Trabalho 3 Inteligência Artificial

NIM

- **Estados**

Para a representação do problema os estados seguem a seguinte estrutura, e (M1, M2, M3), onde M1, M2 e M3 são os 3 molhos de pauzinhos.

- **Estado Terminal**

O estado é terminal quando não existirem mais pauzinhos em nenhuma dos molhos, e (0,0,0).

- **Função de utilidade**

A função de utilidade apenas apresenta o valor 1 ou -1 visto não existirem empates.

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

- **Algoritmos**

O único algoritmo implementado foi o minmax, sendo que não foi possível implementar outros algoritmos com sucesso nos problemas propostos.

Utilizando assim o algoritmo minmax fornecido nas aulas o resultado obtido para o estado inicial proposto (3 molhos com 1, 3, 2 pauzinhos) foi o seguinte: “**retiraMolho3(2)” ,** sendo possível retirar N pauzinhos de apenas um molho, a melhor jogada obtida seria retirar 2 pauzinhos do molho 1.

Assim como nas aulas o algoritmo em questão funciona utilizado o comando “g(<problema>)”, no ficheiro “minmax.pl”.

- **Agente Inteligente**

Utilizando o algoritmo minmax com o ficheiro “minmaxAgent.pl”, da mesma forma que anteriormente, será simulado um jogo onde o computador irá escolher as melhores jogadas a cada estado.

```
| ?- g(nim).
compiling /Users/joaoverdilheiro/Deskto
/Users/joaoverdilheiro/Desktop/Escola_2
retiraMolho3(2)
retiraMolho2(2)
retiraMolho2(1)
retiraMolho1(1)
```

Jogo do Galo

- **Estados**

Para a representação do problema os estados seguem a seguinte estrutura, e ([C1, C2, C3, C4, C5, C6, C7, C8, C9] , J), onde C1..C9 representa cada uma das casas do jogo do galo e J o jogador (para saber se é atribuído “x” ou “o” á sua jogada.

- **Estado Terminal**

O estado terminal sucede quando existir qualquer linha de três símbolos iguais de um jogador, ou quando todas as casas tiverem preenchidas sem que nenhum dos jogadores tenha vencido.

```
terminal(e([0,0,0,_,_,_,_,_,_],_)):- 0==x; 0 == o.
terminal(e([_,_,_,0,0,0,_,_,_],_)):- 0==x; 0 == o.
terminal(e([_,_,_,_,_,_,0,0,0],_)):- 0==x; 0 == o.

terminal(e([0,_,_,0,_,_,0,_,_],_)):- 0==x; 0 == o.
terminal(e([_,0,_,_,0,_,_,0,_],_)):- 0==x; 0 == o.
terminal(e([_,_,0,_,_,0,_,_,0],_)):- 0==x; 0 == o.

terminal(e([0,_,_,_,0,_,_,_,0],_)):- 0==x; 0 == o.
terminal(e([_,_,0,_,0,_,0,_,_],_)):- 0==x; 0 == o.

terminal(e(L,_)):- \+ member(v,L).
```

- **Função de utilidade**

A função de utilidade apenas apresenta o valor 1 ou -1 visto não existirem empates.

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

- **Algoritmos**

O único algoritmo implementado foi o minmax, sendo que não foi possível implementar outros algoritmos com sucesso nos problemas propostos.

Assim como nas aulas o algoritmo em questão funciona utilizado o comando “g(<problema>)”, no ficheiro “minmax.pl”. A melhor jogada segundo o algoritmo utilizado seria jogar na casa (5), para o estado inicial:

e([x, o, v
v, v, v
x, o, o], x).

- **Agente Inteligente**

Utilizando o algoritmo minmax com o ficheiro “minmaxAgent.pl”, da mesma forma que anteriormente, será simulado um jogo onde o computador irá escolher as melhores jogadas a cada estado.

```
| ?- g(galo).
compiling /Users/joaoverdilhe
/Users/joaoverdilheiro/Deskt
casa(5,x)
casa(6,o)
casa(4,x)
```