UNIVERSIDADE DE ÉVORA

Inteligência Artificial

Jogos de dois jogadores- Jogos com informação completa determinísticos

Docente: Irene Pimenta Rodrigues

Alunos: Rui Rodrigues nº14445
João Marques nº31514
João Chiola nº32456

Implementação

Para jogar ao jogo do galo aceder á diretoria e correr no prolog o comando ‘[jogos].’ e em seguida ‘esc.’

1. Escolha uma estrutura de dados para representar os estados dos dois jogos.

A estrutura utilizada para representar o estado do jogo é representado por um tuplo com uma lista de posições do tabuleiro, onde cada posição contém uma referencia(um valor não instanciado), um caractere ”x” ou ”o” que indica uma jogada e a última peça que foi jogada.

Em baixo podem ver-se as representações do s estados iniciais para o “Jogo do Galo”:

```
estado_inicial(([(p(1,1), _),(p(1,2), _),(p(1,3), _),
                 (p(2,1), _),(p(2,2), _),(p(2,3), _),
                 (p(3,1), _),(p(3,2), _),(p(3,3), _)],_)).

estado_inicial(([g,g,g,p,p,p,p,g,p,p,g,g,g,p,g,g,g,p,p,p], _, _, _)).
```

2. Defina o predicado terminal(estado) que sucede quando o estado é terminal para cada jogo.

Um estado é terminal se há uma linha, coluna ou diagonal completa, seja ela com x’s ou com o’s, ou no ultimo dos casos quando estão todas as posições preenchidas e não há ganhadores(empate).

```
terminal((E, _)):-
    linhas(E);colunas(E);diagonais(E);empate(E).
```

3. Defina uma função de utilidade que para um estado terminal que deve retornar o valor do estado (ex: -1 perde, 0 empata, 1 ganha), para cada jogo.

A função de utilidade verifica a profundidade na árvore de pesquisa e os casos em que o estado é terminal, com a excepção do empate. Os valores devolvidos pela mesma podem ser 1, 0 ou -1 sendo que 0 representa empate, 1 ganha e -1 perde.

```
valor((E, _), 1, _):-(linhas(E);colunas(E);diagonais(E)),ganhador(o),!.
valor((E, _), -1, _):-(linhas(E);colunas(E);diagonais(E)),ganhador(x),!.
valor((E, _), 0, _):- empate(E),!.
```

4. Use a implementação da pesquisa minimax dada na aula prática para escolher a melhor jogada num estado.

Foi utilizada a implementação dada pela professora nas aulas para a realização do trabalho (escolher a melhor jogada num estado).

```
g(Jogo):- [Jogo], estado_inicial(Ei), minimax_decidir(Ei,Op),nl,
write(Op),nl.

% decide qual é a melhor jogada num estado do jogo
% minimax_decidir(Estado, MelhorJogada)

% se é estado terminal não há jogada
minimax_decidir(Ei,terminou):- terminal(Ei).

%Para cada estado sucessor de Ei calcula o valor minimax do estado
%Opf é o operador (jogada) que tem maior valor

minimax_decidir(Ei,Opf):-
    findall(Es-Op, op1(Ei,Op,Es),L),
    length(L,S),
    incMais(S),
    findall(Vc-Op,(member(E-Op,L), minimax_valor(E,Vc,1)),L1),
    escolhe_max(L1,Opf).

% se um estado é terminal o valor é dado pela função de utilidade
%minimax_valor(Ei,Val,P):- terminal(Ei), valor(Ei,Val,P).
minimax_valor(Ei,Val,P):- terminal(Ei), !, valor(Ei,Val,P).


%Se o estado não é terminal o valor é:
% -se a profundidade é par, o maior valor dos sucessores de Ei
% -se aprofundidade é impar o menor valor dos sucessores de Ei
minimax_valor(Ei,Val,P):-
    findall(Es,op1(Ei,_,Es),L),
    length(L,S),
    incMais(S),
    P1 is P+1,
    findall(Val1,(member(E,L),minimax_valor(E,Val1,P1)),V),
    seleciona_valor(V,P,Val).


% Se a profundidade (P) é par, retorna em Val o maximo de V
seleciona_valor(V,P,Val):- X is P mod 2, X=0,!, maximo(V,Val).

% Senão retorna em Val o minimo de V
seleciona_valor(V,_,Val):- minimo(V,Val).


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).


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).

minimo([A|R],Val):- minimo(R,A,Val).

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

5. Implemente a pesquisa Alfa-Beta e compare os resultados (tempo e espaço) em exemplos com os dois jogos

Através dos resultados observados, pode-se concluir que o minimax, apesar de demorar mais tempo a efectuar as jogadas, faz sempre uma jogada óptima, enquanto que o corte α-β não faz a melhor jogada óptima mas leva menos tempo a efectuar a mesma. Foram realizadas 10 observações de tempo e nº de nós para jogadas com 1, 3 e 5 posições preenchidas. A média dos resultados obtidos pode ser observada através da tabela 1 para o algoritmo minimax e na tabela 2 para o corte α-β.

Tabela1 (minimax):

Nº Posições Preenchidas	Tempo (ms)	Nº de Nós
1	4324	59704
3	81.2	1109
5	6	51

Tabela 2 (α-β):

Nº Posições Preenchidas	Tempo (ms)	Nº Nós
1	29	337
3	5	18
5	3	9

6. Defina uma função de avaliação que estime o valor de cada estado do jogo, use os dois algoritmos anteriores com corte em profundidade e compare os resultados (tempo e espaço), com exemplos dos dois jogos.

A função de avaliação feita para este jogo foi fazer o cálculo do número de 1 peça isolada, somando ao número de 2 peças juntas, fazer a mesma soma para o oponente e subtrair um valor ao outro. Quando se soma o numero de 2 peças, dá-se um valor

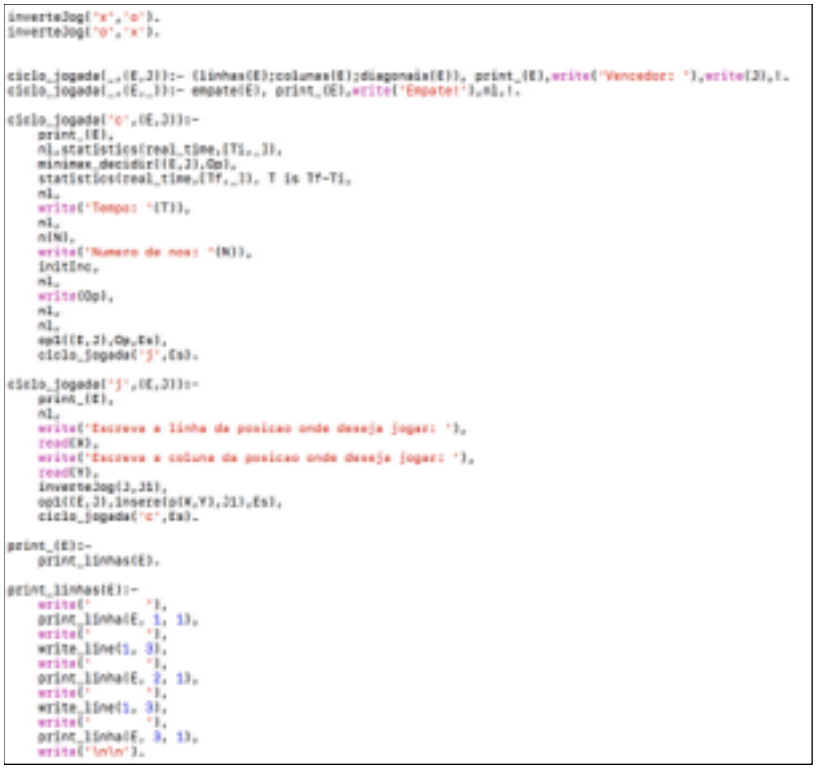
extra a este numero, multiplicando-o por 2.

```
func_aval((E,J), Val,_):-
    inverteJog(J, J2),
    aval(E,J2,Val).

aval(E, J, Val):-
    find_all_1peca(E, J, V1),
    find_all_2pecas(E, J, V2),
    Val1 is V1+(3*V2),
    inverteJog(J, J2),
    find_all_1peca(E, J2, V3),
    find_all_2pecas(E, J2, V4),
    Val2 is V3+(3*V4),
    Val is (Val1-Val2).
```

7.Implemente um agente inteligente que joga o jogo que escolheu usando a pesquisa definida na alínea anterior.

Implementou-se um agente inteligente que joga contra o jogador até que se verifique um estado terminal, mas apenas para o jogo do galo, não sendo possível para o 4 em Linha.



```
inverteJog('x','o').
inverteJog('O','x').


ciclo_jogada(_,(E,J)):- (linhas(E);colunas(E);diagonais(E)), print_(E),write('Vencedor: '),write(J),!.
ciclo_jogada(_,(E,_)):- empate(E), print_(E),write('Empate!'),nl,!.

ciclo_jogada('c',(E,J)):-
    print_(E),
    nl,statistics(real_time,[T1,_]),
    minimax_decidir((E,J),Op),
    statistics(real_time,[Tf,_]), T is Tf-T1,
    nl,
    write('Tempo: '(T)),
    nl,
    n(N),
    write('Numero de nos: '(N)),
    initInc,
    nl,
    write(Op),
    nl,
    nl,
    opl((E,J),Op,Es),
    ciclo_jogada('j',Es).

ciclo_jogada('j',(E,J)):-
    print_(E),
    nl,
    write('Escreva a linha da posicao onde deseja jogar: '),
    read(X),
    write('Escreva a coluna da posicao onde deseja jogar: '),
    read(Y),
    inverteJog(J,J1),
    opl((E,J),insere(p(X,Y),J1),Es),
    ciclo_jogada('c',Es).

print_(E):-
    print_linhas(E).

print_linhas(E):-
    write('    '),
    print_linha(E, 1, 1),
    write('    '),
    write_line(1, 3),
    write('    '),
    print_linha(E, 2, 1),
    write('    '),
    write_line(1, 3),
    write('    '),
    print_linha(E, 3, 1),
    write('\n\n').
```

8.Apresente uma tabela com o nº de nós expandidos para diferentes estados do jogo (10 no mínimo) com os vários algoritmos.

R: Não foi possível, para um mesmo jogo, expandir os nós com ambos os algoritmos, uma vez que o ciclo de jogo contém alguns bugs.

Quatro em linha

1. Escolha uma estrutura de dados para representar os estados dos dois jogos.

```
estado_inicial(([(p(1,1), _),(p(1,2), _),(p(1,3), _),(p(1,4), _),(p(1,5), _),(p(1,6), _),(p(1,7), _),
                 (p(2,1), _),(p(2,2), _),(p(2,3), _),(p(2,4), _),(p(2,5), _),(p(2,6), _),(p(2,7), _),
                 (p(3,1), _),(p(3,2), _),(p(3,3), _),(p(3,4), _),(p(3,5), _),(p(3,6), _),(p(3,7), _),
                 (p(4,1), _),(p(4,2), _),(p(4,3), _),(p(4,4), _),(p(4,5), _),(p(4,6), _),(p(4,7), _),
                 (p(5,1), _),(p(5,2), _),(p(5,3), _),(p(5,4), _),(p(5,5), _),(p(5,6), _),(p(5,7), _),
                 (p(6,1), _),(p(6,2), _),(p(6,3), _),(p(6,4), _),(p(6,5), _),(p(6,6), _),(p(6,7), _)], _)).
```

2. Defina o predicado terminal(estado) que sucede quando o estado é

```
terminal((E,_)):-
    linhas(E);
    colunas(E);
    diagonais(E);
    empate(E).
```

3. Defina uma função de utilidade que para um estado terminal que deve retornar o valor do estado (ex: -1 perde, 0 empata, 1 ganha), para cada jogo.

```
valor((E, _), 1, _):-(linhas(E);colunas(E);diagonais(E)),ganhador(o),!.
valor((E, _), -1, _):-(linhas(E);colunas(E);diagonais(E)),ganhador(x),!.
valor((E, _), 0, _):- empate(E),!.
```