Inteligência Artificial

Relatório do 3º trabalho prático
2022/2023

Jogos de dois jogadores e Jogos com informação completa determinísticos

Trabalho realizado por:
- Rodrigo Alves, nº48681
- Luís Simões, nº48726

Introdução

Com a resolução deste trabalho é pretendido que consigamos representar dois jogos diferentes – o Quatro em Linha e outro jogo à nossa escolha que seja diferente do jogo do galo e do jogo do 31 – com as estruturas de dados que achemos necessárias, sempre como um problema de pesquisa no espaço de estados.

Resolução dos Exercícios

Exercício 1 - Quatro em Linha

(a)

A estrutura de dados para representar os estado do jogo consiste numa lista de listas, em que cada lista representa uma linha do tabuleiro:

```
estado_inicial([
        [v, v, v, v, v, v, v],
        [p, b, b, p, b, p, p],
        [p, b, p, p, p, b, b],
        [b, b, p, b, p, b, b],
        [p, p, b, p, b, p, p],
        [b, b, p, p, b, b, b]]).
```

(b)

O predicado terminal sucede quando num estado há uma sequência de 4 “p” (peça preta) ou 4 “b” (peças pretas) seguidos, quer seja numa linha horizontal, vertical ou diagonal:

```
10  terminal(E) :- linha(E).
11  terminal(E) :- coluna(E).
12  terminal(E) :- diagonal(E).
13
14  % Linhas
15  linha([[X,X,X,X,_,_,_],_,_,_,_,_]) :- X \= v.
16  linha([[_,X,X,X,X,_,_],_,_,_,_,_]) :- X \= v.
17  linha([[_,_,X,X,X,X,_],_,_,_,_,_]) :- X \= v.
18  linha([[_,_,_,X,X,X,X],_,_,_,_,_]) :- X \= v.
19
20  linha([_,[X,X,X,X,_,_,_],_,_,_,_]) :- X \= v.
21  linha([_,[_,X,X,X,X,_,_],_,_,_,_]) :- X \= v.
22  linha([_,[_,_,X,X,X,X,_],_,_,_,_]) :- X \= v.
23  linha([_,[_,_,_,X,X,X,X],_,_,_,_]) :- X \= v.
24
25  linha([_,_,[X,X,X,X,_,_,_],_,_,_]) :- X \= v.
26  linha([_,_,[_,X,X,X,X,_,_],_,_,_]) :- X \= v.
27  linha([_,_,[_,_,X,X,X,X,_],_,_,_]) :- X \= v.
28  linha([_,_,[_,_,_,X,X,X,X],_,_,_]) :- X \= v.
29
30  linha([_,_,_,[X,X,X,X,_,_,_],_,_]) :- X \= v.
31  linha([_,_,_,[_,X,X,X,X,_,_],_,_]) :- X \= v.
32  linha([_,_,_,[_,_,X,X,X,X,_],_,_]) :- X \= v.
33  linha([_,_,_,[_,_,_,X,X,X,X],_,_]) :- X \= v.
34
35  linha([_,_,_,_,[X,X,X,X,_,_,_],_]) :- X \= v.
36  linha([_,_,_,_,[_,X,X,X,X,_,_],_]) :- X \= v.
37  linha([_,_,_,_,[_,_,X,X,X,X,_],_]) :- X \= v.
38  linha([_,_,_,_,[_,_,_,X,X,X,X],_]) :- X \= v.
39
40  linha([_,_,_,_,_,[X,X,X,X,_,_,_]]) :- X \= v.
41  linha([_,_,_,_,_,[_,X,X,X,X,_,_]]) :- X \= v.
42  linha([_,_,_,_,_,[_,_,X,X,X,X,_]]) :- X \= v.
43  linha([_,_,_,_,_,[_,_,_,X,X,X,X]]) :- X \= v.
```

```
45  % Colunas
46  coluna([[X,_,_,_,_,_,_],[X,_,_,_,_,_,_],[X,_,_,_,_,_,_], [X,_,_,_,_,_,_],_,_]) :- X \= v.
47  coluna([_,[X,_,_,_,_,_,_],[X,_,_,_,_,_,_],[X,_,_,_,_,_,_], [X,_,_,_,_,_,_],_,_]) :- X \= v.
48  coluna([_,_,[X,_,_,_,_,_,_],[X,_,_,_,_,_,_],[X,_,_,_,_,_,_], [X,_,_,_,_,_,_],_]) :- X \= v.
49
50  coluna([[_,X,_,_,_,_,_],[_,X,_,_,_,_,_],[_,X,_,_,_,_,_], [_,X,_,_,_,_,_],_,_]) :- X \= v.
51  coluna([_,[_,X,_,_,_,_,_],[_,X,_,_,_,_,_],[_,X,_,_,_,_,_], [_,X,_,_,_,_,_],_,_]) :- X \= v.
52  coluna([_,_,[_,X,_,_,_,_,_],[_,X,_,_,_,_,_],[_,X,_,_,_,_,_], [_,X,_,_,_,_,_],_]) :- X \= v.
53
54  coluna([[_,_,X,_,_,_,_],[_,_,X,_,_,_,_],[_,_,X,_,_,_,_], [_,_,X,_,_,_,_],_,_]) :- X \= v.
55  coluna([_,[_,_,X,_,_,_,_],[_,_,X,_,_,_,_],[_,_,X,_,_,_,_], [_,_,X,_,_,_,_],_,_]) :- X \= v.
56  coluna([_,_,[_,_,X,_,_,_,_],[_,_,X,_,_,_,_],[_,_,X,_,_,_,_], [_,_,X,_,_,_,_],_]) :- X \= v.
57
58  coluna([[_,_,_,X,_,_,_],[_,_,_,X,_,_,_],[_,_,_,X,_,_,_], [_,_,_,X,_,_,_],_,_]) :- X \= v.
59  coluna([_,[_,_,_,X,_,_,_],[_,_,_,X,_,_,_],[_,_,_,X,_,_,_], [_,_,_,X,_,_,_],_,_]) :- X \= v.
60  coluna([_,_,[_,_,_,X,_,_,_],[_,_,_,X,_,_,_],[_,_,_,X,_,_,_], [_,_,_,X,_,_,_],_]) :- X \= v.
61
62  coluna([[_,_,_,_,X,_,_],[_,_,_,_,X,_,_],[_,_,_,_,X,_,_], [_,_,_,_,X,_,_],_,_]) :- X \= v.
63  coluna([_,[_,_,_,_,X,_,_],[_,_,_,_,X,_,_],[_,_,_,_,X,_,_], [_,_,_,_,X,_,_],_,_]) :- X \= v.
64  coluna([_,_,[_,_,_,_,X,_,_],[_,_,_,_,X,_,_],[_,_,_,_,X,_,_], [_,_,_,_,X,_,_],_]) :- X \= v.
```

```
% Diagonais principais (da esquerda para a direita)
diagonal([[X,_,_,_,_,_],[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],_]) :- X \= v.
diagonal([_,[X,_,_,_,_,_],[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],_]) :- X \= v.
diagonal([_,_,[X,_,_,_,_,_],[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_]]) :- X \= v.

diagonal([[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],_]) :- X \= v.
diagonal([_,[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],_]) :- X \= v.
diagonal([_,_,[_,X,_,_,_,_],[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_]]) :- X \= v.

diagonal([[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X],_]) :- X \= v.
diagonal([_,[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X],_]) :- X \= v.
diagonal([_,_,[_,_,X,_,_,_],[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X]]) :- X \= v.

diagonal([[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X],[_,_,_,_,_,_],_]) :- X \= v.
diagonal([_,[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X],[_,_,_,_,_,_],_]) :- X \= v.
diagonal([_,_,[_,_,_,X,_,_],[_,_,_,_,X,_],[_,_,_,_,_,X],[_,_,_,_,_,_]]) :- X \= v.

% Diagonais secundárias (da direita para a esquerda)
diagonal([[_,_,_,_,_,X],[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],_]) :- X \= v.
diagonal([_,[_,_,_,_,_,X],[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],_]) :- X \= v.
diagonal([_,_,[_,_,_,_,_,X],[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_]]) :- X \= v.

diagonal([[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],_]) :- X \= v.
diagonal([_,[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],_]) :- X \= v.
diagonal([_,_,[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_]]) :- X \= v.

diagonal([[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],_]) :- X \= v.
diagonal([_,[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],_]) :- X \= v.
diagonal([_,_,[_,_,_,_,X,_],[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_]]) :- X \= v.

diagonal([[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],[X,_,_,_,_,_],_]) :- X \= v.
diagonal([_,[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],[X,_,_,_,_,_],_]) :- X \= v.
diagonal([_,_,[_,_,_,X,_,_],[_,_,X,_,_,_],[_,X,_,_,_,_],[X,_,_,_,_,_]]) :- X \= v.
```

(c)

A função de utilidade define que determina qual o valor que um estado terminal deve retornar(ex: -1 perde, 0 empata, 1 ganha):

```
% Função de utilidade (0 = empate, 1 = ganha_b, -1 = ganha_p)
valor([L1,L2,L3,L4,L5,L6],0,_):-  \+ member(v,L1),
                                    member(v,L2),
                                    member(v,L3),
                                    member(v,L4),
                                    member(v,L5),
                                    member(v,L6),!.

valor(E,V,P):-
    terminal(E),
    X is P mod 2,
    (X == 1,V = 1;X == 0,V = -1).
```

Neste caso, é empate quando já não existem “v” (espaços vazios no tabuleiro) e é associado o valor 0, dá valor 1 quando um estado E é terminal e a profundidade P da árvore é um número ímpar e dá valor -1 quando um estado E é terminal e a profundidade P da árvore é um número par.

(d)

Usando a jogada minimax dada na aula prática para escolher a melhor jogada num estado, usámos o ficheiro que foi dado na aula mas fizémos algumas alterações no código para corresponder ao nosso jogo. Estas mudanças foram devidas ao nosso operador que tinha

aridade 4.

Oper/4:

```
113   % oper(estado,jogada_na_coluna,estado seguinte)
114   oper(E,joga_coluna(Y),p, En) :-
115       joga_vazio(Y,E,p,En).
116
117   oper(E,joga_coluna(Y),b, En) :-
118       joga_vazio(Y,E,b,En).
```

Jogo_vazio/4:

```
120   % 1ª Coluna
121   joga_vazio(1,
122       [[v, C12, C13, C14, C15, C16, C17],
123       [C21, C22, C23, C24, C25, C26, C27],
124       [C31, C32, C33, C34, C35, C36, C37],
125       [C41, C42, C43, C44, C45, C46, C47],
126       [C51, C52, C53, C54, C55, C56, C57],
127       [C61, C62, C63, C64, C65, C66, C67]],
128       J,
129       [[J, C12, C13, C14, C15, C16, C17],
130       [C21, C22, C23, C24, C25, C26, C27],
131       [C31, C32, C33, C34, C35, C36, C37],
132       [C41, C42, C43, C44, C45, C46, C47],
133       [C51, C52, C53, C54, C55, C56, C57],
134       [C61, C62, C63, C64, C65, C66, C67]]):- C21 \= v, C31 \= v, C41 \= v, C51 \= v, C61 \= v.
135
136   joga_vazio(1,
137       [[C11, C12, C13, C14, C15, C16, C17],
138       [v, C22, C23, C24, C25, C26, C27],
139       [C31, C32, C33, C34, C35, C36, C37],
140       [C41, C42, C43, C44, C45, C46, C47],
141       [C51, C52, C53, C54, C55, C56, C57],
142       [C61, C62, C63, C64, C65, C66, C67]],
143       J,
144       [[C11, C12, C13, C14, C15, C16, C17],
145       [J, C22, C23, C24, C25, C26, C27],
146       [C31, C32, C33, C34, C35, C36, C37],
147       [C41, C42, C43, C44, C45, C46, C47],
148       [C51, C52, C53, C54, C55, C56, C57],
149       [C61, C62, C63, C64, C65, C66, C67]]):- C31 \= v, C41 \= v, C51 \= v, C61 \= v.
150
151   joga_vazio(1,
152       [[C11, C12, C13, C14, C15, C16, C17],
153       [C21, C22, C23, C24, C25, C26, C27],
154       [v, C32, C33, C34, C35, C36, C37],
155       [C41, C42, C43, C44, C45, C46, C47],
156       [C51, C52, C53, C54, C55, C56, C57],
157       [C61, C62, C63, C64, C65, C66, C67]],
158       J,
159       [[C11, C12, C13, C14, C15, C16, C17],
160       [C21, C22, C23, C24, C25, C26, C27],
161       [J, C32, C33, C34, C35, C36, C37],
162       [C41, C42, C43, C44, C45, C46, C47],
163       [C51, C52, C53, C54, C55, C56, C57],
164       [C61, C62, C63, C64, C65, C66, C67]]):- C41 \= v, C51 \= v, C61 \= v.
165
166   joga_vazio(1,
167       [[C11, C12, C13, C14, C15, C16, C17],
168       [C21, C22, C23, C24, C25, C26, C27],
169       [C31, C32, C33, C34, C35, C36, C37],
170       [v, C42, C43, C44, C45, C46, C47],
171       [C51, C52, C53, C54, C55, C56, C57],
172       [C61, C62, C63, C64, C65, C66, C67]],
173       J,
174       [[C11, C12, C13, C14, C15, C16, C17],
175       [C21, C22, C23, C24, C25, C26, C27],
176       [C31, C32, C33, C34, C35, C36, C37],
177       [J, C42, C43, C44, C45, C46, C47],
```

(e)

Para o Alfa-Beta aproveitamos alguns dos predicados do Minimax que eram semelhantes e mudámos os seus nomes e fizémos uma pesquisa para conseguirmos fazer o resto.

Como o algoritmo Minimax vai visitar um número muito superior de estados, e por isso o seu tempo de compilação vai ser superior, no entanto a diferença não foi muito grande.

Devido a erros de full stack overlflow, alguns exemplos não irão funcionar, tais como, quando o tabuleiro está vazio no início, entre outros com diversas combinações existentes.

(f)

Tentámos fazer a função de avaliação que estime o valor de cada estado do jogo use os dois algoritmos anteriores com corte em profundidade mas, infelizmente, não conseguimos aplicar aos nossos algoritmos e ao nosso jogo e, por isso, como tinha pouquíssimo desenvolvimento decidimos não incluir nos ficheiros entregues e, por isso, não conseguimos comparar esta alínea como pedido.

(g)

Para o agente inteligente que joga o jogo que escolhemos usando as estratégias minimax e corte alfa-beta escrevemos o seguinte ficheiro:

```
%Executar o jogo consultando este ficheiro e executando 'joga.'
%Para usar o algoritmo minimax:
%   -descomentar a parte do minimax neste predicado e no predicado joga_pc e comentar a parte do alfabeta
%Para usar o algoritmo alfabeta:
%   -descomentar a parte do alfabeta neste predicado e no predicado joga_pc e comentar a parte do minimax
joga :- /*[minimax],*/[alfabeta], g(quatroemlinha), estado_inicial(Ei), printTable(Ei),joga(Ei).

pede_coluna(Coluna):-
    read(Coluna).

joga_pc(Ei):-terminal(Ei), write('Jogo terminado.').
joga_pc(Ei):- /*minimax_decidir(Ei,Op),*/alfabeta(Ei,Op),
              write(Op),nl,
              oper(Ei,Op,b, Enn),
              printTable(Enn),nl,
              joga(Enn).

joga(Ei):- terminal(Ei),write('Jogo terminado.').
joga(Ei) :- nl,
              write('Escolha a coluna em que quer jogar: '), nl,
              write('por exemplo: 1.'),nl,
              pede_coluna(Coluna),
              oper(Ei, joga_coluna(Coluna),p, En),
              printTable(En),nl,
              joga_pc(En).

printTable([]).
printTable([Row|Rows]) :-
    writeRow(Row),
    nl,
    printTable(Rows).

writeRow([]).
writeRow([v|Cells]) :-
    write('v '),
    writeRow(Cells).

writeRow([p|Cells]) :-
    write('p '),
    writeRow(Cells).

writeRow([b|Cells]) :-
    write('b '),
    writeRow(Cells).
```

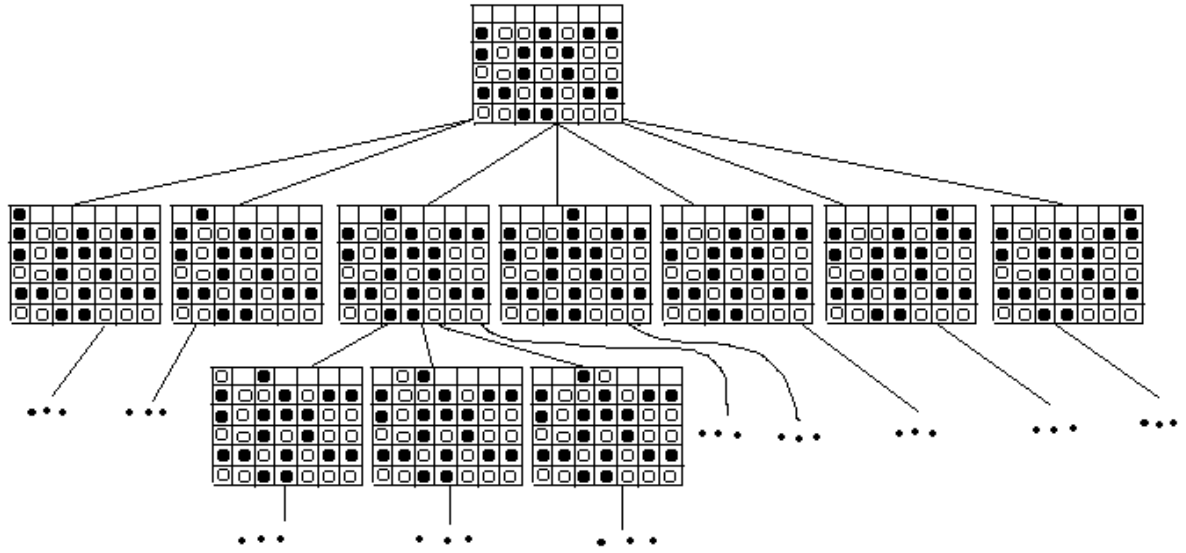
Para o usar basta iniciarmos o gprolog, consultar o ficheiro com ‘[agenteinteligente].’, usar o predicado ‘joga.’ e o jogo é iniciado automaticamente, depois é pedida a jogada que

pretende efetuar. Se fizer uma jogada inválida tem de começar de novo.

Como podemos observar acima nos comentários feitos no início do ficheiro temos que alterar o ficheiro. No caso, temos que comentar ou descomentar o algoritmo que queremos usar.

(h)

Para este exercício usámos o estado inicial acima como primeiro estado e decidimos fazer a profundidade 1 e profundidade 2 da árvore uma vez que estes já cobrem os 10 estados pedidos no mínimo.



	Número de nós expandidos	
	Profundidade 1	**Profundidade 2**
Minimax	7	42 (6 para cada nó expandido à profundidade 1)
Alfa-Beta	7	42 (6 para cada nó expandido à profundidade 1)

O número de nós expandidos para os dois algoritmos vai ser o mesmo, o que vai variar são o número de nós que precisam de ser avaliados uma vez que alguns dos nós não vão ser avaliados com o corte Alfa-Beta devido aos valores que vão ter atribuídos.

Exercício 2 - Jogo das 10 bolas

Para este exercício tínhamos que escolher um jogo de dois jogadores que fosse diferente do jogo do galo e do jogo do 31. Por isso, escolhemos um jogo que consiste no seguinte: Existem 10 (ou mais) peças em linha. Cada jogador pode retirar uma ou duas peças da esquerda para a direita. O jogador que retirar a última peça perde.

(a)

A estrutura de dados para representar os estado do jogo baseia-se na existência de um tuplo com as peças ou bolas com as quais queremos começar o jogo:

```
4   estado_inicial(e([p,p,p,p,p,p,p,p,p,p])).
```

(b)

O predicado terminal(estado) que sucede quando um estado é terminal, ou seja, quando há apenas uma peça no tabuleiro:

```
6   terminal(e([_])).
```

(c)

A função de utilidade para um estado terminal que definimos é seguinte:

```
10  valor(E, V, P) :-
11      terminal(E),
12      X is P mod 2,
13      (X == 1, V = 1 ; X == 0, V = -1).
```

Neste predicado se o estado E for terminal e a profundidade P for ímpar o computador ganha. Contudo, se o estado E for terminal e se a profundidade P for par então o computador perde e nós ganhamos.

(d)

Para o agente inteligente que joga o jogo que escolhemos usando as estratégias minimax e corte alfa-beta escrevemos o seguinte ficheiro:

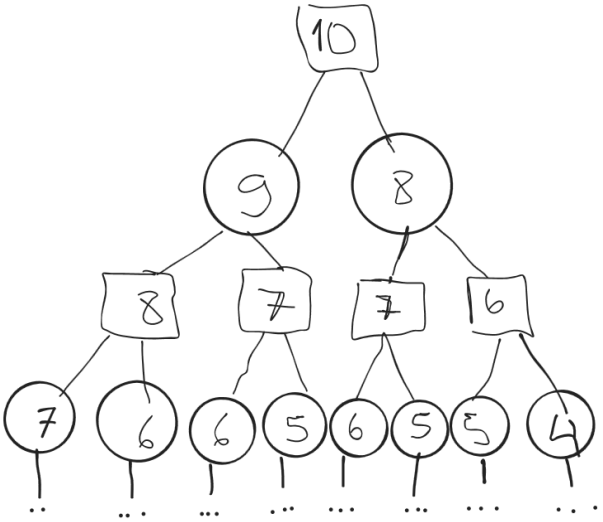
```
%Executar o jogo consultando este ficheiro e executando 'joga.'
%Para usar o algoritmo minimax:
%   -descomentar a parte do minimax neste predicado e no predicado joga_pc e comentar a parte do alfabeta
%Para usar o algoritmo alfabeta:
%   -descomentar a parte do alfabeta neste predicado e no predicado joga_pc e comentar a parte do minimax
joga :- /*[minimax],*/[alfabeta], g(bolas), estado_inicial(Ei), printState(Ei),joga_pc(Ei).

pede_coluna(NumeroDeBolas):-
    read(NumeroDeBolas).

joga_pc(Ei):- Ei = e([]), write('Jogo terminado.').
joga_pc(Ei):- /*minimax_decidir(Ei,Op),*/alfabeta(Ei,Op),
              write(Op),nl,nl,
              oper(Ei,Op,Enn),
              printState(Enn),
              joga(Enn).

joga(Ei):- Ei = e([]),write('Jogo terminado.').
joga(Ei) :-
              write('Escolha o número de bolas que quer retirar: '), nl,
              write('por exemplo: 1.'),nl,
              pede_coluna(NumeroDeBolas),nl,
              oper(Ei,retirar(NumeroDeBolas),En),
              printState(En),
              joga_pc(En).

printState(e([])).
printState(e(Estado)) :-
    write('Estado: '),
    printLinha(Estado),nl,nl.

printLinha([]).
printLinha([H|T]) :-
    write(H),
    write(' '),
    printLinha(T).
```

Para o usar basta iniciarmos o gprolog, consultar o ficheiro com ‘[agenteinteligente].’, usar o predicado ‘joga.’ e o jogo é iniciado automaticamente, depois é pedida a jogada que pretende efetuar. Se fizer uma jogada inválida tem de começar de novo.

Como podemos observar acima nos comentários feitos no início do ficheiro temos que alterar o ficheiro. No caso, temos que comentar ou descomentar o algoritmo que queremos usar.

(e)

Para este exercício usámos o estado inicial acima como primeiro estado e decidimos fazer a profundidade 1, profundidade 2, profundidade 3 da árvore uma vez que estes já cobrem os 10 estados pedidos no mínimo.
Decidimos representar cada nó da árvore com o número de peças que este tem em vez de estarmos a representar as peças todas.



	Número de nós expandidos			
	Profundidade 1	**Profundidade 2**	**Profundidade 3**	**Total**
Minimax	2	4	8	14
Alfa-Beta	2	4	8	14

O número de nós expandidos para os dois algoritmos vai ser o mesmo, o que vai variar são o número de nós que precisam de ser avaliados uma vez que alguns dos nós não vão ser avaliados com o corte Alfa-Beta devido aos valores que vão ter atribuídos.

Conclusão

Com a realização deste trabalho ficámos a ter mais conhecimento sobre a resolução de problemas como um problema de pesquisa no espaço de estados, pois aprofundámos e usámos uma vertente prática dos problemas dados nas aulas.

Para além disso, foi interessante a escolha de um jogo de dois jogadores feito por nós, uma vez que nos dá alguma liberdade de escolha e torna o trabalho menos chato na nossa opinão.

Pensamos ter atingido a maioria dos objetivos do trabalho, principalmente para o problema do jogo do 4 em linha, onde apesar de termos stack overflow em alguns tabuleiros, pensamos ter atingido e percebido a totalidade das perguntas do mesmo com excepção da alínea (f).

Já na segunda parte do nosso trabalho, tivemos muito mais facilidade uma vez que

escolhemos um jogo que pensamos ter implementado mais facilmente e com melhor eficiência por ser um jogo mais simples que o Quatro em Linha.