INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Inteligência Artificial

LEIC-T

Variante do jogo Solitaire

Grupo 5

Número	Nome
77906	António Sarmento
83391	Andreia Valente

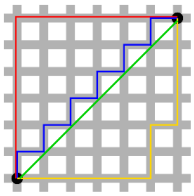
Enunciado do Problema

Considerando um tabuleiro LxC, em que cada posição tem um berlinde, um espaço vazio ou um espaço inválido, tem-se como objetivo encontrar a sequência de jogadas que deixa apenas um berlinde no tabuleiro, sendo que este não tem de ficar no centro do tabuleiro.

Os testes foram realizados sobre tabuleiros fornecidos no enunciado usando os algoritmos de procura DFS, *Greedy* e A*. Obtiveram-se número de nós expandidos e o número de nós gerados em cada algoritmo.

Análise Teórica e Heurística Escolhida

A heurística implementada para resolver o problema de procura informada baseia-se na métrica de distância Manhattan.



No seu conceito original supõe-se que se pode mover uma certa peça para qualquer uma das 4 direções adjacentes possíveis, sem constrangimentos. A distância Manhattan informa-nos que, para resolver o tabuleiro, o mínimo número de movimentos necessários para atingir o *goal state* é a soma da distância verticais e horizontais de cada peça à sua posição desejada.

A heurística implementada, com complexidade O(LxC), varia da implementação original da distância Manhattan na medida em que percorre uma vez o tabuleiro para procurar a primeira posição válida desocupada e soma, num segundo varrimento do tabuleiro, a distância Manhattan entre esse espaço em branco com todas as outras peças ainda no tabuleiro. Por fim, subtrai o custo mínimo encontrado na segunda procura de forma a obter a distância total sem a peça final. O varrimento do tabuleiro tem complexidade O(LxC).

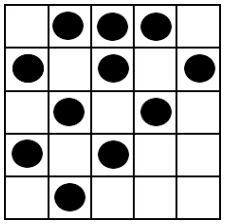
A variante da *Manhattan distance* usada como heurística não é admissível porque, na variante de Solitaire em estudo, cada peça move-se obrigatoriamente duas posições em cada movimento, portanto a heurística sobrestima o número de movimentos necessários até ao *goal state*. A heurística ser não admissível implica que seja também não consistente. A não admissibilidade não é relevante porque não existe uma solução ótima para o problema apresentado, uma vez que todas as soluções necessitam do mesmo número de movimentos para serem atingidas.

O algoritmo de procura DFS tem complexidade temporal teórica $O(L \times C \times r^m)$ enquanto *Greedy* e A* têm complexidade teórica $O(L \times C \times b^m)$, sendo que 'r' é o fator de expansão, 'm' é a profundidade e 'b' é o fator de ramificação.

Resolução de problemas e exemplos

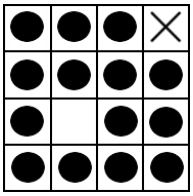
Foram efetuados testes sobre os quatro tabuleiros disponibilizados no enunciado. Cada resultado apresentado é a média de 5 testes:

Exemplo 1 (5x5) – 11 peças



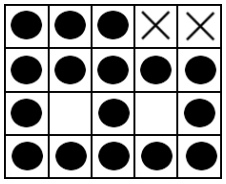
	Tempo de execução (s)	Nº de nós gerados	Nº de nós expandidos
DFS	0.00111	20	12
Greedy	0.00142	20	14
A*	0.00138	20	14

Exemplo 2 (4x4) – 14 peças



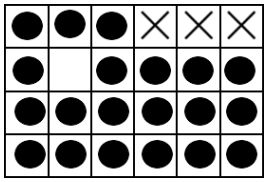
	Tempo de execução (s)	Nº de nós gerados	Nº de nós expandidos
DFS	0.28807	6002	5984
Greedy	0.28199	4309	4268
A*	0.27121	3884	3882

Exemplo 3 (4x5) – 16 peças



	Tempo de execução (s)	Nº de nós gerados	Nº de nós expandidos
DFS	2.84104	53 664	53 636
Greedy	0.31428	4 444	4 407
A*	0.13582	1 841	1 792

Exemplo 4 (4x6) – 20 peças



	Tempo de execução (s)	Nº de nós gerados	Nº de nós expandidos
DFS	1248.54645	14 760 576	14 760 524
Greedy	0.27070	2 281	2 202
A*	0.13584	878	744

Análise de resultados

Completude:

- Todos os métodos de pesquisa encontraram uma solução válida.
- Tanto a procura DFS como a procura Greedy demonstram ser completas, ou seja, não apresentam ciclos infinitos, uma vez que quanto mais o agente atua no ambiente, menos ações são permitidas (menos peças para serem retiradas do tabuleiro), eventualmente chegando a zero – o tabuleiro fica apenas com uma peça.
- A* é também uma pesquisa completa dado que o número de nós na fronteira é sempre finito e, portanto, o número de nós com $f \leq f(G)$ nunca irá ser infinito.

Eficiência:

- O fator que afeta primariamente a eficiência dos algoritmos, face à heurística implementada, depende das posições das peças no tabuleiro e, portanto, não é observável o impacto do número de peças e do tamanho do tabuleiro nos resultados.
- Complexidade espacial: a procura DFS é a procura que exige menos ramos, sendo a mais espacialmente eficiente. A procura Greedy e A* guardam em memória todos os nós da árvore, ocupando mais espaço.
- Complexidade temporal: ao aumentar o número de peças e o tamanho do tabuleiro, observa-se que a procura DFS aumenta exponencialmente o seu tempo de execução. A procura Greedy e A* têm a mesma função de maximização, sendo esse tempo reduzido pela heurística. No entanto, na procura Greedy as decisões de expansão não são revistas e, por procurar uma solução subótima, no pior caso, expande mais nós do que os nós necessários para atingir a solução ótima.

Teoricamente, a pesquisa A* deveria gerar e expandir menos nós que a pesquisa Greedy, o que se verifica nos resultados anteriores. Podemos concluir que a função de custo usada é a indicada.

Conclusão sobre a heurística implementada:

Pode ser verificado, como enunciado na introdução teórica, que a heurística não é admissível, dado que em todos os exemplos testados, o número de nós gerados é maior do que o número de nós expandidos menos um.