1

# Jogos Adversariais

2

## Jogos Adversariais

- Jogos: considerados desafios à inteligência humana - geralmente envolvem problemas de difícil solução
- interesse de IA: aspecto competitivo
- Mundo abstrato (artificial):
  - estados podem ser representados de maneira completa
  - possuem um conjunto limitado de ações bem definidas que o agente pode executar
- Primeiro jogo de xadrez para computador: 1950 (Turing and Shannon)

3

## Jogos de xadrez

- requer inteligência
- regras simples
- acesso total ao estado do mundo
- Problema de busca complexo:
  - $b \approx 35$ nós;
  - profundidade da solução: 50 para cada jogador
  - espaço de busca $\approx 35^{100}$

4

## Características

- Incerteza
  - Sobre a decisão do jogador oponente
  - Tempo limitado de se tomar decisões - calcular a exata consequência de seus movimentos envolve incerteza
- Tomar decisões baseada em experiência passada:
  - muito parecido com problemas do mundo real comparado aos problemas que vimos até agora
- Limite de tempo penaliza jogadores ineficientes

5

## Jogos como um problema de busca

- O agente oponente introduz *incerteza*
  - deve lidar com o problema de contingência
  - suposição: sempre que for possível o oponente escolherá o movimento que mais lhe favoreça (oponente como um agente racional)
- Questões:
  - Como fazer o melhor uso do tempo para fazer boas decisões?
  - Como encontrar o melhor movimento?
  - Como deve ser a função de avaliação?

6

## Jogo de duas pessoas - adversarial

- Duas pessoas:
  - MAX (primeiro a jogar)
  - MIN
- Final de jogo: pontos para o vencedor ou penalidades para o perdedor
- Pode ser definido como um problema de busca

7

## Jogos adversariais como busca

- **Estado Inicial**: ??
- **Operadores**: ??
- **Teste do gol**: ??
- **Função de avaliação**: ??

8

## Jogos adversariais como busca

- **Estado Inicial**: configuração do tabuleiro e quem será o próximo a jogar
- **Operadores**: definem os movimentos legais que o jogador pode realizar
- **Teste do gol**: determina se um estado é o término do jogo (estados terminais)
- **Função Utilidade (ou Função Lucro)**: fornece um valor numérico para os estados terminais (ganhou, perdeu ou empatou)

9

## Estratégia ótima ou racional para MAX

- É função de MAX usar a árvore de busca para determinar o melhor movimento: maximiza a sua chance de vitória.
- MAX não pode fazer uma busca sem considerar as possíveis escolhas do oponente
- Questão: Como encontrar uma estratégia que levará a um estado de nó terminal ***independentemente das decisões tomadas pelo oponente*** que inclua as melhores decisões de MAX ?

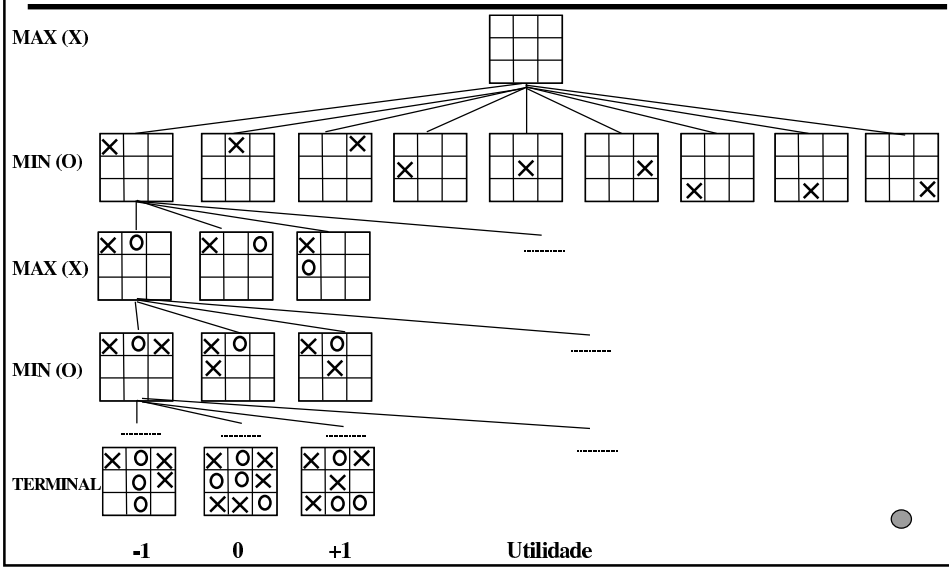
10

## Estratégia ótima ou racional para MIN

- MAX faz as seguintes suposições sobre o comportamento de MIN:
  - MIN irá se mover para minimizar a chance de vitória de MAX
  - MIN é inteligente: nenhum erro por distração ou por fatores psicológicos
  - MIN se comporta como um agente racional: faz a coisa certa

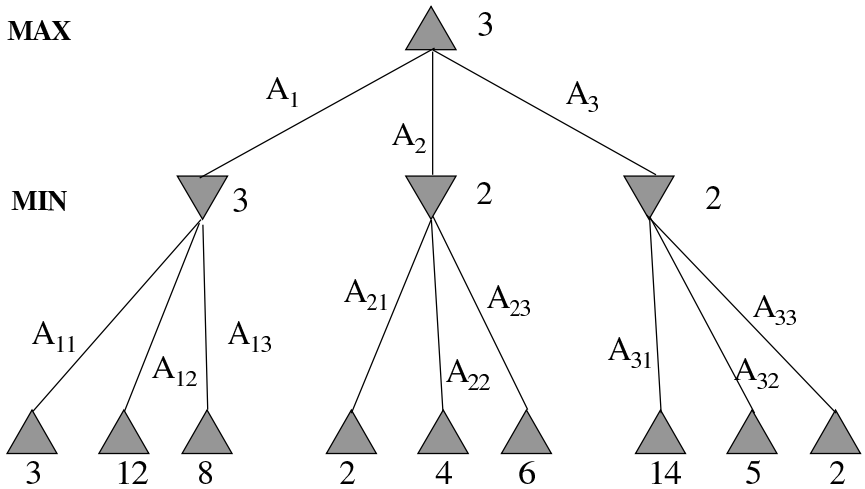
11

# Árvore de busca - Jogo da Velha




12

# Árvore *two-ply* jogo




*ply*: par de movimentos entre os dois adversários

13

## MINIMAX algoritmo

- Determina a estratégia ótima para MAX => qual é o próximo melhor movimento
- Decisão MINIMAX: maximiza a utilidade sob a ***suposição*** que o oponente jogará de maneira correta com o objetivo de minimizar a função utilidade

14

## MINIMAX algoritmo

① gera a árvore do jogo **inteira**, até os estados terminais

② aplica a função utilidade para cada estado terminal

③ a partir dos estados terminais determinar a utilidade dos nós um nível acima (MIN escolhe o menor *f-utilidade* => suposição de que MIN fará a coisa certa)

④ continua a subir na árvore um nível por vez

⑤ atinge a raiz da árvore para MAX escolher o seu próximo movimento

15

## Minimax-Decisão

**Função** Minimax-Decisão(*jogo*) **retorna** *um operador*
  **para cada** *op* **em** Operadores[jogo] **faça**
    Valor[*op*] ⟵ Minimax-Valor((Aplica(*op, jogo*), *jogo*)
  **fim**
  **retorna** o operador de maior valor

Retorna o operador que corresponde ao melhor movimento possível (o melhor valor da função utilidade) sob a suposição que o oponente joga para minimizar a função utilidade.

16

## Minimax-Valor

**Função** Valor-Minimax (*estado, jogo*) **devolve** *um valor utilidade*
  **if** Teste-Terminal[*jogo*](*estado*) **então**
    **devolve** Utilidade[*jogo*](*estado*)
  **senão se** MAX move em *estado* **então**
    **devolve** o maior de Valor-Minimax de Sucessores (*estado*)
  **senão**
    **devolve** o menor de Valor-Minimax de Sucessores (*estado*)

17

## MINIMAX: suposições

- É possível realizar a busca até nós terminais? (DFS ==> $O(b^m)$ )
- para a maioria dos jogos isso é impraticável
- xadrez ($b$=35, 50 movimentos por jogador): $35^{100}$ => $10^{22}$ séculos para gerar a árvore completa de busca
- busca até os estados terminais é impossível
- Solução: ???

18

## MINIMAX: problema

- Deve-se limitar o tempo da busca para se fazer um movimento (restrição do mundo real)
- Solução: Shannon propôs cortar a árvore antes e aplicar uma função de avaliação heurística nas folhas da árvore:
  - substitui a função de utilidade pela função de avaliação EVAL
  - substitui o teste terminal por um teste de corte

# Funções de avaliação

- Retorna uma estimativa da utilidade esperada do jogo a partir de uma dada posição
- Livros introdutórios de xadrez calculam um *valor material* de cada peça:
  - peão = 1
  - cavalo ou bispo = 3
  - torre = 5
  - rainha = 9
- O lado com uma vantagem segura de valor material de um peão ou mais, provavelmente será o vencedor



# Funções de avaliação

- Como medir a qualidade da medida da chance de vencer?

# Qualidade da função de avaliação

① *EVAL* deve **concordar** com a avaliação de utilidade dos estados terminais

② não deve consumir muito tempo para avaliar níveis da árvore (compromisso entre precisão e custo)

③ deve refletir precisamente sobre as *chances* de vencer o jogo



# Incerteza no jogo

- corte na árvore de busca gera ***incerteza*** sobre a chance de vencer ➔ função de avaliação pode ser especificada através de probabilidade
- ***função linear***: incorporam características do estado que podem ser julgadas de maneira independente (ex.: valor material):

$$w_1f_1 + w_2f_2 + w_3f_3 + ... + w_nf_n$$

Por exemplo, no jogo de xadrez: $w_i$ = peso de cada peça; $f_1$ = quantidade de peças no tabuleiro

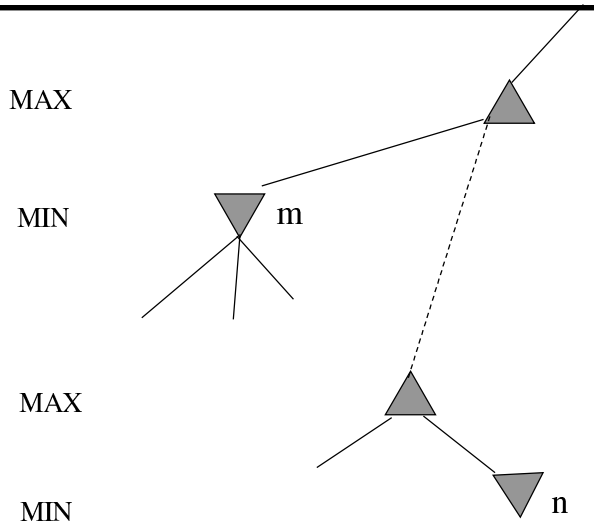
23

## Corte na busca

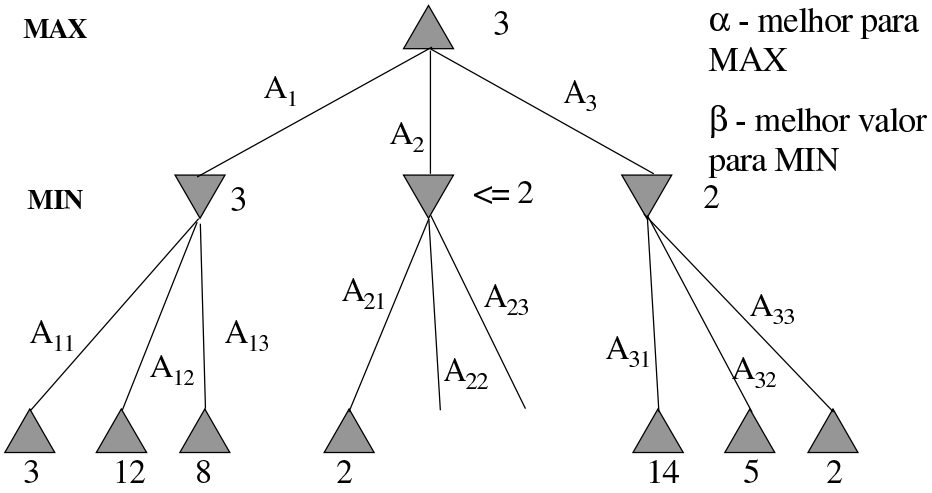
- algoritmo de busca: ≈ 1000 estados/seg
- xadrez: 150 seg/movimento
  - busca: 150.000 estados => 4 jogadas (nível de jogador novato)
- fixar uma profundidade (tempo permitido entre jogadas, de acordo com as regras do jogo)
- IDS: retorna o resultado da busca completa de maior profundidade a qualquer momento
- Problema: a aproximação da função de avaliação pode não estimar certas consequências:
  - efeito horizontal

24

## Alfa-Beta corte



# Alfa-Beta corte (heurística no corte)






# Alfa-Beta corte

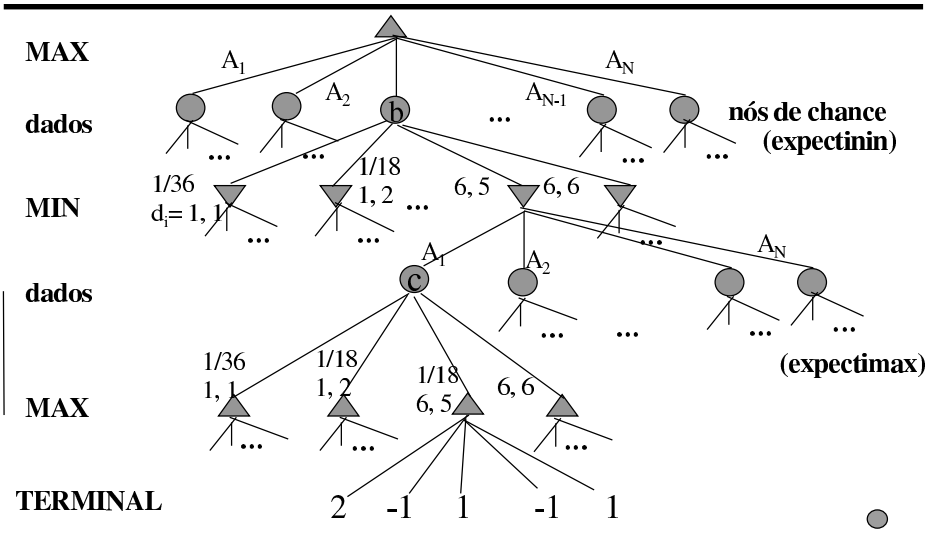
- corte (pruning): processo de eliminar um ramo da árvore de busca sem examiná-la
- aplicado ao jogo de xadrez: aumenta a capacidade de busca (respeitando o limite de tempo de uma competição) para 8 jogadas
- depende da ordem em que nós são avaliados: ordem aleatória não garante reduzir a complexidade.
- melhor com uma função de ordenação. IDS pode orientar a escolha de ordem das buscas
- no xadrez pode reduzir *b* de 35 para 6 (capturar < ameaçar < movimento para frente < movimento para trás)

# Jogos de sorte

- eventos inprevisíveis (jogar dados)
- backgamon: lance de 2 dados antes de cada jogada determina os movimentos possíveis para cada jogador
  - 36 combinações dos 2 dados
  - probabilidade: 1/36 para valores duplos (1-1 a 6-6); 1/18 para o restante (simétricos)
- deve-se incluir *nós de chance* além de nós de MAX e MIN.
- podemos apenas calcular um *valor provável*



# Árvore de busca com probabilidade



29

## *Expectimaxing*

Seja:

- $C$ , um nó chance
- $d_i$ *é* , um lance dos 2 dados
- $P(d_i)$ , a probalilidade de obter $d_i$
- $S(C, d_i)$, o conjunto dos sucessores de C aplicando os movimentos legais para o lance de dado $d_i$

Então o valor estimado *(expected value)* de C é:

$$\exp ectimax(C) = \sum_i P(d_i) max_{s \in S(C,d_i)} (utilidade(s))$$

30

## Jogos - evolução

- previsão feita por Turing ...
- xadrez: venceu Kasparov em jogos com rodadas de 5 e 25 minutos; em campeonatos fica entre os 100 primeiros
- 1970: programas complicados com truques de cortes
- campeonato de programas de xadrez ACM (desde 1970): alpha-beta search + BD de jogadas de campeões e algoritmos de jogadas infalíveis
  - arquiteturas híbrida de tomada de decisão

## Jogos - evolução

- Belle: primeiro computador dedicado para xadrez. Circuitos integrados implementam a geração de movimentos e avaliação de configuração
  - primeiro programa *master-level*
- Hitech: gera $10^4$ posições por jogada.
  - função mais precisa de avaliação proposta até hoje. Primeiro a vencer de um campeão de xadrez (Denker). Primeiro a vencer um campeão (1987)



## Jogos - estado da arte

- O melhor jogo de xadrez da atualidade: Deep Thought 2 (IBM + pesquisadores de CMU)
  - utiliza uma função simples de avaliação + alpha-beta prunning
  - poder: 10-processadores com capacidade de examinar 1/2 bilhão de posições por jogada
  - atinge profundidade de 10 a 11 jogadas; profundidade maior para *casos identificados* (já encontrou soluções de profundidade 37)
  - próxima versão: atingirá profundidade de 14 jogadas com a adição de um chip dedicado

# Jogos - estado da arte

- Chinook: em 1994 o programa que joga damas se tornou o campeão do mundo
  - sistemas anteriores empregavam aprendizagem de máquina para descobrir a melhor função de avaliação
  - emprega alpha-beta search + um BD com soluções perfeitas
- Othello: muito melhor que humanos. Variação de xadrez com espaço de busca bem menor: 5 a 15 movimentos possíveis
- Backgammon: poder na função de avaliação;
- Go com $b$= 360 não pode ser tratado como um problema de busca. Possível solução: uso de PSMs