Inteligência Artificial 2021/22

Relatório de Projeto – Grupo 56

Introdução:

Este projeto tem como objetivo desenvolver um programa que dado como input um problema de numbrix, devolve a sua solução. Para obter a solução recorre-se aos algoritmos DFS, BFS, Greedy e A*.

Escolha de Ações:

Definimos uma ação como um conjunto de inserções de uma única posição no board. Ou seja, uma ação implica a alteração de uma ou mais posições no board.

Consideramos que a melhor abordagem para a escolha de ações selecionar as posições preenchidas de início e a partir destas obter a sequência de menor tamanho entre duas destas posições (por exemplo: se um board inicial tivesse as posições 1, 7 e 9, seriam escolhidos 7 e 9).
Após escolher a sequência de menor tamanho, calculamos todas as hipóteses de caminho que o board pode ter entre essas duas posições. Ao escolher sempre as sequências de menor tamanho permite à árvore de procura ter um tamanho mais reduzido.

Posteriormente implementamos uma funcionalidade que não dava apenas prioridade ao tamanho da sequência mas também ao número de sequências possíveis que esta devolvia. Ou seja, se uma sequência for de tamanho reduzido mas devolver muitas possibilidades, é preferível escolher uma sequência maior mas que ofereça um espectro menor de possibilidades no board.

Cálculo de sequências possíveis:

Utilizamos a class *TreeNode* para expandir um certo início de sequência até ao seu fim. Uma das técnicas utilizadas para aliviar o tamanho da árvore gerada foi o de calcular a distância de Manhattan do nó atual até ao nó de destino, caso fosse impossível, dá essa tentativa de caminho como impossível.

As sequências possíveis dos números iniciais e finais do board (1 e o quadrado do tamanho do board) calculadas após todas as outras, pois nestas sequências não sabemos com segurança onde fica o extremo (1 ou o quadrado do tamanho do board).

Análise de dados:

Algoritmo *DFS*:

Nº teste	1	2	3	4	5	6	7	8 e 9	10
Tempo Real	0.68s	0.080s	0.078s	0.77s	0.079s	0.111s	0.369s	0.327s	0.3s
Nós gerados	2	51	15	59	42	32	145	174	175
Nós expandidos	2	47	12	44	29	29	97	167	169

Algoritmo *BFS*:

Nº teste	1	2	3	4	5	6	7	8 e 9	10
Tempo Real	0.066s	0.116s	0.077s	0.116s	0.107s	0.118s	0.741s	0.395s	0.371s
Nós gerados	2	136	15	204	71	32	234	212	215
Nós expandidos	2	136	15	204	71	32	234	212	215

Algoritmo *Greedy*:

Nº teste	1	2	3	4	5	6	7	8 e 9	10
Tempo Real	0.061s	0.1s	0.075s	0.078s	0.090s	0.112s	0.3s	0.117s	0.1s
Nós gerados	2	105	14	51	70	25	117	44	40
Nós expandidos	2	94	13	14	29	22	36	30	25

Algoritmo *A**:

Nº teste	1	2	3	4	5	6	7	8 e 9	10
Tempo Real	0.071s	0.109s	0.076s	0.080s	0.088s	0.111s	0.207s	0.120s	0.099s
Nós gerados	2	125	14	51	70	25	107	44	40
Nós expandidos	2	114	13	14	28	22	45	30	25

Embora todos os algoritmos acima sejam completos, na nossa submissão, escolhemos **DFS**, pois apesar dos algoritmos que recorrem à heurística terem tempos ligeiramente melhores e gerarem menos nós, excedemos a memória fornecida pelo mooshak mais facilmente quando utilizamos qualquer um destes algoritmos.
A heurística que utilizamos calcula para um certo board a soma do número de todos os vizinhos ocupados que todos os espaços vazios do board têm.

Embora tenhamos testado outras heurísticas esta foi a que nos deu resultados mais consistentes.
Os testes foram corridos numa máquina com Ubuntu 20.04.4 LTS, processador AMD Ryzen 9 5900X @ 4.5Ghz e 32GB de RAM.

Francisco Sousa - 95579
Sara Aguincha - 95674