Trabalho Prático 03 - Algoritmos I

Lucas Rodrigues Pereira

Junho 2025

1 Introdução

O trabalho resolve o problema de otimização combinatória que consiste em, dado um tabuleiro $N \times M$ com uma rainha 'R', k peões 'P' e obstáculos '-', determinar a sequência mínima de movimentos da rainha para capturar todos os peões, respeitando as regras de movimento do xadrez e evitando os obstáculos. Implementam-se duas abordagens: um algoritmo exato e uma heurística eficiente, cuja comparação evidencia o compromisso entre exatidão e desempenho em problemas NP-difíceis. O estudo propõe a modelagem do problema em um grafo e a análise de algoritmos para otimização combinatória.

2 Modelagem

Considerando o problema em questão, podemos modela-lo de forma a abstrair o objetivo prático, obter o menor número de movimentos da rainha para capturar todos os peões, em conceitos e técnicas já conhecidas, visando buscar auxilio de ferramentas algorítmicas poderosas. Em situações onde uma peça esteja isolada, ou seja, um tabuleiro inválido, a resposta para o problema sempre será -1. Sabendo disso:

Proposição 1. *É sempre possível modelar um tabuleiro válido dado em um grafo completo e ponderado K.*

Demonstração. Como o tabuleiro é válido, podemos garantir que sempre exista um caminho de qualquer peça a qualquer outra, logo é possível garantir a existência de arestas entre quaisquer pares de vértices de K, sendo as peças tais vértices. □

Portanto, considerando somente situações onde a solução não é trivial, tabuleiros válidos, modelamos o problema como um grafo completo e ponderado K tal que:

- A rainha e cada um dos peões no tabuleiro são vértices de K.
- Uma aresta e que conecta um vértice v_i a v_j, $i \neq j$, tem um peso w, onde w é o mínimo de movimentos possíveis partindo da posição no tabuleiro de v_i a v_j, dentro do padrão de movimento da rainha no jogo de Xadrez.

3 Solução

Tendo em vista o modelo, e assumindo $P \neq NP$, podemos verificar que o problema proposto pertence a uma classe de problemas intratáveis em tempo polinomial da seguinte forma:

Teorema 1. *O problema da rainha no xadrez é NP-difícil, conforme demonstrado por redução polinomial ao Problema do Caixeiro Viajante (TSP).*

Demonstração. Considere a seguinte redução:

1. **Instância do TSP**: Dado um grafo completo $G = (V, E)$ com pesos $w : E \to \mathbf{R}^+$ e um limite L, determine se existe um ciclo hamiltoniano com peso total $\leq L$.
2. **Transformação**:
   (a) Para cada vértice $v_i \in V$, posicione um peão P_i em coordenadas $(i, 0)$ no tabuleiro.
   (b) Posicione a rainha R na origem $(0, 0)$.
   (c) Para cada aresta $e = (v_i, v_j)$ com peso w, construa um obstáculo padrão que force a rainha a realizar exatamente w movimentos para ir de P_i a P_j.
   (d) Defina $k = |V|$.
3. **Equivalência**:
   - Sim: Se existir um caminho TSP com custo $\leq L$, a rainha pode capturar todos os peões em $\leq L$ movimentos seguindo a ordem do caminho.
   - Não: Se não existir tal caminho TSP, nenhuma sequência de captura da rainha terá $\leq L$ movimentos.

A transformação é polinomial pois:

- A disposição dos peões é $O(k)$.
- A construção dos obstáculos é $O(k^2)$ (para todas as arestas).
- A configuração do tabuleiro mantém tamanho polinomial.

Como o TSP é NP-completo e nossa redução é polinomial, o problema da rainha é NP-difícil. A verificação de uma solução proposta pode ser feita em tempo polinomial, portanto o problema é NP-completo. □

Como consequência direta da NP-completude demonstrada, soluções exatas se tornam inviáveis a instâncias maiores do problema, enquanto aproximações são possíveis em troca de precisão. Portanto, dois algoritmos foram propostos, um exato utilizando uma abordagem que usa uma DP com memoization, outro aproximativo que usa como heurística o vizinho mais próximo.

3.1 Algoritmos

Para a implementação dos dois algoritmos o grafo do problema foi representado como uma matriz de adjacência.

3.1.1 Algoritmo Exato: Programação Dinâmica

1. Inicialize uma tabela para armazenar soluções parciais.
2. Comece na posição da rainha com nenhum peão capturado.
3. Para cada estado possível:
   - Se todos os peões foram capturados, retorne custo zero.
   - Para cada peão não capturado:
     - Calcule o custo mínimo recursivamente.
     - Armazene o melhor resultado.
4. Retorne o menor custo encontrado ou -1 se inválido.

3.1.2 Algoritmo Aproximativo: Vizinho Mais Próximo

1. Comece na posição da rainha.
2. Enquanto houver peões não capturados:
   - Vá para o peão mais próximo ainda não visitado.
   - Some o custo do movimento ao total.
3. Retorne o custo total ou -1 se algum peão for inalcançável.

4 Análise de Complexidade

4.1 Tempo

4.1.1 Programação Dinâmica

Complexidade: $O(|V|^2 \cdot 2^{|V|})$
Justificativa: Temos $2^{|V|}$ estados possíveis da máscara de bits, $|V|$ vértices de partida diferentes e $|V|$ iterações para verificar cada peão não capturado, portanto $O(|V|^2 \cdot 2^{|V|})$.

4.1.2 Vizinho Mais Próximo

Complexidade: $O(|V|^2)$
Justificativa: Temos $|V|$ etapas, uma para cada peão, em cada etapa, verifica até $|V|$ peões restantes. Logo, $O(|V|^2)$.

4.2 Espaço

4.2.1 Programação Dinâmica

Complexidade: $O(|V| \cdot 2^{|V|})$
Justificativa: Temos a tabela de memoização com $|V| \times 2^{|V|}$ entradas e uma pilha recursiva, que consome no máximo, $O(|V|)$ espaço adicional. Logo, $O(|V| \cdot 2^{|V|})$.

4.2.2 Vizinho Mais Próximo

Complexidade: $O(|V|)$
Justificativa: Temos um vetor binário para marcar peões visitados, $|V|$ posições. Ao final, $O(|V|)$.

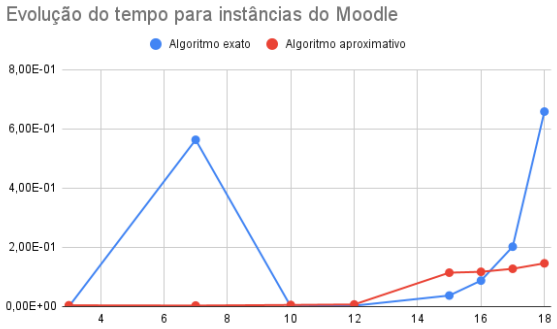
5 Discussão dos Algoritmos

Como apresentado, o problema trabalhado é intratável, portanto duas soluções foram proposta. Visando criar um comparativo entre as soluções em termos da qualidade da solução e tempo de resposta sobre instâncias com diferentes magnitudes, uma série de experimentos foi realizada.

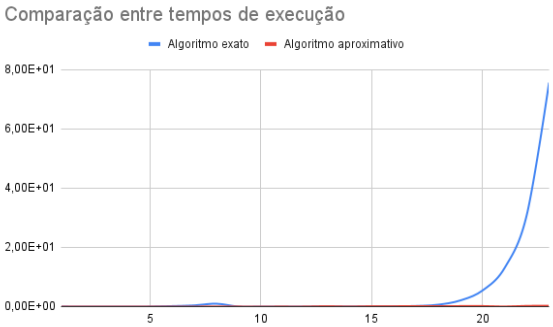
Para a obtenção dos dados apresentados abaixo foram utilizados testes disponíveis no Moodle e tabuleiros gerados proceduralmente com as seguintes características:

- Dimensões constantes, 100×100.
- Uma quantidade k de peões distribuídos aleatoriamente em posições válidas.
- Uma probabilidade de 20% de uma posição ser intransponível.

5.1 Comparativo de tempo



(a) Evolução do tempo para instâncias do Moodle
(Eixo X: Magnitude das instâncias; Eixo Y: Tempo em milissegundos)

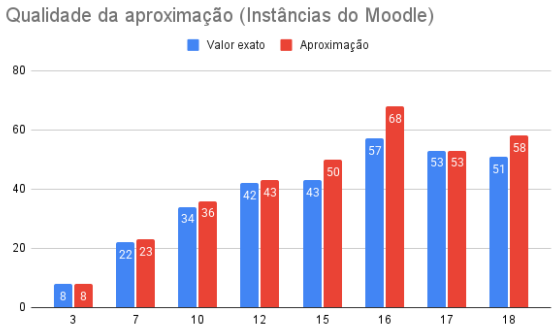


(b) Comparação entre tempos de execução
(Eixo X: Magnitude das instâncias; Eixo Y: Tempo em milissegundos)

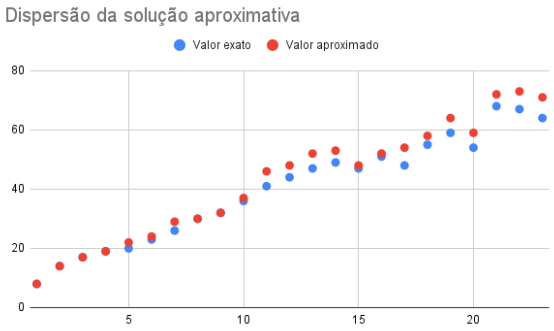
Figura 1: Análise de desempenho dos algoritmos implementados.

Como esperado, a solução exata cresce extremamente rápido em relação a mínimas variações da entrada. A partir da comparação entre tempos de execução, podemos perceber a discrepância entre as soluções, sendo tão absurdamente grande, que não é, nem ao menos, possível observar o comportamento polinomial da solução aproximativa.

5.2 Comparativo da qualidade da resposta



(a) Qualidade da aproximação
(Eixo X: Magnitude das instâncias; Eixo Y: Movimentos para captura)



(b) Dispersão da solução aproximativa
(Eixo X: Magnitude das instâncias; Eixo Y: Movimentos para captura)

Figura 2: Análise comparativa da qualidade das soluções aproximadas

É possível perceber que a solução aproximada gera boas respostas em relação as instâncias testadas, porém, pela limitação de tempo da resposta exata, torna-se inviável avaliar a solução aproximada para instâncias realmente grandes, além disso, o algoritmo aproximativo implementado não nos dá nenhuma garantia em relação a resposta, ou seja, ele pode ser muito pior que o melhor caminho. Em relação a outras aproximações, temos heurísticas que nos garante algo sobre a resposta, como por exemplo o algoritmo aproximativo que utiliza uma MST para obtenção de um caminho barato, nem sempre ótimo, mas que garante ser pelo menos menor que duas vezes o caminho ótimo. Entretanto, em comparação, a complexidade para implementação desses algoritmos é maior, e em certas instâncias e cenários onde se pode arcar com a não garantia da solução ideal, a heurística de vizinho mais próximo se mostra promissora.

6 Considerações Finais

O trabalho analisou duas abordagens para o problema da rainha: a solução exata por meio de programação dinâmica, que mostrou-se adequada para instâncias pequenas, e a heurística, vizinho mais próximo, que ofereceu um bom tempo para casos maiores, entretanto sem garantias de qualidade. Os resultados reforçam que a escolha do algoritmo deve considerar os ganhos e as perdas entre exatidão e eficiência em problemas de otimização combinatória.

Referências

[1] TARDOS, Eva; KLEINBERG, Jon. **Algorithm Design**. 1. ed. Pearson, 2006.

[2] WIKIPÉDIA. **Problema do caixeiro-viajante**. Disponível em: `https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante`.

[3] WIKIPÉDIA. **Nearest neighbour algorithm**. Disponível em: `https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm`.

[4] GEEKSFORGEEKS. **Approximate solution for TSP using MST**. Disponível em: https://www.geeksforgeeks.org/dsa/approximate-solution-for-travelling-salesman-problem-using-mst/.

[5] GEEKSFORGEEKS. **TSP using Dynamic Programming**. Disponível em: https://www.geeksforgeeks.org/dsa/travelling-salesman-problem-using-dynamic-programming/.