UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIAS

Jogo do três em linha

Cadeira de Inteligência Artificial

3º Trabalho 2019/2020

Prof. Paulo Quaresma

Trabalho realizado por:
Fernando Prates 34197
Miguel Luís 37555

Problema

Num espaço de estados de 5x4 em que cada posição pode estar vazia(‘ ‘), ocupada pelo jogador Azul(‘A’) ou Vermelho(‘V’), o objectivo é aplicar os algoritmos minimax com alfabeta para descubrir a próxima melhor jogada possível.

Predicados:

```
tabuleiro([_,_,_,_,_]).%5 colunas
estado_inicial([[' ',' ',' ','A'],[' ',' ',' ','V'],['A','V','A','A'],[' ',' ','A','V'],[' ','V','V','A']]).
i([[' ',' ',' ',' '],[' ',' ',' ',' '],[' ',' ',' ',' '],[' ',' ',' ',' '],[' ',' ',' ',' ']]).
```

Joga:

```
joga:-
    %estado_inicial(Ei),
    i(Ei),
    ciclo_jogada('A',Ei).

ciclo_jogada('A',Estado):-
    write(Estado),nl,
    minimax_decidir(Estado,Op1),
    write('minimax:'),write(Op1),nl,
    alfabeta(Estado,Op2),
    write('alfabeta:'),write(Op2),nl,
    write('Insira a Coluna que deseja jogar:'),
    read(X),
    jogada(Estado,'A',X,Novoestado),
    ciclo_jogada('V',Novoestado).

ciclo_jogada('V',Estado):-
    write('Vez do computador'),nl,
    alfabeta(Estado,Op),
    jogada(Estado,'V',Op,Novoestado),
    ciclo_jogada('A',Novoestado).
```

Chama o predicado ciclo_jogada que inicia assim um jogo com o computador, sendo este o Vermelho e o utilizador o Azul. Dá ao Jogador duas dicas da próxima jogada usando os algoritmos minimax e alfabeta. Chama minimax_decidir() e alfabeta().

alfabeta:

```
alfabeta(Ei,terminou):-terminal(Ei).

alfabeta(Ei,Opf) :-
    findall(Vc-Nr_coluna, (jogada(Ei,'A',Nr_coluna,Tabuleiro), alfabeta_min(Tabuleiro,Vc,'A',-10000,10000)), L),
    escolhe_max(L,Opf).
alfabeta_min(Ei,Val,_,_,_) :-
    terminal(Ei),
    valor(Ei,Val), !.
alfabeta_min(Ei,Val,P,Alfa,Beta) :-
    muda_jogador(P,J),
    V is 10000,
    findall(Tabuleiro, jogada(Ei,J,_,Tabuleiro), L),
    processa_lista_min(L, J, V, Alfa, Beta, Val), !.


processa_lista_min([], _, V, _, _, V).
processa_lista_min([H|T], P, V, A, B, V1) :-
    alfabeta_max(H, V2, P, -10000, 10000),
    min(V, V2, V3),
    (V3 < A, V1 is V3; min(B, V3, B1), processa_lista_min(T, P, V3, A, B1, V1)).


alfabeta_max(Ei,Val,_,_,_) :-
    terminal(Ei),
    valor(Ei,Val), !.

alfabeta_max(Ei,Val,P,Alfa,Beta) :-
    muda_jogador(P,J),
    V is -10000,
    findall(Tabuleiro, jogada(Ei,J,_,Tabuleiro), L),
    processa_lista_max(L, J, V, Alfa, Beta, Val), !.

processa_lista_max([], _, V, _, _, V).
processa_lista_max([H|T], P, V, A, B, V1) :-
    alfabeta_min(H, V2, P, -10000, 10000),
    max(V, V2, V3),
    (V3 >= B, V1 is V3; max(A, V3, A1), processa_lista_max(T, P, V3, A1, B, V1)).
```

Dando um estado, dá-nos a melhor jogada possível e reduz o numero de estados visitados do algoritmo minimax.
O algoritmo tem dois valores, máximo e mínimo. O máximo é a o valor máximo que o jogador conseguir e mínimo é o mínimo do jogador oponente.
São inicializadas a valores extremos.
O algoritmo escolhe o máximo dos mínimos de todas a jogadas do adversário. Acha todas as jogadas possíveis e calcula o seu Valor.
Este algoritmo difere do minimax apenas nos estados visitados em que encontra um nó que tem menor valor do que já obtido, sendo este não necessário explorar.

Minimax:

```
minimax_decidir(Ei,terminou):-terminal(Ei).

minimax_decidir(Ei,Opf):-
    findall(Vc-Nr_coluna,(jogada(Ei,'A',Nr_coluna,Tabuleiro),minimax_valor(Tabuleiro,Vc,'A')),L),
    escolhe_max(L,Opf).

minimax_valor(Ei,Val,_):-terminal(Ei),valor(Ei,Val).

minimax_valor(Ei,Val,P):-
    muda_jogador(P,J),
    findall(Val1,(jogada(Ei,J,_,Es),minimax_valor(Es,Val1,J)),V),
    seleciona_valor(V,P,Val).

seleciona_valor(V,P,Val) :-
    P ='A',
    maximo(V,Val).
seleciona_valor(V,_,Val):- minimo(V,Val).
```

Jogada:

```
jogada([H|T],Cor,1,[H1|T]):-poe_na_coluna(H,Cor,H1).
jogada([C1,H|T],Cor,2,[C1,H1|T]):-poe_na_coluna(H,Cor,H1).
jogada([C1,C2,H|T],Cor,3,[C1,C2,H1|T]):-poe_na_coluna(H,Cor,H1).
jogada([C1,C2,C3,H|T],Cor,4,[C1,C2,C3,H1|T]):-poe_na_coluna(H,Cor,H1).
jogada([C1,C2,C3,C4,H],Cor,5,[C1,C2,C3,C4,H1]):-poe_na_coluna(H,Cor,H1).
```

Valores:

```
%Azul ganha
valor(Tabuleiro,1):-tabuleiro(Tabuleiro),colunas(Tabuleiro,'A'),!.
valor(Tabuleiro,1):-
    tabuleiro(Tabuleiro),
    linhas(Tabuleiro,'A'),!.
valor(Tabuleiro,1):-tabuleiro(Tabuleiro),diagonal(Tabuleiro,'A'),!.

%Vermelho ganha
valor(Tabuleiro,-1):-colunas(Tabuleiro,'V'),!.
valor(Tabuleiro,-1):-
    tabuleiro(Tabuleiro),
    linhas(Tabuleiro,'V'),!.
valor(Tabuleiro,-1):-diagonal(Tabuleiro,'V'),!.
valor(_,0).
```

Predicados avaliadores:

```
cheio([C1,C2,C3,C4,C5]) :-
    append(C1,C2, C12),
    append(C12, C3, C123),
    append(C123,C4,C1234),
    append(C1234,C5,CF),
    \+member(' ', CF).

colunas([H|T],Cor):-(tres_em_coluna(H,Cor);colunas(T,Cor)).
tres_em_coluna([Cor,Cor,Cor,_],Cor):-Cor\=' '.
tres_em_coluna([_,Cor,Cor,Cor],Cor):-Cor\=' '.


linhas([H1,H2,H3|T],Cor):-(tres_em_linha(H1,H2,H3,Cor);linhas([H2,H3|T],Cor)).
tres_em_linha([Cor,_,_,_],[Cor,_,_,_],[Cor,_,_,_],Cor):-Cor\=' '.
tres_em_linha([_,Cor,_,_],[_,Cor,_,_],[_,Cor,_,_],Cor):-Cor\=' '.
tres_em_linha([_,_,Cor,_],[_,_,Cor,_],[_,_,Cor,_],Cor):-Cor\=' '.
tres_em_linha([_,_,_,Cor],[_,_,_,Cor],[_,_,_,Cor],Cor):-Cor\=' '.

diagonal([H1,H2,H3|T],Cor):-tres_em_diagonal(H1,H2,H3,Cor);diagonal([H2,H3|T],Cor).
tres_em_diagonal([Cor,_,_,_],[_,Cor,_,_],[_,_,Cor,_],Cor):-Cor\=' '.
tres_em_diagonal([_,Cor,_,_],[_,_,Cor,_],[_,_,_,Cor],Cor):-Cor\=' '.
tres_em_diagonal([_,_,Cor,_],[_,Cor,_,_],[Cor,_,_,_],Cor):-Cor\=' '.
tres_em_diagonal([_,_,_,Cor],[_,_,Cor,_],[_,Cor,_,_],Cor):-Cor\=' '.
```

Predicados auxiliares :

```
coluna_cheia(Coluna):- \+ member(' ',Coluna).

maximo([A|R],Val):- maximo(R,A,Val).
maximo([],A,A).
maximo([A|R],X,Val):- A < X,!, maximo(R,X,Val).
maximo([A|R],_,Val):- maximo(R,A,Val).

minimo([A|R],Val):- minimo(R,A,Val).
minimo([],A,A).
minimo([A|R],X,Val):- A > X,!, minimo(R,X,Val).
minimo([A|R],_,Val):- minimo(R,A,Val).

escolhe_max([A|R],Val):- escolhe_max(R,A,Val).
escolhe_max([],_-Op,Op).
escolhe_max([A-_|R],X-Op,Val) :- A < X,!, escolhe_max(R,X-Op,Val).
escolhe_max([A|R],_,Val):- escolhe_max(R,A,Val).

muda_jogador('V','A').
muda_jogador('A','V').

max(A,B,B) :- A < B, !.
max(A, _, A).
min(A,B,A) :- A < B, !.
min(_, B, B).
```

Testes e Exemplos

```
?- joga.
[[ , , ,A],[ , , ,V],[A,V,A,A],[ , ,A,V],[ ,V,V,A]]
minimax:2
alfabeta:2
Insira a Coluna que deseja jogar:
```

Dando o tabuleiro vazio este demora imenso tempo a responder, por isso fizemos umas prévias jogadas no tabuleiro inicial. Obviamente a jogada a se fazer por parte do jogador Azul é na segunda coluna pois faz logo três em linha. Conseguimos assim verificar que os algoritmos dão ambos a mesma resposta embora o alfabeta seja relativamente mais rápido e eficiente.