MATERIAL DE ESTUDO - BUSCAS INFORMADAS E HEURÍSTICAS NO 8-PUZZLE

Revisão dos conceitos e exemplo resolvido com o 8-Puzzle

PARTE 1 - CONCEITO DE HEURÍSTICA E PROPRIEDADES

1. O que é uma função heurística h(n)?

Uma função heurística h(n) é uma estimativa do custo que ainda falta para sair de um estado n e chegar ao estado objetivo. Em uma busca informada, essa estimativa é usada para decidir quais estados parecem mais promissores e, por isso, devem ser analisados primeiro. A heurística não precisa informar o custo exato. Ela serve como uma orientação para a busca.

No 8-Puzzle, por exemplo, uma heurística pode estimar o quanto o tabuleiro atual está distante do tabuleiro objetivo. Quanto menor for h(n), mais próximo da solução o estado parece estar.

2. Heurística admissível e heurística consistente

Uma heurística é admissível quando nunca superestima o custo real mínimo até a solução. Em outras palavras, o valor calculado por h(n) deve ser menor ou igual ao custo verdadeiro que ainda falta para chegar ao objetivo. Se h*(n) representa esse custo real, a condição pode ser escrita como h(n) <= h*(n).

Uma heurística é consistente, ou monótona, quando a estimativa de um estado não é maior do que o custo para chegar a um estado vizinho somado à estimativa desse vizinho. Para um sucessor n' de n, a condição é h(n) <= c(n,n') + h(n'). Essa propriedade evita que a estimativa apresente quedas incoerentes ao longo de um caminho.

A admissibilidade é importante porque sustenta a garantia de solução ótima do A*. Em busca em grafo, a consistência evita variações incoerentes da estimativa entre estados vizinhos e permite lidar adequadamente com estados repetidos. No 8-Puzzle, a Distância de Manhattan é admissível e consistente quando cada movimento custa 1.

PARTE 2 - HEURÍSTICAS PARA O 8-PUZZLE

3. Peças Fora do Lugar e Distância de Manhattan

A heurística h1(n), chamada Peças Fora do Lugar (Misplaced Tiles ou Hamming Distance), conta quantas peças numeradas estão em posições diferentes das posições ocupadas no estado objetivo. O espaço vazio, representado por 0, normalmente não entra na contagem. Se quatro peças estiverem em posições incorretas, então h1(n) = 4.

A heurística h2(n), chamada Distância de Manhattan (Taxicab ou City Block Distance), calcula quantos deslocamentos horizontais e verticais cada peça precisa fazer, no mínimo, para alcançar sua posição no estado objetivo. As distâncias de todas as peças de 1 a 8 são somadas, também desconsiderando o espaço vazio.

A Distância de Manhattan usa mais informação do que apenas verificar se uma peça está certa ou errada. Uma peça pode estar fora do lugar e precisar de um, dois ou mais deslocamentos. Essa diferença aparece no valor da heurística.

4. Dominância entre heurísticas

Dizemos que uma heurística domina outra quando, para todos os estados, ela fornece estimativas maiores ou iguais às da outra sem deixar de ser admissível. Uma heurística dominante costuma ser mais informada, pois consegue diferenciar melhor estados que estão mais ou menos próximos da solução.

No 8-Puzzle, a Distância de Manhattan domina a heurística de Peças Fora do Lugar. Toda peça fora do lugar possui distância de Manhattan de pelo menos 1. Além disso, algumas peças podem estar a duas ou mais posições de distância do objetivo. Por isso, h2(n) é sempre maior ou igual a h1(n), considerando as mesmas regras e desconsiderando o espaço vazio.

PARTE 3 - ALGORITMOS DE BUSCA INFORMADA

5. Busca Gulosa (Greedy Best-First Search)

Na Busca Gulosa, a função de avaliação considera somente a heurística: f(n) = h(n). Isso significa que o algoritmo escolhe para expansão o estado que aparenta estar mais próximo do objetivo, sem considerar quanto já foi gasto para chegar até ele.

Esse comportamento pode fazer a busca avançar rapidamente quando a heurística indica um bom caminho, mas não existe garantia de que o caminho encontrado será o menor. A Busca Gulosa pode escolher uma alternativa que parece boa no início e depois exigir muitos movimentos. Em espaços de busca gerais, ela também não é considerada completa sem condições adicionais. No 8-Puzzle, que possui um espaço finito, uma implementação que controle estados repetidos pode encontrar uma solução quando ela for alcançável, mas ainda não há garantia de solução ótima.

6. Busca A* (A-Star)

No A*, a função de avaliação é f(n) = g(n) + h(n). O termo g(n) representa o custo já percorrido desde o estado inicial até o estado atual. O termo h(n) representa a estimativa do custo que ainda falta para chegar ao objetivo.

O A* equilibra passado e futuro. Ele não escolhe somente o estado que parece mais perto da meta, como a Busca Gulosa, nem considera apenas o custo já percorrido. Ao somar g(n) e h(n), o algoritmo prioriza estados com menor custo total estimado. No 8-Puzzle, com custo 1 por movimento e Distância de Manhattan, o A* pode encontrar um caminho ótimo quando a implementação trata corretamente os estados repetidos.

PARTE 4 - EXERCÍCIO PRÁTICO E COMPARATIVO

7. Cálculo das heurísticas no estado fornecido

Considere os seguintes estados do 8-Puzzle:

Observação importante sobre o notebook da aula. O estado objetivo desta atividade é (1, 2, 3 / 8, 0, 4 / 7, 6, 5), diferente do GOAL usado no notebook, que é (1, 2, 3 / 4, 5, 6 / 7, 8, 0). Todos os cálculos abaixo seguem o objetivo da atividade. Para reproduzir exatamente este exemplo no código, basta ajustar GOAL e start; a lógica de BFS, DFS e IDS permanece a mesma.

Estado atual

2	8	3
1	6	4
7	0	5

Estado objetivo

1	2	3
8	0	4
7	6	5

a) h1(n) - Peças Fora do Lugar

Comparando o estado atual com o objetivo, as peças 2, 8, 1 e 6 estão em posições diferentes. As peças 3, 4, 5 e 7 já estão nas posições corretas. O espaço vazio (0) não é contado.

h1(n) = 4

b) h2(n) - Distância de Manhattan

Para o cálculo abaixo, as posições são identificadas por linha e coluna, numeradas de 0 a 2, seguindo os índices usados no Python. A distância de cada peça é a diferença entre as linhas somada à diferença entre as colunas.

Peça	Posição atual	Posição objetivo	Distância
1	(1,0)	(0,0)	1
2	(0,0)	(0,1)	1
3	(0,2)	(0,2)	0
4	(1,2)	(1,2)	0
5	(2,2)	(2,2)	0
6	(1,1)	(2,1)	1
7	(2,0)	(2,0)	0

8	(0,1)	(1,0)	2

h2(n) = 1 + 1 + 0 + 0 + 0 + 1 + 0 + 2 = 5

c) Cálculo de f(n) no A*

Se o custo já percorrido for g(n) = 4 e a heurística utilizada for a Distância de Manhattan, então h(n) = 5. Aplicando a fórmula do A*:

$$f(n) = g(n) + h(n)$$
$$f(n) = 4 + 5$$
$$\mathbf{f(n) = 9}$$

8. Análise comparativa entre BFS, DFS, IDDFS/IDS e A*

A comparação pode ser feita de forma conceitual, considerando principalmente a quantidade de estados que cada estratégia tende a explorar, o uso de memória e a capacidade de encontrar o menor caminho. Nesta etapa, o notebook trabalhado em sala chega até IDS; A* e Busca Gulosa são estudadas conceitualmente e serão implementadas posteriormente.

A BFS realiza a busca por níveis. No 8-Puzzle, quando cada movimento possui o mesmo custo, ela consegue encontrar uma solução com o menor número de movimentos. O problema é que precisa manter uma grande quantidade de estados na fronteira. Conforme a profundidade aumenta, o número de possibilidades cresce rapidamente, fazendo com que o consumo de memória seja alto.

A DFS segue um caminho em profundidade antes de tentar outras alternativas. Em geral, utiliza menos memória do que a BFS, pois não precisa manter todos os estados de um mesmo nível ao mesmo tempo. Porém, pode entrar primeiro em um caminho pouco promissor e explorar muitos estados antes de chegar à solução. Também não garante o menor caminho.

A IDDFS, também chamada IDS, executa buscas em profundidade com limites crescentes. Primeiro tenta profundidade 0, depois 1, 2, 3 e assim por diante até encontrar a solução. Seu uso de memória é semelhante ao da DFS e ela consegue encontrar a solução de menor profundidade quando os custos dos movimentos são iguais. A desvantagem é repetir parte do trabalho a cada novo limite de profundidade.

O A* utiliza informação sobre o problema para escolher os estados. Com a Distância de Manhattan, ele prioriza tabuleiros com menor custo total estimado e costuma reduzir a exploração em comparação com buscas cegas. Isso não significa que sempre explorará menos estados em toda instância ou ordem de expansão. Além disso, o A* pode exigir bastante memória, porque mantém estados da fronteira e estados já explorados.

Algoritmo	Nós explorados	Uso de memória	Menor caminho
BFS	Pode ser alto	Alto	Sim, com custos iguais
DFS	Muito variável	Baixo	Não
IDDFS / IDS	Pode repetir estados	Baixo	Sim, com custos iguais

A* + Manhattan	Tende a ser menor	Pode ser alto	Sim, com heurística adequada

Um ponto importante é diferenciar o número de movimentos da solução do número de estados explorados. Dois algoritmos podem encontrar uma solução com a mesma quantidade de movimentos, mas um deles pode ter analisado muito mais estados até chegar ao resultado. É justamente nesse aspecto que uma boa heurística pode tornar a busca mais eficiente.

CONCLUSÃO

As buscas cegas não utilizam conhecimento específico sobre o problema para decidir qual estado deve ser explorado primeiro. Já as buscas informadas usam uma heurística para orientar essa escolha. No 8-Puzzle, a Distância de Manhattan é mais informada do que a contagem de Peças Fora do Lugar porque considera o quanto cada peça está distante de sua posição correta. O A* combina essa estimativa com o custo já percorrido e, por isso, tende a evitar parte das explorações desnecessárias realizadas pelas buscas cegas.

REFERÊNCIAS BIBLIOGRÁFICAS

RUSSELL, Stuart J.; NORVIG, Peter. Artificial Intelligence: A Modern Approach. 4. ed. Hoboken: Pearson, 2021.

HART, Peter E.; NILSSON, Nils J.; RAPHAEL, Bertram. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, v. 4, n. 2, p. 100-107, 1968.

KORF, Richard E. Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence, v. 27, n. 1, p. 97-109, 1985.