Trabalho 1 Inteligência Artificial

Problema 1 (g1)

Espaço de Estados e operadores de transição

Para a representação do problema apresentado foi escolhido o estado (linha, coluna), onde é representado a posição do agente A. No caso apresentado a representação do estado inicial é (7,2), e a saída seria (1,5), para os diferentes cálculos foi antes utilizada a saída (3,4) devido a alguns problemas de memoria (global stack overflow).

```
%(Linha, Coluna)

%estado inicial
estado_inicial((7,2)).

%estado final
estado_final((3,4)).

%op(Estado_act,operador,Estado_seg,Custo)
op((L,C),esquerda ,(Ls,Cs),1):-
                        free(L, C),
                        Ls is L + 0, Cs is C - 1, lim(Ls, Cs).
op((L,C),direita ,(Ls,Cs),1):-
                        free(L, C),
                        Ls is L + 0, Cs is C + 1, lim(Ls, Cs).
op((L,C),cima ,(Ls,Cs),1):-
                        free(L, C),
                        Ls is L - 1, Cs is C + 0, lim(Ls, Cs).
op((L,C),baixo ,(Ls,Cs),1):-
                        free(L, C),
                        Ls is L + 1, Cs is C + 0, lim(Ls, Cs).

% Limite do tabuleiro
lim(L, C) :- L>0, C>0, L<8, C<8.

% Casa com X
free(L, C) :- member( (L, C) , [ (1,3), (2,1), (2,3), (2,7), (4,4), (5,4), (6,4) ] ),!, fail.
free(L,C).
```

Análise dos Algoritmos

Os algoritmos utilizados tanto na pesquisa informada como não informada foram retirados dos algoritmos disponibilizados durante as aulas.

Pesquisa não informada

Para executar este tipo de pesquisas basta utilizar o ficheiro “pni.pl” com o comando “pesquisa(<problema>, <algoritmo>)”, sendo os algoritmos possíveis: “it” (profundidade iterativa), “largura”, “profundidade”.

Durante a análise dos diferentes algoritmos foi apenas possível retirar conclusões relativamente ao algoritmo de largura e profundidade iterativa, devido a problemas relacionados com global stack overflow, não foi possível retirar qualquer conclusão relativamente á pesquisa em profundidade.

Algoritmo	Estados Visitados	Estados em Memória	Profundidade	Custo
Largura	813	1799	6	6
Profundidade Iterativa	45103	13	6	6

Comparando a pesquisa em largura com profundidade iterativa podemos concluir que ambas encontram a solução á mesma profundidade e custo. No entanto embora a pesquisa em largura tenha menos estados visitados, tem mais estados em memória.

Pesquisa Informada

Para executar este tipo de pesquisas basta utilizar o ficheiro “pi.pl” com o comando “pesquisa(<problema>, <algoritmo>)”, sendo os algoritmos possíveis: “a” (A*) ou “g” (Gready). É também possível alterar o tipo de heurística no final do ficheiro relativo ao problema, bastando alterar a heurística (“h”) comentada.

Heurísticas

Para representar duas heurísticas admissíveis para o problema em questão foi utilizada a distância de Manhattan e a distância euclidiana.

```
%heuristica para estimar distancia h(Estado,Valor)

h(A, B):- h1(A, B).

%h(A, B):- h2(A, B).


% manhattan

h1((X,Y),N):- estado_final((W,Z)),
              modulo_dif(X,W,M),
              modulo_dif(Y,Z,O),
              N is M+O.

modulo_dif(A,B,C):- A>B, C is A-B.
modulo_dif(A,B,C):- C is B-A.


% euclidiana

h2((X,Y),N):- estado_final((W,Z)),
              modulo_dif(X,W,M),
              modulo_dif(Y,Z,O),
              N is round(sqrt(M ** 2 + O ** 2)).
```

Algoritmo	Estados Visitados	Estados em Memória	Profundidade	Custo
A*, euclidiana	408	324	6	6
A*, Manhattan	210	236	6	6
Gready, euclidiana	10	27	6	6
Gready, Manhattan	145	215	8	8

Analisando todos os algoritmos, concluímos que o melhor algoritmo é o gready com a heurística calculada pela distância euclidiana.

Problema 2 (g2)

Espaço de Estados e operadores de transição

Para a representação do problema apresentado foi escolhido o estado (linha jogador, coluna jogador, linha caixa, coluna caixa), onde é representado a posição do agente A. No caso apresentado a representação do estado inicial é ((7, 2), (6, 2)), e a saída seria ((_,_),(1,5)), sendo que a posição do jogador não importa desde que a caixa se encontre no ponto pretendido. para os diferentes cálculos foi antes utilizada a saída((_,_),(6,3)), devido a alguns problemas de memoria (global stack overflow).

```
11  %op(Estado_act,operador,Estado_seg,Custo)
12
13  op( ((Lj, Cj) , (Lc, Cc) ) , esquerda, ( (Ljs, Cjs), (Lcs, Ccs) ), 1):-
14                              free(Lj, Cj),
15                              Ljs is Lj, Cjs is Cj - 1, lim(Ljs, Cjs),
16                              move(Ljs, Cjs, Lc, Cc, Lcs, Ccs, 0, -1),
17                              lim(Lcs, Ccs),
18                              free(Lcs, Ccs).
19
20  op( ((Lj, Cj) , (Lc, Cc) ) , direita, ( (Ljs, Cjs), (Lcs, Ccs) ), 1):-
21                              free(Lj, Cj),
22                              Ljs is Lj, Cjs is Cj + 1, lim(Ljs, Cjs),
23                              move(Ljs, Cjs, Lc, Cc, Lcs, Ccs, 0, 1),
24                              lim(Lcs, Ccs),
25                              free(Lcs, Ccs).
26
27  op( ((Lj, Cj) , (Lc, Cc) ) , cima, ( (Ljs, Cjs), (Lcs, Ccs) ), 1):-
28                              free(Lj, Cj),
29                              Ljs is Lj - 1, Cjs is Cj, lim(Ljs, Cjs),
30                              move(Ljs, Cjs, Lc, Cc, Lcs, Ccs, -1, 0),
31                              lim(Lcs, Ccs),
32                              free(Lcs, Ccs).
33
34  op( ((Lj, Cj) , (Lc, Cc) ) , baixo, ( (Ljs, Cjs), (Lcs, Ccs) ), 1):-
35                              free(Lj, Cj),
36                              Ljs is Lj + 1, Cjs is Cj, lim(Ljs, Cjs),
37                              move(Ljs, Cjs, Lc, Cc, Lcs, Ccs, 1, 0),
38                              lim(Lcs, Ccs),
39                              free(Lcs, Ccs).
40
41  move(L, C, L, C, Lcs, Ccs, X, Y):- Lcs is L + X, Ccs is C + Y, !.
42  move(L, C, Lc, Cc, Lcs, Ccs, X, Y):- Lcs is Lc,
43                                       Ccs is Cc.
44
```

Análise dos Algoritmos

Pesquisa não informada

Durante a análise dos diferentes algoritmos foi apenas possível retirar conclusões relativamente ao algoritmo de largura e profundidade iterativa, devido a problemas relacionados com global stack overflow, não foi possível retirar qualquer conclusão relativamente á pesquisa em profundidade.

Algoritmo	Estados Visitados	Estados em Memória	Profundidade	Custo
Largura	17	36	3	3
Profundidade Iterativa	185	6	3	3

Comparando a pesquisa em largura com profundidade iterativa podemos concluir que ambas encontram a solução á mesma profundidade e custo. No entanto embora a pesquisa em largura tenha menos estados visitados, tem mais estados em memória.

Pesquisa Informada

Heurísticas

Para representar duas heurísticas admissíveis para o problema em questão foi utilizada a distância de Manhattan e a distância euclidiana.

```
%heuristica para estimar distancia h(Estado,Valor)

h(A, B):- h1(A, B).

%h(A, B):- h2(A, B).


% manhattan entre a caixa e a saida

h1( ((_,_), (X,Y)) ,N):- estado_final( ( ( _,_) , (W,Z) ) ),
                         modulo_dif(X,W,M),
                         modulo_dif(Y,Z,0),
                         N is M+0.

modulo_dif(A,B,C):- A>B, C is A-B.
modulo_dif(A,B,C):- C is B-A.


% euclidiana entre a caixa e a saida

h2(((_,_), (X,Y)) ,N):- estado_final( ( ( _,_) , (W,Z) )),
                         modulo_dif(X,W,M),
                         modulo_dif(Y,Z,0),
                         N is round(sqrt(M ** 2 + 0 ** 2)).


```

Algoritmo	Estados Visitados	Estados em Memória	Profundidade	Custo
A*, euclidiana	11	15	3	3
A*, Manhattan	8	11	3	3
Gready, euclidiana	6	8	3	3
Gready, Manhattan	5	5	3	3

Analisando todos os algoritmos, concluímos que o melhor algoritmo é o gready com a heurística calculada pela distância de Manhattan.