TEORIA DOS JOGOS.

Jogos que envolvem adversários são diferentes dos jogos vistos até agora. O jogador não tem que se preocupar apenas em chegar ao objetivo final, mas também evitar que algum oponente chegue antes dele.

A área de Jogos, mais especificamente a elaboração de jogadores automáticos têm sido campo de pesquisa em IA, devido a diversos fatores simplicidade das regras, por serem abstrações e poderem ser facilmente representados por um problema de busca. É considerado um campo desafiador onde deve-se manter uma relação entre tamanho e limitação de tempo: necessidade de fazer lance sem certeza de que é o melhor e oponente imprevisível (incerteza devido ao outro jogador).

4.1 JOGOS DE DUAS PESSOAS

Considere dois jogadores: MAX e MIN: MAX começa jogando primeiro e eles alternam movimentos até que o jogo termine. No final do jogo, pontos são dados para o vencedor e possíveis penalidades para o perdedor.

FORMULANDO E RESOLVENDO PROBLEMAS:

O **estado inicial**, o qual inclui as posições do tabuleiro e uma indicação da vez, ou seja, mantém informação sobre o jogador da vez.

Estado final, posições nas quais o jogo acaba (para o jogo da velha, não somente quando as nove posições estiverem preenchidas).

Um **conjunto de operadores**, os quais definem que movimentos legais um jogador pode fazer.

Um **teste de término**, o qual indica quando o jogo terminou. Os estados nos quais o jogo terminou é chamado de estados terminais.

A função utilidade (*payoff function*) valor numérico do resultado (pontuação). Em jogos que o resultado é "ganhar", "empatar" ou "perder", a função utilidade é +1, 0, -1, respectivamente. Em outros jogos, diferentes funções de utilidade podem ser definidas.

4.2 ALGORITMO MINIMAX

Idéia: maximizar a utilidade (ganho) supondo que o adversário vai tentar minimizá-lo. O agente é o MAX e o adversário é o MIN: o jogador MAX escolhe seu movimento entre nove posições depois o MIN escolhe e assim por diante.

O algoritmo Minimax foi desenvolvido para determinar a melhor estratégia para MAX, e decidir qual é o melhor movimento a ser feito. O algoritmo tem os seguintes passos:

1. Gera a árvore inteira, do estado atual (raiz) até todos os estados terminais.

2. Aplica a função de utilidade nas folhas (estados terminais).

3. Propaga os valores dessa função subindo a árvore através do minimax, ou seja, utiliza a função de utilidade dos terminais para determinar o nível acima na árvore de busca. Nos movimentos do MIN assuma que ele realize a melhor jogada para ele (menor ganho para MAX), então o valor mínimo é dado ao pai:

4. Quando acima é a vez de MIN jogar, escolher o que levaria para o retorno mínimo e quando acima e a vez de MAX fazer um movimento, escolher o que levaria para o retorno Maximo. Continue subindo até a raiz, nos movimentos do MAX o maior valor da função de utilidade é atribuído ao pai.

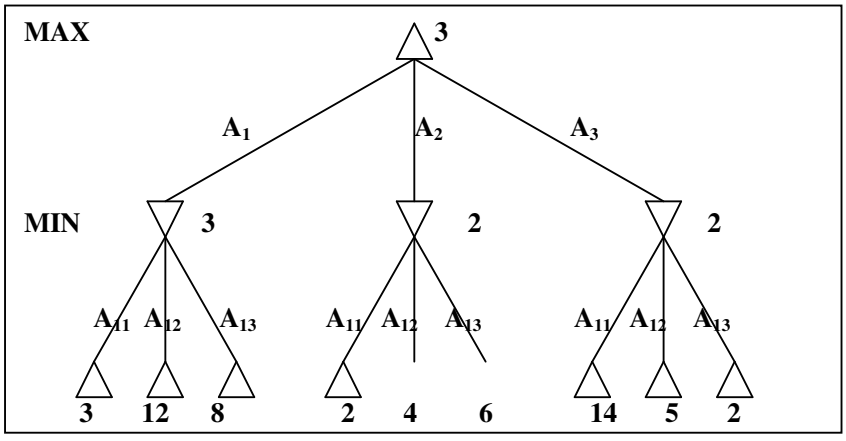
5. Escolha o movimento que leve a um nodo de maior valor.

Exemplo: mesmo para um jogo simples como o "jogo da velha" é muito complexo desenhar a árvore, então considere o seguinte jogo mais simplificado (um movimento e uma resposta).

Movimento possíveis do MAX: A1, A2 e A3.

Movimento possíveis do MIN: A11, A12 e A13.

E a função de utilidade dos terminais varia de 2 a 14



$\Rightarrow$ A solução de examinar todas as possibilidades utilizando o minimax é impraticável devido ao custo $O(b^m)$

Para melhorar (limitar o tempo):

- Deve-se trocar a função utilidade (que necessita ir até as folhas da árvore) por uma função heurística (função de avaliação).
- Podar a árvore onde a busca seria irrelevante: poda alfa-beta.

4.3 FUNÇÕES DE AVALIAÇÃO:

Idéia: durante o jogo, dada uma posição, a função de avaliação retorna uma aproximação/estimativa da "utilidade".

Obs.: jogadores de xadrez: funções para calcular as chances de vitória baseadas no posicionamento do jogo: pesos para as peças, "Uma boa distribuição dos peões", segurança do rei.

FUNÇÕES HEURÍSTICAS PARA O JOGO DA VELHA:

Número de possibilidades para cada jogador:

X		
	O	

h = 6 – 5 = 1

FUNÇÕES HEURÍSTICAS PARA O XADREZ:

Função linear ponderada de pesos:

$$h(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s).$$

Onde w é o peso da peça e f o número de peças.

4.4 PODA ALFA-BETA:

Idéia: não expandir desnecessariamente nós durante o minimax.

Parâmetros:

α = melhor valor no caminho para MAX

β = melhor valor no caminho para MIN

Teste de expansão:

α não pode diminuir (não pode ser menor do que um ancestral)

β não pode aumentar(não pode ser maior do que um ancestral)

ALGORITMO:

Se α é maior do que a melhor pontuação para o MIN (ou seja, a menor pontuação) então não explora os filhos, pois o MAX escolhe o maior, e este valor já é suficientemente grande para o MIN não escolher este ramo da árvore.

Semelhantemente, se β é maior do que a melhor solução, não explora os filhos...