Análise Comparativa de Algoritmos de Busca Aplicados ao Quebra-Cabeça dos 8 Números

Patryck Fragoso Dias

[1]Curso de Ciência da Computação – Universidade Tuiuti do ParanáEmail: patryck.dias@utp.edu.br

Abstract. *Este trabalho apresenta uma análise comparativa entre algoritmos de busca cega (Busca em Largura e Busca em Profundidade) e busca heurística (Busca Gulosa e A*) aplicados ao problema clássico do Quebra-Cabeça dos 8 Números. Foram realizadas experiências com três instâncias distintas do problema, considerando tempo de execução, uso de memória e qualidade da solução obtida. Os resultados mostram a eficiência dos algoritmos heurísticos em relação aos demais.*

1. Introdução

O Quebra-Cabeça dos 8 Números é um problema clássico de Inteligência Artificial onde o objetivo é reorganizar um tabuleiro 3x3 até atingir uma configuração final desejada. Este problema é um exemplo clássico de problemas de busca, no qual é necessário explorar o espaço de estados em busca de uma solução. A abordagem para resolver este tipo de problema pode ser dividida em duas categorias: busca cega (sem heurísticas) e busca heurística (que usa informações sobre o problema para guiar a busca). O objetivo deste trabalho é comparar o desempenho de quatro algoritmos de busca, dois cegos (Busca em Largura e Busca em Profundidade) e dois heurísticos (Busca Gulosa e A*), considerando o tempo de execução, o uso de memória e a qualidade da solução.

2. Algoritmos Utilizados

Os algoritmos de busca implementados para resolver o Quebra-Cabeça dos 8 Números são divididos em duas categorias: cega e heurística.

2.1. Busca em Largura (BFS)

A Busca em Largura (BFS) é um algoritmo de busca cega que explora todos os nós em ordem de profundidade. Ele garante encontrar a solução ótima, mas com alto custo de memória, uma vez que precisa armazenar todos os nós gerados.

2.2. Busca em Profundidade (DFS)

A Busca em Profundidade (DFS) é outra abordagem cega que explora o máximo possível em um ramo antes de retroceder. Embora tenha baixo custo de memória, pode ser ineficiente em termos de tempo, pois não garante a solução ótima e pode entrar em ciclos infinitos.

2.3. Busca Gulosa

A Busca Gulosa é um algoritmo heurístico que utiliza uma função heurística para priorizar a expansão dos nós mais próximos da solução. Embora seja rápido, ele não garante encontrar a solução ótima e pode, em alguns casos, levar a soluções subótimas.

2.4. A*

O algoritmo A* combina a função heurística com o custo acumulado até o estado atual. Ele é geralmente o mais eficiente, pois encontra a solução ótima enquanto tenta minimizar o custo computacional, equilibrando a exploração e a exploração dos nós.

3. Implementação e Configuração dos Testes

A implementação foi realizada em Python, utilizando as bibliotecas `time` para medição do tempo de execução e `tracemalloc` para monitoramento do uso de memória. A heurística utilizada para a Busca Gulosa e A* foi a distância de Manhattan, que calcula a soma das distâncias verticais e horizontais entre os blocos atuais e os blocos na configuração final.

Foram testadas três instâncias diferentes do problema, com configurações variadas de blocos no tabuleiro. O código foi ajustado para garantir que a busca nos estados fosse eficiente, sem entrar em ciclos infinitos, especialmente para o DFS.

4. Resultados

Os resultados foram obtidos para cada instância do problema, com tempo de execução, uso de memória e o número de passos necessários para atingir a solução. A tabela abaixo apresenta os resultados de cada algoritmo nas três instâncias:

Tabela 1. Comparativo entre algoritmos nas três instâncias.

Algoritmo	Tempo (s)	Memória (KB)	Passos	Instância
BFS	0.00013	0.81	2	1
DFS	0.00008	0.70	2	1
Gulosa	0.00012	0.61	2	1
A*	0.00008	0.52	2	1
BFS	0.00065	4.07	4	2
DFS	0.50799	229.58	26	2
Gulosa	0.00036	1.09	4	2
A*	0.00025	0.99	4	2
BFS	0.00367	3.52	6	3
DFS	0.87697	215.63	20	3
Gulosa	0.00031	1.34	6	3
A*	0.00025	1.35	6	3

5. Discussão

Os resultados mostram que os algoritmos heurísticos, especialmente o A*, são mais eficientes em termos de tempo e uso de memória. O A* foi capaz de encontrar a solução ótima em todas as instâncias testadas, com o menor tempo de execução e uso de memória eficiente. Por outro lado, a Busca em Profundidade, embora tenha sido rápida em algumas instâncias, apresentou grande ineficiência em termos de uso de memória e número de passos em instâncias mais complexas. A Busca Gulosa, apesar de ser rápida, não foi tão eficiente quanto o A*, principalmente porque não garante soluções ótimas.

6. Conclusão

Concluímos que os algoritmos heurísticos, especialmente o A*, são os mais indicados para resolver o Quebra-Cabeça dos 8 Números de forma eficiente. Eles não apenas encontraram a solução ótima em todas as instâncias, mas também apresentaram um melhor equilíbrio entre tempo de execução e uso de memória. Como futuras melhorias, seria interessante comparar esses algoritmos com outras técnicas, como o IDA*, ou aplicá-los em versões maiores do quebra-cabeça.