Inteligência Artificial Relatório de Projeto

Grupo 12

Francisco Augusto - 99218

Luís Marques - 99265

Introdução:

O problema do Takuzu consiste no preenchimento de um tabuleiro n x n, tendo em consideração um conjunto de restrições. Este projeto consistia numa tentativa de encontrar um programa que encontra a solução a qualquer tabuleiro Takuzu da forma mais eficiente possível.

Heurística:

A heurística que utilizamos na realização deste projeto consiste na preferência de tabuleiros com mais espaços preenchidos, desde que os mesmos não quebrem quaisquer restrições. Dessa forma se o algoritmo tiver que escolher entre duas jogadas vai preferir a jogada que origina o tabuleiro com mais espaços preenchidos, simultaneamente desprezando quaisquer jogadas que levem ao preenchimento incorreto de um espaço vazio. Por exemplo se num tabuleiro de 15 x 15 tivermos 3 jogadas possíveis (A, B e C, respetivamente). Considerando que a jogada A leva a que o tabuleiro fique com 54 espaços, a jogada B fica com 82 espaços vazios, e a jogada C fica com 46 espaços, mas apresenta uma sequência de 3 0's, então a heurística escolheria a jogada A.

Resultados:

Analisando os dados relativos à procura em largura primeiro (BFS) podemos verificar este é o algoritmo que gera o maior número de nós, nos tabuleiros 3, 4, 5 e 6, o que deve ser devido a uma situação que o algoritmo não é capaz de tomar decisão. Também é notório que à medida que o input vai aumentando, também se verifica um grande aumento do tempo de execução do algoritmo.

Relativamente ao algoritmo de procura em profundidade primeiro (DFS), o mesmo também apresenta um grande aumento de tempo à medida que o tamanho do input aumenta. Tal como no algoritmo anterior, verifica-se que o algoritmo expande nós nos testes referidos anteriormente, apesar de aparentar expandir menos nós.

A procura gananciosa também demonstra ter um aumento com o aumento do tamanho do input, e segue o mesmo padrão dos algoritmos anteriores, não necessitando de expandir nós exceto nos testes 3, 4,5 e 6.

Por fim o algoritmo A*, também segue o mesmo raciocínio, tendo um grande aumento do tempo de execução em acompanhamento do tamanho do input, apesar de os ramos que expandem mais nós serem os testes 3, 4, 5 e 6.

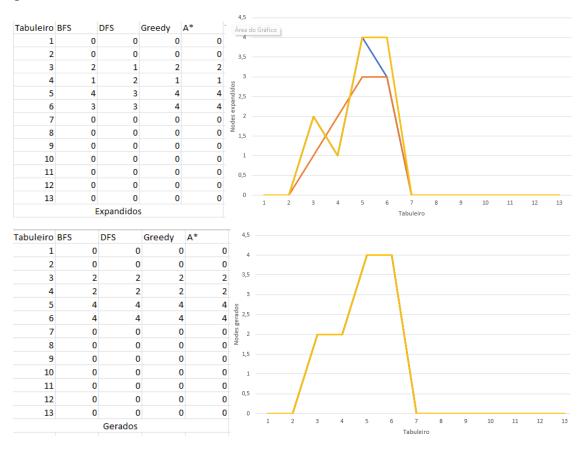
Relativamente aos tempos de execução, podemos realçar que nos inputs utilizados nenhum algoritmo é preferível, tendo os quatro apresentado tempos bastante semelhantes, salvo algumas exceções.

Completude:

Um algoritmo é considerado completo caso o mesmo apresente sempre uma solução a um dado problema, ou caso a mesma não exista, seja capaz de identificar a inexistência da mesma. Sendo assim, podemos, com base nos inputs utilizados, afirmar que o algoritmo com as 4 procuras é completo, o que se encontra de acordo com os dados teóricos.

Eficiência:

Com base nos resultados obtidos não há grande discrepância entre os tempos de execução dos algoritmos. Apesar disso, consideramos que com base nos nós expandidos durante a procura, a procura em profundidade primeiro revela-se como a mais eficiente, dado expandir menos nós que as restantes. Dessa forma, fazendo uma análise geral dos dados, podemos organizar os algoritmos em ordem de eficácia da seguinte forma: Procura em profundidade primeiro, procura em largura primeiro, A* e procura gananciosa.



Tabuleiro	BFS	DFS	Greedy	A*
1	0,177	0,154	0,156	0,144
2	0,15	0,15	0,162	0,16
3	0,234	0,237	0,236	0,251
4	0,244	0,297	0,253	0,271
5	0,374	0,341	0,38	0,372
6	0,489	0,465	0,537	0,534
7	0,45	0,452	0,486	0,458
8	0,422	0,404	0,466	0,439
9	1,015	1,035	1,045	1,006
10	1,554	1,503	1,521	1,539
11	1,698	1,755	1,78	1,843
12	2,707	2,598	2,638	2,618
13	5,292	5,212	5,502	5,479
		Time		

