UNIVERSIDADE DE ÉVORA

4º Trabalho de Inteligência Artificial

Yaroslav Kolodiy `139859` Eduardo Medeiros `139873`

Maio, Ano Letivo 2019/2020

Inteligência Artificial

Prof. Paulo Quaresma

Índice

1 Respostas

1.1 Nota Prévia

Nota prévia:

O programa *ex1.pl* deve ser compilado juntamente com o código de *queens.pl*.

Após compilado, o predicado pesquisa/1 deve ser chamado com o tamanho do tabuleiro de modo a iniciar-se a resolução do problema. Logo de seguida é mostrado o tabuleiro, i.e. uma lista em que cada índice representa uma coluna e o valor desse índice representa a linha onde está uma rainha. À frente do "tabuleiro"é mostrada a rainha escolhida, i.e. o índice da mesma, e a posição para onde esta foi movida.

1.2 Abordagem

Abordagem:

Dada a lista referida anteriormente é calculado o número de ataques que ocorre nesse "tabuleiro", i.e. é verificado se existem rainhas nas mesmas diagonais umas das outras, ou se existe mais do que uma rainha na mesma linha.

Depois de calculado o valor dos próximos estados, através da tentativa de mover todas as rainha de um modo aleatório para outra posição na respectiva coluna, o algoritmo decide avançar para um estado cujo o número de ataques seja menor que no estado atual. Deste modo o número de ataques no "tabuleiro"tenderá para 0, ou seja, para um estado final.

Se um estado é considerado final, isto significa que a número de ataques nele presente é 0. No entanto isto é verificado através de restrições e não usando a heuristica.

1.3 Tempo médio de N

Abaixo encontra-se a o tempo médio obtido através de 10 execuções do programa para cada N.

Caso se pretenda ver o tempo de cada execução, o predicado time/1 de ser chamado com o tamanho do tabuleiro.

4. 9.3 ms
5. 10.3 ms
6. 110.8 ms
7. 141.8 ms
8. 244.8 ms
9. 315.8 ms
10. 888.7 ms
11. 1138.0 ms
12. 1328.2 ms
13. 1373.2 ms
14. 1953.0 ms
15. 2678.3 ms
16. 2589.3 ms
17. 1860.7 ms
18. 4553.8 ms
19. 4651.3 ms
20. 7294.9 ms

2 Anexos

2.1 Anexo 1 - ex1.pl

```
1  :- dynamic(tamanho/1).
2  :- dynamic(estado_inicial/1).
3  :- use_module(library(clpfd)).
4
5  %estado inicial consoante o tamanho
6  criar_estado_inicial():-
7      tamanho(T),
8      length(N, T),
9      T1 is T +1,
10     maplist(random(1,T1), N),
11     asserta(estado_inicial(N)).
12
13 % Dado um estado E move a rainha X para a posi   o Y,
       formando En.
14 mover([_|[]],1,Y,[Y]).
15 mover([_|T],1,Y,[Y|T]).
16
17 mover([E|T],X,Y,[E|En]):-
18     X1 is X-1,
19     mover(T,X1,Y,En).
20
21 % realizar joagada
22 % op(estadio_inicial, [rainha_a_mover,
       posicao_para_onde_mover], estado_resultante,
       peso_da_op)
23 op(E,[X,Y],En,1) :-
24     tamanho(N),!,
25     [X,Y] ins 1..N, labeling([max(X),min(Y)],[X,Y]), %
           gera valores de X e y entre 1 e o tamanho. Nao
           meche no taboleiro
26     mover(E,X,Y,En).
27
28 % verificar se e estado final
29 estado_final(E) :-
30     tamanho(N),
31     queens(N,E).
32
```

```
33 print_elements([]).                    % print de um elemento
34 print_elements([Element|T]):-
35     write(Element),print_elements(T).
36
37 print_rows([]).                         % print da matrix
38 print_rows([H|T]) :- print_elements(H), nl, print_rows(
       T).
39
40 % retirado da net http://blog.ivank.net/prolog-matrices
       .html
41 % trans(+M1, -M2) - transpose of square matrix
42 % 1. I get first column from Tail and make a first row
       (NT) from it
43 % 2. I transpose "smaller matrix" Rest into NRest
44 % 3. I take T and make it to be a first column of NTail
45
46 trans([[H|T] |Tail], [[H|NT] |NTail]) :-
47     firstCol(Tail, NT, Rest), trans(Rest, NRest),
           firstCol(NTail, T, NRest).
48 trans([], []).
49
50 % firstCol(+Matrix, -Column, -Rest)  or  (-Matrix, +
       Column, +Rest)
51
52 firstCol([[H|T] |Tail], [H|Col], [T|Rows]) :- firstCol(
       Tail, Col, Rows).
53 firstCol([], [], []).
54
55 %cira tabuleiro para display
56 criar_tabuleiro([], []).
57 criar_tabuleiro([H|T], [Lista1|TL]) :-
58     tamanho(N),
59     N1 is N-1,
60     create_list_of_Zeros(N1,Lista),
61     nth1(H, Lista1, ' Q ',Lista),
62     criar_tabuleiro(T,TL).
63
64 create_list_of_Zeros(1,[' _ _ ']).
65 create_list_of_Zeros(X,[' _ _ '|T]) :-
66     X1 is X -1,
67     create_list_of_Zeros(X1,T).
```

```
68
69 time(A) :-
70         statistics(walltime, [TimeSinceStart | [
               TimeSinceLastCall]]),
71         pesquisa(A),
72         statistics(walltime, [NewTimeSinceStart | [
               ExecutionTime]]),
73         write('Execution took '), write(ExecutionTime),
               write(' ms.'), nl.
74
75 % algoritmo pesquisa
76 pesquisa_local_hill_climbingSemCiclos(E, _) :-
77          estado_final(E),
78          write(E), write(' '), nl, nl,
79      criar_tabuleiro(E, T),
80      trans(T, T1),
81      print_rows(T1).
82
83 pesquisa_local_hill_climbingSemCiclos(E, L) :-
84          write(E), write(' '),
85          expande(E,LSeg),
86          sort(3, @=<, LSeg, LOrd),
87          obtem_no(LOrd, no(ES, Op, _)),
88          \+ member(ES, L),
89          write(Op), nl,
90          (pesquisa_local_hill_climbingSemCiclos(ES,[E|L
               ]) ; write(undo(Op)), write(' '), fail).
91
92 expande(E, L):-
93          findall(no(En,Opn, Heur),
94                  (op(E,Opn,En,_), heur(En, Heur)),
95                  L).
96
97 obtem_no([H|_], H).
98 obtem_no([_|T], H1) :-
99          obtem_no(T, H1).
100
101 % cria um tabuleiro com tamnho N.
102 % cria um estado inicial para mesmo.
103 % efetua movimentos de modo a que as rainhas no estado
      inicial deixem de se atacar.
```

```
104 pesquisa(N) :-
105     asserta(tamanho(N)),
106         criar_estado_inicial(),
107         estado_inicial(E),!,
108         pesquisa_local_hill_climbingSemCiclos(E, []).
109
110 % Dado um estado devolve o n meor de ataques nesse
        estado.
111 heur([],0).
112 heur([Queen|Others],R) :-
113     heur(Others,R0),
114     sum_d1(Queen,Others,1,R1),
115     sum_d2(Queen,Others,1,R2),
116     sum_L(Queen,Others,R3),
117     R is R0+R1 +R2+R3.
118
119 % Diagonal Y=-X. Verifica se h  ataques na mesma e
        devolve a sua soma.
120 sum_d1(_,[],_,0).
121 sum_d1(Y,[Y1|Ylist],Xdist,S):-
122     Y2 is Y1-Y,
123     Y2 \= Xdist,
124     Dist1 is Xdist + 1,
125     sum_d1(Y,Ylist,Dist1,S).
126
127 sum_d1(Y,[Y1|Ylist],Xdist,S):-
128     Y2 is Y1-Y,
129     Y2 is Xdist,
130     Dist1 is Xdist + 1,
131     sum_d1(Y,Ylist,Dist1,S1),
132     S is S1 +1.
133
134 % Diagonal Y=X. Verifica se h  ataques na mesma e
        devolve a sua soma.
135 sum_d2(_,[],_,0).
136 sum_d2(Y,[Y1|Ylist],Xdist,S):-
137     Y2 is Y-Y1,
138     Y2 \= Xdist,
139     Dist1 is Xdist + 1,
140     sum_d2(Y,Ylist,Dist1,S).
141
```

```
142 sum_d2(Y,[Y1|Ylist],Xdist,S):-
143     Y2 is Y-Y1,
144     Y2 is Xdist,
145     Dist1 is Xdist + 1,
146     sum_d2(Y,Ylist,Dist1,S1),
147     S is S1 +1.
148
149 % Linha. Verifica se h  ataques na mesma e devolve a
        sua soma.
150 sum_L(_,[],0).
151 sum_L(Y,[Y1|Ylist],S):-
152     Y \= Y1,
153     sum_L(Y,Ylist,S).
154
155 sum_L(Y,[Y1|Ylist],S):-
156     Y is Y1,
157     sum_L(Y,Ylist,S1),
158     S is S1 +1.
```

2.2 Anexo 2 - queens.pl

```
1
2  :- use_module(library(clpfd)).
3
4  ok([]).
5  ok([R|Rs]) :- ok(Rs, R, 1), ok(Rs).
6
7  ok([], _, _).
8  ok([Rj|Rs], Ri, I) :-
9       I1 is I+1,
10      ok(Rs, Ri, I1),
11      Ri #\= Rj, Ri #\= Rj+I, Ri+I #\= Rj.
12
13 queens(N, R) :-
14      length(R, N),
15      R ins 1..N,
16      ok(R).
```