FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

4 em Linha

INTELIGÊNCIA ARTIFICIAL

Docente: Inês Dutra

up201405425 João Rodrigues

up201403403 Pedro Almeida

Introdução:

No nosso segundo trabalho realizado no âmbito da cadeira de Inteligência Artificial foi proposto resolver um problema que está inserido no tema de "Jogos com Oponentes".

Como em todos os jogos, e estes não fogem a exceção, o objetivo é sempre ganhar e existe sempre uma jogada no qual neste tipo de jogos é sempre condicionada por clima de incerteza e insegurança na jogada realizada porque basta um erro e poder ser fatal fazendo com que o jogador que fez a jogada possa sair derrotado. As principais diferenças neste tipo de problema em relação ao anterior trabalho é o facto de ter um espaço e tempo de procura muito maior comparado com os problemas de pesquisa.

Neste tipo de jogos (xadrez, gamão, jogo das Damas, jogo do galo entre outros) é usual usar-se algoritmos apropriados para eles e neste trabalho o nosso foco foram os algoritmos MiniMax e AlfaBeta. Estes algoritmos são caracterizados por:

- - Estado Inicial: a configuração do "tabuleiro" inicial e a sua respetiva primeira jogada;
- - Uma função is_solution onde vê se é um estado final;
- - Uma função move que gera os sucessores com o ajuda de uma outra função posso_jogar que permite verificar se a jogada é possível.
- - Uma função pontos que recebe um tabuleiro e retorna o número de pontos acumulados no mesmo.

O jogo "4 em linha" foi o jogo escolhido pela professora para a execução deste trabalho com recurso ao uso dos algoritmos MiniMax e Alfabeta. Á media que se realiza uma jogada é sempre escolhida a melhor jogada possível. De seguida é avaliado cada jogada e os seus pontos respetivos e é aplicado um dos algoritmos possíveis para a escolha mais eficaz da próxima jogada deste tipo de jogos.

Minimax

O minimax irá escolher as jogadas com maior probabilidade de levar o jogador à vitória, sabendo à partida que o adversário vai ter uma jogada que não pode ser adivinhada na sua totalidade. Os passos seguidos pelo minimax são os seguintes:

- Gera uma árvore de pesquisa desde o primeiro nó até aos nós finais.
- Aplica a função dos pontos para determinar a utilidade de cada nó a escolher.
- Usa a função utilidade dos nós terminais para determinar através de um processo de backup a utilidade dos nós no nível imediatamente acima na árvore de procura;
- Se for um nível de valor mínimo a escolher a jogada o valor calculado é o mínimo dos filhos desse mesmo nó e se for o máximo a jogar o valor calculado é o valor máximo dos nós filhos.
- Continua a usar este processo nível a nível até atingir o nó inicial.

- Quando chega ao nó inicial faz a respetiva jogada conforme o valor determinado para esse nó.

A complexidade temporal do MiniMax é de $O(b^m)$, sendo m a profundidade máxima da árvore e b o fator de ramificação.

O pseudo-código do MiniMax segue-se em baixo:

```
function MINIMAX_DECISION(state): returns an action
     inputs: state → estado corrente no jogo
     v ← MAX-VALUE(state)
     return the action in SUCCESSORS(state) with value v
function MAX-VALUE(state) returns a utility value
if TERMINAL_TEST(state) then
   return UTILITY(state)
end if
v ← -infinito
for s in SUCCESSORS(state) do
   v ← MAX(v, MIN-VALUE(s))
end for
return v
function MIN-VALUE(state) returns a utility value
if TERMINAL_TEST(state) then
   return UTILITY(state)
end if
v ← infinito
for s in SUCCESSORS(state) do
   v ← MIN(v, MAX-VALUE(s))
end for
return v
```

AlfaBeta

No algoritmo AlfaBeta, que é muito idêntico ao MiniMax, considera-se um nó em que se estiver em um nível máximo e quando é detetado a oportunidade de selecionar um nó melhor que o de nível atual ou em algum nível acima desse, então o nó nunca será alcançado durante o jogo. Como tal, quando tivermos a mínima informação possível vai nos ser possível "podar" os nós e os seus descendentes.

Este algoritmo, tal como o minimax, é do género da pesquisa em profundidade porque em cada momento da pesquisa é apenas considerado os nós ao longo de um ramo da árvore de pesquisa.

Alfa representa a melhor escolha possível a realizar até ao momento, ao longo de um ramo, para MAX. Beta é o valor da melhor escolha possível de realizar até ao momento, ao longo de um ramo, para MIN.

Até encontrar os piores valores de alfa e beta ele procura e vai atualizando os valores recursivamente, a partir do momento em que encontra esses mesmo valores para a sua execução ele corta essa mesma sub-árvore.

De seguia segue-se o algoritmo em pseudo-código:

Jogos: Algoritmo Alfa-Beta

```
function ALPHA-BETA-SEARCH(state): returns an action
     inputs: state → estado corrente no jogo
     v ← MAX-VALUE(state,-inf,+inf)
     return the action in SUCCESSORS(state) with value v
function MAX-VALUE(state,alfa,beta) returns a utility value
inputs: state, alfa → melhor alternativa para MAX, beta → melhor alternativa para MIN
if TERMINAL_TEST(state) then
   return UTILITY(state)
end if
v ← -infinito
for s in SUCCESSORS(state) do
   v ← MAX(v, MIN-VALUE(s,alfa,beta))
   if (v >= beta ) then
      return v // momento da poda
   end if
   alfa ← MAX(alfa,v)
end for
return v
function MIN-VALUE(state,alfa,beta) returns a utility value
if TERMINAL_TEST(state) then
   return UTILITY(state)
end if
v ← +infinito
for s in SUCCESSORS(state) do
   v ← MIN(v, MAX-VALUE(s,alfa,beta))
   if (v <= alfa ) then
      return v // momento da poda
   end if
   beta ← MIN(beta,v)
end for
return v
```

4 em linha:

Definição: “O jogo é uma espécie de jogo de tabuleiro, que consiste em fichas vermelhas e azuis; um tabuleiro bilateral e vertical, com diversos furos, e um botão deslizante amarelo na parte inferior do tabuleiro, para remoção das fichas. O objetivo do jogo é ir colocando as fichas, até que o jogador consiga colocar 4 fichas em linha (podendo ser na horizontal, na vertical ou na diagonal), e impedir que o adversário consiga o mesmo. Há muitas variações no tamanho da placa, sendo o mais comum 7x6, seguido por 8×7, 9×7 e 10×7.” [1]

- O tabuleiro é uma matriz 6x7;
- Dois jogadores escolhem onde querem jogar que neste trabalho optamos por escolher entre o “x” e o “o”;
- As jogadas só podem ser feitas se as posições onde vão ser colocados o “x” ou o “o” não estiverem ocupadas;
- O objetivo deste jogo é colocar quatro peças numa linha contínua vertical, horizontal ou diagonalmente;
- MinMax e AlfaBeta aplicados ao jogo 4 em linha.

Neste trabalho escolhemos C++ porque é uma linguagem com a qual estamos mais confortáveis e é mais fácil de manipular no que respeita nos acesso à memória a ser usada pelo nosso programa.

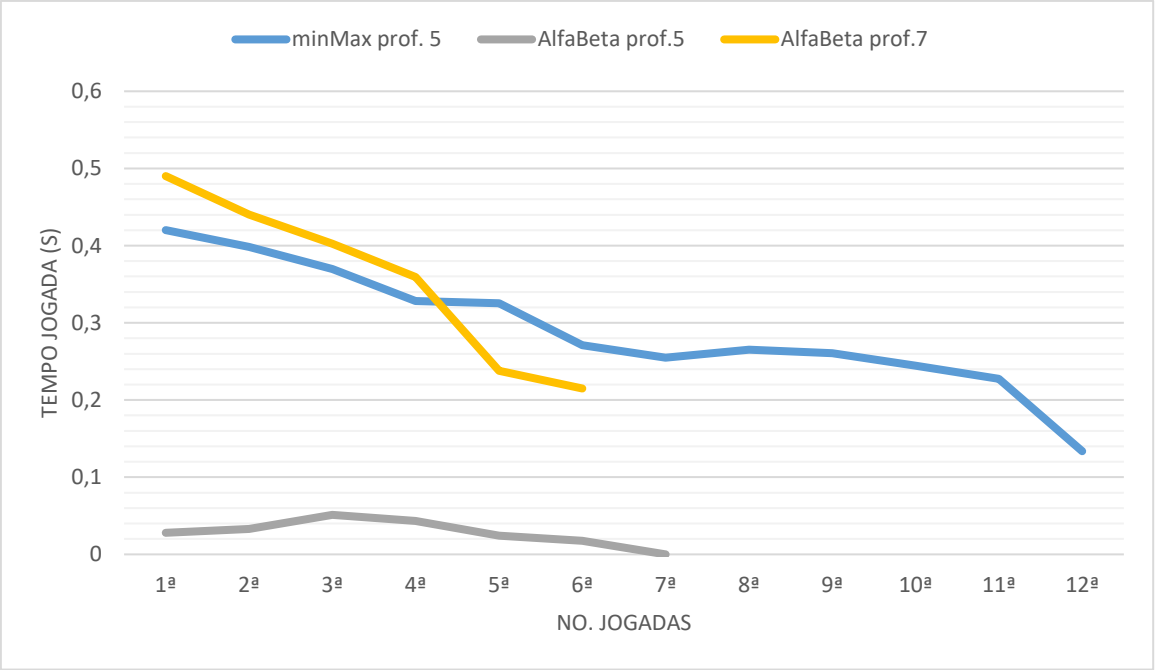
Entre os algoritmos já mencionados o Alfa-Beta, é uma otimização do Minimax estes dois algoritmos, podem ser implementados de duas maneiras cada um, das quais uma delas os algoritmos percorrem a árvore até ao fim ou então podemos limitar a profundidade tudo depende da função utilidade.

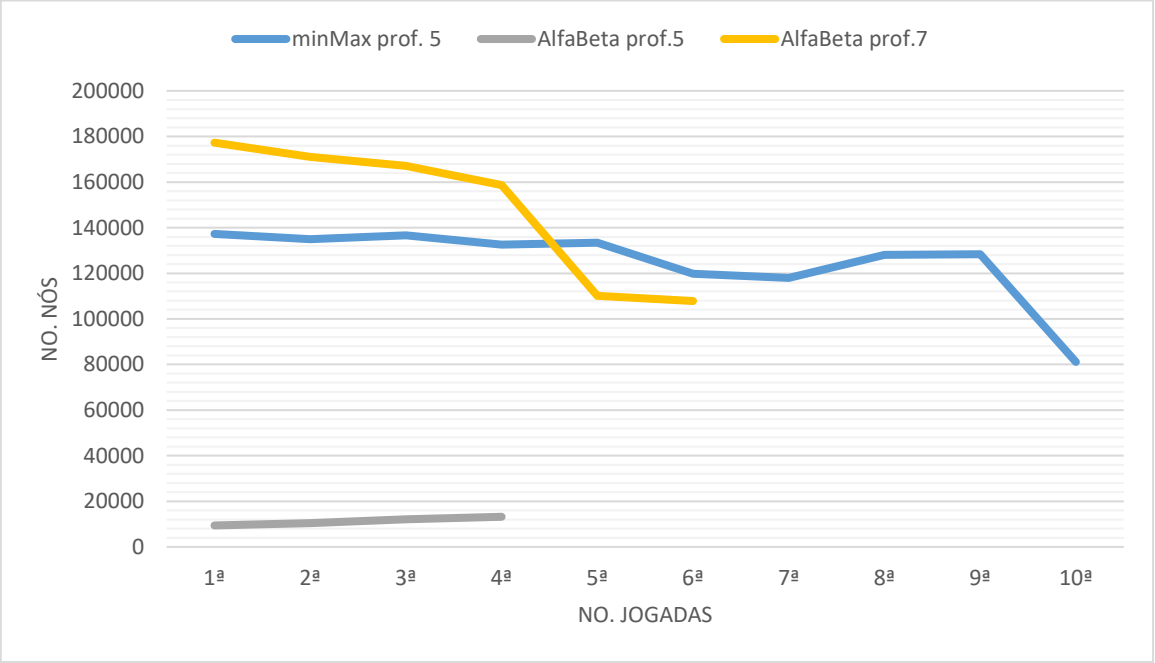
Nesta implementação as principais estruturas usadas foram:

- uma matriz 6 x 7 em que íamos guardando sempre cada nova jogada e os seus respetivos pontos.
- uma função jogada_h que verificava se a jogada era válida e se fosse fazia a respetiva jogada inserindo o ‘o’ na matriz;
- uma função jogada_pc que verificava se a jogada era válida e se fosse fazia a respetiva jogada inserindo o ‘x’ na matriz;
- uma função print que imprime a matriz em cada interação;
- uma função pontos que calcula o número de pontos por jogada;
- uma função move que gera o novo nó;
- uma função min_v e max_v para ser usada no minmax;
- uma função min_v1 e max_v1 para ser usada no alfabeta;
- uma função terminal que verifica se o nó é solução final;

Definimos uma profundidade inicial de 5.

Resultados





Segue em anexo vários ficheiros com nome tests_*.txt, que mostra o tempo e número de nós a cada jogada. Ambos os algoritmos testados com profundidade 5 e 7, a começar com o pc e a começar com o "humano".

Análise dos resultados

Após a análise minuciosa dos resultados obtidos pela execução dos dois algoritmos Minimax e AlfaBeta concluímos o seguinte:

- O Minimax é o algoritmo que expande um número maior de nós nomeadamente na jogada inaugural onde a diferença entre o número de nós expandidos é muito superior em relação ao AlfaBeta;
- Em termos de tempo de resposta, o AlfaBeta é mais rápido, especialmente a primeira jogada, onde a diferença de tempo é muito grande;

Posto isto concluímos que o algoritmo mais rápido e eficaz para o "4 em linha" é o Alfa-Beta.

Bibliografia

1. https://pt.wikipedia.org/wiki/Lig_4 (ultima consulta 23 Março 2017)