Inteligência Artificial

Projeto 1 - Relatório

2018/2019

Diogo Marante 86409

João Daniel Silva 86445

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procura Gananciosa - Greedy | | | | |
|  | Número de peças | Tempo de Execução (s) | Número de nós expandidos | Número de nós gerados |
| m4x4 | 14 | 0.006068468094 | 20 | 56 |
| m4x5 | 16 | 0.5789146423 | 5798 | 5852 |
| m5x5 | 11 | 0.002222537994 | 10 | 21 |
| m4x6 | 20 | 0.05686926842 | 172 | 306 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procura em Profundidade Primeiro - DFS | | | | |
|  | Número de peças | Tempo de Execução (s) | Número de nós expandidos | Número de nós gerados |
| m4x4 | 14 | 0.1879117489 | 5986 | 6003 |
| m4x5 | 16 | 1.759344339 | 53636 | 53664 |
| m5x5 | 11 | 0.0007915496826 | 13 | 20 |
| m4x6 | 20 | NULL | NULL | NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procura A\* (A-Star) | | | | |
|  | Número de peças | Tempo de Execução (s) | Número de nós expandidos | Número de nós gerados |
| m4x4 | 14 | 0.006244897842 | 16 | 64 |
| m4x5 | 16 | 0.4389920235 | 3445 | 3550 |
| m5x5 | 11 | 0.001917600632 | 10 | 21 |
| m4x6 | 20 | 0.03278660774 | 39 | 179 |

Análise dos Resultados

Obtivemos os resultados acima executando as funções de procura fornecidas pelo corpo docente, disponíveis no ficheiro search.py. As matrizes executadas foram as disponibilizadas no enunciado. Para obtermos as medições de tempo, utilizámos a biblioteca *time* do Python, e para a informação dos *nodes,* através da dos atributos implementados na classe *InstrumentedProblem.*

(Comparar resultados da RNL com o do meu PC?, dizendo que a capacidade do CPU influencia o tempo de execução)

Os factores que influenciam o número de nós gerados são a dimensão do problema e o número de peças no tabuleiro. O tabuleiro 4x6 é o maior e tem o maior número de peças, sendo por isso o que mais demora a ser resolvido por qualquer algoritmo. Apesar do tabuleiro 5x5 ser o segundo maior, é o que tem o menor número de peças, por isso é muito mais rapidamente resolvido que os restantes.

Em relação aos algoritmos, a procura em profundidade é a mais ineficiente de todas, tanto em tempo de execução e em número de nós gerados. Na matrix 4x6 torna-se impraticável a sua aplicação.

A procura A\* é mais eficiente. A nossa heurística consiste no cálculo das jogadas necessárias para completar o jogo e do número de peças que não se podem mover. Se existirem muitas peças isoladas, esse é um estado que mais dificilmente estará perto da solução final, e por isso a sua escolha deve ser desfavorecida. Assumim2os que cada peça isolada poderá se mover numa jogada seguinte, quando outra peça se mova para uma posição que lhe permita a jogada. Na matriz 4x6, a procura A\* gera ~ 64% do número de nós da procura greedy, a**pesar de demorar quase o mesmo tempo (92%)**.