Inteligência Artificial

Estratégias de Busca Parte 5

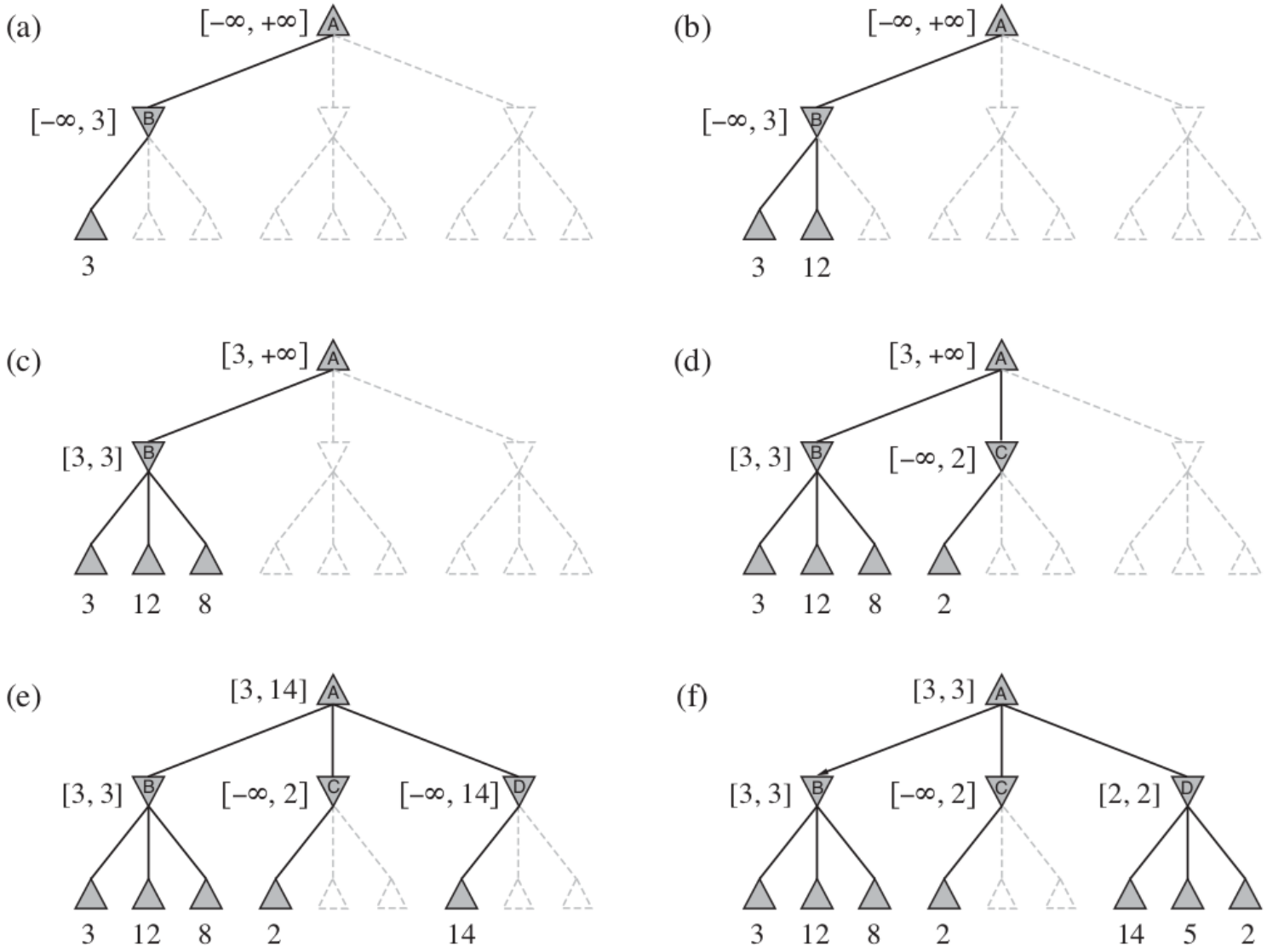
Prof. Jefferson Morais

Busca Com Competição - Poda Alfa-Beta

- **Minimax** é impraticável para muitos jogos
  - O número de estados de jogo que a busca tem de examinar é exponencial em relação ao número de movimentos.
  - É possível reduzir o expoente efetivamente pela metade e ainda encontrar a decisão ótima
  - **Artifício**: calcular a decisão minimax correta sem examinar todos os nós na árvore de jogo

- Poda (Prunning)
  - Deixar de considerar grandes partes da árvore de jogo
  - Elimina ramificações que não influenciam a decisão final
  - Vamos analisar o algoritmo poda alfa-beta

Busca Com Competição - Poda Alfa-Beta



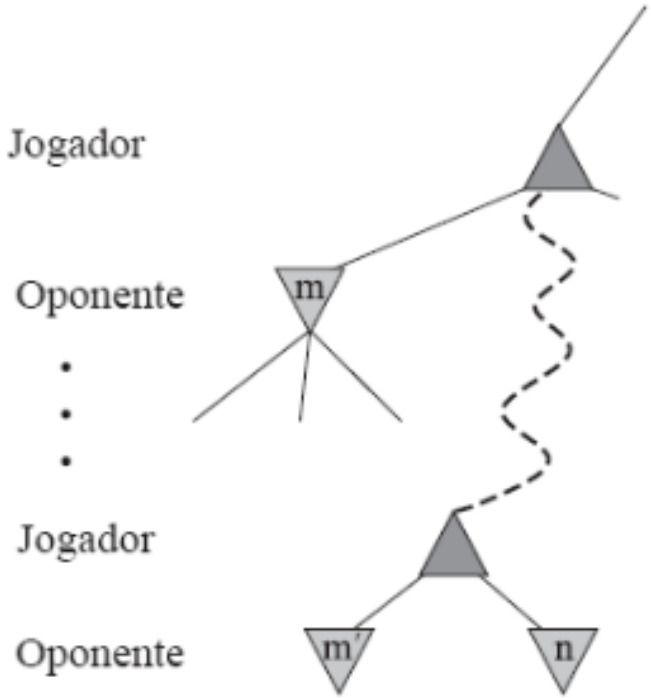
Busca Com Competição - Poda Alfa-Beta

- O exemplo pode ser visto como uma simplificação da fórmula de VALOR-MINIMAX
  - Sejam **x** e **y** valores dos dois sucessores não avaliados do nó **C**
  - Seja **z** o mínimo entre **x** e **y**
  - Então o valor do nó raiz é dado por

$$\begin{aligned}
\textsc{Minimax}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\
&= \max(3, \min(2,x,y), 2) \\
&= \max(3,z,2) \qquad \text{where } z = \min(2,x,y) \leq 2 \\
&= 3.
\end{aligned}$$

Busca Com Competição - Poda Alfa-Beta

- Generalizando
  - Considere o nó **n**
  - Se m ou m’ é melhor que n para o jogador, nunca chegaremos a n em um jogo



Busca Com Competição - Poda Alfa-Beta

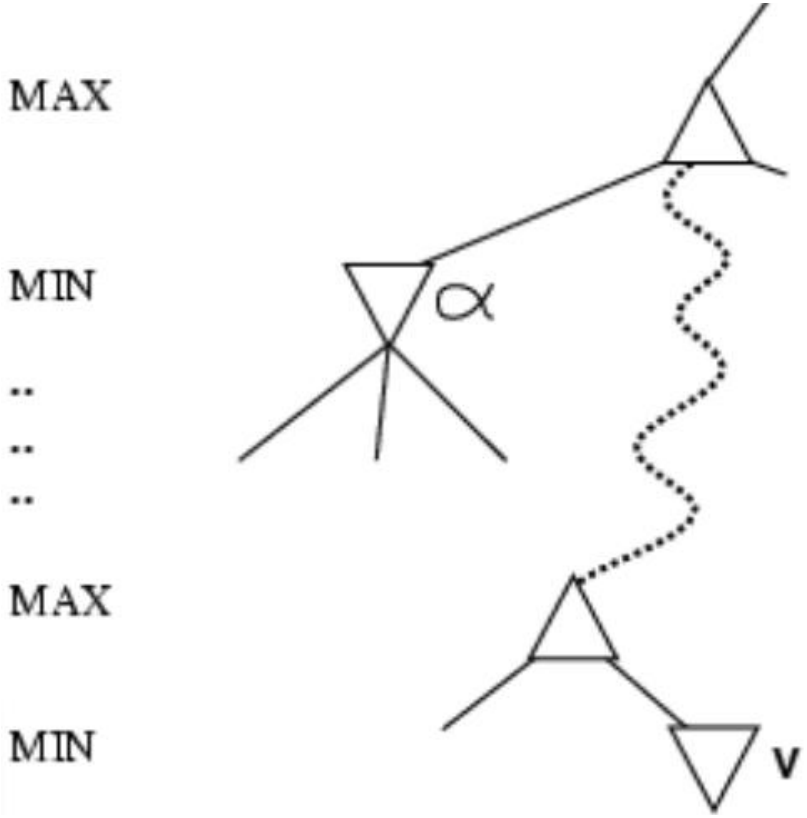
- Poda alfa-beta recebe esse nome por causa de seus parâmetros
  - **Alfa (α): valor mais alto**
    - o valor da melhor escolha encontrado em qualquer ponto de escolha ao longo do caminho para **MAX**
  - **Beta (β): valor mais baixo**
    - o valor da melhor escolha encontrado em qualquer ponto de escolha ao longo do caminho para **MIN**

- Os valores de **α** e **β** são atualizados à medida que o algoritmo prossegue

- O algoritmo poda as ramificações restantes em um nó tão logo se sabe que o valor do nó corrente é pior que o valor corrente de **α** ou **β**

Busca Com Competição - Poda Alfa-Beta

- **α** é o valor da melhor escolha encontrado até então para a escolha de **MAX**
  - Se **v** é pior que **α**, **MAX** irá evitá-lo
  - Poda o ramo inteiro
- **Se já achou algo melhor, por que piorar?**



Busca Com Competição - Poda Alfa-Beta

function ALPHA-BETA-SEARCH($state$) **returns** an action
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in ACTIONS($state$) with value v

---

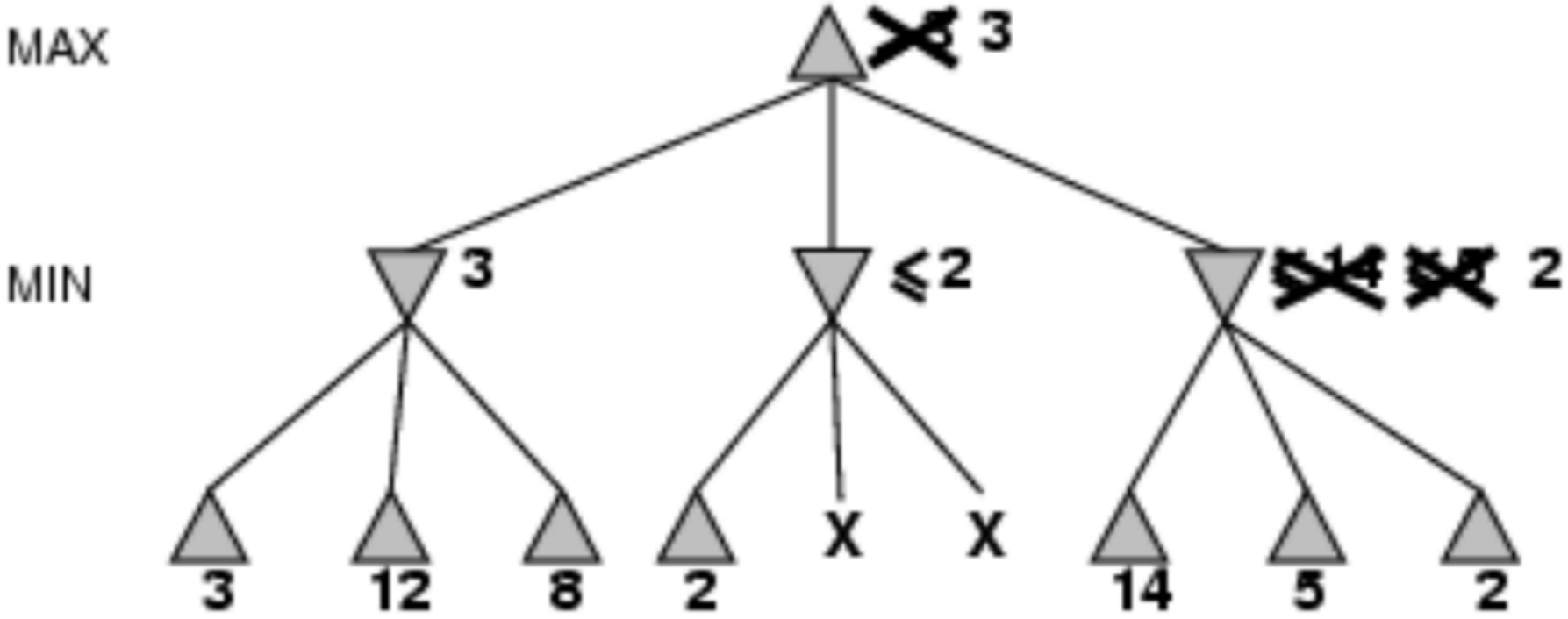
function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Busca Com Competição - Poda Alfa-Beta

- A efetividade da poda alfa-beta é altamente dependente da **ordem** em que os estados são examinados



Se terceiro sucessor (2) tivesse sido gerado primeiro, os outros dois (14, 5) poderiam ter sido podados, pois a solução nesse ramo já seria pior que a da ação que leva ao 3.

Busca Com Competição - Poda Alfa-Beta

- Se supusermos que isso pode ser feito perfeitamente
  - Alfa-beta precisará examinar apenas **$O(b^{m/2})$** nós para escolher o melhor movimento, contra **$O(b^m)$** para minimax
  - O fator de ramificação efetivo se tornará $\sqrt{b}$, em vez de **b**
    - No caso do xadrez, 6 em vez de 35
- Se sucessores forem examinados em ordem aleatória
  - O número total de nós examinados será cerca de **$O(b^{3m/4})$** para um valor moderado de **b**

Busca Com Competição - Poda Alfa-Beta

- Considerações sobre o algoritmo
  - 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
- Curiosidades
  - **Xadrez**
    - **Minimax**: 5 jogadas à frente
      - Humano médio: 6 a 8
    - **Alfa-beta**: 10 jogadas à frente
      - Desempenho de especialista

Curiosidade: filme que conta a história da disputa entre o Kasparov vs Deep Blue "Game Over: Kasparov and the Machine"

Decisões imperfeitas em tempo real

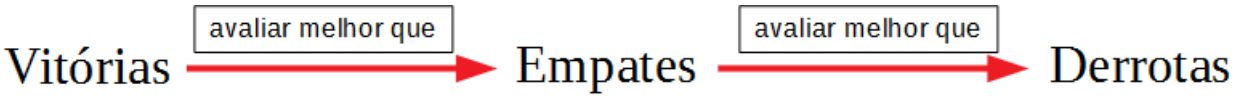
- O algoritmo minimax gera o espaço de busca do jogo inteiro
- O algoritmo alfa-beta poda grandes partes desse espaço
  - Ainda assim, o algoritmo faz uma busca até os estados terminais, pelo menos para uma parte do espaço de busca
  - Em problemas reais, essa profundidade não é prática (limite de tempo)
- São ineficientes para jogos que possuam muitos passos para os estados terminais (i.e., quase todos os jogos interessantes!)
- Solução
  - Substituir a função utilidade por uma função de avaliação (**heurística**), a qual fornece uma **estimativa** da utilidade esperada da posição

Funções de Avaliação

- Uma função de avaliação retorna uma estimativa da utilidade esperada do jogo

- Nota: o desempenho do algoritmo está diretamente relacionado com a função de avaliação projetada

- Boas práticas para projetar boas funções de avaliação
  - Deve ordenar os estados terminais do mesmo modo que a verdadeira função utilidade



  - A computação não deve demorar muito tempo
  - No caso de estados não terminais, a função de avaliação deve estar fortemente relacionada com as chances reais de vitória