Faculdade de Ciências da Universidade do Porto

Inteligência Artificial CC2006 2015/2016 - Trabalho 1 - Jogo dos 8

INTRODUÇÃO

Este trabalho teve como objetivo a implementação e comparação de várias estratégias de procura para solucionar o jogo dos 8.

Este jogo consiste num tabuleiro com 9 peças, numeradas de 1 a 8 de maneira não sequencial e com uma peça vazia. A finalidade principal do jogo é ordenar as peças utilizando a casa vazia para as movimentar, chegando à uma matriz final pré-definida pelo utilizador no inicio do jogo.

ESTRATÉGIAS DE PROCURA

PROCURA NÃO GUIADA

1. Busca em Profundidade

Na busca em profundidade expande-se completamente cada ramo da árvore antes dos ramos vizinhos, ou seja, por cada nó expandido sucede-se a expansão do seu filho. O algoritmo é bastante económico no que diz respeito ao espaço utilizado, pois a cada iteração guarda apenas os nós que ainda não foram expandidos e o caminho da solução. Temporalmente, pode ser bastante rápido em encontrar a solução quando o problema apresenta várias resoluções ou existem muitos caminhos que conduzem à mesma. Este método de procura não termina nos casos em que a árvore de procura contém ciclos.

- Espaço: b*m
- Tempo: O(b^m), b = fator de ramificação e m = profundidade máxima

2. Busca em Largura

O algoritmo consiste em construir uma árvore de estados a partir do estado inicial, gerando em cada nível todos os estados sucessores de cada um dos estados, isto é, expande todos os nós do nível onde se encontra antes de passar para o nível a seguir. O algoritmo requer bastante espaço porque guarda todas as jogadas possíveis duma profundidade, antes de passar para a profundidade seguinte.

- Espaço: b*l

- Tempo: O(b^l), b = factor de ramificação e l = profundidade da solução.

3. Busca Iterativa Limitada em Profundidade

A busca em profundidade iterativa procura combinar as virtudes da busca em profundidade e da busca em largura. Essas virtudes são a relativamente pouca necessidade de memória e a capacidade de examinar todo o espaço de estados encontrando sempre a solução ótima.

- Espaço: b*m
- Tempo: O(b^m), b = fator de ramificação e m=profundidade máxima

PROCURA GUIADA

1. Busca Gulosa

Expande os nós que se encontram mais próximos do objetivo, desta maneira é provável que a busca encontre uma solução rapidamente, mas não a a solução óptima. A implementação do algoritmo é semelhante a da busca em profundidade, no entanto utiliza-se uma função heurística para decidir o nó que deve ser expandido.

- Espaço: O(b^m)
- Tempo: O(b^m)

2. A*

De forma semelhante à busca gulosa, este algoritmo usa uma função heurística, mas essa função é 'aperfeiçoada' pois também considera a profundidade, mas é preciso ter em conta que esse aperfeiçoamento só garante soluções optimas se a função heurística for crescente e monótona.

- Espaço: O(b^m)
- Tempo: O(b^m)

3. DESCRIÇÃO DA IMPLEMENTAÇÃO

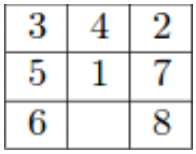
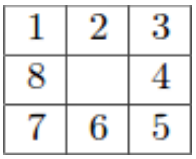
A linguagem de programação utilizada foi o JAVA, pois é a linguagem com a qual estamos familiarizados e providencia muitas bibliotecas.

Em todos os métodos usamos uma *HashSet* para guardar os nós visitados. Na procura em profundidade e procura em profundidade iterativa usamos uma *Stack* (pilha) para guardar os nós expandidos e saber qual o proximo a explorar. Por outro lado, na profundidade em largura usamos uma *Queue* (fila). Na procura gulosa e A* usamos uma 'PriorityQueue' que guarda os nós expandidos ordenados pelo custo.

O que nos fez escolher estas estruturas foi o facto de serem as mais eficientes para os problemas em causa.

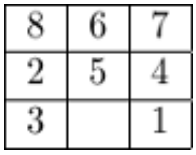
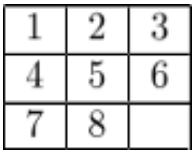
4. RESULTADOS

Para testar a implementação dos algoritmos começamos por utilizar a configuração composta pelas seguintes tabelas, cujo o número de movimentos esperado, conforme indicado pela professora, é 23.

Estado inicial	Estado final
	

Estratégia	Tempo (segundos)	Espaço (Nós)	Espaço(Mb)	Profundidade da Solução
DFS	0.357	167779	49.962	45887
BFS	0.283	116989	26.468	23
iDFS	0.909	572126	68.877	23
Greedy	0.012	269	1.800	47
A*	0.108	13154	7.326	23

Após investigar quais as tabelas cuja solução requer mais passos, descobrimos que existe um grupo de tabelas, das quais faz parte a configuração a seguir apresentada, que só têm solução após um mínimo de 31 movimentos.

Estado inicial	Estado final
	

Estratégia	Tempo (segundos)	Espaço (Nós)	Espaço(Mb)	Profundidade da Solução
DFS	0.217	104549	31.818	54365
BFS	0.412	181641	80.472	31
iDFS	8.806	5787060	286.543	31
Greedy	0.007	163	1.800	51
A*	0.066	6013	4.200	31

COMENTÁRIOS FINAIS E CONCLUSÕES

Analisando os dados conseguimos perceber que os algoritmos baseados em métodos informados de pesquisa se destacam em tempo de execução e espaço utilizado. Sendo a busca gulosa a mais "poupada" em recursos, mas não garantindo uma solução ótima. O A* sendo apenas ligeiramente mais "caro" garante sempre a solução ótima.

Trabalho Realizado Por:

José Gabriel - 201007482

Raul Ferreira - 200905641