Estudo de Algoritmos de Busca para Solução do 8-Puzzle: Comparação de Heurísticas e Desempenho

1st Bárbara Braga Gualberto Correa
Centro Federal de Educação Tecnológica (CEFET/MG)
Divinópolis, Brasil
barbarabgual@gmail.com

2nd Julia Mello Lopes Gonçalves
Centro Federal de Educação Tecnológica (CEFET/MG)
Divinópolis, Brasil
juliamellolopesgoncalves@gmail.com

***Resumo*—Neste artigo, comparamos os algoritmos Breadth-First Search (BFS), Depth-First Search (DFS) e A* (com heurísticas de Manhattan e Euclidiana) na resolução do problema do 8-puzzle. Foram realizados experimentos com 30 quebra-cabeças aleatórios, avaliando o desempenho em termos de tempo de execução, consumo de memória, completude, optimalidade e nós explorados. Os resultados mostraram que o A* com heurística de Manhattan teve o melhor desempenho, sendo eficiente em termos de tempo e uso de memória, enquanto o BFS garantiu a solução ótima, porém com alto custo de memória. O DFS, por sua vez, mostrou-se subótimo, explorando soluções mais profundas e ineficientes. Este estudo reforça a importância da escolha de heurísticas adequadas para a resolução de problemas de busca e abre espaço para a exploração de outros algoritmos em problemas de maior complexidade.**

***Index Terms*—Breadth-First Search, Depth-First Search, A*, Algoritimos de Busca, Heurística, 8Puzzle**

I. Introducão

A resolução de problemas de busca muitas vezes se beneficia de métodos heurísticos e meta-heurísticos, que são amplamente utilizados em Inteligência Artificial. Heurísticas são ferramentas que ajudam a encontrar soluções aproximadas com base em informações disponíveis, direcionando a busca de forma mais eficiente [3]. Segundo Pearl, essas técnicas são fundamentais para guiar a solução de problemas complexos, como o 8-puzzle [6].

O 8-puzzle é um quebra-cabeça clássico de busca em Inteligência Artificial, onde o objetivo é transformar o estado inicial em um estado objetivo, deslizando o bloco vazio na grade 3x3 [7]. Este problema pode ser descrito como um jogo de agente único, que pode ser modelado pela descrição PEAS (Desempenho, Ambiente, Atuadores e Sensores). O desempenho do agente é avaliado pela eficiência com que ele atinge o estado objetivo no menor número de movimentos, com a ajuda de uma função heurística que calcula o valor baseado no estado atual do jogo. O ambiente é composto pelo agente, o estado inicial e o estado objetivo, enquanto os atuadores são responsáveis pelos movimentos de ”Deslizar para Esquerda”, ”Deslizar para Direita”, ”Deslizar para Cima”e ”Deslizar para Baixo”. A função heurística atua como o ”sensor”do agente, fornecendo uma métrica para medir a distância entre o estado inicial e o objetivo [2].

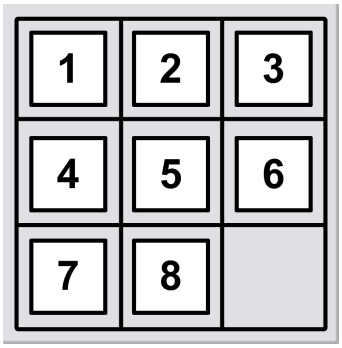

Figura 1. Representação visual do jogo 8-puzzle, mostrando o estado objetivo. O objetivo é reorganizar os blocos numerados deslizando-os até que atinjam a configuração final.

A Figura 1 ilustra visualmente o jogo do 8-puzzle e apresenta um exemplo de estado objetivo utilizado neste estudo.

Além de seu uso na IA, o 8-puzzle tem aplicações práticas, como no design de jogos, no desenvolvimento de algoritmos de planejamento e busca em robótica, e como ferramenta de ensino para introdução a conceitos de algoritmos e heurísticas. Também pode ser utilizado como modelo em problemas logísticos e em criptografia, devido à similaridade entre a solução do quebra-cabeça e o processo de otimização ou descriptografia.

Este artigo compara diferentes algoritmos de busca informada, como Breadth-First Search, Depth-First Search e A*, utilizando heurísticas de Distância de Manhattan e Euclidiana. O objetivo é avaliar o desempenho dessas abordagens em termos de tempo de execução e número de nós explorados, com experimentos realizados em 30 puzzles gerados aleatoriamente. A análise visa destacar as vantagens e limitações de cada algoritmo, bem como suas possíveis aplicações em cenários semelhantes.

II. Fundamentação Teórica

Nesta seção, são apresentados os conceitos teóricos fundamentais para a compreensão dos algoritmos de busca utilizados neste estudo.

A. Breadth-First Search (BFS)

O BFS é um algoritmo de busca em largura que explora sistematicamente todos os níveis do espaço de estados antes de avançar para o próximo nível [10]. Sua principal característica é que ele é completo e ótimo em grafos não ponderados, encontrando sempre o caminho mais curto quando os custos de transição entre nós são iguais. Contudo, o BFS pode ser ineficiente em termos de consumo de memória, especialmente em grafos grandes, pois mantém todos os nós de cada nível na memória durante a busca [9].

B. Depth-First Search (DFS)

O DFS é um algoritmo de busca em profundidade que explora um caminho até seu final antes de retroceder e explorar alternativas. Embora seja mais eficiente em termos de memória do que o BFS, o DFS não é garantido ser completo ou ótimo, pois pode seguir longos caminhos subótimos antes de encontrar uma solução, ou até mesmo entrar em loops em grafos cíclicos [1].

C. A*

O A* é o algoritmo de busca heurística mais utilizado para problemas de busca de caminhos e travessia de grafos. Ele combina a busca de custo uniforme com uma função de avaliação heurística, que fornece um custo estimado para alcançar o objetivo a partir do estado atual, permitindo obter soluções otimizadas de maneira eficiente. No problema do 8-puzzle, o A* examina o estado inicial e gera os estados sucessores movendo uma peça por vez. Ele calcula o valor $f(n)$ para cada estado sucessor utilizando a equação de custo do caminho:

$$f(n) = g(n) + h(n) \tag{1}$$

onde $g(n)$ é o custo desde o estado inicial e $h(n)$ é o custo heurístico até o estado objetivo. O A* seleciona o estado com o menor valor de $f(n)$ e o expande até atingir o estado objetivo. Para o 8-puzzle, as heurísticas de Manhattan e Euclidiana são empregadas nesta pesquisa, cada uma fornecendo uma abordagem única para estimar o custo de alcançar o objetivo.

1) Heurística de Manhattan: A heurística de Manhattan é amplamente utilizada para resolver o problema do 8-puzzle. Ela calcula a distância em termos de movimentos necessários para alinhar cada bloco à sua posição objetivo, somando as distâncias absolutas nos eixos horizontal e vertical, conforme a equação abaixo:

$$d_M(k) = |x_k - x_{kg}| + |y_k - y_{kg}| \tag{2}$$

onde (x_k, y_k) são as coordenadas do bloco no estado atual, e (x_{kg}, y_{kg}) são suas coordenadas no estado objetivo.

2) Heurística Euclidiana: A heurística Euclidiana calcula a distância direta entre dois blocos no plano do 8-puzzle usando o Teorema de Pitágoras [5]. A distância entre o bloco atual e sua posição objetivo é dada por:

$$d_E(k) = \sqrt{(x_k - x_{kg})^2 + (y_k - y_{kg})^2} \tag{3}$$

onde (x_k, y_k) e (x_{kg}, y_{kg}) correspondem às coordenadas do bloco no estado atual e no estado objetivo, respectivamente.

III. Metodologia

Nesta seção, detalhamos o processo seguido para implementar, executar e avaliar os algoritmos de busca aplicados aos quebra-cabeças. A metodologia inclui a descrição do ambiente de teste, a geração dos quebra-cabeças, a execução dos algoritmos, e os critérios de avaliação utilizados.

A. Ambiente de Teste

Os experimentos foram conduzidos em um computador equipado com um processador AMD Ryzen 5 1600 Six-Core, operando a 3,2 GHz, e com 16 GB de RAM. A implementação dos algoritmos foi realizada na linguagem de programação Python, devido à sua robustez e ampla disponibilidade de bibliotecas para manipulação de dados e grafos.

B. Geração dos Quebra-cabeças

Para avaliar os algoritmos de maneira consistente, foram gerados aleatoriamente 30 quebra-cabeças distintos, todos com o formato padrão de 8-puzzle (3x3). Cada quebra-cabeça foi projetado com diferentes configurações iniciais das peças, garantindo uma distribuição variada da peça sem número (espaço vazio). Essas configurações abrangem desde puzzles mais simples até os mais complexos, com diferentes níveis de dificuldade e possibilidades de movimento, permitindo testar os algoritmos em uma ampla gama de cenários.

C. Execução dos Algoritmos

Quatro algoritmos de busca foram implementados e executados uma vez em cada um dos quebra-cabeças gerados: Breadth-First Search, Depth-First Search e A* com duas heurísticas diferentes, Distância de Manhattan e Distância Euclidiana. No total, foram realizadas 120 execuções (30 quebra-cabeças x 4 algoritmos). Esse procedimento foi escolhido para avaliar o desempenho dos algoritmos em uma ampla variedade de configurações, assegurando a consistência e a reprodutibilidade dos experimentos.

Durante a execução, foram monitoradas as seguintes métricas:

- Tempo de execução: Medido em segundos, indica o tempo total necessário para o algoritmo encontrar a solução do quebra-cabeça.
- Consumo de memória: Avalia a quantidade máxima de memória RAM consumida pelo algoritmo durante a execução, medido em KB.
- Completude: Verifica se o algoritmo foi capaz de encontrar uma solução para o quebra-cabeça. Um algoritmo é

considerado completo se encontra uma solução sempre que ela existe.
- Optimalidade: Verifica se o algoritmo encontrou o caminho mais curto possível. Um algoritmo é considerado ótimo se sempre encontra a solução de menor custo quando esta existe.
- Nós explorados: Contabiliza a quantidade de nós que o algoritmo processa durante a busca, fornecendo uma métrica de eficiência espacial.

Essas métricas fornecem uma visão abrangente do desempenho de cada algoritmo, permitindo analisar tanto a eficiência temporal quanto a espacial, bem como a completude e a capacidade de encontrar soluções de maneira ótima.

IV. Resultados

Nesta seção, apresentamos e analisamos os resultados obtidos a partir da execução dos algoritmos BFS, DFS e A* (com heurísticas de Manhattan e Euclidiana) nos 30 quebra-cabeças gerados. Os resultados foram comparados com base em quatro critérios principais: tempo de execução, consumo de memória, completude, optimalidade e nós explorados. A seguir, discutimos cada critério em detalhe, realizando uma análise comparativa dos algoritmos e interpretando as variações observadas nos diferentes cenários testados.

A. Tempo de Execução

O tempo de execução é uma métrica fundamental para avaliar a eficiência dos algoritmos de busca. Conforme ilustrado na Tabela I e na Figura 2, o algoritmo A* com heurística de Manhattan apresentou o melhor desempenho, com um tempo médio de 0,64 segundos por quebra-cabeça. Isso ocorre devido à precisão dessa heurística, que reflete diretamente as regras do 8-puzzle, onde apenas movimentos horizontais e verticais são permitidos. Como resultado, o A* com Manhattan é capaz de guiar a busca de forma eficiente pelos caminhos mais promissores, minimizando o número de estados explorados e reduzindo o tempo de execução.

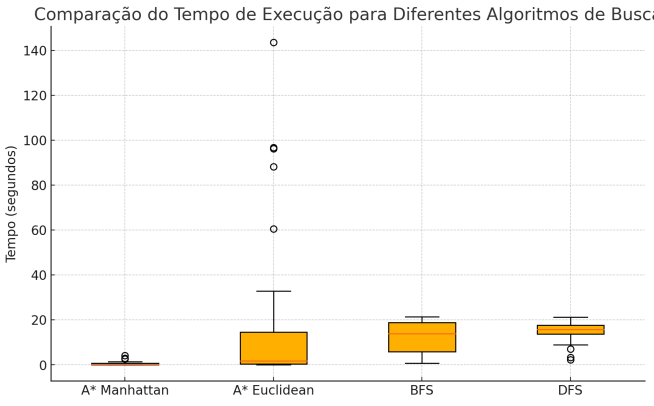

Figura 2. Comparação do Tempo de Execução para Diferentes algoritmos de busca.

Em contraste, o A* com heurística Euclidiana, que tem uma média de 20,07 segundos por quebra-cabeça, mostrou uma maior variação nos tempos de execução, com outliers significativos. A heurística Euclidiana considera a distância direta (em linha reta) entre os blocos e suas posições finais, o que inclui movimentos diagonais, não permitidos no 8-puzzle. Isso resulta em estimativas menos precisas do custo real e, consequentemente, em uma exploração mais extensa do espaço de estados, além de envolver cálculos matemáticos mais complexos, como raízes quadradas, o que contribui para o tempo elevado.

Os algoritmos não informados, Breadth-First Search (BFS) e Depth-First Search (DFS), tiveram desempenhos intermediários. O BFS apresentou um tempo médio de 12,75 segundos, com tempos mais estáveis, mas ligeiramente superiores ao A* com heurística de Manhattan. Como o BFS explora todos os nós em cada nível antes de avançar, ele garante a solução ótima, mas sacrifica o tempo de execução em problemas com grandes espaços de estados. O DFS, por sua vez, teve um tempo médio de 14,94 segundos. Embora o tempo médio seja maior que o do BFS, o DFS apresentou menos variação nos tempos de execução, conforme evidenciado pela Figura 2, o que indica que o DFS foi mais consistente, porém menos eficiente em geral.

Esses resultados demonstram a importância da escolha de heurísticas adequadas para algoritmos de busca informada. A heurística de Manhattan provou ser altamente eficaz para o 8-puzzle, ao passo que a heurística Euclidiana, embora admissível, introduz sobrecargas adicionais que afetam negativamente o desempenho. Tanto o BFS quanto o DFS apresentaram tempos previsíveis, porém, ainda assim, superiores ao A* com Manhattan. O gráfico de boxplot na Figura 2 ilustra essas diferenças de maneira clara, destacando a variação dos tempos de execução entre os algoritmos.

B. Consumo de Memória

O consumo de memória foi uma métrica importante avaliada durante a execução dos algoritmos. Como esperado, o Breadth-First Search (BFS) apresentou o maior consumo de memória, com uma média de aproximadamente 73.172 KB, devido ao fato de explorar amplamente o espaço de busca, armazenando todos os nós gerados até encontrar a solução. Isso se reflete na necessidade de armazenar múltiplos caminhos em diferentes níveis da árvore de busca, característica comum em algoritmos em largura como o BFS.

Por outro lado, o Depth-First Search (DFS) teve um desempenho de memória relativamente melhor que o BFS, com uma média de cerca de 66.760 KB, embora com uma variação significativa no consumo. Isso se deve à natureza do DFS de explorar profundamente um caminho antes de retroceder, resultando em menos nós sendo armazenados simultaneamente. No entanto, em quebra-cabeças mais complexos, como mostrado na Figura 3, o DFS pode consumir mais memória ao explorar caminhos longos e reverter frequentemente.

Os algoritmos A* com heurísticas de Manhattan e Euclidiana apresentaram um consumo de memória significativamente menor em comparação com os algoritmos não informados. A média para o A* com Manhattan foi de aproximadamente

Tabela I
TEMPO DE EXECUÇÃO (S) PARA DIFERENTES ALGORITMOS EM 30 QUEBRA-CABEÇAS

8-Puzzle	A* Manh.	A* Eucl.	BFS	DFS
1	0,020	0,054	5,845	7,079
2	0,079	0,377	4,737	19,144
3	0,301	2,768	16,729	10,417
4	0,021	0,038	1,636	16,485
5	4,130	96,676	21,303	15,430
6	0,297	2,367	17,143	2,306
7	0,108	0,488	6,842	15,071
8	0,066	0,295	5,554	15,489
9	1,467	96,251	21,349	16,305
10	0,018	0,115	4,666	17,671
11	0,074	0,159	5,858	20,389
12	0,995	4,648	17,468	13,673
13	2,994	143,519	21,370	20,350
14	1,053	32,912	19,291	15,734
15	1,369	60,476	20,628	19,995
16	0,050	0,255	6,189	21,159
17	0,481	15,003	19,651	17,747
18	0,150	0,623	9,058	14,179
19	0,266	6,508	17,641	17,579
20	0,082	0,716	10,595	19,344
21	0,727	24,246	19,992	14,570
22	0,108	1,188	10,312	16,365
23	0,161	1,248	11,819	18,177
24	0,530	4,583	15,079	16,220
25	0,237	1,126	12,634	12,478
26	0,009	0,017	0,763	15,344
27	0,269	3,966	17,241	11,719
28	0,021	0,064	2,539	3,238
29	2,547	88,210	21,158	8,856
30	0,612	13,301	17,429	15,830
Média	0,641	20,073	12,751	14,945

704 KB, enquanto a versão com a heurística Euclidiana teve um consumo médio de 2.445 KB. O menor uso de memória pelos algoritmos A* pode ser explicado pelo uso de heurísticas que guiam a busca de forma mais eficiente, restringindo a exploração a áreas mais promissoras do espaço de busca, o que reduz a quantidade de nós armazenados em memória.

A Figura 3 apresenta o boxplot comparativo do consumo de memória entre os algoritmos. Observa-se que o BFS possui a maior dispersão de valores, com altos picos de consumo de memória, enquanto os algoritmos A* têm um comportamento mais controlado, com menor variação. O DFS também apresenta um consumo elevado, embora mais estável em comparação ao BFS.

Esses resultados deixam claro que o consumo de memória é significativamente menor nos algoritmos A*, o que pode ser atribuído ao uso das heurísticas, que limitam a exploração a estados promissores. Por outro lado, tanto o BFS quanto o DFS, apesar de resolverem o problema, demandam um uso de memória consideravelmente maior, especialmente o BFS, que explora exaustivamente todos os nós antes de avançar para o próximo nível da árvore de busca.

C. Completude

Todos os algoritmos testados — BFS, DFS, A* com heurística de Manhattan e A* com heurística Euclidiana — foram capazes de encontrar a solução para os 30 quebra-cabeças do 8-puzzle gerados aleatoriamente, confirmando sua

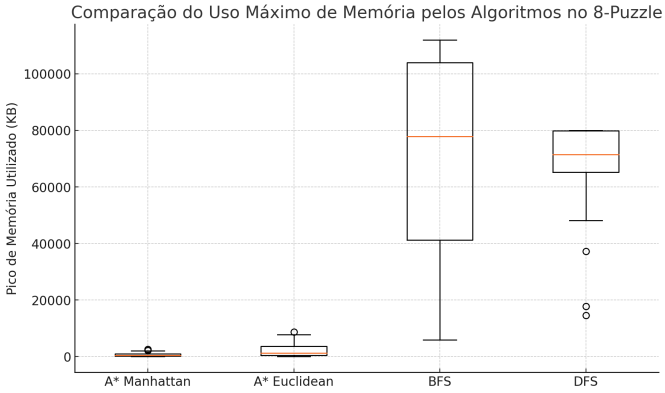

Figura 3. Comparação de Consumo de Memória entre Algoritmos

completude no contexto deste estudo. Mesmo o DFS, que não garante completude em todos os cenários, conseguiu resolver cada quebra-cabeça gerado. Os fatores que contribuíram para a completude observada são:

- Simplicidade dos Quebra-cabeças: Os quebra-cabeças 8-puzzle utilizados possuem um espaço de busca relativamente pequeno e bem estruturado, o que favorece a completude de todos os algoritmos, inclusive o DFS, que poderia, em outros cenários, não encontrar uma solução.
- Possibilidade de Movimentação Limitada: No 8-puzzle, os movimentos permitidos são restritos a direções específicas (esquerda, direita, cima e baixo). Essa característica, juntamente com a natureza do problema, torna a busca mais controlável e, portanto, aumenta a probabilidade de que todos os algoritmos testados encontrem uma solução.
- Estado Objetivo Bem Definido: A estrutura dos quebra-cabeças garante que existe uma solução alcançável a partir do estado inicial, o que foi suficiente para assegurar que cada algoritmo conseguisse atingir o estado objetivo sem se perder em caminhos inválidos ou loops.

Essa análise ressalta que, apesar das limitações teóricas do DFS em espaços de busca mais complexos, no contexto do 8-puzzle e com os quebra-cabeças testados, ele se comportou de maneira completa, assim como os demais algoritmos informados (A* Manhattan e Euclidiana) e o BFS.

D. Optimalidade

A avaliação da optimalidade dos algoritmos foi realizada comparando a profundidade dos caminhos encontrados com a profundidade obtida pelo BFS, que, por definição, sempre encontra o caminho mais curto em grafos não ponderados. Tanto o A* com heurística de Manhattan quanto o A* com heurística Euclidiana demonstraram ser ótimos em todos os 30 quebra-cabeças, encontrando sempre o caminho de menor profundidade, equivalente ao do BFS.

Por outro lado, o DFS apresentou um desempenho significativamente subótimo. Embora tenha conseguido encontrar uma solução para todos os quebra-cabeças, os caminhos explorados foram consideravelmente mais profundos em comparação aos

algoritmos A* e BFS. Isso ocorre porque o DFS explora um caminho profundamente até o final, sem garantir que esse caminho seja o mais curto, resultando em soluções com profundidades muito maiores do que o necessário.

E. Nós Explorados

O número de nós explorados por cada algoritmo foi registrado para avaliar sua eficiência na busca por uma solução. A Tabela II apresenta os resultados para os 30 quebra-cabeças testados.

Tabela II
NÚMERO DE NÓS EXPLORADOS PELOS ALGORITMOS EM 30 QUEBRA-CABEÇAS

8-Puzzle	A* Manh.	A* Eucl.	BFS	DFS
1	119	211	51.039	64.794
2	393	924	40.614	177.233
3	1.070	2.885	145.075	95.070
4	116	181	13.200	152.933
5	4.806	14.221	176.118	149.432
6	1.098	2.758	157.406	22.215
7	516	1.102	60.935	144.766
8	358	810	49.846	147.617
9	2.890	13.891	177.391	140.494
10	119	400	39.549	171.729
11	371	537	54.068	208.834
12	2.340	3.934	160.905	119.076
13	3.949	16.272	177.926	192.359
14	2.316	8.479	164.022	143.826
15	2.692	11.152	173.921	186.311
16	273	725	52.533	204.491
17	1.396	6.110	165.978	163.375
18	631	1.276	76.205	128.344
19	941	4.285	144.666	162.362
20	381	1.299	89.408	186.901
21	1.716	7.688	167.713	131.572
22	488	1.823	84.208	147.021
23	623	1.787	92.607	168.311
24	1.417	3.580	120.933	142.885
25	829	1.755	101.143	109.309
26	32	69	6.255	137.069
27	918	3.318	144.181	102.898
28	111	249	21.647	28.260
29	3.688	13.135	175.942	80.156
30	1.598	5.879	146.474	142.133
Média	1.273	4.357	107.730	138.392

Os resultados indicam uma clara diferença entre os algoritmos informados (A* com heurísticas de Manhattan e Euclidiana) e os algoritmos não informados (BFS e DFS):

- A* com heurística de Manhattan: Esse algoritmo explorou o menor número de nós, com uma média de 1.273 nós por quebra-cabeça. Isso reflete a eficiência da heurística de Manhattan, que guia o algoritmo de forma mais direta para a solução, limitando a quantidade de estados explorados.

- A* com heurística Euclidiana: Embora tenha explorado mais nós do que a versão com Manhattan, com uma média de 4.358 nós, o A* Euclidiano ainda apresentou melhor desempenho em relação aos algoritmos não informados. A heurística Euclidiana considera distâncias diagonais, que não são permitidas no 8-puzzle, resultando em mais estados explorados.

- BFS (Breadth-First Search): O BFS, que explora todos os nós em cada nível da árvore de busca antes de avançar, teve um número significativamente maior de nós explorados, com uma média de 107.730 nós. Como o BFS não usa heurísticas, ele precisa verificar todos os caminhos possíveis até encontrar a solução, resultando em uma exploração mais extensa do espaço de busca.

- DFS (Depth-First Search): O DFS foi o algoritmo que mais explorou nós, com uma média de 138.392 nós. Isso se deve à sua abordagem de seguir um caminho em profundidade até o final antes de retroceder e tentar caminhos alternativos. Como resultado, o DFS frequentemente explora muitos estados irrelevantes, especialmente em quebra-cabeças mais complexos.

Esses resultados demonstram a importância do uso de heurísticas para reduzir a quantidade de nós explorados em algoritmos de busca. O A* com Manhattan é claramente o mais eficiente neste aspecto, enquanto o DFS, sem o auxílio de heurísticas, explora um número significativamente maior de nós, o que compromete sua eficiência.

V. CONCLUSÃO

Neste trabalho, foram implementados e analisados quatro algoritmos de busca (BFS, DFS, A* com heurísticas de Manhattan e Euclidiana) aplicados ao problema do 8-puzzle. A avaliação foi realizada com base em quatro critérios principais: tempo de execução, consumo de memória, completude, optimalidade e número de nós explorados. Os experimentos mostraram resultados importantes e divergentes entre os algoritmos, ressaltando as vantagens de algoritmos informados como o A* e as desvantagens de algoritmos não informados como o DFS e o BFS.

O algoritmo A* com a heurística de Manhattan demonstrou a melhor eficiência em termos de tempo de execução, consumo de memória e número de nós explorados. Isso reflete a adequação dessa heurística ao problema do 8-puzzle, onde apenas movimentos horizontais e verticais são permitidos, tornando a busca mais direta e econômica. Em comparação, o A* com a heurística Euclidiana apresentou desempenho inferior em tempo e memória, pois considera movimentos diagonais que não são permitidos no 8-puzzle, o que leva a uma estimativa menos precisa dos custos e a mais estados explorados. No entanto, ambos os algoritmos A* garantiram sempre a solução ótima, encontrando o caminho de menor custo em todos os quebra-cabeças testados.

Os algoritmos não informados, como o BFS e o DFS, também conseguiram resolver todos os quebra-cabeças, confirmando sua completude. No entanto, o BFS teve um consumo de memória significativamente elevado, explorando um número maior de nós para garantir a solução ótima. Já o DFS, embora eficiente no uso de memória em alguns casos, explorou um número excessivo de nós, resultando em soluções subótimas em termos de profundidade e tempo de execução.

Para trabalhos futuros, sugerimos a investigação de outros algoritmos de busca, como Iterative Deepening A* (IDA*) e algoritmos baseados em aprendizagem, como o Deep Q-Learning, que poderiam trazer melhorias em termos de eficiência tanto em memória quanto em tempo de execução. Além disso, ampliar os experimentos para quebra-cabeças

maiores, como o 15-puzzle, seria um passo importante para avaliar a escalabilidade dos algoritmos testados. A análise de algoritmos em problemas de maior dimensão poderia fornecer dados adicionais sobre as limitações e potencial de cada técnica em espaços de busca mais complexos.

REFERÊNCIAS

[1] AWERBUCH, B. A new distributed Depth-First-Search algorithm. Information Processing Letters, 1985. Available at: https://www.sciencedirect.com/science/article/abs/pii/0020019085900833.

[2] A. K. Mishra and P. C. Siddalingaswamy, "Analysis of tree based search techniques for solving 8-puzzle problem,"in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2017, pp. 1-5, doi: 10.1109/IPACT.2017.8245012.

[3] S. Costa, T. Retes, and G. Botelho, "Avaliação dos algoritmos A* e Genético na resolução do 8 Puzzle," in *Proceedings*, Oct. 2018.

[4] GEETAPRIYAA, S., PILLAI, Niranjana R., K, Aswin C., MENON, Maya. Graph-Based Algorithm For Mobile Robot Navigation In A Known Environment. IEEE Xplore, 2019. Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-87, Department of Computer Science and Engineering, India.

[5] Iordan, E., 2016. A Comparative Study of Three Heuristic Functions used to Solve the 8-Puzzle. British Journal of Mathematics & Computer Science. 16, 1-18. doi:10.9734/BJMCS/2016/24467.

[6] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[7] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education Limited, Malaysia, 2016.

[8] R. Jain, "Investigating the Impact of Different Search Strategies (Breadth First, Depth First, A*, Best First, Iterative Deepening, Hill Climbing) on 8-Puzzle Problem Solving - A Case Study,"*Journal*, vol. 8, pp. 633-641, Feb. 2023, doi: 10.1729/Journal.33807.

[9] PATEL, A. R., and PATEL, A. Comparative Analysis of Stereo Matching Algorithms. IEEE, Oct 2019. Available at: https://ieeexplore.ieee.org/document/8992965 doi:10.1109/UEMCON47517.2019.8992965.

[10] SIMIC, Milos. The Informed Vs. Uninformed Search Algorithms. 2021.