IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

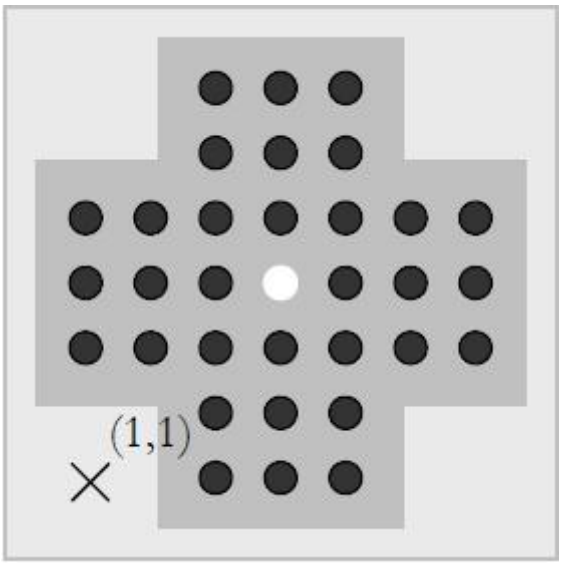
INTELIGÊNCIA ARTIFICIAL

Docentes

Prof. Joaquim Filipe

Eng. Cédric Grueau

Projecto 1 – MANUAL TÉCNICO



"Puzzle Solitário"

Número	Nome	Turma
052206031	Zulmira Ceita	3INF01
070221158	Pedro Santos	

Setúbal

2009/2010

ÍNDICE:

1 OBJECTIVOS

Neste projecto desenvolvido no âmbito da unidade curricular de Inteligência Artificial o objectivo da aplicação é desenvolver um programa, em LISP, utilizando métodos de procura em espaço de estados, para indicar a sequência de passos que conduzem de uma posição inicial do puzzle até a uma posição terminal, atingindo o objectivo, mediante as restrições inerentes ao jogo.

É também importante consolidar os conhecimentos teóricos acerca dos diferentes métodos de procura em espaço de estados de forma analisar quais as melhores decisões a tomar perante os diferentes problemas que possam surgir.

Para mais informações sobre as características do jogo é favor consultar o enunciado do projecto de forma a inteirar-se da forma como o jogo se processa e de como a aplicação o irá resolver.

2 CONTEÚDO DA APLICAÇÃO

A aplicação é distribuída na forma de um workspace do Eclipse 3.4.2 (Ganymede) a correr o CUSP 0.9.375 (com SBCL 1.0.20).

A aplicação que permite executar o "Puzzle Solitário" está contida no package Projecto1 que inclui os 3 ficheiros .lisp: "Projecto.lisp", "Procura.lisp", "Puzzle.lisp" as quais tratam da interacção com o utilizador, algoritmos de procura e resolução específica do problema, respectivamente.

Para além dos ficheiros inerentes ao ambiente do Eclipse foram desenvolvidos por nós os seguintes ficheiros:

- **Problemas.dat**

É o ficheiro que contém os tabuleiros possíveis, isto é, os problemas a resolver. É uma lista com as listas de cada tabuleiro.

- **solucoes.dat**

Neste ficheiro é onde fica registado as estatísticas da resolução do jogo segundo os diferentes algoritmos, o tabuleiro inicial e o final.

- **puzzle.lisp**

Este ficheiro responsável pela gestão das regras do domínio da aplicação. É neste ficheiro onde encontram-se os movimentos possíveis para jogar(operadores), o objectivo, os sucessores e algumas funções de apoio de implementação a estas funções principais.

- **Projecto.lisp**

Neste ficheiro foram criadas as funções que interagem com o utilizador imprimindo as opções no ecrã e recebendo as opções escolhidas, lendo e escrevendo no ficheiro. Estas funções foram desenvolvidas de forma a servir de ponte entre o utilizador e a resolução do problema.

- **Procura.lisp**

Foi definida a estrutura do estado do tabuleiro, as heurísticas, a procura genérica e todo o desenvolvimento dos algoritmos. A heurística é um valor que é guardado qualquer que seja o algoritmo de procura a ser utilizado, no entanto só o A* e o IDA* fazem uso dela. Os restantes algoritmos têm utilizam uma heurística de valor 0 para não atrasar o processamento

Para mais informações sobre a forma como o projecto foi implementado é favor consultar o código fonte que se encontra comentado de forma a facilitar a leitura do mesmo por outras pessoas.

3 ALGORITMOS

Neste projecto desenvolvemos alguns algoritmos (bem conhecidos) tal como foi pedido no enunciado do projecto. Ao longo das próximas secções vamos descrever algumas das suas características, vantagens, desvantagens e alguns comentários tendo em conta os resultados que obtermos.

4 BREADTH-FIRST

A procura em largura (Breadth-first em Inglês) é um método de procura não-informada (ou desinformada) que expande e examina sistematicamente todos os nós de uma árvore, em procura de uma solução. Em outras palavras, podemos dizer que seu algoritmo realiza uma procura exaustiva numa árvore inteira, sem considerar o seu alvo de procura, até que ele o encontre. Este algoritmo não utiliza uma heurística.

Do ponto de vista do algoritmo, todos os nós filhos obtidos pela expansão de um nó são adicionados a uma fila (FIFO). Normalmente, nós que ainda não foram examinados por seus vizinhos são colocados num container (como por exemplo uma fila ou lista ligada) que é chamado de "aberto". Uma vez examinados, são colocados num container "fechado".

A procura em largura é completa apenas se a árvore pesquisada tem um número finito de ramos – o algoritmo irá encontrar o alvo da procura caso ele exista (ou seja, ele alcança todos os nós de uma árvore).
A procura em largura é ótima se o custo dos passos forem idênticos – o algoritmo encontrará a solução no nível mais baixo de uma árvore de procura, não necessariamente a melhor. No caso em que os passos possuem custos diferentes, a solução mais “rasa” não é necessariamente a melhor.

A procura em largura tem complexidade linear espacial do tamanho (arestas somadas a vértices) da árvore/grafo pesquisada(o), já que ela precisa armazenar todos os nós expandidos na memória.
Devido a esta complexidade este algoritmo gasta muito tempo para resolver problemas complexos. No nosso caso apenas o caso de estudo **a** é resolvido rapidamente e o **b** demora cerca de 10 minutos. Todos os outros casos de estudo não são resolvidos em tempo útil.

5 DEPTH-FIRST E DEPTH-LIMITED

A procura em profundidade (Depth-first em Inglês) é um algoritmo de procura em profundidade realiza uma procura não-informada que progride através da expansão do primeiro nó filho da árvore de procura, e se aprofunda cada vez mais, até que o alvo da procura seja encontrado ou até que ele se depare com um nó que não possui filhos (nó folha). Então a procura retrocede (backtrack) e começa no próximo nó. Numa implementação não-recursiva, todos os nós expandidos recentemente são adicionados a uma pilha, para realizar a exploração.

A complexidade espacial de um algoritmo de procura em profundidade é muito menor que a de um algoritmo de procura em largura. A complexidade temporal de ambos algoritmos são proporcionais ao número de vértices somados ao número de arestas dos grafos aos quais eles atravessam.
Quando ocorrem procuras em grafos muito grandes, que não podem ser armazenadas completamente na memória, a procura em profundidade não termina, em casos onde o comprimento de um caminho numa árvore de procura é infinito. O simples artifício de “lembrar quais nós já foram visitados“ não funciona, porque pode não haver memória suficiente. Isso pode ser resolvido estabelecendo-se um limite de aumento na profundidade da árvore.
Visto que este algoritmo expande os nós sempre do mesmo ramo a sua complexidade temporal/espacial não é muito elevada acabando por explorar um ramo à exaustão até que encontra uma soluçã. No nosso

caso encontra solução para todos os casos de estudo mas são soluções com imensos passos quando comparadas com as soluções óptimas.

Outra variante do Depth-first é o Depth-limited que impõe um limite "artificial" à profundidade, isto é, ao chegar a um nó com uma determinada profundidade mesmo que ainda seja possível continuar a explorar volta-se atrás continuando a explorar outro ramo até essa mesma profunidade. Desta forma garante-se que, caso o custo seja unitário, na pior das hipóteses encontra-se uma solução com o custo igual ao limite.

6 A* E IDA*

A procura A* é um tipo de procura informada, ao contrário dos algoritmos já apresentados, que utiliza heurísticas de forma a direccionar a procura de uma solução. Recorrendo a uma heurística e estimando o custo de cada nó até a solução (composto por g (custo até ao nó em questão) e h (custo restante até à solução) é possível ordenar os nós a serem expandidos e assim explorar os que são mais prováveis de levar a uma solução. Desta forma reduz-se o número de nós a expandir tornando a pesquisa mais rápida.

O IDA* pega no mesmo conceito que o A* (utilização de heurísticas para conduzir mais rapidamente à solução) mas tenta reduzir a quantidade de memória utilizada através do uso de limiares de custo. Isto é só mantém em memória, e expandem-se, nós que tenham um valor inferior ao limite estimado no nó inicial. Caso se chegue a um caso em que não se encontram nós que satisfaçam esta condição reinicia-se a procura escolhendo como limite o valor mais baixo por entre os nós descartados. Repete-se este procedimento até encontrar a solução. Desta forma o número de nós mantidos em memória é reduzido substancialmente, no entanto, caso sejam necessários muitos reinicios do algoritmo pode torná-lo mais lento que o A*.

Estes algoritmos encontram a solução óptima caso a heurísitca utilizada seja admíssivel.

No nosso caso, dependendo da heurísitica, foi possível resolver todos os problemas. Sendo que a heurísitica 2 é capaz de resolver todos os casos de estudo com soluções sub-óptimas.

7 HEURÍSTICAS

As heurísiticas acrescentam conhecimento sobre o domínio da aplicação aos algoritmos de procura. São regras empíricas sobre o domínio de aplicação que podem acelerar a procura. No entanto, é de notar que são específicas de cada domínio de aplicação e dependendo da qualidade das mesmas podem, ou não, levar a bons resultados.

As heurísticas são admissíveis quanto o valor que estimam como custo até à solução corresponde exactamente, ou é inferior, ao valor real desse custo. No caso de algortimos informados estes conduzem à solução óptima caso a heurísitica utilizada seja admíssivel. Caso contrário a solução encontrada é subóptima.

8 HEURÍSTICA 1 – (NÚMERO DE PEÇAS NO TABULEIRO -1)

A heurística 1 foi a que foi dada no enunciado do projecto e foi primeiramente implementada. Trata-se de uma heurística bastante simples e eficaz, em determinados casos. É também uma heurística admissível pois o número de passos necessários para terminar um jogo é sempre, na melhor das hipóteses, igual ao número de peças no tabuleiro se for possível apenas realizar capturas. Caso sejam necessários movimentos o número de passos necessários vai ser maior mas a heurística mantém admissível pois estima um número menor de passos.

O lado negativo desta heurística é nos casos em que as peças se encontram dispostas de uma forma em que são necessários alguns movimentos simples para além das capturas para terminar o jogo. Nestes casos esta heurística não fornece muita informação de como se proceder para terminar o problema da forma mais eficiente possível levando a uma procura pouco informada (muitos nós expandidos de forma errada) para esses casos de estudo e tornando-a inexequível em tempo útil.

9 HEURÍSTICA 2 – (SOMA DAS DISTÂNCIAS DAS PEÇAS AO CENTRO DO TABULEIRO) + (NÚMERO DE PEÇAS NO TABULEIRO -1)

Numa tentativa de solucionar os casos de estudo que ofereciam um maior desafio à heurística 1 (c, d e f) decidiu-se tentar desenvolver uma nova heurística. Visto que o maior problema da heurística dada é que não conduz a que as peças tenham tendência a juntar-se, de forma a que seja possível realizar capturas e assim reduzir o número de peças no tabuleiro até chegar à solução.

Decidimos tentar melhorar a heurística dada reaproveitando-a ao adicionar a soma da distância das peças ao centro do tabuleiro. Desta forma as peças têm tendência a encontrar-se no centro do tabuleiro promovendo as capturas. Além disso continua-se a "premiar" cada vez que uma peça é capturada ao manter a (soma das peças no tabuleiro – 1) como parte da equação.

No entanto esta heurística não é admissível pois o custo calculado até à solução fica em quase todos os casos acima do valor real. Mesmo assim esta heurística permitiu que os tabuleiros c, d, e f tenham sido resolvidos com o A* e IDA* em tempo útil. No entanto, devido à falta de admissibilidade da heurística as solução apresentadas são sub-óptimas.

10 COMPARAÇÃO DOS TABULEIROS OBTIDOS ATRAVÉS DOS ALGORITMOS A* E IDA* MEDIANTE A UTILIZAÇÃO DAS HEURÍSTICAS H1 E H2

Algoritmo A*

- **Caso de estudo a)**

Algoritmo: A-STAR Heurística: HEURISTICA1 Limite/Limiar: NIL Movimentos: ((CC 4 2) (CB 4 5)) Penetrância: 0.071428575 Nós Gerados: 28 Nós Expandidos: 3 Profundidade da Solução: 2 Factor de ramificação médio: 14.0	Algoritmo: A-STAR Heurística: HEURISTICA2 Limite/Limiar: NIL Movimentos: ((CC 4 2) (CB 4 5)) Penetrância: 0.0952381 Nós Gerados: 21 Nós Expandidos: 2 Profundidade da Solução: 2 Factor de ramificação médio: 10.5

Heurísticas h1 e h2 não geram o mesmo resultado, visto que a h2 não é admissível, isto é, a solução encontrada não é óptima e por isso o número de nós gerados e expendidos é inferior em relação a h1.

- **Caso de estudo b)**

Algoritmo: A-STAR Heuristica: HEURISTICA1 Limite/Limiar: NIL Movimentos: ((CD 4 4) (CC 4 2) (CD 3 4) (CE 6 4) (CC 4 4)) Penetrância: 0.031446543 Nós Gerados: 159 Nós Expandidos: 11 Profundidade da Solução: 5 Factor de ramificação médio: 31.8	Algoritmo: A-STAR Heuristica: HEURISTICA2 Limite/Limiar: NIL Movimentos: ((CB 4 3) (C 4 1) (CB 4 5) (CC 4 2) (CD 4 4) (D 3 4) (E 6 4) (CE 5 4)) Penetrância: 0.04278075 Nós Gerados: 187 Nós Expandidos: 15 Profundidade da Solução: 8 Factor de ramificação médio: 23.375

A parte a profundidade, a h2 tem valores superiores e tudo isso deve-se ao facto dela não ser admissível.

- **Caso de estudo e)**

Algoritmo: A-STAR Heurística: HEURISTICA1 Limite/Limiar: NIL Movimentos: ((CC 2 3) (CC 3 4) (CD 3 6) (CE 4 3) (CD 1 3) (CC 4 4) (CB 5 4) (CB 5 6) (CE 6 4) (CE 7 3) (CC 5 2) (CE 5 4) (CC 3 3) (CD 2 5) (CC 4 5)) Penetrância: 0.012701101 Nós Gerados: 1181 Nós Expandidos: 54 Profundidade da Solução: 15 Factor de ramificação médio: 78.73333	Algoritmo: A-STAR Heurística: HEURISTICA2 Limite/Limiar: NIL Movimentos: ((CC 6 3) (CC 2 3) (CD 5 5) (CC 5 3) (CE 3 5) (CE 5 5) (CD 3 3) (CB 3 5) (C 1 3) (CB 1 5) (C 1 3) (D 1 4) (D 2 4) (CC 3 3) (B 3 5) (CD 3 4) (CC 5 3) (E 7 5) (CE 6 5) (CB 4 6) (E 7 3) (C 6 3) (E 6 4) (CE 5 4)) Penetrância: 0.036036037 Nós Gerados: 666 Nós Expandidos: 33 Profundidade da Solução: 24 Factor de ramificação médio: 27.75

O mesmo verifica-se para o caso de estudo e).

Para os casos de estudo c), d), f) e g) a heurística 1 não resolve porque não fornece informação suficiente para resolver o problema em tempo útil.

Algoritmo IDA*

- **Caso de estudo a)**

Algoritmo: IDA-STAR
Heurística: HEURISTICA1
Limite/Limiar: 2
Movimentos: ((CC 4 2) (CB 4 5))
Penetrância: 0.4
Nós Gerados: 5
Nós Expandidos: 3
Profundidade da Solução: 2
Factor de ramificação médio: 2.5

Algoritmo: IDA-STAR
Heuristica: HEURISTICA2
Limite/Limiar: 3
Movimentos: ((CC 4 2) (CB 4 5))
Penetância: 0.5
Nós Gerados: 4
Nós Expandidos: 3
Profundidade da Solução: 2
Factor de ramificação médio: 2.0

Neste caso não há muita diferença porque é um problema relativamente simples de encontrar a solução.

- **Caso de estudo b)**

Algoritmo: IDA-STAR
Heurística: HEURISTICA1
Limite/Limiar: 5
Movimentos: ((CD 4 4) (CC 4 2) (CD 3 4) (CE 6 4) (CB 4 5))
Penetrância: 0.2
Nós Gerados: 25
Nós Expandidos: 18
Profundidade da Solução: 5
Factor de ramificação médio: 5.0

Algoritmo: IDA-STAR
Heuristica: HEURISTICA2
Limite/Limiar: 11
Movimentos: ((CB 4 3) (C 4 1) (CB 4 5) (CC 4 2) (CE 4 4) (E 5 4) (D 2 4) (CD 3 4))
Penetância: 0.10958904
Nós Gerados: 73
Nós Expandidos: 13
Profundidade da Solução: 8
Factor de ramificação médio: 9.125

No caso de estudo b) é bastante notável que a h1 é muito eficiente, fazendo uma análise quantitativa dos dados.

- **Caso de estudo e)**

Algoritmo: IDA-STAR
Heurística: HEURISTICA1
Limite/Limiar: 15
Movimentos: ((CC 6 3) (CD 5 5) (CD 4 3) (CE 7 3) (CC 5 3) (CD 4 5) (CE 7 5) (CD 3 4) (CC 5 4) (CE 5 6) (CB 3 6) (CB 3 4) (CD 1 3) (CC 3 2) (CE 3 4))
Penetrância: 0.001545595
Nós Gerados: 9705
Nós Expandidos: 9572
Profundidade da Solução: 15
Factor de ramificação médio: 647.0

Algoritmo: IDA-STAR
Heurística: HEURISTICA2
Limite/Limiar: 45
Movimentos: ((CC 2 3) (CC 6 3) (CE 3 5) (CC 3 3) (CE 5 3) (CB 5 5) (CD 3 5) (CE 6 5) (C 1 3) (CB 1 5) (E 7 3) (CE 6 3) (CD 3 3) (E 5 3) (CC 4 5) (B 4 7) (B 4 6) (D 1 3) (D 2 3) (CD 3 3) (CB 4 5) (CE 5 3))
Penetância: 0.040515654
Nós Gerados: 543
Nós Expandidos: 30
Profundidade da Solução: 22
Factor de ramificação médio: 24.681818

- **Caso de estudo g)**

Algoritmo: IDA-STAR
Heurística: HEURISTICA1
Limite/Limiar: 31
Movimentos: ((CD 2 4)(CE 5 4)(CE 7 4) (CB 4 6)(CE 6 5)(CB 5 7) (CC 5 4)(CC 5 2)(CE 7 3) (CB 5 4) (CC 5 1) (CE 4 4)(CD 4 3) (CC 4 1)(CD 3 7) (CB 5 7) (CD 4 5) (CE 7 5) (CD 3 3) (CE 6 3) (CC 3 1) (CD 2 5) (CE 5 5) (CB 3 6) (CB 3 4) (CD 1 3) (CE 4 3) (CB 1 5) (CD 1 3) (CC 3 2) (CE 3 4))
Penetrância: 0.0066724066
Nós Gerados: 4646
Nós Expandidos: 4417
Profundidade da Solução: 31
Factor de ramificação médio: 149.87097

Algoritmo: IDA-STAR
Heurística: HEURISTICA2
Limite/Limiar: 116
Movimentos: ((CD 2 4) (CC 3 2) (CD 1 3) (CB 1 5) (CE 5 2) (CE 3 5) (CB 3 7) (CE 5 7) (CE 5 6) (C 1 3) (CB 1 5) (CB 5 4) (CC 5 1) (CE 7 4) (CD 3 1) (CB 3 3) (CB 3 5) (CB 3 7) (B 7 5) (CC 7 3) (CD 5 3) (CB 5 5) (CE 7 5) (E 7 3) (CD 5 3) (CD 3 3) (E 7 3) (CE 6 3) (B 5 5) (CD 3 5) (CB 5 5) (CE 5 3) (E 5 1) (CD 3 1) (E 5 1) (C 4 1) (C 4 2) (CD 3 3) (C 1 3) (E 5 3) (CC 4 3) (B 4 5) (D 1 4) (D 2 4) (CD 3 4))
Penetrância: 0.038330495
Nós Gerados: 1174
Nós Expandidos: 62
Profundidade da Solução: 45
Factor de ramificação médio: 26.088888

Neste caso, a diferença do número de nós expandidos entre as duas heurísticas é bastante significante, porque a h2 encontra muito mais rápido a solução apesar de não ser óptima.

Nota: Para todos os outros casos que não foram apresentados, a heurística 1 não resolve, sendo que a heurística 2 resolve não garantindo a solução óptima.

Apesar de não ser admissível a heurística 2, é a que resolve os casos abaixo indicados:

- **Caso de estudo c)**

Algoritmo: IDA-STAR
Heuristica: HEURISTICA2
Limite/Limiar: 55
Movimentos: ((D 1 3) (CB 1 5) (CD 1 3) (C 3 1) (CC 3 2) (CE 5 1) (E 7 5) (CC 7 3) (CE 7 5) (B 5 7) (CB 5 6) (CD 3 7) (C 3 1) (E 5 7) (E 5 4) (B 4 7) (B 4 6) (CB 4 5) (C 4 3) (CD 3 4) (E 5 4) (D 3 2) (C 4 2) (CC 4 3))
Penetância: 0.05491991
Nós Gerados: 437
Nós Expandidos: 38
Profundidade da Solução: 24
Factor de ramificacão médio: 18.208334

- **Caso de estudo d)**

Algoritmo: IDA-STAR
Heuristica: HEURISTICA2
Limite/Limiar: 79
Movimentos: ((C 1 3) (CB 1 5) (CE 3 5) (CE 5 5) (CE 7 5) (C 1 3) (CB 1 5) (C 1 3) (CE 3 3) (CC 1 3) (CE 5 3) (CE 7 3) (B 5 7) (CB 5 6) (CD 3 7) (CC 5 3) (E 5 7) (E 5 5) (CD 3 5) (E 5 5) (C 5 1)(CD 3 1) (CC 5 1) (E 5 3) (CD 3 3) (E 5 3) (B 4 7) (CB 4 6)(B 1 5) (CC 4 3) (B 4 5) (D 1 4) (D 2 4) (CD 3 4))
Penetância: 0.040621266
Nós Gerados: 837
Nós Expandidos: 48
Profundidade da Solução: 34
Factor de ramificação médio: 24.617647

- **Caso de estudo f)**

Algoritmo: IDA-STAR
Heuristica: HEURISTICA2
Limite/Limiar: 64
Movimentos: ((C 1 3) (CD 1 4) (E 7 5) (CE 6 5) (CD 3 5) (CC 3 3) (E 7 3) (CE 6 3) (E 6 4) (CB 5 5) (CE 5 3) (E 5 7) (CB 4 7) (CD 3 5)(E 5 5) (E 5 1) (CC 4 1) (CD 3 3) (B 1 5) (E 5 3) (D 3 7) (B 4 7) (CC 4 5) (B 4 7) (B 4 6) (D 3 1) (C 4 1) (CB 4 3) (CB 4 5) (C 4 3) (C 4 1) (C 4 2) (CC 4 3) (B 4 5) (D 1 4) (D 2 4) (CD 3 4))
Penetância: 0.029042386
Nós Gerados: 1274
Nós Expandidos: 62
Profundidade da Solução: 37
Factor de ramificação médio: 34.432434

11 QUADRO COMPARATIVO

No seguinte quadro comparativo é possível ter uma ideia dos problemas resolvidos (e em que condições) por cada um dos diferentes algoritmos desenvolvidos.

Tabuleiros	Breadth-first	Depth-first	Depth-limited	A* (heurística1)	A* (heurística2)	IDA* (heurística1)	IDA* (heurística2)
a	Sim	Sim(3)	Sim(2)	Sim	Sim(3)	Sim	Sim(3)
b	Sim(1)	Sim(3)	Sim(2)	Sim	Sim(3)	Sim	Sim(3)
c	Não	Sim(3)	Sim(2)	Não	Sim(3)	Não	Sim(3)
d	Não	Sim(3)	Sim(2)	Não	Sim(3)	Não	Sim(3)
e	Não	Sim(3)	Sim(2)	Sim	Sim(3)	Sim	Sim(3)
f	Não	Sim(3)	Sim(2)	Não	Sim(3)	Não	Sim(3)
g	Não	Sim(3)	Sim(2)	Não	Sim(3)	Sim	Sim(3)

Legenda

(1) – Resolve mas demora mais de 1 minuto
(2) – Resolve em tempo útil dependendo do limite introduzido
(3) – Resolve com uma solução sub-óptima

Para consultar mais resultados para além dos que foram apresentados neste relatório é favor consultar o ficheiro "solucoes.dat" incluído no pacote de entrega. Nesse ficheiro poderá encontrar muitos mais exemplos da resolução de problemas feita pelo projecto que não foram incluídas directamente neste manual para o manter o mais legível possível.

12 CONCLUSÃO

Este projecto que desenvolvemos ao longo das últimas semanas foi bastante enriquecedor no que toca aos assuntos abordados e são sem dúvida uma mais valia na futura vida académica e profissional.

Ainda ao analisarmos os resultados do projecto para cada caso de estudo concluímos, como esperado, que os algoritmos não informados, Depth-First e Breadth-First, são os que geram maior número de nós, ocupando mais memória que os restantes algoritmos. Ainda assim, o algoritmo Depth-First é capaz de devolver uma solução para todos os tabuleiros, enquanto que o Breadth-First, devido à sua grande expansão, esgota a memória disponível não sendo possível chegar a uma solução a partir do tabuleiro "c". Consequentemente, os algoritmos A* e IDA* são os que devolvem a melhor solução do problema, resolvendo o maior número de tabuleiros e em menor tempo.

Comparativamente ao IDA*, o algoritmo A* tem por vezes um melhor desempenho, uma vez que não é dotado de um limite de expansão para um dado nó, logo a solução de devolvida pelo IDA* acaba por passar por muitas mais iterações.

Todas estas conclusões estão já patentes no resto do relatório e no ficheiro de soluções enviado em anexo.

13 BIBLIOGRAFIA

- http://en.wikipedia.org/wiki/Breadth-first
- http://en.wikipedia.org/wiki/Depth-first_search
- http://en.wikipedia.org/wiki/Depth-limited_search
- http://en.wikipedia.org/wiki/A*_search_algorithm
- http://en.wikipedia.org/wiki/IDA*
- http://pt.wikipedia.org/wiki/Busca_em_largura
- http://pt.wikipedia.org/wiki/Busca_em_profundidade
- http://pt.wikipedia.org/wiki/Algoritmo_A*