Jogo Puzzle

Davi Caetano Tavares Ramos
1398113

A implementação apresenta três algoritmos de busca: profundidade limitada (DFS-L), amplitude (BFS) e A*.

Esses algoritmos foram aplicados ao 8 puzzle, cujo objetivo é transformar o estado inicial em 1 2 3 4 5 6 7 8 0. O ambiente de teste fixou o tabuleiro inicial [[1, 2, 3], [4, 0, 5], [6, 7, 8]], atribuindo um custo unitário a cada deslocamento do espaço em branco. Todos os algoritmos partem da mesma representação de estado, armazenam o caminho percorrido e medem o número de nós expandidos, a profundidade da solução e o tempo decorrido com “time.perf_counter()” para permitir a comparação direta.

A busca em profundidade limitada foi configurada com um limite de 50, limite que é o suficiente para encontrar uma solução, mas não necessariamente para explorar todo o espaço. Sua operação sempre expande o sucessor mais profundo até atingir o teto de profundidade, o que produz margens de memória restritas, mas pode alongar o caminho até o objetivo.

A busca em largura visita nós em camadas equidistantes da raiz; a fila “deque” garante totalidade e uma optimização em ambientes de custo unitário, mas o algoritmo tende a consumir mais memória porque deve manter simultaneamente toda a fronteira do nível atual.

A* combina um custo cumulativo g(n) com uma estimativa h(n) do custo restante.

Para avaliar o impacto da qualidade heurística, implementamos a distância de Manhattan, que soma as distâncias horizontal e vertical de cada peça até sua posição objetivo, e a contagem de peças mal colocadas, ambas admissíveis de acordo com a definição clássica de heurística otimista e consistentes no sentido da desigualdade triangular.

Os experimentos produziram a seguinte saída:

Estado inicial:

1 2 3

4 5

6 7 8

DFS limit=50 | tempo = 0.9809 s | nós expandidos = 21473 | profundidade da solução = 46

BFS | tempo = 0.3537 s | nós expandidos = 4693 | profundidade da solução = 14

A* Manhattan | tempo = 0.0144 s | nós expandidos = 128 | profundidade da solução = 14

A* Misplaced | tempo = 0.0451 s | nós expandidos = 321 | profundidade da solução = 14

Profundidade: 0

1 2 3

4 5

6 7 8

Profundidade: 1

1 2 3

4 5

6 7 8

Profundidade: 2

1 2 3

4 5 8

6 7

Profundidade: 3

1 2 3

4 5 8

6 7

Profundidade: 4

1 2 3

4 5 8

 6 7

Profundidade: 5

1 2 3

  5 8

4 6 7

Profundidade: 6

1 2 3

5   8

4 6 7

Profundidade: 7

1 2 3

5 6 8

4   7

Profundidade: 8

1 2 3

5 6 8

4 7

Profundidade: 9

1 2 3

5 6

4 7 8

Profundidade: 10

1 2 3

5 6

4 7 8

Profundidade: 11

1 2 3

 5 6

4 7 8

Profundidade: 12

1 2 3

4 5 6

 7 8

Profundidade: 13

1 2 3

4 5 6

7 8

Profundidade: 14

1 2 3

4 5 6

7 8

DFS-L expandiu 21.473 nós, levou 0,981 s e retornou um caminho de profundidade 46; BFS expandiu 4.693 nós, levou 0,354 s e encontrou a solução ótima de profundidade 14; A* com Manhattan exigiu apenas 128 expansões e 0,014 s, enquanto A* com peças mal posicionadas exigiu 321 expansões e 0,045 s, ambas retornando profundidade 14. A diferença de desempenho aponta para dois aspectos.

Primeiro, heurísticas admissíveis e mais bem informadas reduzem drasticamente o fator de ramificação efetivo: Manhattan está mais próximo do custo real porque considera quantos movimentos cada peça ainda requer, enquanto a contagem de peças fornece apenas um limite inferior aproximado, daí o motivo para cerca de 2,5 vezes mais expansões.

Segundo, embora o BFS também seja ótimo neste cenário de custo uniforme, ele paga o preço de visitar todos os nós nas primeiras treze camadas antes de atingir o objetivo, gerando sobrecarga temporal e espacial que A* evita ao priorizar nós com menor f(n) = g(n) + h(n).

A busca com profundidade limitada, mesmo encontrando uma solução, ilustra o risco de trajetórias longas: 46 movimentos contra o ótimo de 14. Esse alongamento se deve à estratégia de descer por um único ramo até forçar o backtracking, característica que, combinada à ausência de heurísticas e ao limite arbitrário, impede a garantia de otimalidade e pode causar explosões de custos quando o fator de ramificação é alto.

O método que “melhor desenvolveu o problema” foi, portanto, o A* com distância de Manhattan. Com o menor tempo, com a menor expansão de nós e produzindo a solução ótima. Em segundo lugar ficou o A* com peças mal posicionadas, demonstrando que mesmo uma heurística simples já traz ganhos substanciais em comparação ao BFS. O BFS, por sua vez, serve como referência para completude e otimalidade sem informações heurísticas, enquanto o DFS limitado se mostrou menos eficaz para esta instância.

Dado o mesmo ponto de partida e custo uniforme, A* expande, no máximo, o conjunto de nós igualmente promissores de acordo com a heurística escolhida, e nenhuma estratégia ótima e completa pode se sair melhor com a mesma estimativa. Conclui-se que, para o 8-Puzzle e as heurísticas admissíveis, a distância de Manhattan oferece o melhor compromisso entre qualidade da solução e custo computacional dentre os algoritmos selecionados.

Fontes: https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/ e pdfs disponibilizados no canvas.