Estudo comparativo de métodos de busca para resolução de puzzles

Julia Veloso Dias
1314675@sga.pucminas.br
Instituto de Ciências Exatas e Informática (ICEI)
Belo Horizonte, Minas Gerais, Brasil

ABSTRACT

O estudo investiga a aplicação de vários algoritmos de busca na resolução de quebra-cabeças deslizantes, um problema clássico em inteligência artificial e ciência da computação que envolve a rearrumação de peças numeradas em uma grade. Três abordagens principais de busca - A* (A Estrela), Busca Gulosa e Busca Uniforme - são exploradas e comparadas no contexto de quebra-cabeças deslizantes, com foco em diferentes tamanhos e complexidades de quebra-cabeças.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

A* (A Estrela), Busca Gulosa, Busca Uniforme, quebra-cabeças deslizantes

ACM Reference Format:
Julia Veloso Dias. 2024. Estudo comparativo de métodos de busca para resolução de puzzles. In *Proceedings of PUC Minas, Inteligência Artificial (Atividade Prática 2)*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Os algoritmos de busca desempenham um papel fundamental na resolução de problemas em inteligência artificial. Este trabalho explora a aplicação de diferentes algoritmos de busca na resolução de quebra-cabeças deslizantes, uma classe de problemas clássicos que envolve a manipulação de peças numeradas em uma grade. O objetivo do código visa analisar e contrastar a eficiência e o desempenho dos algoritmos A*, Busca Gulosa e Busca Uniforme quando aplicados a instâncias do problema de quebra-cabeças deslizantes. A pesquisa investiga como diferentes heurísticas impactam o comportamento dos algoritmos e compara as compensações entre otimalidade e custo computacional.

Atividade Prática 2, Nov 2023, Belo Horizonte, MG, Brasil

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 METODOLOGIA

Os algoritmos foram implementados e testados em quebra-cabeças deslizantes de mesmo tamanho e complexidade, pois é necessário fazer a análise de performance de cada um com o mesmo cenário. O desempenho de cada algoritmo é avaliado com base no tempo de execução e quantidade de passos dados para se chegar ao resultado esperado.

Foi utilizado um estudo em um quebra-cabeça de matriz 3x3 e variação de números de 0 a 8, sendo o 0 uma representação do espaço vazio que esse tipo de problema exige para que possam ser feitas as movimentações. A quantidade de movimentação é definida de uma forma diferente de acordo com o método de busca implementado, gerando resultados distintos.

Dessarte, o estudo busca demonstrar essa diferente avaliação de métodos de forma comparativa. Foram realizados testes em dois cenários: o primeiro é em que a casa vazia, representada pelo 0 na matriz de números do quebra-cabeça deslizante, está na primeira casa; o segundo cenário é no qual a casa vazia é a última casa, e o padrão dos outros números é variável na matriz inicial e, para a matriz resultante (objetivo a ser alcançado), ambos apresentam ordem crescente dos números entre 1 e 8, definidos para teste.

Esses experimentos foram conduzidos na linguagem Python, utilizando um computador com a seguinte configuração: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz.

3 REFERENCIAL TEÓRICO

Nesta seção serão contemplados os principais conceitos envolvidos neste estudo, esses tais, conceitos e descriação do funcionamento dos algoritmos utilizados para realização da comparação de desempenho dos diferentes métodos.

A* (A Estrela)

A* representa uma classe de algoritmos essenciais para a busca de caminhos e travessia de grafos em diversos contextos. A seleção de uma função A específica determina o algoritmo a ser utilizado dentro dessa família. O funcionamento desse algoritmo é fundamentado na manutenção de uma fila de prioridade contendo nós a serem expandidos, sendo que o nó com a menor estimativa de custo é priorizado. A estimativa de custo para um nó consiste na soma do custo para alcançar esse nó a partir do ponto inicial e o custo estimado para atingir o nó de destino partindo desse ponto. A projeção do custo é calculada por meio de uma função heurística, que oferece uma estimativa da distância restante até o nó de destino. A* expande os nós em ordem crescente da estimativa de custo até alcançar o nó de destino.[3]

O algoritmo A* segue uma abordagem best-first, onde o custo associado a um estado é calculado pela função $f(n) = g(n)+h(n)$, em que: $g(n)$ representa o custo do caminho percorrido do estado inicial até o nó n e $h(n)$ é o custo estimado do caminho ou o custo da função heurística do nó n até o nó de destino. Dessa forma, $f(n)$ reflete o menor custo total possível para qualquer caminho que passe pelo nó n até o estado de destino, considerando $h(n)$ como o custo estimado restante do percurso. Em situações em que o valor f de vários nós é igual, o algoritmo resolve o impasse selecionando o nó com valores inferiores de estimativa heurística $h(n)$ (ordenamento dos nós). Esse processo continua até que o nó a ser expandido corresponda ao nó de destino, marcando a conclusão bem-sucedida do algoritmo. O A* destaca-se como uma abordagem eficiente e flexível na busca por soluções otimizadas em uma variedade de contextos.[4]

Distância de Manhattan. A Distância de Manhattan é uma função heurística amplamente empregada na resolução do problema do quebra-cabeça N. Essa heurística fundamenta-se na ideia de que o quebra-cabeça pode ser solucionado movendo cada peça ao longo do caminho mais curto até sua posição final. Na modelagem da Distância de Manhattan, a solução ótima consiste em um conjunto de sub-soluções, uma para cada peça. Uma sub-solução é qualquer caminho mais curto para uma peça específica, indo de sua posição atual até sua posição final. Essa abordagem efetiva destaca a estratégia de decompor a resolução do quebra-cabeça em soluções parciais, focadas em cada peça individualmente. Essa heurística simplifica o processo de busca por soluções otimizadas, reforçando a eficácia do modelo de Distância de Manhattan na resolução do problema N.[2]

Número de peças fora do lugar. A heurística Número de peças fora do lugar *(Number of misplaced tiles)* é uma métrica utilizada na resolução de quebra-cabeças, onde cada peça que não está em sua posição correta é contada. Essa abordagem considera que, a partir de qualquer configuração inicial, uma peça pode ser retirada e movida para qualquer posição desejada. O algoritmo associado a essa heurística busca encontrar a solução movendo cada peça da posição atual para a posição em sua configuração desejada. O comprimento do caminho de menor custo é, portanto, determinado pela contagem de peças que não estão em suas posições desejadas. [4]

Busca Gulosa *Best-First* (*Greedy Search*)

O algoritmo Busca Gulosa *Best-First*é o extremo lógico do A* ponderado. Em vez de escalonar $h(n)$ em relação a $g(n)$, a busca gulosa ignora completamente $g(n)$. Em uma busca best-first, o algoritmo avalia cada nó no grafo com base em uma função heurística que estima quão próximo o nó está do estado final desejado. O algoritmo então seleciona o nó que é estimado como o mais próximo do estado final e o expande, gerando seus nós vizinhos. Esse processo continua até que o estado final seja alcançado ou nenhum outro nó possa ser expandido. [1]

Busca Uniforme *(Uniform Cost Search)*

O algoritmo de custo uniforme é uma variação do algoritmo A* usado para encontrar o caminho mais curto em um grafo, partindo de um nó inicial em direção a um nó de destino. Nele, os nós são expandidos em ordem crescente de custo. Começando com um custo de 0 para o nó inicial e infinito para os demais, o algoritmo armazena o nó inicial no bucket 0. Ao esvaziar os buckets em ordem crescente de custo, um nó retirado verifica se pode propagar custos para nós adjacentes. Se o custo original do nó adjacente for maior que o custo recalculado (soma do custo do nó gerador e o custo da aresta), o nó vai para o bucket correspondente, aguardando processamento. O algoritmo continua esvaziando buckets e propagando custos até atingir o nó de destino ou processar todos os nós. Essa abordagem serve como uma técnica de busca heurística em situações onde o uso de informações heurísticas não é possível ou demorado. [5]

4 RESULTADOS DOS ALGORITMOS IMPLEMENTADOS

Nessa seção será discutido como foi realizado a implementação dos métodos de busca, com base nas seguintes matrizes 3x3 com objetivo de demonstrar o desenvolvimento da implementação dos códigos.

Inicialmente foi apresentado o seguinte quebra-cabeça de 8 puzzles:

$$\begin{bmatrix} 7 & 2 & 4 \\ 5 & 0 & 6 \\ 8 & 3 & 1 \end{bmatrix}$$

Para alcançar a seguinte solução:

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

A* (A Estrela) com Distância de Manhattan

O algoritmo A* foi implementado utilizando duas heurísticas diferentes para a mesma situação, para alcançar o referido resultado, a primeira resolução proposta consistiu na aplicação da heurística Distância de Manhattan. Essa heurística implica na soma das distâncias horizontais e verticais entre as posições dos elementos no estado atual e no estado objetivo. Importante salientar que esta heurística é admissível, isto é, nunca superestima o custo necessário para atingir o objetivo.

A partir desse ponto, o procedimento adentra a iteração dos nós, representativos dos números do quebra-cabeça. O algoritmo A* mantém duas listas: uma de nós abertos e outra de nós fechados. Em cada iteração, o nó com o menor custo total (soma do custo real e da heurística) é selecionado, expandido e, consequentemente, os estados vizinhos são gerados por meio de uma função que encapsula as ações possíveis (tais como movimentos para cima, baixo, esquerda ou direita).

Posteriormente, é invocado o método de reconstrução do caminho. Esse método percorre a árvore de busca de maneira reversa, iniciando a partir do nó final (estado objetivo) e seguindo os ponteiros dos pais de cada nó.

Dado a matriz anterior, foi obtido os seguintes resultados com a implementação descrita: foram necessários 26 passos para alcançar o objetivo, enquanto o tempo decorrido para a resolução foi de aproximadamente 0.63 segundos. Durante esse processo, um total de 6.778 nós foram visitados, evidenciando a eficiência do algoritmo na exploração do espaço de busca para encontrar a solução desejada.

A* (A Estrela) com Número de peças fora do lugar *(Number of misplaced tiles)*

A segunda heurística integrada ao algoritmo A* consiste na contagem do número de peças fora do lugar, uma estratégia frequentemente aplicada na resolução de quebra-cabeças utilizando esse algoritmo.

A fase inicial envolve a avaliação da quantidade de peças que não estão posicionadas corretamente. Essa heurística oferece uma estimativa da distância entre o estado atual e o estado objetivo. Com base nisso, os nós, que representam os diferentes estados na busca, são computados pelo algoritmo A*. Este algoritmo mantém duas listas: uma de nós abertos e outra de nós fechados. A cada iteração, o nó com o menor custo total (soma do custo real e da heurística) é cuidadosamente selecionado e expandido.

Segue-se, então, a fase de recostrução do caminho, um componente crucial para obter uma representação ordenada das ações necessárias para atingir o estado objetivo. Esse método percorre a árvore de busca de maneira reversa, indo do nó final até o nó inicial.

Dado a matriz anterior, foi obtido os seguintes resultados com a implementação descrita: a solução alcançada demandou um total de 26 passos, assim como com a heurística de Manhattan, porém o tempo decorrido para a resolução foi de aproximadamente 6.03 segundos. Durante esse processo, o algoritmo visitou 71.830 nós, evidenciando a abrangência e eficácia da abordagem na exploração do espaço de busca.

Busca Gulosa *Best-First* (*Greedy Best-First Search*)

Outra estratégia de busca implementada para resolver o quebra-cabeça proposto foi o algoritmo *Greedy Best-First Search*. Trata-se de uma abordagem heurística que prioriza a expansão de nós com base em uma heurística específica, sem considerar o custo total acumulado. No contexto do quebra-cabeça deslizante, a implementação utilizou a Distância Manhattan como heurística, quantificando a distância de cada peça em relação à sua posição correta no estado objetivo. Essa métrica é calculada como a soma das distâncias horizontais e verticais entre as posições das peças nos estados atual e objetivo, fornecendo uma estimativa da proximidade ou distância de um estado em relação à solução desejada.

O nó desempenha um papel central na representação dos estados no espaço de busca. Cada nó contém informações cruciais, como o estado atual, a referência ao nó pai, o custo acumulado até o momento e a heurística associada. A busca gulosa utiliza a heurística para determinar quais nós expandir, priorizando aqueles que parecem mais promissores com base na estimativa de proximidade ao estado objetivo. Isso é realizado por meio de um método responsável por gerar estados vizinhos a partir do estado atual do quebra-cabeça deslizante, representando as ações possíveis, ou seja, os movimentos permitidos no jogo. A expansão do nó é então avaliada.

A escolha da heurística de Distância Manhattan e a abordagem gulosa proporcionam uma busca direcionada, priorizando caminhos que parecem mais promissores com base na estimativa de proximidade ao objetivo.

Dado a matriz anterior, foi obtido os seguintes resultados com a implementação descrita: a solução encontrada exigiu um total de 36 passos, com o tempo decorrido para a conclusão baixo de aproximadamente 0.01 segundos. Durante esse processo, o algoritmo visitou um número reduzido de nós, alcançando um total de 123. Esses resultados destacam a rapidez e a eficiência do *Greedy Search* na exploração do espaço de busca, proporcionando uma solução efetiva com um

tempo de execução notavelmente baixo e uma quantidade mínima de nós visitados.

Busca Uniforme *(Uniform Cost Search)*

A Busca Uniforme, aplicada neste contexto, destaca-se pela sua abordagem sistemática na exploração dos nós com o menor custo acumulado. O nó desempenha um papel fundamental na representação dos estados no espaço de busca, contendo informações essenciais, como o estado atual, a referência ao nó pai e o custo acumulado até o momento.

Para implementar a Busca Uniforme, utiliza-se uma fila de prioridade que ordena os nós com base no custo total acumulado, priorizando aqueles com menor custo. Esse procedimento ocorre por meio de um método responsável por gerar estados vizinhos a partir do estado atual do quebra-cabeça. Cada estado vizinho é obtido movendo a célula vazia (representada pelo valor 0) para uma posição adjacente. Por exemplo, ao mover a célula vazia para cima, realiza-se uma troca entre o valor na posição atual da célula vazia e o valor na posição acima dela. Esse processo é repetido para todas as direções possíveis, gerando os estados vizinhos essenciais para a busca do caminho ótimo.

A Busca Uniforme destaca-se por assegurar uma solução ótima, sendo particularmente útil em situações em que o custo da solução é um critério crucial.

Dado a matriz anterior, foi obtido os seguintes resultados com a implementação descrita: a solução encontrada demandou de 20 passos, demonstrando a eficácia dessa abordagem na resolução do problema. Além disso, o tempo decorrido para a conclusão foi de aproximadamente 10.39 segundos, durante os quais a Busca Uniforme visitou um total de 133.788 nós. Esses resultados destacam a capacidade desta estratégia em alcançar soluções eficientes, mesmo em cenários onde o custo da solução é um critério crucial, evidenciando seu desempenho robusto na exploração do espaço de busca.

5 VALIDAÇÃO E COMPARAÇÃO DOS RESULTADOS

Para validar e comparar os resultados, foram definidos 10 cenários de teste, apresentados na Tabela 1. O objetivo é realizar uma análise média do desempenho dos algoritmos A* com ambas as heurísticas, Busca Gulosa e Busca Uniforme em diferentes situações. Ao final, será feita uma comparação de desempenho, levando em consideração os resultados dessa média.

Ao realizar os testes com os quebra-cabeças apresentados, foi possível alcançar as médias apresentadas na Figura 1.

Os resultados apresentados na tabela fornecem uma visão abrangente do desempenho dos diferentes algoritmos de busca. No contexto do problema em questão, três métricas fundamentais foram avaliadas: a média de nós visitados, a média de tempo de execução e a média de passos necessários para atingir a solução.

O modelo A* com a heurística de Distância de Manhattan demonstra eficiência, mantendo uma média relativamente baixa de nós visitados (22,2) e um tempo de execução razoavelmente rápido (0,1634). Além disso, o número médio de passos para a solução é de 3271,1, indicando um desempenho sólido.

No entanto, ao empregar a heurística baseada no Número de Peças Fora do Lugar no modelo A*, observamos uma pequena redução na eficiência. Embora a média de nós visitados permaneça similar (21,8), o tempo de execução aumenta para 0,9732, e o número médio de passos atinge 21925,4, sugerindo uma possível complexidade adicional.

A Busca Uniforme apresenta um tempo de execução notavelmente baixo (0,0073), mas à custa de uma média significativamente mais alta de nós visitados (52,6). Esta abordagem destaca a rapidez da busca, mas também evidencia a exploração extensiva de estados.

A Busca Gulosa, por outro lado, demonstra uma eficiência impressionante em termos de nós visitados (16,6), mas o tempo de execução (3,614) e o número de passos (136212,4) são substancialmente elevados. Isso sugere que a busca gulosa pode não ser a escolha ideal quando se busca um equilíbrio entre eficiência e tempo.

Em comparação, o A* com a heurística de Distância de Manhattan emerge como uma opção balanceada, combinando uma eficiência notável com um tempo de execução razoável.

Table 1: Cenários para realização de testes

Cenários de Teste
7, 2, 4, 5, 0, 6, 8, 3, 1
7, 2, 4, 5, 6, 8, 3, 1, 0
2, 8, 4, 5, 1, 6, 3, 7, 0
7, 2, 8, 4, 3, 5, 1, 6, 0
5, 2, 6, 3, 1, 0, 7, 4, 8
1, 2, 3, 6, 5, 8, 7, 4, 0
7, 3, 8, 2, 6, 4, 1, 0, 5
4, 7, 0, 8, 3, 1, 5, 2, 6
6, 2, 4, 8, 7, 0, 5, 1, 3
5, 7, 2, 6, 1, 8, 3, 4, 0

Figure 1: Resultados das médias avaliadas para três métricas: a média de nós visitados, a média de tempo de execução e a média de passos necessários para atingir a solução.

Modelo	*Média de nós visitados*	*Média de tempo*	*Média de passos*
A* Distância de Manhattan	22,2	0,1634	3271,1
A* Número de Peças Fora do Lugar	21,8	0,9732	21925,4
Busca Gulosa	52,6	0,0073	334,1
Busca Uniforme	16,6	3,614	136212,4

6 CONCLUSÃO

A análise comparativa destes resultados destaca que cada algoritmo possui suas próprias vantagens e limitações. A Busca Uniforme garante soluções ótimas, mas pode ser computacionalmente mais custosa. A Busca Gulosa oferece rapidez, porém não garante otimalidade. A* com diferentes heurísticas apresenta um equilíbrio entre eficiência e otimalidade, permitindo escolher a abordagem mais adequada conforme as prioridades do problema. A compreensão destes resultados é crucial para a seleção informada da estratégia mais apropriada, considerando as demandas específicas de cada contexto de aplicação.

7 CÓDIGO DESENVOLVIDO

Para esse trabalho o código desenvolvido com devidos métodos pode ser acessado através do link para o github: *8-puzzle*.

REFERENCES

[1] J. E. Doran and D. Michie. 1966. Experiments with the graph traverser program. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* (1966), 235–259.

[2] O. Hansson, A. E. Mayer, and M. Yung. 1985. *Generating admissible heuristics by criticizing solutions to relaxed models.* Technical Report. Columbia University, New York, NY, USA.

[3] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[4] D. Nayak. 2014. Analysis and implementation of admissible heuristics in 8 puzzle problem. (2014).

[5] B.J.H. Verwer, P.W. Verbeek, and S.T. Dekker. 1989. An efficient uniform cost algorithm applied to distance transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 4 (1989), 425–429. https://doi.org/10.1109/34.19041