Jogos com Oponentes (Jogo dos 4)

Trabalho realizado por Bruno Silva(up201503818) e Diogo Lourenço(up201508214)

Introdução

Os jogos com oponentes consistem em dois jogadores que jogam de forma alternada em que o objetivo é os jogadores fazerem as suas jogadas de forma a chegarem a um estado final. Para implementar algoritmos inteligentes para derrotarem oponentes é necessário ter em conta uma componente que diferencia este problema de outros como uma simples procura (busca em profundidade, A*…), essa componente é a incerteza.

Quando um computador joga contra um humano não sabe qual será a próxima jogada que essa pessoa vai tomar, por este motivo os algoritmos para implementar as jogadas inteligentes, para além de possuírem espaços de buscas muito grandes, também precisam de os formular a cada nova jogada, como exemplo temos o Xadrez com uma média de fator de ramificação de 35, ou o Go, que jogado num tabuleiro 19x19 chega a um fator de ramificação médio de 250.

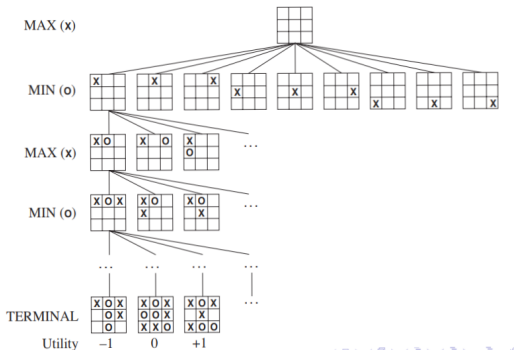
Desta forma, para criarmos algoritmos que permitem a escolha de uma maneira inteligente, podemos usar uma estratégia que dado uma jogada de um jogador, calcula todas as possíveis jogadas que daí derivam e de entre essas escolham aquela que dará o melhor resultado, isto leva-nos ao algoritmo Minimax.

Algoritmo Minimax

Este algoritmo é usado para a implementação de jogadas inteligentes num determinado jogo, dado um certo estado do jogo, o algoritmo escolhe a melhor jogada de todas as possíveis. Para isso, ele divide as jogadas em dois casos, jogadas do adversário, atribuindo a função Min e as jogadas da máquina, atribuindo-lhe a função Max.

Cada uma destas funções é chamada recursivamente e alternadamente, sendo que vão criando nós de uma árvore. Nos nós em que o estado do jogo é considerado final, o algoritmo calcula a sua utilidade, guardando esse valor para ser usado pelas funções Min e Max, de forma a subirem na árvore até a sua raiz.

A função Max escolhe o nó sucessor com maior valor, e a função Min escolhe o que tiver menor valor e ambas passam isso para os respetivos nós pai, que por sua vez também fará o mesmo sendo que apenas é trocada a função, isto é, se o filho utiliza a função Min, o pai terá que utilizar a função Max e vice-versa. Este processo será repetido até chegar ao nó raiz da árvore.



Exemplo do algoritmo Minimax aplicado ao jogo Tic-Tac-Toe

A complexidade espacial deste algoritmo é O(b.d) e a complexidade temporal é $O(b^d)$, onde b é o fator de ramificação e d é a profundidade máxima da árvore.

Alpha-Beta Pruning

No algoritmo, Minimax, temos uma complexidade temporal, assim esse algoritmo pode demorar muito tempo até devolver o resultado. Uma forma de reduzir essa complexidade é eliminar nós de uma árvore de pesquisa, dado que alguns dos nós originais são desnecessários. Isto é o que o Alpha-Beta Pruning faz, conseguindo os mesmos resultados do Minimax, mas reduzindo a sua complexidade temporal.

Para além do algoritmo usado pelo Minimax, o Alpha-Beta introduz dois novos valores de controlo, α (alpha), sendo esta a melhor escolha para a função Max e β (beta), a melhor escolha para a função Min.

Os nós desnecessários são eliminados quando o valor recebido da função utilidade dos nos filhos for menor ou igual a α no caso em que está a ser usada a função Min e maior ou igual a β no caso em que é usada a função Max.

Este algoritmo tem uma complexidade espacial de O(b.d) e uma complexidade temporal de $O(b^d)$, contudo no pior das hipóteses esta complexidade pode ser igual à do Minimax.

Jogo dos 4

Este é um jogo para dois jogadores que consiste num tabuleiro com 6 linhas e 7 colunas, 21 peças de uma cor para um dos jogadores e 21 peças de cor diferentes da anterior para outro jogador.

Durante o jogo, os jogadores vão depositando alternadamente as respetivas peças no topo da tabela com o objetivo de formar uma linha de 4 peças consecutivas da sua cor, sendo que esta linha pode ser na horizontal, vertical ou diagonal. Quando as peças são depositadas caem até ao fundo da tabela, sendo se uma peça é depositada na primeira

coluna, quando se coloca outra peça na mesma coluna esta fica na linha superior. O jog termina quando um dos jogadores alcança o objetivo, ou quando já não há mais onde jogar, ou seja, todas as peças foram usadas.

Minimax e Alpha-Beta Pruning no Jogo dos 4

Para aplicarmos estes algoritmos ao jogo dos 4, é necessária a criação de uma função capaz de encontrar a melhor jogada possível. Para isso aplica-se uma busca em profundidade na arvore gerada pelas funções Min e Max. Esta árvore é gerada sempre que o jogador executa a sua jogada, sendo o estado atual a raiz da árvore e os seus filhos serão as possiveis jogadas que daí derivarem e assim sucessivamente.

O jogo está implementado de forma a que se comece com um tabuleiro “vazio” (cada célula contém o ‘-‘) que em código está implementado na forma de matriz 6x7, sendo que é dado a escolher ao jogador qual o algoritmo que pretende usar, se pretende jogar com as peças representadas por X ou pelas O e ainda se pretende jogar ou não em primeiro lugar. Quando começa o jogo propriamente dito o jogador na sua vez pode escolher uma das 7 colunas para a sua jogada, caso esta não esteja completa, seguidamente será a vez do computador jogar, esta sequência é repetida alternadamente até que se atinja um estado final. Internamente, quando o jogo é iniciado, é chamada a função MINIMAX no caso do jogador ter escolhido o algoritmo Minimax ou a função ALPHA-BETA, no caso de ter escolhido o algoritmo Alpha-Beta, por sua vez, estas funções chamam a função MIN e esta chama a função MAX, sedo estas aplicadas recursivamente até chegar a um nó terminal (nós cujo estado do jogo é com algum vencedor, quatro em linha, ou com o tabuleiro cheio) ou até chegar ao limite de profundidade 9. Estas situações seguidamente chamam a função UTILITY que gera valores de pontuação em relação ao estado atual do jogo segundo as seguntes regras:

512 - vitória do computador
-512 - vitória do jogador
0 - empate, mistura de X’s e O’s ou se tiver vazio
50 - 3 em linha do computador
10 - 2 em linha do computador
1 - por cada peça do computador
-50 - 3 em linha do jogador
-10 - 2 em linha do jogador
-1 - por cada peça do jogador

Dado que as funções foram sendo chamadas recursivamente, estes valores vão ser retornados até a raiz permitindo ao computador escolher a melhor jogada.

Conclusão

Concluimos que o algoritmo Minimax tem um ótimo desempenho no que toca a jogos com oponentes mas quando o espaço de procura é muito grande pode demorar bastante tempo para encontrar uma solução. Para combater esse problema estabelecemos uma profundidade máxima mas com isso podemos perder a otimalidade. O Alfa-Beta, pelo

contrário, observasse uma grande melhoria em termos de tempo e número de nós expandidos.

Em suma, o algoritmo Alfa-Beta Prunning é um melhoramento mais eficiente do Minimax, porque utiliza menor espaço no que respeita à expansão de nós, o que permite chegar a uma solução mais rapidamente. Mas se o universo de procura for aceitável os dois algoritmos garante sempre que o computador ganhe ou no pior dos casos empate.

Bibliografia

Russell, S. and Norvig, P. (1995). In A modern Approach - Artificial Intelligence. Pearson

Education Inc.

http://www.dcc.fc.up.pt/~ines/aulas/1718/IA/jogos_new.pdf - slides das aulas sobre jogos com oponentes

https://www.cs.cornell.edu/courses/cs312/2002sp/lectures/rec21.htm – visitado a 21 de março 2018