Universidade De Évora
Departamento de Informática
Inteligência Artificial
Ano letivo 2019 - 2020

Inteligência Artificial: 2º Trabalho Prático

Alunos:
Luis Ressonha - 35003
Rúben Teimas - 39868

Docentes:
Paulo Quaresma

4 de Abril de 2020

1 Introdução

Neste 2º trabalho é nos pedido que, à semelhança do 1º, representemos o problema sendo que este consiste num agente que se encontra num labirinto *4 X 4*, na posição *(1,1)*, e pretende sair desse mesmo labirinto, sendo uma das nossas tarefas encontrar esse caminho.

Ao contrário do 1º trabalho, o agente não tem qualquer conhecimento do espaço onde se encontra, *i.e*, não sabe que portas estão bloqueadas nem a a localização da saída.

Dadas estas restrições estamos então presente um problema de exploração, o que exige que utilizemos um algortimo de pesquisa local.

O algortimo de pequisa local por nós escolhido foi o algortimo *Hill Climbing* sem ciclos.

2 Representação do Problema em Prolog

A representação do problema é essencialmente semelhante à do 1º trabalho, sendo definido o estado inicial e final (desconhecido para o agente) do agente e as respetivas operações (as mesmas do 1º trabalho).

A ordem pela qual as operações foram definidas não foi aleatória, pois embora o agente desconheça o labirinto, nós não o que virá a influenciar o caminho gerado.

```
1  % Estados dados por um tuplo com as coordenadas da sala
2  estado_inicial((1, 1)).
3  estado_final((1, 4)).
4
5  % bloqueadas(sala1, sal2).
6  bloqueada((1,1), (1,2)).
7  bloqueada((1,2), (1,1)).
8  bloqueada((2,1), (2,2)).
9  bloqueada((2,2), (2,1)).
10 bloqueada((3,1), (4,1)).
11 bloqueada((4,1), (3,1)).
12 bloqueada((3,2), (3,3)).
13 bloqueada((3,3), (3,2)).
14 bloqueada((4,2), (4,3)).
15 bloqueada((4,3), (4,2)).
16
17 % op(sala_atual, direcao, sala_seguinte, custo).
18 op((X, Y), esquerda, (W, Y), 1):-
19     X > 1,
20     W is X-1,
21     \+ bloqueada((X, Y), (W, Y)).
22
23 op((X, Y), baixo, (X, Z), 1):-
24     Y < 4,
25     Z is Y+1,
26     \+ bloqueada((X, Y), (X,Z)).
27
28 op((X, Y), direita, (W, Y), 1):-
29     X < 4,
30     W is X+1,
31     \+ bloqueada((X, Y), (W, Y)).
32
33 op((X, Y), cima, (X, Z), 1):-
34     Y > 1,
35     Z is Y-1,
36     \+ bloqueada((X, Y), (X,Z)).
```

3 Respostas

3.1 Algoritmo de Pesquisa

O algoritmo de pesquisa utilizado foi o algoritmo *Hill Climbing* sem ciclos. O código deste algoritmo foi-nos facultado pelo docente, tendo nós feito uma pequena alteração para que pudessemos contar o número de nós visitados até chegar à saída.

Como a localização da saída é desconhecida pelo agente é impossivel calcular uma heuristica com base na saída, como tal optámos por definir a heuristica com um valor constante *(1)*.

```
1  % Predicado que servir  para contar os estados visitados e em
       mem ria
2  :- dynamic(visitados/1).
3
4  pesquisa_local_hill_climbingSemCiclos(E, _) :-
5      retract(visitados(X)),
6      X1 is X+1,
7      asserta(visitados(X1)),
8      estado_final(E),
9      write(E), write('␣').
10 pesquisa_local_hill_climbingSemCiclos(E, L) :-
11     write(E), write('␣'),
12     expande(E,LSeg),
13     sort(3, @=<, LSeg, LOrd),
14     obtem_no(LOrd, no(ES, Op, _)),
15     \+ member(ES, L),
16     write(Op), nl,
17     (pesquisa_local_hill_climbingSemCiclos(ES,[E|L]) ; write(
           undo(Op)), write('␣'), fail).
18
19 expande(E, L):-
20     findall(no(En,Opn, Heur),
21                 (op(E,Opn,En,_), heur(En, Heur)),
22                 L).
23 obtem_no([H|_], H).
24 obtem_no([_|T], H1) :-
25     obtem_no(T, H1).
26
27 pesquisa :-
28     asserta(visitados(0)),
29     estado_inicial(S0),
30     pesquisa_local_hill_climbingSemCiclos(S0, []),
31     nl,
32     write("Visitados:␣" ),
33     retract(visitados(Y)),
34     write(Y), nl.
35 % Heuristica admissivel
36 heur(_, 1).
```

Sendo a heuristica igual para todos os movimentos, o caminho seguido pelo agente irá depender da ordem pela qual definimos as operações. A ordem pela qual as definimos foi *esquerda->baixo->direita->cima* acreditando que resultará na solução ótima.

Utilizando este algortimo de pesquisa com a nossa representação do problema obtivémos o caminho:

(1,1) direita > (2,1) direita > (3,1) baixo > (3,2) esquerda > (2,2) esquerda > (1,2) baixo > (1,3) baixo > (1,4)

Tendo visitado um total de **8 salas**.

Caso tivessemos definido as operações na ordem (*cima -> direita -> baixo -> esquerda*), a ordem dos ponteiros do relógio, o caminho encontrado teria a certa altura um beco sem saída que teria sido resolvido pelo algoritmo voltando para trás até um "ponto de divergência"no qual o agente pudesse optar por outro caminho. Esta representação do problema faria com que o agente visitasse **14 salas**, sendo 2 delas, visitadas 2 vezes (caminho para trás).

4 Conclusão

Com este trabalho conseguimos obter um melhor conhecimento dos algoritmos de pesquisa local e em que situações devem ser utilizados.