Comparação de Algoritmos de Busca Aplicados ao Problema do 8-Puzzle

Murilo Henrique Baruffi[1]

[1]Universidade Tuiuti do Paraná – Curitiba – Paraná

murilo.baruffi@utp.edu.br

Resumo. *Este trabalho apresenta a implementação e avaliação comparativa dos algoritmos BFS, DFS, A* e Busca Gulosa aplicados ao problema do 8-Puzzle. Foram analisados o tempo de execução, o uso de memória e a qualidade das soluções encontradas. Os resultados demonstram que o algoritmo A* se destaca pela eficiência e qualidade das soluções, enquanto DFS apresentou limitações consideráveis em cenários mais complexos.*

1. Introdução

O 8-Puzzle é um clássico problema de busca no qual nove peças (numeradas de 1 a 8 e um espaço vazio) devem ser reorganizadas em uma configuração alvo a partir de um estado inicial. Este tipo de problema é amplamente utilizado para testar algoritmos de busca, devido à sua simplicidade de enunciado, mas complexidade de solução [1].

Neste trabalho, comparamos quatro algoritmos de busca: Busca em Largura (BFS), Busca em Profundidade (DFS), Busca A* e Busca Gulosa. Cada algoritmo foi avaliado quanto ao tempo de execução, memória utilizada e qualidade da solução (número de movimentos).

2. Descrição da Implementação e Configuração dos Testes

As implementações foram feitas em linguagem Python. Para os algoritmos A* e Busca Gulosa, utilizamos a heurística da distância de Manhattan como função de avaliação.

Cada algoritmo foi testado em múltiplas instâncias do 8-Puzzle, variando a dificuldade dos estados iniciais. Para garantir a confiabilidade dos testes, cada configuração foi executada cinco vezes, computando-se a média dos resultados.

As métricas analisadas foram:

- **Tempo de execução** (em segundos)
- **Memória utilizada** (em kilobytes)
- **Qualidade da solução** (número de movimentos para resolver)

3. Resultados

A Tabela 1 apresenta os resultados médios obtidos nos experimentos. Cada linha corresponde a uma instância diferente do problema, variando em dificuldade, e cada coluna exibe as métricas de interesse para cada algoritmo.

Observa-se que o algoritmo A* apresenta o melhor equilíbrio entre tempo de execução e qualidade da solução, enquanto a Busca Gulosa é ligeiramente mais rápida,

Tabela 1. Comparação dos algoritmos para o problema do 8-Puzzle

Instância	BFS			DFS			Gulosa			A*		
	Movimentos	Tempo (s)	Memória (KB)	Movimentos	Tempo (s)	Memória (KB)	Movimentos	Tempo (s)	Memória (KB)	Movimentos	Tempo (s)	Memória (KB)
1	2	0.0000	2.25	44	0.1078	1379.97	2	0.0000	0.84	2	0.0000	0.88
2	4	0.0000	6.13	50	0.0715	981.51	4	0.0000	1.09	4	0.0000	1.42
3	6	0.0025	27.72	50	0.1307	1459.68	6	0.0000	1.96	6	0.0000	2.40
4	8	0.0031	49.41	44	0.0954	1157.73	8	0.0010	5.45	8	0.0000	3.11
5	6	0.0010	9.90	48	0.0880	1067.45	6	0.0000	1.95	6	0.0000	1.95
6	6	0.0010	21.73	48	0.0648	977.06	6	0.0000	1.96	6	0.0010	2.88
7	8	0.0020	37.02	8	1.0030	10221.75	8	0.0000	2.34	8	0.0000	3.41
8	3	0.0010	3.08	49	0.0774	1001.25	3	0.0000	0.98	3	0.0000	0.98
9	13	0.0299	885.30	47	1.1113	14546.56	13	0.0000	3.64	13	0.0000	6.73
10	13	0.0398	1131.92	-1	1.4912	16314.80	37	0.0112	122.71	13	0.0020	17.13

porém gera soluções menos ótimas. A Busca em Profundidade foi a menos eficiente em termos de qualidade da solução, muitas vezes encontrando soluções extremamente longas ou falhando em encontrar uma solução.

Além disso, a quantidade de memória consumida pelo BFS cresce rapidamente com a dificuldade do problema, tornando-o inviável para instâncias maiores ou com pouco espaço disponível.

4. Discussão

Os resultados confirmam que o algoritmo A* é o mais eficiente para o problema do 8-Puzzle, considerando a combinação de rapidez e precisão. A heurística da distância de Manhattan mostrou-se adequada para guiar a busca de forma otimizada, fornecendo soluções rápidas e curtas.

O algoritmo BFS, embora garantido para encontrar a solução ótima, sofre com a explosão de memória, sendo inadequado para casos com alta profundidade de busca. Já o DFS mostrou-se capaz de economizar memória, mas ao custo de encontrar caminhos extremamente longos ou até mesmo de não encontrar a solução (como visto na instância 10).

A Busca Gulosa foi muito eficiente em termos de tempo, mas devido a considerar apenas a heurística, e não o custo acumulado, frequentemente encontrou caminhos subótimos, o que é aceitável em aplicações onde o tempo é mais crítico do que a qualidade da solução.

Outro ponto interessante observado foi a consistência do A*, que mesmo em instâncias mais difíceis manteve tempos de execução baixos e qualidade alta, reforçando seu uso como melhor escolha geral para problemas semelhantes.

5. Conclusão e Trabalhos Futuros

A partir dos testes realizados, conclui-se que o algoritmo A* é a melhor opção para resolver o problema do 8-Puzzle entre as alternativas estudadas. Seu bom desempenho deve-se à utilização balanceada da heurística e do custo acumulado.

Como trabalhos futuros, propõe-se investigar variações de heurísticas para o A*, como o número de peças fora do lugar ou heurísticas ponderadas. Também seria interessante a implementação de técnicas de otimização de memória, como o A* com limite de memória (Memory-bounded A*), ou a busca iterativa aprofundada A* (IDA*).

Além disso, futuros trabalhos podem abordar o desempenho dos algoritmos em variantes mais complexas, como o 15-Puzzle ou 24-Puzzle, bem como realizar a análise

em ambientes restritos de hardware, simulando condições reais de sistemas embarcados e aplicações de tempo real.

Referências

[1] S. Russell e P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3ª edição, 2016.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, e C. Stein. *Introduction to Algorithms*. MIT Press, 3ª edição, 2009.

[3] P. Hart, N. Nilsson, e B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.