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

3º Trabalho Prático
Três em Linha

Alunos:
Luís Ressonha - 35003
Rúben Teimas - 39868

Docente:
Paulo Quaresma

29 de Abril de 2020

Indice

1 Introdução

Este terceiro trabalho prático tem como objetivo representar o jogo "Três em Linha" (uma versão simplificada do "Quatro em Linha") como um problema de pesquisa no espaço de estados.

O jogo deve ser realizado sobre um tabuleiro de 5 colunas e 4 linhas, contudo, devido às restrições da memória da *Stack* do *Prolog*, optámos por remover uma linha e uma coluna, ficando assim um tabuleiro *4X3*.

Para a resolução do problema foi-nos proposto que usássemos os algoritmos *Minimax* e *Minimax* com corte *Alfa-Beta* e comparássemos os resultados.

2 Respostas às perguntas do enunciado

(a) **Escolha uma estrutura de dados para representar os estado do jogo:**

R: A estrutura que escolhemos para representar os estados do jogo foi uma lista de listas, em que cada uma destas listas corresponde a uma coluna do tabuleiro. Achámos pertinente representar os estados pelas colunas e não pelas linhas dadas as características do jogo, no qual se usam as colunas.

```
/*
(1,1)(2,1)(3,1)
(1,2)(2,2)(3,2)
(1,3)(2,3)(3,3)
(1,4)(2,4)(3,4)
*/
estado_inicial([[v,v,x],
                [v,o,o],
                [o,x,o],
                [x,o,x]]).
```

(b) **Defina o predicado terminal(Estado) que sucede quando o Estado é terminal.**

R: Para representar o predicado **terminal/1** definimos 4 predicados auxiliares que representam as ocasiões nas quais o jogo chega ao fim, sendo elas: conseguir 3 símbolos seguidos numa linha, conseguir 3 símbolos seguidos numa coluna, conseguir 3 símbolos seguidos numa diagonal ou ter o tabuleiro cheio, i.e, ficar sem posições vazias. A forma que escolhemos para representar o predicado **terminal/1**, ainda que funcione, não é óptima, dado que se aumentássemos o tabuleiro em mais 2 colunas e 2 linhas, tornar-se ia impraticável representá-lo. Esta representação deve-se maioritariamente às nossas dificuldades com a linguagem *Prolog*.

```
% 1    coluna
colunas([[X,X,X],_,_,_], X):- X \= v.
colunas([[X,X,X],_,_,_], X):- X \= v.
% 2    coluna
colunas([_,[X,X,X],_,_], X):- X \= v.
colunas([_,[X,X,X],_,_], X):- X \= v.
% 3    coluna
colunas([_,_,[X,X,X],_], X):- X \= v.
colunas([_,_,[X,X,X],_], X):- X \= v.
% 4    coluna
colunas([_,_,_,[X,X,X]], X):- X \= v.
```

```
12 colunas([_,_,_,[X,X,X]], X):- X \= v.
13
14 % 1   linha
15 linhas([[X|_],[X|_],[X|_],_], X):- X \= v.
16 linhas([_,[X|_],[X|_],[X|_]], X):- X \= v.
17
18 % 2   linha
19 linhas([[_,X|_],[_,X|_],[_,X|_],_], X):- X \= v.
20 linhas([_,[_,X|_],[_,X|_],[_,X|_]], X):- X \= v.
21
22 % 3   linha
23 linhas([[_,_,X],[_,_,X],[_,_,X],_], X):- X \= v.
24 linhas([_,[_,_,X],[_,_,X],[_,_,X]], X):- X \= v.
25
26 % Diagonais Ascendentes
27 diagonais([[_,_,X],[_,X,_],[X,_,_],_], X):- X \= v.
28 diagonais([_,[_,_,X],[_,X,_],[X,_,_]], X):- X \= v.
29
30 % Diagonais Descendentes
31 diagonais([[X,_,_],[_,X,_],[_,_,X],_], X):- X \= v.
32 diagonais([_,[X,_,_],[_,X,_],[_,_,X]], X):- X \= v.
33
34 % Predicado em que o tabuleiro ta cheio
35 cheio([C1, C2, C3, C4]):-
36 append(C1, C2, C12),
37 append(C3, C4, C34),
38 append(C12, C34, FB),
39 \+ member(v, FB).
40
41 terminal(F):- linhas(F,_).
42 terminal(F):- colunas(F,_).
43 terminal(F):- diagonais(F,_).
44 terminal(F):- cheio(F).
```

(c) **Defina uma função de utilidade que, para um estado terminal, deve retornar o valor do estado.**

R: Para um estado terminal, a função de utilidade pode retornar os valores 0, 1 ou -1. Retorna 1 em caso de vitoria de "x", retorna -1 em caso de vitoria de "o"e retorna 0 em caso de empate.

```
1
2 valor(F, 1):- linhas(F, x).
```

```
valor(F, 1):- colunas(F, x).
valor(F, 1):- diagonais(F, x).
valor(F, -1):- linhas(F, o).
valor(F, -1):- colunas(F, o).
valor(F, -1):- diagonais(F, o).
valor(_,0).
```

(d) **Use a implementação da pesquisa minimax para escolher a melhor jogada num estado.**

R: Utilizando a pesquisa minimax sobre a nossa representação do problema, conseguimos obter a melhor jogada possível. Por exemplo, se o tabuleiro estiver no estado da *Figura 1*, ao executarmos o predicado *joga/0*, onde é chamado o minimax sobre a nossa representação, o resultado vai ser o estado da *Figura 2*.

```
?- printBoard.
v v o x
v o x o
x o o x
```

Figura 1: Estado

```
v x o x
v o x o
x o o x
```

Figura 2: Resultado

Embora tenha sido obtida sempre a melhor jogada usando o algoritmo *minimax*, o tempo de execução nem sempre foi o mais eficiente, tendo sido um processo bastante demorado em casos como tendo o tabuleiro vazio.

(e) **Implemente a pesquisa Minimax com corte alfa-beta e compare os resultados (tempo e número de nós visitados).**

R: Através dos resultados observados, podemos confirmar que o *minimax* demora muito mais tempo a que o *corte alfa-beta*, estando no mesmo estado, ainda que ambos os algoritmos façam uma jogada óptima. Este acréscimo de

tempo por parte do algoritmo *minimax* deve-se à quantidade superior de nós que visita, sendo que o algoritmo *minimax* com *corte alfa-beta* resolve esse problema utilizando os parâmetros *alfa* e *beta*.

Após executar os dois algoritmos com o tabuleiro vazio tivemos os seguintes resultados:

- Utilizando o algoritmo "minimax", obtivemos 1 107 696 nós visitados em aproximadamente 25 segundos;
- Utilizando o algoritmo "minimax com corte alfa-beta"obtivemos 277 166 nós visitados em aproximadamente 2 segundos.

(f) **Implemente um agente inteligente que joga o jogo "Três em Linha".**

R: Infelizmente não conseguimos concluir a etapa final deste trabalho, maioritariamente devido às dificuldades em utilizar a linguagem *Prolog*.

Em teoria não nos parecia de todo uma tarefa difícil de fazer. A nossa ideia passava por fazer retracts e asserts ao estado inicial à medida que o utilizador jogava e novamente quando era chamada o predicado *joga*, onde seria devolvida a melhor jogada por parte do agente, isto aconteceria até que um dos estados fosse terminal.
Na prática não foi tão fácil de concretizar pelo que acabámos por não o incluir no trabalho.

Para a criação do agente seria utilizado, lógicamente, o algoritmo *minimax* com *corte alfa-beta* devido ao número reduzido de nós que visita.

Ainda que não tenhamos conseguido responder a esta pergunta, achamos que o balanço foi positivo pois compreendemos o funcionamento de ambos os algoritmos e as situações em que devem ser usados, ficando só a faltar o domínio da ferramenta designada: *Prolog*.