U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Folding Blocks

Métodos de Pesquisa Heurística para Resolução de Jogo do Tipo Solitário

João Araújo, 201705577
Jorge Pacheco, 201705754
Leonor Sousa, 201705377

Inteligência Artificial
Mestrado Integrado em Engenharia Informática e Computação

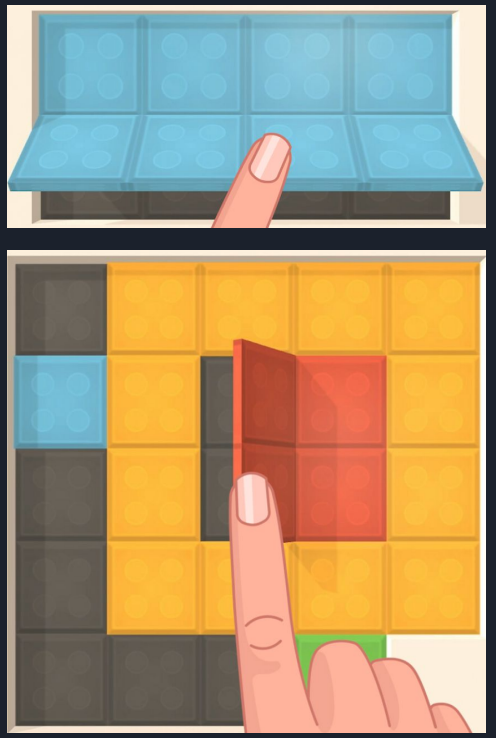
Especificação do Trabalho a Efetuar

O jogo Folding Blocks é um jogo do tipo solitário.

Cada nível do jogo é representado por um tabuleiro que, inicialmente tem algumas células coloridas (de uma ou mais cores) e outras cinzentas. Para a especificação do trabalho, convém considerar grupos de células, em que cada grupo corresponde ao conjunto de todas as células da mesma cor.

O jogo permite movimentos que correspondem a duplicar um grupo de células coloridas, aplicando-lhes uma simetria no tabuleiro, sendo que todas as células duplicadas têm que ficar em locais anteriormente ocupados por células cinzentas.

O objetivo de cada nível consiste em preencher o tabuleiro com células coloridas.



Pesquisa Efetuada

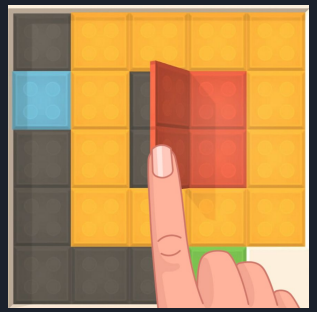
- Encontrámos uma solução/implementação possível para o problema, em python. LINK
- Encontrámos algumas tips que podem ajudar a encontrar o melhor método de resolução/a melhor heurística a ser usada. LINK LINK
    - Dividir o tabuleiro em secções e tentar resolver uma secção de cada vez
    - Começar por “duplicar” grupos de células maiores, visto que são, por norma, os mais difíceis de encaixar no tabuleiro.
    - Detetar a simetria nos níveis, adquirindo uma estratégia diferente consoante haja simetria (ou falta dela)

Formulação do Problema como um Problema de Pesquisa

Representação do Estado: matriz retangular de células M, em que:

- 0 -> célula cinzenta
- A-Z ->célula colorida, em que cada letra representa uma cor diferente
- _ ->célula “vazia” (buraco no tabuleiro)

Estado Inicial: matriz com células de vários tipos, problema com resolução
Teste Objetivo: matriz sem células cinzentas/sem símbolos 0.
Exemplo:



```
M=[[0, A, A, A, A],
   [B, A, 0, C, A],
   [0, A, 0, C, A],
   [0, A, A, A, A],
   [0, 0, 0, D, _]]
```

Estado Inicial

```
M=[[B, A, A, A, A],
   [B, A, C, C, A],
   [B, A, C, C, A],
   [B, A, A, A, A],
   [D, D, D, D, _]]
```

Teste Objetivo

Operadores

Operador	Pré-condição	Efeito	Custo
B_k (baixo)	Para cada $M_{ij}=k$, a posição $M_{(2*imax-i+1)j}=0$	Para cada $M_{ij}=k$, a posição $M_{(2*imax-i+1)j}=k$	1
C_k (cima)	Para cada $M_{ij}=k$, a posição $M_{(2*imin-i-1)j}=0$	Para cada $M_{ij}=k$, a posição $M_{(2*imin-i-1)j}=k$	1
E_k (esquerda)	Para cada $M_{ij}=k$, a posição $M_{i(2*jmin-j-1)}=0$	Para cada $M_{ij}=k$, a posição $M_{i(2*jmin-j-1)}=k$	1
D_k (direita)	Para cada $M_{ij}=k$, a posição $M_{j(2*jmax-i+1)}=0$	Para cada $M_{ij}=k$, a posição $M_{i(2*jmax-j+1)}=k$	1

Em que:

- k corresponde ao símbolo que representa um determinado grupo
- i corresponde à coordenada no eixo do y (linha)
- j corresponde à coordenada no eixo do x (coluna).
- imax corresponde ao valor máximo de i para as células do grupo k.
- imin corresponde ao valor mínimo de i para as células do grupo k.
- jmax corresponde ao valor máximo de j para as células do grupo k.
- jmin corresponde ao valor mínimo de j para as células do grupo k.

Heurísticas Utilizadas

Pesquisa Gulosa: a heurística utilizada procura o grupo de células vazias mais pequeno. Depois analisa esse mesmo grupo novamente, nas jogadas possíveis a partir desse estado, dando prioridade às jogadas que diminuem o tamanho desse grupo. Depois repete o processo.

Pesquisa A*: o value é igual ao número de casas por preencher somado à depth (número de jogadas executadas), e procura a move que leva ao value menor. Verifica também se já analisou esse estado/board, e em caso positivo atualiza o value e depth guardados (caso esse novo tenha um value inferior).

Algoritmos Implementados

Pesquisa Cega:

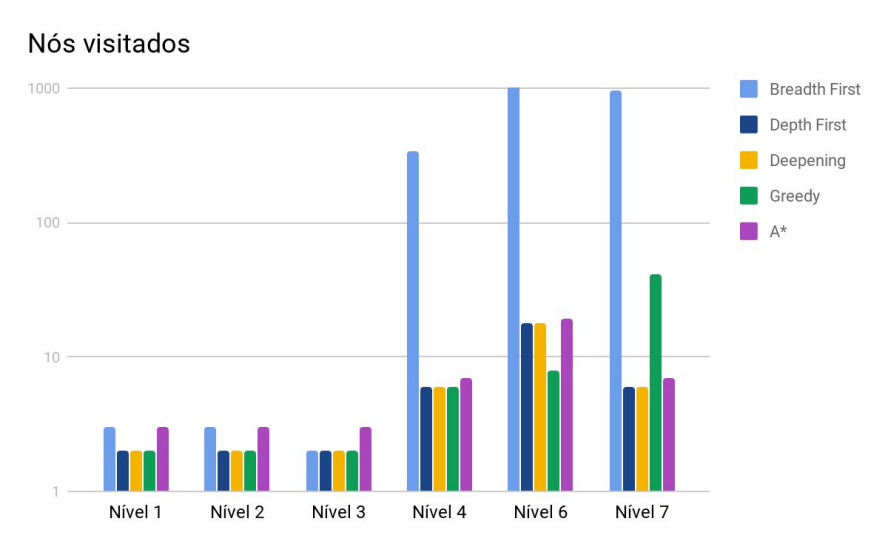
- Pesquisa em Largura (Breadth First)
- Pesquisa em Profundidade (Depth First)
- Pesquisa em Profundidade Iterativa (Iterative Deepening)

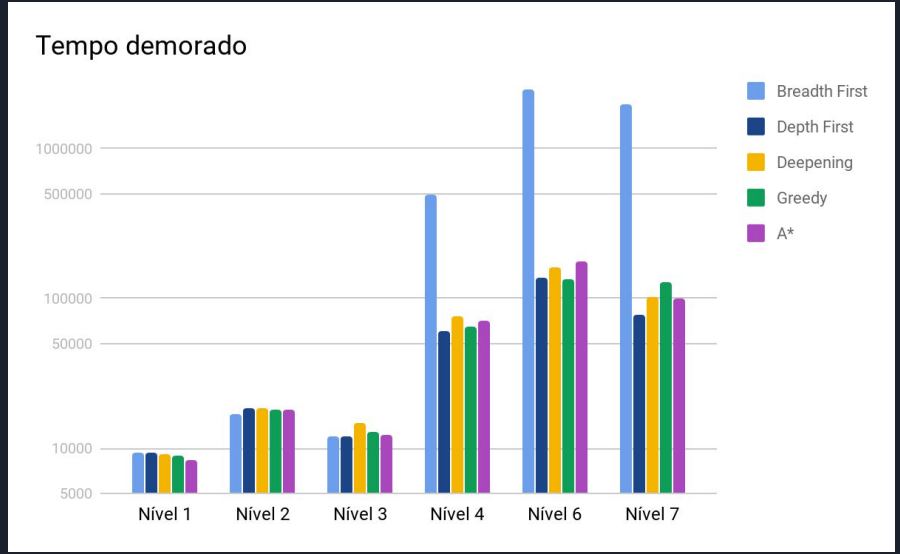
Pesquisa Informada:

- Pesquisa Gulosa (Greedy)
- Pesquisa A* (Graph) [1]

[1] A pesquisa A* Graph difere da A* (simples) num simples ponto: enquanto que na A* (simples) pode-se explorar o mesmo nó mais do que uma vez, tal não acontece na A* Graph.

Resultados





Notas:

- O nível 5 não é apresentado, visto que não tem solução possível;
- No nível 6, o valor de nós visitados (15417) não é visível no gráfico.

Resultados - Solução Encontrada é a Melhor?

Nível 1	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim
Nível 2	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim
Nível 3	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim

Nível 4	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim
Nível 6	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim
Nível 7	Breadth First	Sim
	Depth First	Sim
	Iterative Deepening	Sim
	Greedy	Sim
	A*	Sim

Conclusões

Com a realização deste projeto e análise dos respetivos resultados, é possível concluir que uma pesquisa informada apresenta grandes vantagens perante a pesquisa cega, nomeadamente no uso de recursos computacionais (o número de nós visitado é bastante menor).

De entre a pesquisa informada, a pesquisa A*, normalmente, revela uma melhor eficácia quando comparada com a pesquisa greedy, devendo ser portanto a adotada. Quando se usa uma boa heurística, a pesquisa A* apresenta uma maior possibilidade de se encontrar a melhor solução do que a pesquisa gulosa. Nos resultados obtidos, a maior eficiência da pesquisa A* não se verifica, visto que a heurística utilizada para a pesquisa A* tem uma menor qualidade do que a utilizada na pesquisa gulosa e, provavelmente, devido aos níveis utilizados nos testes.

De acordo com o aprendido durante as aulas, apenas dois dos métodos garantem que a solução encontrada é a melhor: pesquisa em largura e “iterative deepening”. Esta situação não se verifica nos resultados devido, possivelmente, à conjugação entre as heurísticas utilizadas e os níveis testados. Quando se pretende que esta garantia seja assegurada, a melhor opção será utilizar a pesquisa “iterative deepening” que é o método que, em geral, encontra a melhor solução explorando um menor número de nós e em menor tempo.

Referências

- IA_Lecture4_SolvingSearch - Slides fornecidos pelo Prof. Luís Paulo Reis, na página do moodle da unidade curricular;
- Google Play Store, Folding Blocks, https://play.google.com/store/apps/details?id=com.popcore.foldingblocks, consultado a 28-02-2020;
- Codewars, Kata - www.codewars.com/kata, consultado a 10-03-2020;
- Level Winner - www.levelwinner.com, consultado a 10-03-2020;
- GameZebo - www.gamezebo.com, consultado a 10-03-2020;
- Geeks for Geeks - www.geeksforgeeks.org, consultado entre 15-03-2020 e 31-03-2020;
- Stack Overflow - www.stackoverflow.com, consultado entre 15-03-2020 e 31-03-2020;
- CPlusPlus - www.cplusplus.com, consultado entre 15-03-2020 e 31-03-2020;