

Portfólio 2 - Projeto de busca não informada

1 Introdução

Este documento tem como foco apresentar o problema a ser resolvido, a implementação e os resultados de um algoritmo de Busca não informada, desenvolvido na linguagem Python, para a disciplina de Inteligência Artificial.

Foram utilizados a IDE pycharm com a biblioteca itertools para implementar o algoritmo de Busca em profundidade iterativa.

2 Problema

Imagine o "Quebra-Cabeça de 8"(ou 15). Ele consiste em um tabuleiro 3x3 com 8 peças numeradas e um espaço vazio, onde o objetivo é encontrar o número mínimo de movimentos para reorganizar as peças de um estado embaralhado até o estado ordenado.

Os algoritmos de busca tradicionais falham aqui a Busca em Largura seria capaz de encontrar a melhor solução, pois explora o tabuleiro nível por nível andando por todas as possibilidades. No entanto, o número de configurações possíveis é enorme, ultrapassando 180.000 para o 3x3. A BFS precisa armazenar todos os estados visitados em sua fila, o que poderia esgotaria a memória do computador.

Por outro lado, a Busca em Profundidade não sofre pelo problema de memória, pois armazena apenas o caminho atual mas ela não é adequada para este problema porque ela não é ótima, ela busca em um caminho e pode encontrar uma solução com 50 movimentos, mesmo que exista uma com 18 e tambem ela não é completa e pode ficar presa em um ciclo repetindo o mesmo movimento muitas vezes.

É aqui que a Busca em profundidade iterativa resolve o dilema. Ela executa uma série de Buscas em Profundidade, cada uma com um limite maior. Primeiro, ela faz uma DFS com limite 0 ao falhar, ela tenta uma DFS com limite 1 então ela continua esse processo, aumentando o limite para 2, 3, 4, e assim por diante uma hora ela tentará, por exemplo, uma busca com limite 18 e encontrará a solução.

3 Algoritmo desenvolvido de busca não informada: Busca em profundidade iterativa

A Busca em Profundidade Iterativa combina as melhores características de duas outras estratégias de busca, a eficiência de memória da Busca em Profundidade e a garantia de encontrar o caminho mais curto da Busca em Largura (BFS). O seu funcionamento baseia-se na ideia de executar uma série de buscas em profundidade com um limite *path* de profundidade que aumenta progressivamente. Ela funciona exatamente como uma DFS, explorando um caminho o mais fundo possível, mas com uma regra estrita, ela é forçada a parar e retroceder assim que atinge o limite de profundidade estabelecido para aquela iteração.

O processo começa definindo este limite *path* em 0 e executando a busca em profundidade. Se o estado inicial não for o objetivo, a busca falha. Então, o algoritmo descarta todo o progresso, incrementa o limite para 1 e inicia uma busca completamente nova a partir do estado inicial, mas agora permitindo que a busca em profundidade explore um nível mais fundo. Se a solução não for encontrada, ele repete o processo, aumentando o limite para 2, depois 3, e assim por diante. Em cada nova iteração, a busca recomeça do zero, re-explorando os níveis anteriores, mas agora com permissão para ir um passo adiante.

3.1 Exemplo de uso

Primeiro foi criada a estrutura do puzzle depois foram definidos dois JSON's um para o estado inicial e outro para o estado final. A partir deles o algoritmo foi iniciado como mostrado na Figura 1. A biblioteca `itertools` foi utilizada para auxiliar na execução da busca.

```
--- 8-Puzzle Solver ---  
  
Estado Inicial:  
1 2 3  
4 0 5  
7 8 6  
  
Estado Alvo:  
1 2 3  
4 5 6  
7 8 0  
  
Tentando com profundidade limite: 0  
Tentando com profundidade limite: 1  
Tentando com profundidade limite: 2  
  
Solução encontrada na profundidade 2!  
Caminho da solução: ['Direita', 'Baixo']  
Total de movimentos: 2  
  
--- Início ---  
1 2 3  
4 0 5  
7 8 6  
  
--- Movimento 1 (Direita) ---  
1 2 3  
4 5 0  
7 8 6  
  
--- Movimento 2 (Baixo) ---  
1 2 3  
4 5 6  
7 8 0
```

Figura 1: Execução da busca em profundidade iterativa