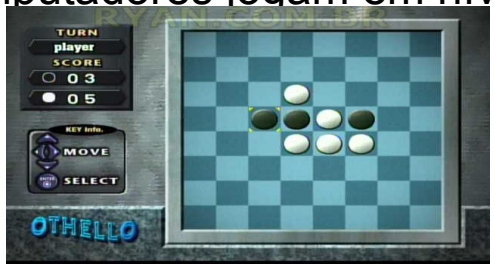
Inteligência Artificial

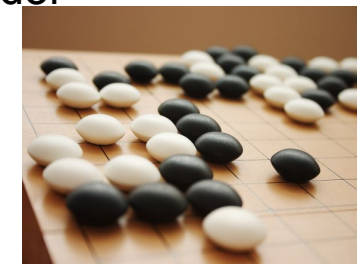
Busca competitiva

Profa. Debora Medeiros

Jogos

- Entre primeiras tarefas empreendidas em IA
  - Máquinas:
    - Ultrapassaram humanos em:
      - Damas
      - Othello
    - Derrotaram humanos algumas vezes em:
      - Xadrez
      - Gamão
    - São competitivos em outros jogos
      - Excecão: Go – computadores jogam em nível amador





Jogos

- São domínios clássicos em IA
  - Estruturados: fáceis de formalizar e representar
    - Ações bem definidas
  - Clara definição de sucesso e fracasso
  - Podem ter sua **complexidade reduzida** ou aumentada
  - Exigem a **tomada de decisões**
    - Muitas vezes em um curto intervalo de tempo
  - Há interação e pode haver **não determinismo**

Jogos x busca clássica

- O oponente é “imprevisível”
  - Dificuldade de levar em consideração todos os movimentos possíveis do oponente

- Limites de tempo
  - tomar uma decisão, mesmo que não seja ótima

Decisões em jogos

- Jogos com dois jogadores
  - MIN e MAX
    - MAX faz primeiro movimento e eles se revezam
- Jogos
  - Metas em conflito
  - Jogos de revezamento de dois jogadores

Jogo como busca

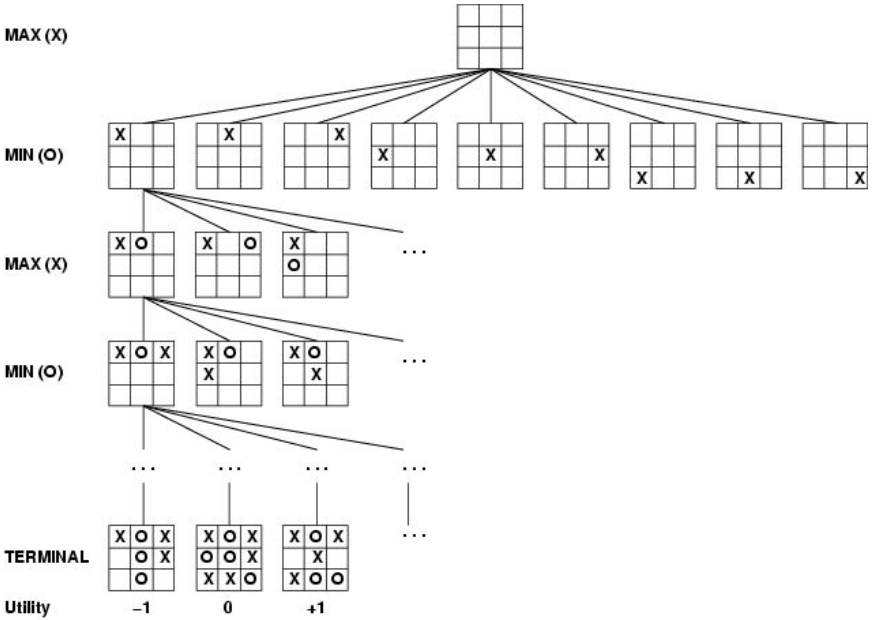
- Problema de busca com componentes:
  - *Estado inicial*:
    - Posição no tabuleiro do jogo e que jogador inicia
  - *Gerando sucessores*:
    - Lista de pares (movimento possível, estado resultante)
  - *Teste de término*:
    - Determina quando o jogo termina (estados terminais)
  - *Função de utilidade (objetivo ou compensação)*:
    - Dá valor numérico aos estados terminais
      - Exemplos:

Xadrez: vitória = +1; derrota = -1; empate = 0
Gamão: 192 a -192

Jogos de soma 0

Árvore de jogo

- Árvore de busca de jogo
  - 2 jogadores, determinístico, turnos
  - Mostra todas as possibilidades de jogo



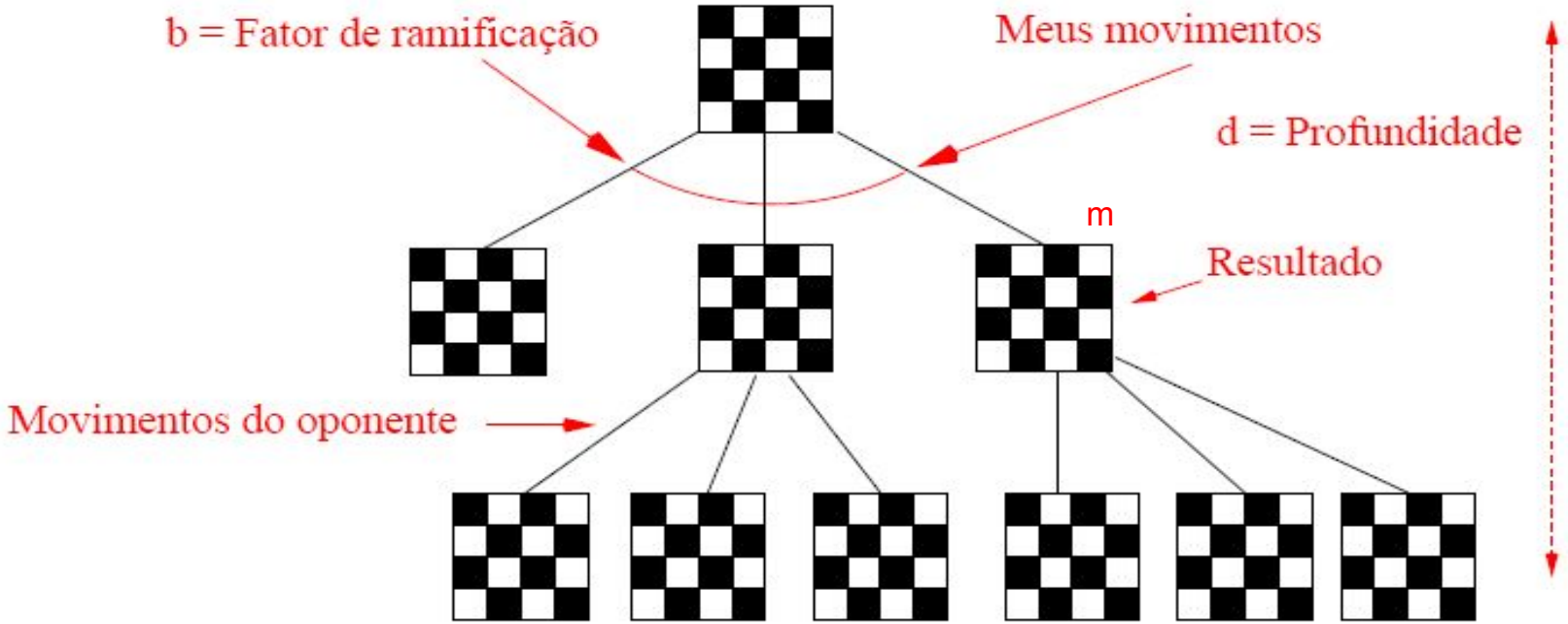
- MAX = X (jogador)
- MIN = O (adversário)

- Utilidade do ponto de vista de MAX
  - Valores altos bons para MAX e ruins para MIN

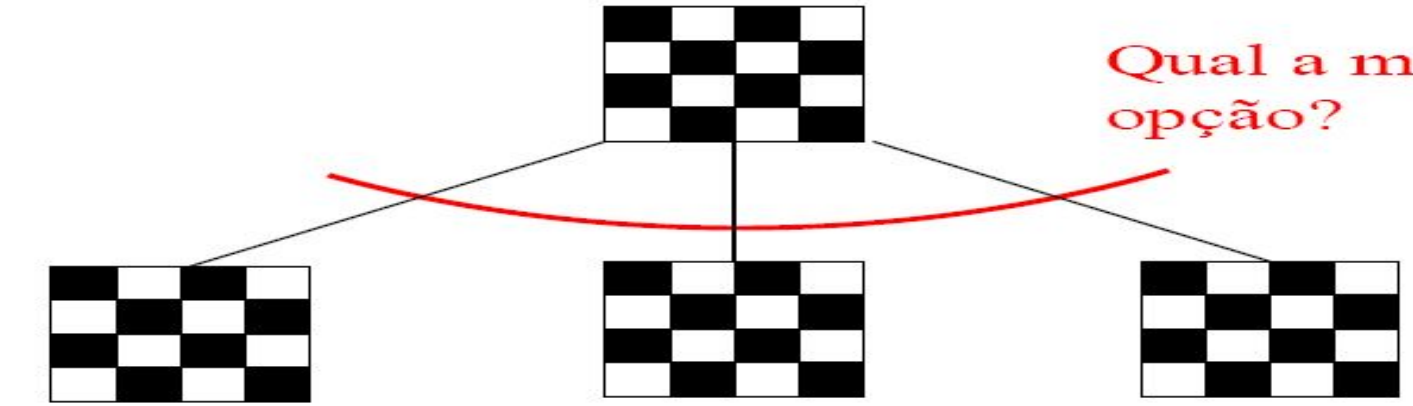
Estratégias de busca

- A solução ótima para MAX depende dos movimentos de MIN, logo:
  - MAX deve encontrar uma *estratégia* que especifique o **movimento de MAX no estado inicial**, e depois o **movimento de MAX nos estados resultantes** de cada movimento de MIN e assim por diante...
  - Estou procurando pelo próximo movimento
    - Esperando que me leve à vitória
    - Meus melhores movimentos dependem dos movimentos do adversário

Jogos


b = Fator de ramificação
Meus movimentos
d = Profundidade
m
Resultado
Movimentos do oponente

Jogos

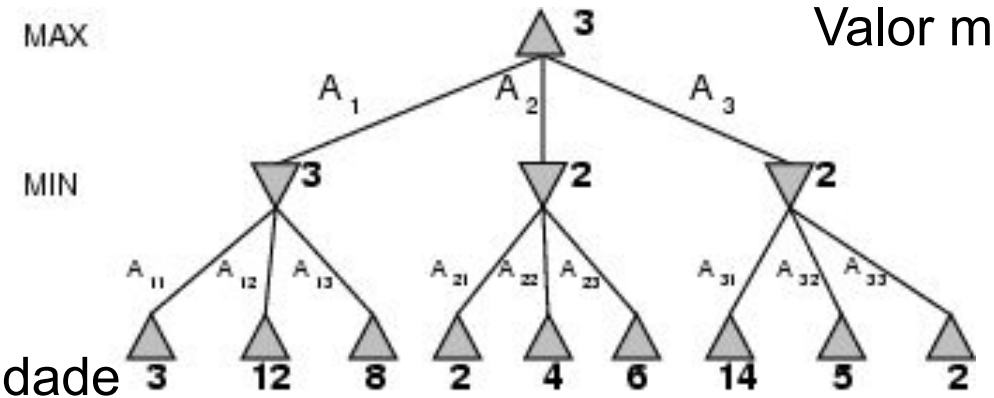


Pontuação

(Probabilidade de vencer a partir deste estado)

Estratégias de busca

- Jogo
  - Deve-se levar em conta o caráter competitivo
    - O que MAX fará é determinado também por MIN
  - Supor o jogo:

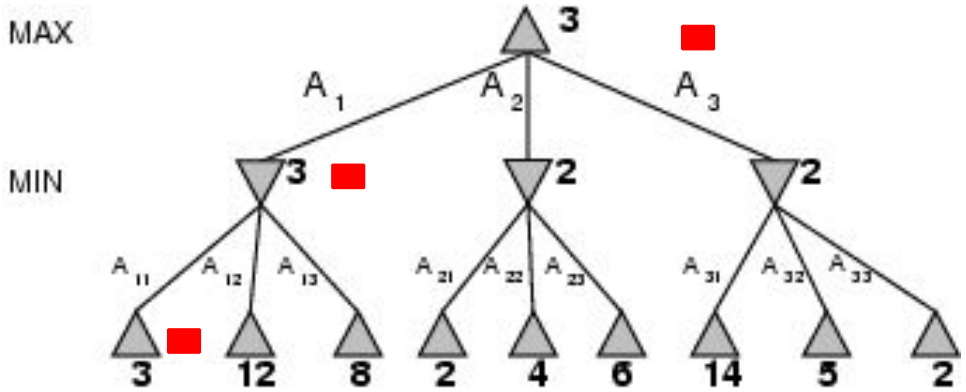


Valor minimax

utilidade

Estratégias de busca

- MAX prefere mover para estado de minimax máximo
  - E MIN prefere valor mínimo
    - Dada uma árvore de jogo, a estratégia ótima pode ser determinada a partir do valor ***minimax*** de cada nó



Ideia: maximizar a utilidade (ganho) supondo que o adversário vai tentar minimizá-la

Minimax faz busca cega em profundidade

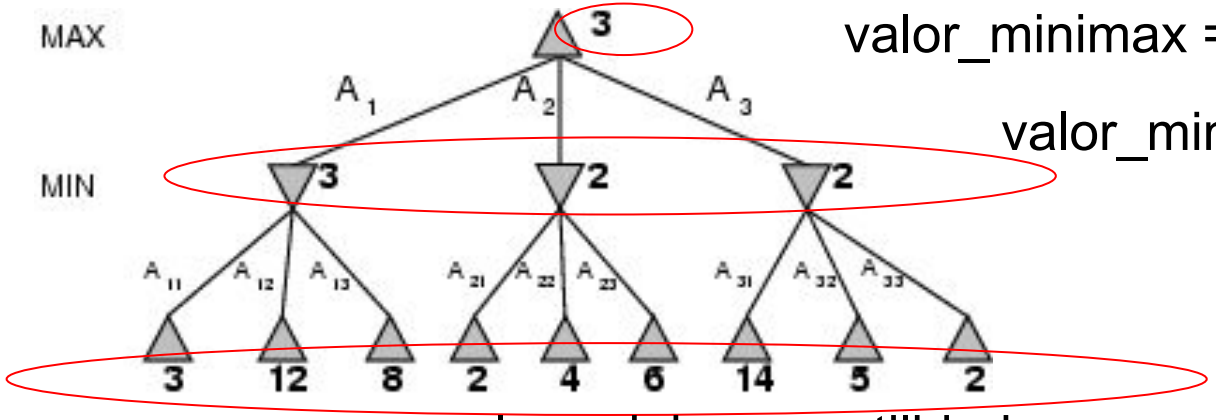
Valor minimax

- Valor minimax de um nó é a utilidade de MAX no estado correspondente
  - Supondo que ambos jogadores têm desempenho ótimo desse estado até fim do jogo

$$
\text{Valor\_minimax}(n) = \begin{cases} \text{utilidade}(n) & \text{se } n \text{ é estado terminal} \\ \max_{S \in \text{Sucessor}(n)} \text{Valor\_minimax}(n) & \text{se } n \text{ é um nó MAX} \\ \min_{S \in \text{Sucessor}(n)} \text{Valor\_minimax}(n) & \text{se } n \text{ é um nó MIN} \end{cases}
$$

O valor minimax (para MAX) é a utilidade de MAX para cada estado, ***assumindo que MIN escolhe os estados mais vantajosos*** para ele mesmo (i.e. os estados com menor valor utilidade para MAX)

Valor minimax



valor_minimax = max sucessores

valor_minimax = min sucessores

valor_minimax = utilidade

A ação *A1* é a escolha ótima para MAX, porque leva ao sucessor com mais alto valor minimax

É a melhor jogada para um jogo determinístico assumindo o melhor oponente

Algoritmo minimax

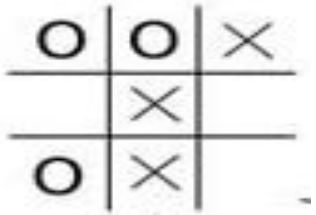
- Exploração completa em profundidade da árvore de jogo
  - Calcula recursivamente valores de utilidade
  - E toma decisão com base nesses valores

Seja m a profundidade máxima da árvore
b o número de movimentos possíveis em cada ponto

Complexidade de tempo = $O(b^m)$
Complexidade de espaço = O(bm)

Exercício

Considere que, em um jogo da velha, chegou-se ao seguinte estado:



O próximo movimento é do X, que será assumido como o MAX (*maximizer*).

a) Monte a árvore de jogo a partir desse estado e diga quais são os valores minimax dos nós.

b) Qual o melhor movimento para MAX?

Desempenho Minimax

- Completo?
- Ótimo?
- Complexidade de tempo? $O(b^m)$
  - m = profundidade máxima
  - b = movimentos válidos em cada estado
- Complexidade de espaço? O(bm)
  - Com busca em profundidade

Para xadrez, b ≈ 35, m ≈100
em jogos “razoáveis”
⇒ Solução exata é inviável

Desempenho Minimax

- Completo? Sim
  - Se árvore é finita
- Ótimo? Sim
  - Contra um oponente ótimo
- Complexidade de tempo? $O(b^m)$
  - m = profundidade máxima
  - b = movimentos válidos em cada estado
- Complexidade de espaço? O(bm)
  - Com busca em profundidade
  - Ou O(m) se gerar um sucessor por vez

Para xadrez, b ≈ 35, m ≈100
em jogos "razoáveis"
⇒ Solução exata é inviável

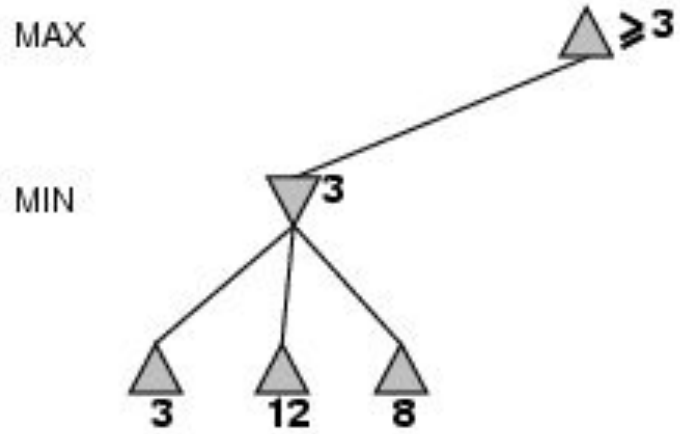
Poda alfa-beta

- Minimax é **impraticável** para muitos jogos
  - Número de estados do jogo a examinar é **exponencial**
  - É possível **reduzir expoente**
    - **Poda**
      - Deixar de considerar grandes partes da árvore de jogo
      - Podando **ramificações que não influenciam a decisão final**

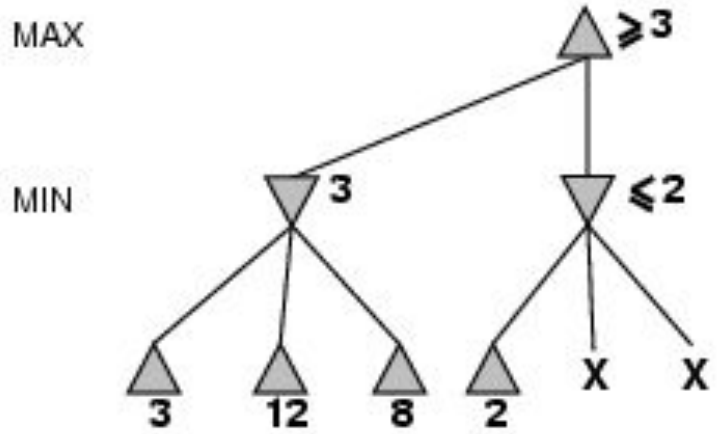
calcular a decisão correta sem examinar todos os nós da árvore (evitar gerar toda a árvore, analisando que subárvores não influenciam na decisão)

retorna o mesmo que minimax, porém sem percorrer todos os estados.

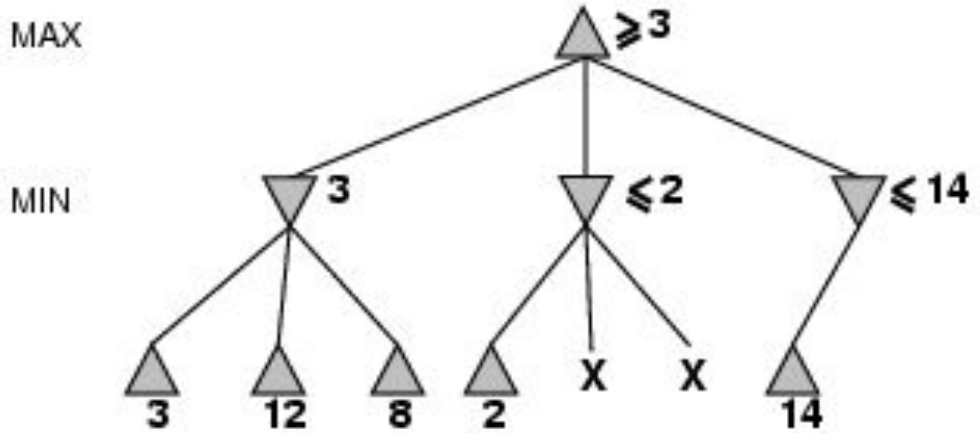
Poda alfa-beta



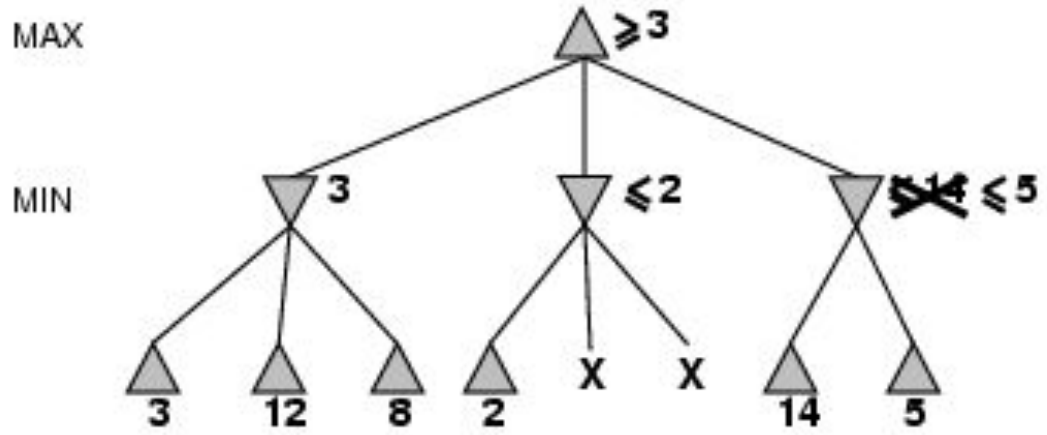
Poda alfa-beta



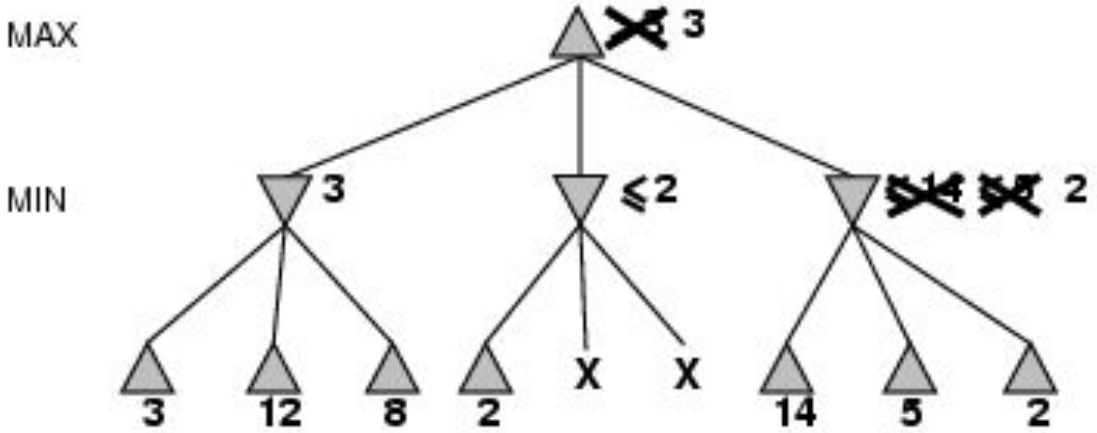
Poda alfa-beta



Poda alfa-beta



Poda alfa-beta



Busca alfa-beta

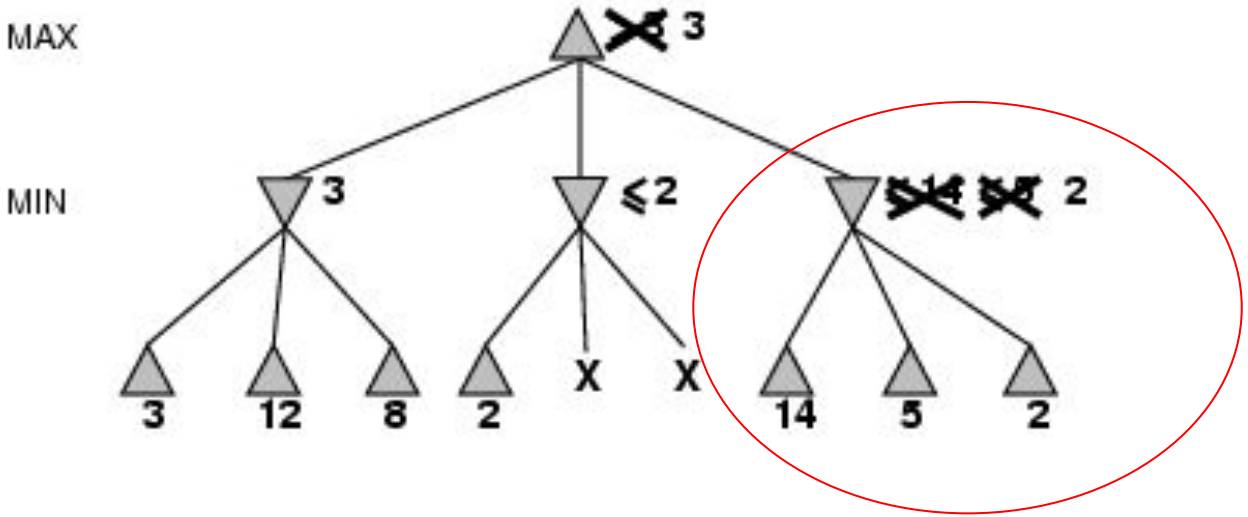
- Alfa
  - Valor da melhor escolha encontrado em qualquer ponto ao longo do caminho de busca para MAX
    - Valor mais alto
- Beta
  - Valor da melhor escolha encontrado em qualquer ponto de escolha do caminho para MIN
    - Valor mais baixo

Busca alfa-beta

- Atualiza valores de alfa e beta à medida que prossegue em profundidade
  - Poda ramificações tão logo sabe que o valor de um nó corrente é pior que o valor corrente de alfa ou beta para MAX ou MIN

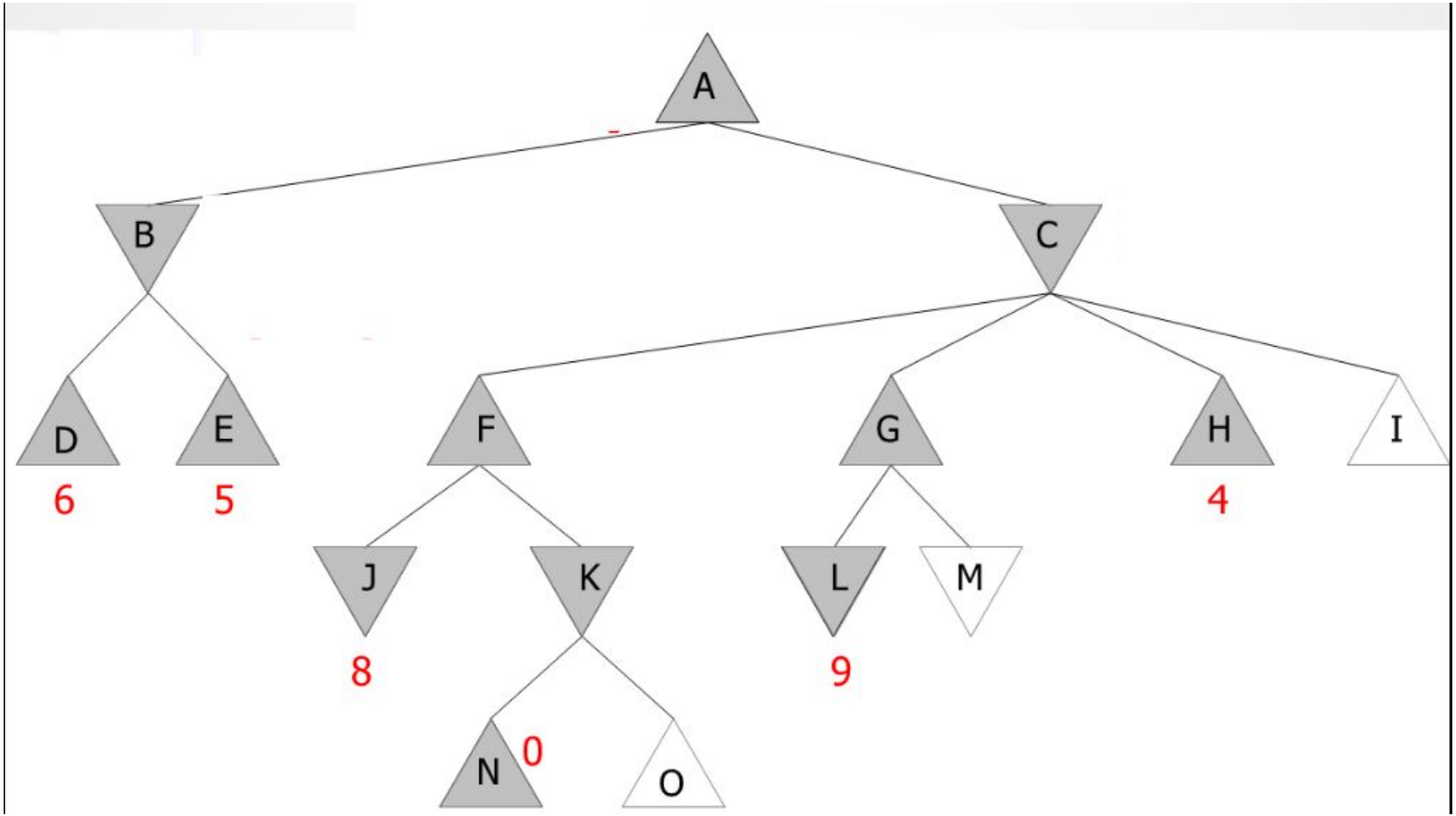
Busca alfa-beta

- Efetividade **depende da ordem** em que sucessores são examinados



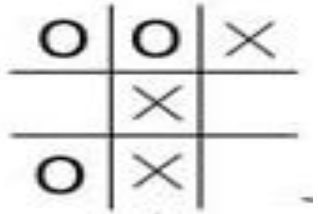
Se terceiro sucessor tivesse sido gerado primeiro, outros dois poderiam ter sido podados

Busca alpha-beta



Exercício

Considere o exercício anterior do jogo da velha, em que parte-se do estado inicial:



O próximo movimento é do X, que será assumido como o MAX (*maximizer*).

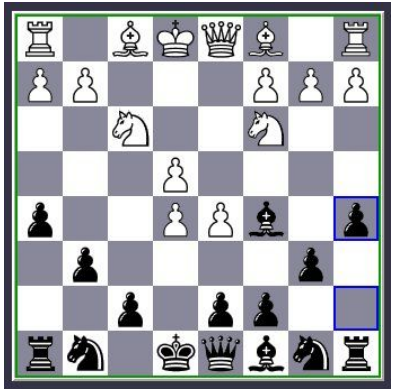
Se você usasse a poda alfa-beta, daria para melhorar a busca em relação ao MiniMax feito anteriormente?

Busca alfa-beta

- Supondo que utiliza melhor ordem
  - Alfa-beta examina $O(b^{m/2})$ nós para escolher melhor movimento
    - Contra $O(b^{m})$ do minimax
    - Pode uma árvore de profundidade duas vezes maior no mesmo tempo
- Examinando em ordem aleatória
  - $O(b^{3m/4})$

Busca alfa-beta

- Exemplo de ordenação para xadrez:
  - 1) capturas
  - 2) ameaças
  - 3) movimentos para frente
  - 4) movimentos para trás



Propriedades alfa-beta

- Poda não afeta resultado final
- Boa ordem de movimento melhora efetividade da poda
- Com “ordem perfeita”, complexidade de tempo = $O(b^{m/2})$
  - Dobra profundidade da busca

Referências

- Cap 5 Livro Russel e Norvig
- Slides:
  - Ana Carolina Lorena, Unifesp
  - Richard Khoury, University of Waterloo
  - Cornell University
  - Fabrício Barth, SENAC
  - FEI
  - UFPE