3	Não
5	< 1	5	Sim
6–50	< 1	70–80% de N	Não

5.5. Síntese Comparativa

A Tabela 7 evidencia contrastes marcantes entre as quatro abordagens avaliadas. O *Backtracking* destaca-se pela completude e pela taxa de sucesso de 100% tanto no Passeio do Cavalo quanto no problema das N-Rainhas para $N = 8$, embora apresente limitações de escalabilidade devido ao crescimento exponencial do espaço de busca. A heurística de Warnsdorff mostra excelente desempenho no Passeio do Cavalo, com tempo muito baixo e taxa de sucesso superior a 95%, mas não garante solução em todos os cenários e não se aplica ao problema das N-Rainhas. O Algoritmo Genético apresenta bom equilíbrio entre tempo e taxa de sucesso, atingindo 100% de acerto em 8-Rainhas e cerca de 50% no Passeio do Cavalo, mas depende de parametrização adequada e não é completo. Por

fim, a Busca Gulosa simples exibe desempenho inferior nas duas tarefas, com baixa taxa de sucesso e pouca capacidade de escalabilidade, sendo útil apenas para casos triviais. Em conjunto, a síntese reforça a complementaridade entre estratégias exatas, heurísticas e metaheurísticas.

Tabela 7. Síntese comparativa entre abordagens.

Critério	Backtracking	Warnsdorff	Alg. Genético	Gulosa simples
Tempo médio	Alto	Baixo	Baixo	Muito baixo
Taxa de sucesso (Cavalo)	100%	>95%	∼50%	0%
Taxa de sucesso (N-Rainhas)	100% (N=8)	—	100% (N=8)	Baixa
Escalabilidade	Limitada	Moderada	Moderada	Baixa
Completude	Sim	Não	Não	Não

Discussão. Os resultados obtidos permitem uma análise mais abrangente das diferenças fundamentais entre as abordagens estudadas, considerando complexidade, eficiência e qualidade das soluções. O *Backtracking* confirmou seu papel como método exato: produz sempre soluções completas, mas apresenta crescimento exponencial de custo conforme o tamanho da instância aumenta. Em tabuleiros pequenos, como os casos testados, o método é extremamente eficiente, mas sua escalabilidade permanece limitada, tornando-o inviável para dimensões maiores sem técnicas adicionais de poda.

A heurística de Warnsdorff demonstrou excelente desempenho no Passeio do Cavalo, combinando baixo custo computacional com alta taxa de sucesso. Entretanto, sua natureza puramente heurística implica ausência de garantias formais de completude: embora funcione muito bem na prática, falha em casos específicos, especialmente quando ocorrem empates mal resolvidos ou estruturas desfavoráveis no tabuleiro. Ainda assim, representa uma solução muito eficiente quando se busca rapidez com alta probabilidade de êxito.

O *Algoritmo Genético* apresentou comportamento intermediário entre métodos exatos e heurísticos: sua eficiência variou de acordo com os parâmetros adotados e com a dificuldade do problema. Em N-Rainhas, exibiu convergência rápida e consistente; já no Passeio do Cavalo, mostrou maior instabilidade e tendência à estagnação, refletindo a sensibilidade de algoritmos evolutivos ao desenho da função de aptidão, diversidade populacional e operadores genéticos. Apesar disso, quando bem parametrizado, o AG é capaz de encontrar boas soluções em tempo reduzido, mesmo sem garantias formais de completude.

Por fim, a Busca Gulosa simples serviu como referência de base, evidenciando suas limitações significativas: baixa taxa de sucesso, alta suscetibilidade a bloqueios e incapacidade de revisar decisões anteriores. Sua eficiência é alta apenas em termos de tempo — não em qualidade de solução — confirmando que estratégias puramente imediatistas raramente são adequadas para problemas combinatórios estruturados.

De forma geral, a análise comparativa evidencia que métodos determinísticos (como Backtracking) oferecem qualidade máxima ao custo de escalabilidade, enquanto heurísticas estruturadas (Warnsdorff) atingem excelente eficiência prática, e metaheurísticas (como AG) ocupam um espaço intermediário, combinando flexibilidade com desempenho dependente de parametrização. Esses resultados sugerem que abordagens híbridas podem ser particularmente promissoras: por exemplo, empregar Warnsdorff para guiar a ordem de expansão do Backtracking, ou inicializar populações do AG com soluções parcialmente estruturadas, pode reduzir substancialmente o custo computacional e melhorar a taxa de sucesso em instâncias maiores. Tais combinações tendem a captar o melhor de cada paradigma, equilibrando exploração sistemática e heurísticas eficazes.

6. Conclusão

Este trabalho apresentou a modelagem, implementação e análise comparativa de três técnicas aplicadas a dois problemas clássicos de natureza combinatória: o **Passeio do Cavalo** e o **Problema das N-Rainhas**. As abordagens investigadas foram *Backtracking*, *Algoritmos Genéticos* e *Busca Gulosa*, com destaque para a regra de Warnsdorff.

Os resultados mostraram diferenças importantes entre os métodos. O *Backtracking* obteve soluções completas em todos os cenários testados, sendo apropriado quando

a obtenção garantida da solução é prioridade. Porém, sua escalabilidade é limitada devido ao crescimento rápido do espaço de busca.

A Heurística de Warnsdorff apresentou bom desempenho no Passeio do Cavalo, com tempo reduzido e elevada taxa de sucesso, embora possa falhar dependendo da posição inicial. Assim, é uma alternativa viável quando se busca rapidez, desde que a completude não seja obrigatória.

No **N-Rainhas**, os *Algoritmos Genéticos* alcançaram 100% de sucesso para $N = 8$ em todas as execuções, produzindo soluções completas e corretas com tempos baixos e poucas gerações. A qualidade das soluções também se manteve estável, já que todas as execuções atingiram configuração sem conflitos, resultando em 100% de acurácia para esse caso.

No **Passeio do Cavalo**, entretanto, o desempenho foi mais limitado. O AG atingiu cerca de 50% de tours completos, conforme observado nos resultados experimentais. Nas execuções sem sucesso, a trajetória média atingiu entre 56 e 59 casas visitadas (aproximadamente 87% do percurso total), indicando que, embora o método tenha dificuldade em fechar o tour, ele ainda produz trajetórias parcialmente válidas. Essa redução na taxa de sucesso é atribuída à maior dificuldade em preservar sequências de movimentos coerentes após operações de recombinação e mutação.

De forma geral, a escolha do método depende dos objetivos. Métodos exatos são úteis quando a solução correta deve ser encontrada, enquanto heurísticas e metaheurísticas oferecem alternativas mais flexíveis em contextos nos quais tempo e simplicidade de implementação são fatores importantes. Além disso, tanto o Problema do Passeio do Cavalo quanto o das N-Rainhas possuem um forte potencial pedagógico: são problemas clássicos, visualmente intuitivos e suficientemente ricos para introduzir, de forma acessível, conceitos fundamentais de computação. Por exemplo, o Passeio do Cavalo permite demonstrar poda, explosão combinatória e heurísticas como Warnsdorff, enquanto o N-Rainhas serve como base para ensinar retrocesso, conflitos entre variáveis e princípios de otimização. Dessa forma, ambos se mostram adequados como exemplos didáticos em disciplinas de algoritmos, inteligência artificial e técnicas evolutivas.

Limitações

Alguns aspectos limitaram o escopo do estudo:

- O *Backtracking* não escala bem com o aumento de N.
- Os *Algoritmos Genéticos* tiveram baixa taxa de sucesso no Passeio do Cavalo, indicando a necessidade de operadores adaptados ao problema.
- A Heurística de Warnsdorff não garante solução em todas as posições iniciais.
- As análises foram restritas a tabuleiros de até 8×8.
- Não foram realizadas medições específicas de uso de memória.
- As implementações em C++ não exploraram paralelização ou ferramentas gráficas.

Perspectivas Futuras

Como desdobramentos, destacam-se:

- Combinação de *Backtracking* com ordenação heurística.

- Aperfeiçoamento dos *Algoritmos Genéticos* com operadores especializados.
- Extensão dos experimentos para tabuleiros maiores e outros problemas combinatórios.
- Desenvolvimento de interfaces de visualização para apoio didático.

Referências

Bell, J. and Stevens, B. (2009). A survey of known results and research areas for n-queens. *Discrete Mathematics*, 309(1):1–31.

Besa, J. J., Johnson, T., Mamano, N., Osegueda, M. C., and Williams, P. (2022). Taming the knight's tour: Minimizing turns and crossings.

Cancela, H. and Mordecki, E. (2006). Counting knight's tours through the randomized warnsdorff rule.

Costa, A. (2018). Estudo de backtracking no problema do passeio do cavalo. Material de Aula, FFCLRP-USP. Acesso em: 3 jan. 2025.

Costa, V. S. and de Sá, V. G. P. (2015). Heurística para o passeio aberto do cavalo em tabuleiros multidimensionais. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 3(1).

Costa, V. S. and de Sá, V. P. (2013). Heurıstica eficiente para o passeio aberto do cavalo a partir de casas arbitrárias em tabuleiros quadrados. *Anais do XLV Simpósio Brasileiro de Pesquisa Operacional*, pages 3041–3052.

Joshuva, J. (2024). Cracking the chessboard challenge: The n-queens problem explained. *DEV Community*.

Martinjak, I. and Golub, M. (2007). Comparison of heuristic algorithms for the n-queen problem. In *2007 29th International Conference on Information Technology Interfaces*, pages 759–764.

Moghimi, O. and Amini, A. (2024). A novel approach for solving the n-queen problem using a non-sequential conflict resolution algorithm. *Electronics*, 13(20).

Mukherjee, S., Datta, S., Chanda, P. B., and Pathak, P. (2015). Comparative study of different algorithms to solve n queens problem. In *IEEE Annual Symposium on Foundations of Computer Science*.

Nascimento, M. O. and Brasil, C. R. S. (2023). Algoritmo evolutivo aplicado ao problema do percurso do cavalo. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Faculdade de Computação, Universidade Federal de Uberlândia. Acesso em: 5 jan. 2025.

Silva, A. T. R., Nazareno, G. S., and Schneider, A. M. (2003). Resolvendo o problema do cavalo do xadrez utilizando algoritmo genético. In *Anais do V Encontro de Estudantes de Informática do Tocantins*, pages 123–130, Palmas, TO. Outubro.

Warnsdorff, H. C. (1823). *Des Rösselsprunges einfachste und allgemeinste Lösung*. Varnhagen, Schmalkalden.

Witt, H. C. (2020). Análise de algoritmos de busca no problema das n-rainhas. Technical report, Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP). Relatório Técnico. Acesso em: 3 jan. 2025.