Análise Comparativa de Estratégias de Busca para o Quebra-Cabeça dos 8 Números

Klaus Siegfried Beckmann

[1]Universidade Tuiuti do Paraná
Curitiba – PR

klaus.beckmann@utp.edu.br

Resumo. *Este trabalho apresenta uma análise comparativa de diferentes estratégias de busca aplicadas ao quebra-cabeça dos 8 números. Foram implementados e avaliados quatro algoritmos: Busca em Largura (BFS), Busca em Profundidade (DFS), Busca Gulosa e A*. O desempenho de cada algoritmo foi analisado com base no tempo de execução, uso de memória, número de nós expandidos e qualidade da solução obtida. Os resultados demonstram que o algoritmo A* apresenta melhor desempenho geral, sendo mais eficiente em termos de nós expandidos e tempo de execução, enquanto a BFS garante soluções ótimas com menor uso de memória para as instâncias testadas.*

1. Introdução

O quebra-cabeça dos 8 números (8-puzzle) é um problema clássico na área de Inteligência Artificial, frequentemente utilizado para ilustrar e comparar diferentes estratégias de busca. O problema consiste em um tabuleiro 3×3 com oito peças numeradas de 1 a 8 e um espaço vazio, que pode ser representado pelo número 0. O objetivo é alcançar uma configuração específica do tabuleiro movendo as peças para posições adjacentes ao espaço vazio.

A complexidade do quebra-cabeça dos 8 números reside no grande espaço de estados possíveis, que pode conter até 9!/2 = 181.440 configurações distintas e alcançáveis. Este cenário torna o problema adequado para avaliar e comparar diferentes estratégias de busca, permitindo analisar suas características em termos de eficiência computacional, uso de memória e qualidade das soluções encontradas.

Neste trabalho, implementamos e analisamos quatro estratégias fundamentais de busca:

- **Busca em Largura (BFS)**: explora todos os nós em um mesmo nível antes de avançar para o próximo nível, garantindo encontrar a solução ótima em termos do número de movimentos.
- **Busca em Profundidade (DFS)**: explora um caminho até sua profundidade máxima antes de retroceder e explorar caminhos alternativos, geralmente consumindo menos memória, mas não garantindo soluções ótimas.
- **Busca Gulosa**: utiliza uma função heurística para estimar a distância até o estado objetivo, selecionando sempre o nó que aparenta estar mais próximo da solução, sem considerar o custo do caminho percorrido.

- **Algoritmo A***: combina o custo do caminho percorrido com uma função heurística, equilibrando a exploração de caminhos promissores com a garantia de encontrar soluções ótimas quando a heurística é admissível.

O objetivo principal deste estudo é avaliar qual dessas estratégias apresenta melhor desempenho para o quebra-cabeça dos 8 números, considerando métricas como tempo de execução, uso de memória, número de nós expandidos e qualidade da solução encontrada. Os resultados desta análise podem fornecer insights valiosos para a escolha de algoritmos adequados em problemas de busca semelhantes.

2. Metodologia

2.1. Implementação

A implementação dos algoritmos foi realizada em Python, utilizando as bibliotecas NumPy para manipulação de arrays, heapq para estruturas de fila de prioridade, e psutil para monitoramento do uso de memória.

O problema do quebra-cabeça dos 8 números foi modelado através de uma classe `Puzzle` que representa o estado do tabuleiro 3×3. Cada estado contém as seguintes funcionalidades:

- Representação do tabuleiro como uma matriz 3×3
- Funções para detectar o espaço vazio e gerar movimentos
- Métodos para verificar se um estado é o objetivo
- Funções heurísticas: distância Manhattan e peças fora do lugar
- Geração de estados sucessores

Para cada algoritmo de busca, implementamos as seguintes características:

1. **Busca em Largura (BFS)**:
    - Estrutura de dados: Fila (FIFO) implementada com `deque`
    - Exploração por níveis, garantindo caminho ótimo
    - Controle de estados já visitados para evitar ciclos
2. **Busca em Profundidade (DFS)**:
    - Estrutura de dados: Pilha (LIFO) implementada com lista
    - Limite de profundidade para evitar buscas infinitas
    - Controle de estados já visitados para evitar ciclos
3. **Busca Gulosa**:
    - Estrutura de dados: Fila de prioridade implementada com `heapq`
    - Prioridade baseada na distância Manhattan até o estado objetivo
    - Controle de estados já visitados para evitar ciclos
4. **Algoritmo A***:
    - Estrutura de dados: Fila de prioridade implementada com `heapq`
    - Função de avaliação f(n) = g(n) + h(n), onde g(n) é o custo do caminho até o nó e h(n) é a distância Manhattan
    - Controle de estados já visitados e atualização de caminhos mais curtos

2.2. Configuração dos Testes

Para avaliar o desempenho dos algoritmos, foram utilizadas 10 instâncias diferentes do quebra-cabeça dos 8 números, obtidas a partir de um arquivo CSV. Cada instância representa uma configuração inicial do tabuleiro.

Para cada algoritmo e cada instância, foram coletadas as seguintes métricas:

- **Tempo de execução**: tempo total necessário para encontrar a solução
- **Uso de memória**: quantidade de memória RAM consumida durante a execução
- **Nós expandidos**: número total de estados explorados
- **Tamanho máximo da fronteira**: número máximo de estados mantidos em memória simultaneamente
- **Comprimento do caminho**: número de movimentos necessários para alcançar a solução

Os testes foram executados em um ambiente controlado com hardware consistente para garantir a comparabilidade dos resultados. Limitamos a profundidade máxima para o algoritmo DFS em 50 níveis para evitar buscas muito longas ou ciclos.

3. Resultados

Os experimentos realizados com as 10 instâncias do quebra-cabeça dos 8 números revelaram diferenças significativas entre as estratégias de busca implementadas. A Tabela 1 apresenta os valores médios das métricas coletadas para cada algoritmo.

Tabela 1. Comparativo de desempenho das estratégias de busca (valores médios)

Algoritmo	Nós Expandidos	Tamanho Máx. Fronteira	Tempo (s)	Memória (MB)	Passos Solução
BFS	356,10	225,70	1,507	43,14	6,90
DFS	40.904,10	38,60	21,770	57,04	32,90
Busca Gulosa	35,80	29,10	0,017	44,99	7,70
A*	15,10	14,00	0,009	44,89	6,90

Para visualizar melhor o contraste entre os algoritmos, especialmente considerando a grande diferença de escala em algumas métricas, apresentamos na Figura 1 um gráfico comparativo para o número de nós expandidos e o comprimento do caminho encontrado.

```
--- Melhor Algoritmo por Métrica ---
Melhor algoritmo para expandidos: A* (média: 15.1000)
Melhor algoritmo para fronteira_max: A* (média: 14.0000)
Melhor algoritmo para tempo: A* (média: 0.0081)
Melhor algoritmo para memoria: BFS (média: 42.8809)
Melhor algoritmo para passos: BFS (média: 6.9000)
```

Figura 1. Comparação entre algoritmos.

3.1. Análise por Instância

Além dos valores médios, analisamos o comportamento de cada algoritmo para instâncias específicas. A Tabela 2 apresenta os resultados para três instâncias representativas: uma simples (instância 1), uma de complexidade média (instância 4) e uma complexa (instância 9).

Tabela 2. Desempenho dos algoritmos em instâncias selecionadas

Algoritmo	Instância 1		Instância 4		Instância 9	
	Nós	**Passos**	**Nós**	**Passos**	**Nós**	**Passos**
BFS	2	2	136	8	1.345	13
DFS	2	2	10.208	48	93.019	49
Busca Gulosa	2	2	22	8	16	13
A*	2	2	13	8	20	13

4. Discussão

A análise dos resultados permite identificar características importantes de cada estratégia de busca implementada e determinar qual delas é mais eficiente para o quebra-cabeça dos 8 números.

4.1. Busca em Largura (BFS)

A BFS apresentou um desempenho equilibrado, garantindo soluções ótimas em todas as instâncias testadas. O número médio de passos para solução (6,9) foi equivalente ao do A*, confirmando a otimalidade das soluções encontradas. Contudo, para instâncias mais complexas, o algoritmo expandiu significativamente mais nós que as abordagens heurísticas, resultando em tempos de execução mais elevados.

Uma característica positiva observada foi o menor consumo médio de memória (43,14 MB) entre todos os algoritmos testados, o que pode ser contra-intuitivo, já que teoricamente a BFS tende a consumir mais memória. Isso pode ser explicado pela implementação eficiente e pelo controle de estados já visitados.

4.2. Busca em Profundidade (DFS)

O DFS mostrou-se o algoritmo menos eficiente para o quebra-cabeça dos 8 números. Mesmo com limite de profundidade de 50 níveis, expandiu em média 40.904 nós, um valor extremamente superior aos demais algoritmos. Além disso, as soluções encontradas estavam longe de ser ótimas, exigindo em média 32,9 passos, quando o caminho ótimo tinha em média 6,9 passos (obtido pela BFS e A*).

O tempo médio de execução (21,77 segundos) e o consumo de memória (57,04 MB) também foram os mais elevados entre os algoritmos testados. Um ponto positivo foi o menor tamanho máximo da fronteira, demonstrando a característica da DFS de manter uma fronteira mais compacta, embora isso não tenha compensado o alto número de expansões de nós.

4.3. Busca Gulosa

A Busca Gulosa apresentou resultados notavelmente bons em termos de eficiência computacional. Ela expandiu, em média, apenas 35,8 nós, um valor muito inferior à BFS (356,1) e ao DFS (40.904,1). O tempo médio de execução (0,017 segundos) foi cerca de 88 vezes menor que o da BFS.

Embora não garanta soluções ótimas teoricamente, nas instâncias testadas a Busca Gulosa encontrou soluções muito próximas do ótimo, com média de 7,7 passos, apenas 0,8 passos a mais que o caminho ótimo. Isso indica que a heurística da distância Manhattan é bastante eficaz para o quebra-cabeça dos 8 números.

4.4. Algoritmo A*

O A* destacou-se como o algoritmo mais eficiente em termos gerais. Expandiu o menor número médio de nós (15,1), muito abaixo dos demais algoritmos, e obteve o menor tempo médio de execução (0,009 segundos). Ao mesmo tempo, garantiu soluções ótimas em todas as instâncias, com média de 6,9 passos, igual à BFS.

O consumo de memória (44,89 MB) foi ligeiramente superior ao da BFS, mas inferior ao DFS. O tamanho máximo da fronteira (14,0) foi o menor entre todos os algoritmos, demonstrando a eficiência da estratégia de exploração direcionada pela combinação de custo do caminho e heurística.

4.5. Análise Comparativa

Considerando as diferentes métricas avaliadas, podemos estabelecer o seguinte ranking dos algoritmos:

1. **A***: Melhor desempenho geral, equilibrando eficiência computacional com qualidade da solução.
2. **Busca Gulosa**: Excelente eficiência, com soluções próximas do ótimo.
3. **BFS**: Garante soluções ótimas com melhor uso de memória, mas menos eficiente em tempo.
4. **DFS**: Desempenho significativamente inferior em todas as métricas, exceto tamanho da fronteira.

A superioridade do A* demonstra a importância de combinar o conhecimento do problema (através da heurística) com a garantia de otimalidade. A Busca Gulosa, embora não garanta teoricamente soluções ótimas, mostrou-se uma alternativa eficiente quando a heurística é bem ajustada ao problema.

É importante notar que a eficácia de cada algoritmo pode variar dependendo da complexidade da instância. Para instâncias simples (como a instância 1), todos os algoritmos apresentaram desempenho similar. Porém, para instâncias mais complexas (como a instância 9), as diferenças tornaram-se muito mais evidentes, com o A* expandindo apenas 20 nós contra 1.345 da BFS e 93.019 do DFS.

5. Conclusão

Este trabalho apresentou uma análise comparativa de quatro estratégias de busca aplicadas ao quebra-cabeça dos 8 números. Os resultados demonstraram que o algoritmo A*

apresenta o melhor desempenho geral, combinando eficiência computacional com a garantia de soluções ótimas. A Busca Gulosa surgiu como uma alternativa viável quando a velocidade é prioritária e soluções aproximadamente ótimas são aceitáveis.

A BFS manteve-se como uma estratégia confiável para garantir soluções ótimas, enquanto o DFS mostrou-se inadequado para este tipo de problema devido ao alto número de expansões de nós e à qualidade inferior das soluções encontradas.

Como trabalhos futuros, sugerimos:

- Implementação de novas heurísticas, como o padrão de conflitos lineares, para melhorar ainda mais o desempenho do A* e da Busca Gulosa.
- Extensão da análise para o quebra-cabeça dos 15 números (15-puzzle), que apresenta um espaço de estados muito maior.
- Avaliação de algoritmos de busca bidirecional, que podem reduzir significativamente o espaço de busca.
- Implementação de técnicas de aprendizado por reforço para resolver o quebra-cabeça, comparando seu desempenho com os métodos de busca tradicionais.
- Análise do impacto de diferentes estruturas de dados e otimizações de implementação no desempenho dos algoritmos.

O estudo realizado contribui para a compreensão das vantagens e limitações de diferentes estratégias de busca, fornecendo insights valiosos para a seleção de algoritmos adequados em problemas de busca em espaços de estados, um componente fundamental em diversas aplicações de Inteligência Artificial.