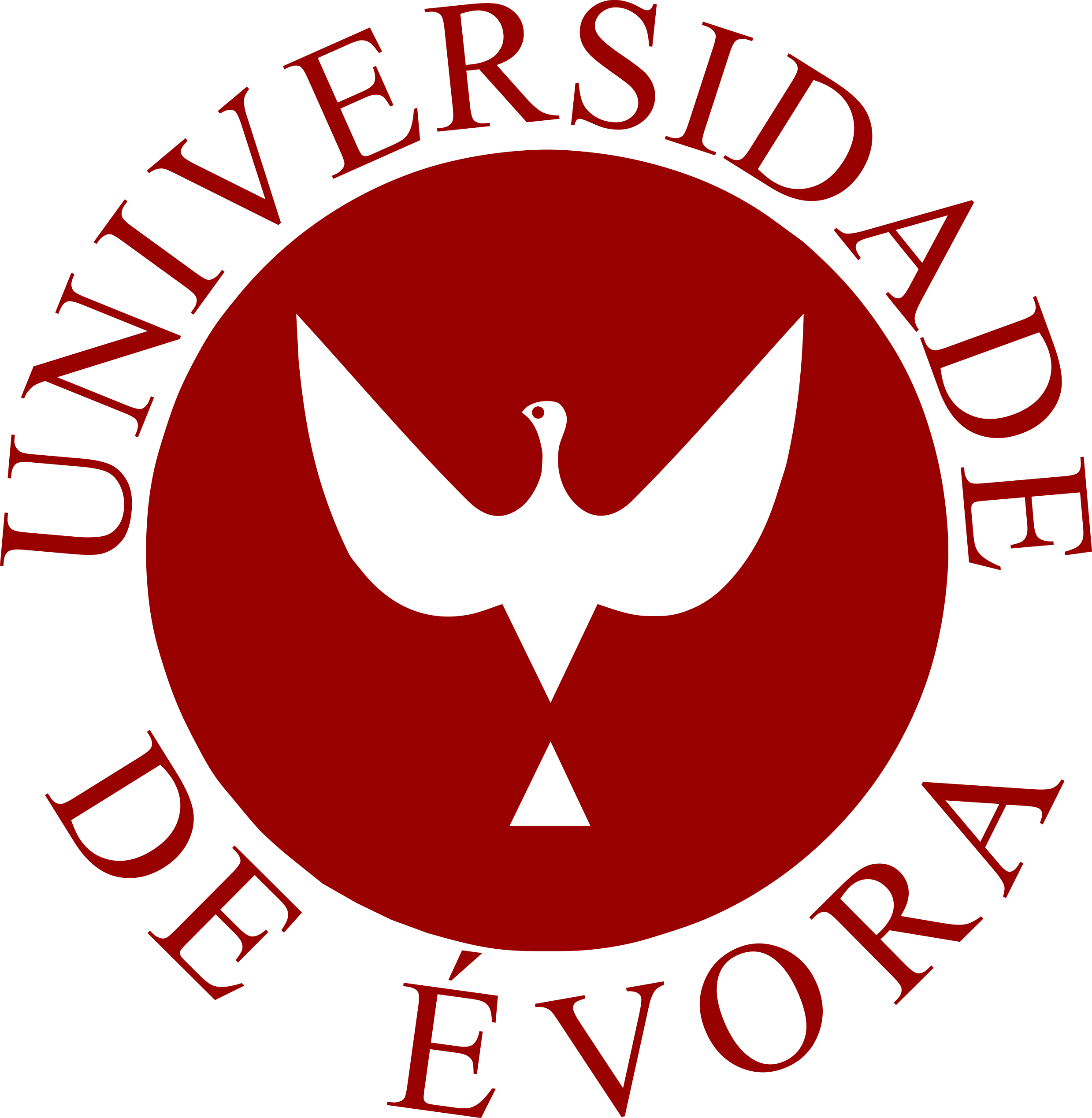
Inteligência Artificial

Trabalho 3

MinMax

Discente:

João Reys Santos nº 29634

Évora 2016

Indice

[Introdução 2](#_Toc424560460)

[1- Estrutura para estado de jogo 2](#_Toc424560461)

[2- Predicado terminal 2](#_Toc424560462)

[3- Função utilidade 3](#_Toc424560463)

[4- Operadores 4](#_Toc424560464)

[5- Execução](#_Toc424560465) 4

# Introdução

Neste trabalho pretende-se simular o jogo do galo como um problema de pesquisa em espaço de estados. Irá ser utilizado o algoritmo minmax para definir qual a melhor jogada a efectuar. Nota: este relatório foi inteiramente baseado no trabalho feito o ano passado com o aluno André Gouveia nº 26918.

# Estrutura para estado de jogo

Definiu-se o estado como um tuplo com duas listas. A 1ª lista guarda as posições por afetar, enquanto que a 2ª lista as posições já afectadas.

Ex:

estado\_inicial(([(1,2,\_),

(2,2,\_),

(3,1,\_),(3,2,\_)],[(1,1,0),(1,3,0),(2,1,1),(2,3,1),(3,3,0)])).

# Predicado terminal

Definiram-se três opções em que terminal sucede.

A 1ª opção sucede quando o jogador 1 vence.

terminal((\_,X)):-

(member((1,1,0),X),member((1,2,0),X),member((1,3,0),X)),!;

(member((2,1,0),X),member((2,2,0),X),member((2,3,0),X)),!;

(member((3,1,0),X),member((3,2,0),X),member((3,3,0),X)),!;

(member((1,1,0),X),member((2,1,0),X),member((3,1,0),X)),!;

(member((1,2,0),X),member((2,2,0),X),member((3,2,0),X)),!;

(member((1,3,0),X),member((2,3,0),X),member((3,3,0),X)),!;

(member((1,1,0),X),member((2,2,0),X),member((3,3,0),X)),!;

(member((1,3,0),X),member((2,2,0),X),member((3,1,0),X)),!.

A 2ª opção sucede quando o jogador 2 vence.

terminal((\_,X)):-

(member((1,1,1),X),member((1,2,1),X),member((1,3,1),X)),!;

(member((2,1,1),X),member((2,2,1),X),member((2,3,1),X)),!;

(member((3,1,1),X),member((3,2,1),X),member((3,3,1),X)),!;

(member((1,1,1),X),member((2,1,1),X),member((3,1,1),X)),!;

(member((1,2,1),X),member((2,2,1),X),member((3,2,1),X)),!;

(member((1,3,1),X),member((2,3,1),X),member((3,3,1),X)),!;

(member((1,1,1),X),member((2,2,1),X),member((3,3,1),X)),!;

(member((1,3,1),X),member((2,2,1),X),member((3,1,1),X)),!.

A 3ª opção sucede quando o já não há jogadas e se deu um empate.

terminal(([],\_)):-

aux =.. [empate,true],

asserta(aux).

# Função utilidade

A função utilidade é o predicado valor que recebe o estado, a profundidade e devolve valor:

* 0 se empate

valor(([],\_),0,\_):- empate(true),!.

* 1 se jogador atual for o 1º

valor(\_,1,P):- X is P mod 2, X=0,!.

* -1 caso contrário.

valor(\_,-1,\_).

# Operadores

Definido operador para cada uma das posições possiveis de jogar.

Modo genérico de operador:

op1(estado actual, nome de operador, estado seguinte, profundidade)

O algoritmo minmax foi ajustado para poder reconhecer este tipo de operador.

op1((X,Y),joga11,(W,[(1,1,J)|Y]),P):-

member((1,1,\_),X),

J is P mod 2,

remover((1,1,\_),X,W).

op1((X,Y),joga12,(W,[(1,2,J)|Y]),P):-

member((1,2,\_),X),

J is P mod 2,

remover((1,2,\_),X,W).

op1((X,Y),joga13,(W,[(1,3,J)|Y]),P):-

member((1,3,\_),X),

J is P mod 2,

remover((1,3,\_),X,W).

op1((X,Y),joga21,(W,[(2,1,J)|Y]),P):-

member((2,1,\_),X),

J is P mod 2,

remover((2,1,\_),X,W).

op1((X,Y),joga22,(W,[(2,2,J)|Y]),P):-

member((2,2,\_),X),

J is P mod 2,

remover((2,2,\_),X,W).

op1((X,Y),joga23,(W,[(2,3,J)|Y]),P):-

member((2,3,\_),X),

J is P mod 2,

remover((2,3,\_),X,W).

op1((X,Y),joga31,(W,[(3,1,J)|Y]),P):-

member((3,1,\_),X),

J is P mod 2,

remover((3,1,\_),X,W).

op1((X,Y),joga32,(W,[(3,2,J)|Y]),P):-

member((3,2,\_),X),

J is P mod 2,

remover((3,2,\_),X,W).

op1((X,Y),joga33,(W,[(3,3,J)|Y]),P):-

member((3,3,\_),X),

J is P mod 2,

remover((3,3,\_),X,W).

remover(X,[X|Z],Z):-!.

remover(X,[Y|Z],[Y|W]):-

remover(X,Z,W).

# Execução

Para executar o programa deverá estar na pasta dos ficheiros minmax.pl e 3t.pl, fazer consulta ao ficheiro minmax.pl e depois fazer a query: g(‘3t.pl’).