Grupo AL023 - Mafalda Mendes, 83502 - Margarida Morais, 86473

Relatório Projeto 1
Inteligência Artificial (2018/19)

Introdução

Este projeto consiste num conjunto de funções que ajudam um agente a resolver diferentes puzzles (tabuleiros) de uma variante do jogo Solitaire, independentemente da sua dimensão e do seu conteúdo.

O tabuleiro usado para este jogo tem dimensões linhas x colunas e pode nem sempre ser quadrado ou retangular. As peças do jogo são representadas pelo carater **"O"**, os espaços livres são representados por **"_"** e os espaços que estão bloqueados são representados por **"X"**.

Existem **quatro condições** para fazer uma **jogada válida**:

1) A peça só se pode mover ortogonalmente (cima, baixo, esquerda, direita);
2) A peça só pode ser movida, se passar por cima de outra peça e eliminá-la do jogo;
3) Uma peça não ficar em cima de outra que já estava nessa posição;
4) Não é possível mover uma peça para um espaço bloqueado ("X").

O **objetivo** do jogo é chegar ao fim com apenas uma peça no tabuleiro, independentemente da posição onde está.

O relatório está organizado em quatro partes, sendo que na primeira é explicado a escolha da **heurística**, na segunda os **testes e resultados** usados para fazer a avaliação da procura, e por fim uma **discussão dos resultados**.

1. Heurística

No caso das procuras informadas (**A*** e **Greedy Search**), foi escolhida uma composição linear de heurísticas, sendo estas:

1) Dar menor peso, e como tal maior preferência, aos tabuleiros cuja posição final da ação leva a peça para algum canto do mesmo;
2) Quanto menos peças tiver um tabuleiro, menor será o peso do nó, dando preferência a esse nó.

A solução do tabuleiro é encontrada, quando a função *goal_test* retorna *True*, ou seja, o número de peças existentes no tabuleiro é 1.

2. Exemplos de Teste

- **Tabuleiro 1** de 5x5 (linhas x colunas):
  [["_","O","O","O","_"],["O","_","O","_","O"],["_","O","_","O","_"],["O","_","O","_","_"],["_","O","_","_","_"]]
- **Tabuleiro 2** de 4x4 (linhas x colunas):
  [["O","O","O","X"],["O","O","O","O"],["O","_","O","O"],["O","O","O","O"]]
- **Tabuleiro 3** de 4x5 (linhas x colunas):
  [["O","O","O","X","X"],["O","O","O","O","O"],["O","_","O","_","O"],["O","O","O","O","O"]]
- **Tabuleiro 4** de 4x6 (linhas x colunas):
  [["O","O","O","X","X","X"],["O","_","O","O","O","O"],["O","O","O","O","O","O"],["O","O","O","O","O","O"]]

3. Resultados dos Testes

Procura-Gananciosa

Tabuleiro	Tempo de execução (s)	Nós gerados	Nós expandidos
1	0.000000	19	11
2	0.015625	78	53
3	4.593750	21 912	21 872
4	0.593750	1 810	1 729

Procura em Profundidade Primeiro

Tabuleiro	Tempo de execução (s)	Nós gerados	Nós expandidos
1	0.015625	20	12
2	0.640625	6 002	5 984
3	7.375000	53 664	53 636
4	3 065.750000 (51 min)	14 760 576	14 760 524

Procura A*

Tabuleiro	Tempo de execução (s)	Nós gerados	Nós expandidos
1	0.000000	19	11
2	0.015625	78	53
3	4.500000	21 912	21 872
4	0.562500	1 810	1 729

4. Discussão dos Resultados*

Neste relatório utilizamos apenas as procuras **A***, **Greedy**, **Depth First Graph Search** para testarmos a complexidade das heurísticas utilizadas e também do programa em si.

Tal como se pode verificar pelos tempos obtidos, vemos que o tempo utilizado pelos dois algoritmos de procura informada (**A*** e **Greedy**) é muito menor que aquele utilizado pela **DFS**. Isto deve-se ao facto de a DFS expandir todos os nós em profundidade até encontrar o objetivo, enquanto os algoritmos de procura informada apenas expandem os nós que têm menor *peso*, recorrendo às heurísticas para diferenciar o *peso* de cada nó.

O aumento dos nós gerados na **DFS** é exponencial, podemos verificar que, adicionando apenas mais uma linha ou coluna no tabuleiro, o **número de nós** aumenta cerca de **8 vezes** do tabuleiro 2 para o tabuleiro 3, e cerca de **275 vezes** do tabuleiro 3 para o tabuleiro 4. O tempo que a **DFS** demora a procurar a solução nestes problemas é explicada pela complexidade do código utilizado nas funções auxiliares, como a ***board_moves***, ***boar_perform_moves***, *etc…* , uma vez que a **DFS** não recorre às heurísticas para fazer a procura.

Tal como na **DFS** em que o aumento de nós deve-se ao aumento das dimensões do tabuleiro ou peças existentes, também na **A*** e no **Greedy** o aumento de nós deve-se à mesma coisa. No entanto, notamos que o aumento dos nós não é exponencial como na **DFS**.

No tabuleiro 3, quando é resolvido recorrendo aos algoritmos de procura **A*** e **Greedy**, podemos verificar que este tabuleiro tem um aumento de tempo em relação aos outros acompanhado também por um aumento significativo no número de nós gerados (e expandidos). Isto deve-se ao facto de a heurística utilizada não ser a melhor para este problema, provavelmente, devido à dificuldade do tabuleiro em relação aos outros.

Verificando sempre os resultados obtidos, foi sempre retornado um último nó cujo tabuleiro correspondia ao estado objetivo.

***Estes testes foram feitos numa máquina Intel Core i7 a 1.80GHz, 16GB**