Inteligência Artificial

Busca competitiva

👨🏼‍🏫 Prof. Chauã Queirolo

👨🏼‍💻 https://github.com/chaua/inteligencia-artificial

Sumário

- Busca competitiva

Introdução

Introdução

- Busca sem informação
  - Baseada somente na organização de estados e a sucessão entre eles
- Busca com informação
  - Utiliza, também, informações a respeito do domínio do problema
  - Função de avaliação: função de custo e função heurística
- Busca local
  - Não interessam as ações para chegar no estado objetivo

Introdução

- Alguns problemas envolvem mais de um agente
  - Sequência de decisões de agentes que **controlamos**
  - Outras decisões de agente que **não controlamos**
  - Troca de objetivos
  - Objetivos conflitantes
- Busca competitiva
  - Considera que há oponentes hostis e imprevisíveis
  - Exemplo: jogos

Jogos

Jogos

- Jogos são populares em IA
  - Jogos estão relacionados com a inteligência
  - Abstraem detalhes complicados
  - São simples de serem representados
  - Restritos a um pequeno número de ações
  - Resultados são definidos por regras precisas
- “Os jogos estão para a IA assim como as corridas estão para os projetos de automóveis” (S. Russel)

Jogos

- Em IA, jogos são de um tipo específico
  - Consideram a **alternância** entre dois jogadores (turnos)
  - Jogos **determinísticos**: cada ação leva a um resultado conhecido
  - **Soma zero**: o ganho de um jogador é a perda do outro
  - **Informações perfeitas**: todos os jogadores conhecem o estado atual do jogo

Xadrez

- Complicado demais para ser resolvido exatamente:
  - Fator de ramificação médio de 35
  - Duram em média 50 movimentos por jogador
  - Árvore de busca tem tipicamente 35^{100} ou 10^{154} nós
  - Grafo possui 10^{40} nós distintos
- Por comparação:
  - 10^{154} é mais do que a quantidade de átomos do universo
  - 200 milhões posições/segundo, mais de 10^{100} anos de processamento
  - Universo tem 10^{10} anos

Problema de busca em jogos

- Tamanho (espaço de busca) + limitação de tempo
- Adversário é imprevisível
  - Cada agente tem que levar em consideração todos os movimentos possíveis de oponente e ter um plano de contingência para eles
- Restrição sobre recursos
  - Difícil encontrar a meta
  - O agente tem que tomar uma decisão, mesmo que não seja ótima (decisão aproximada)

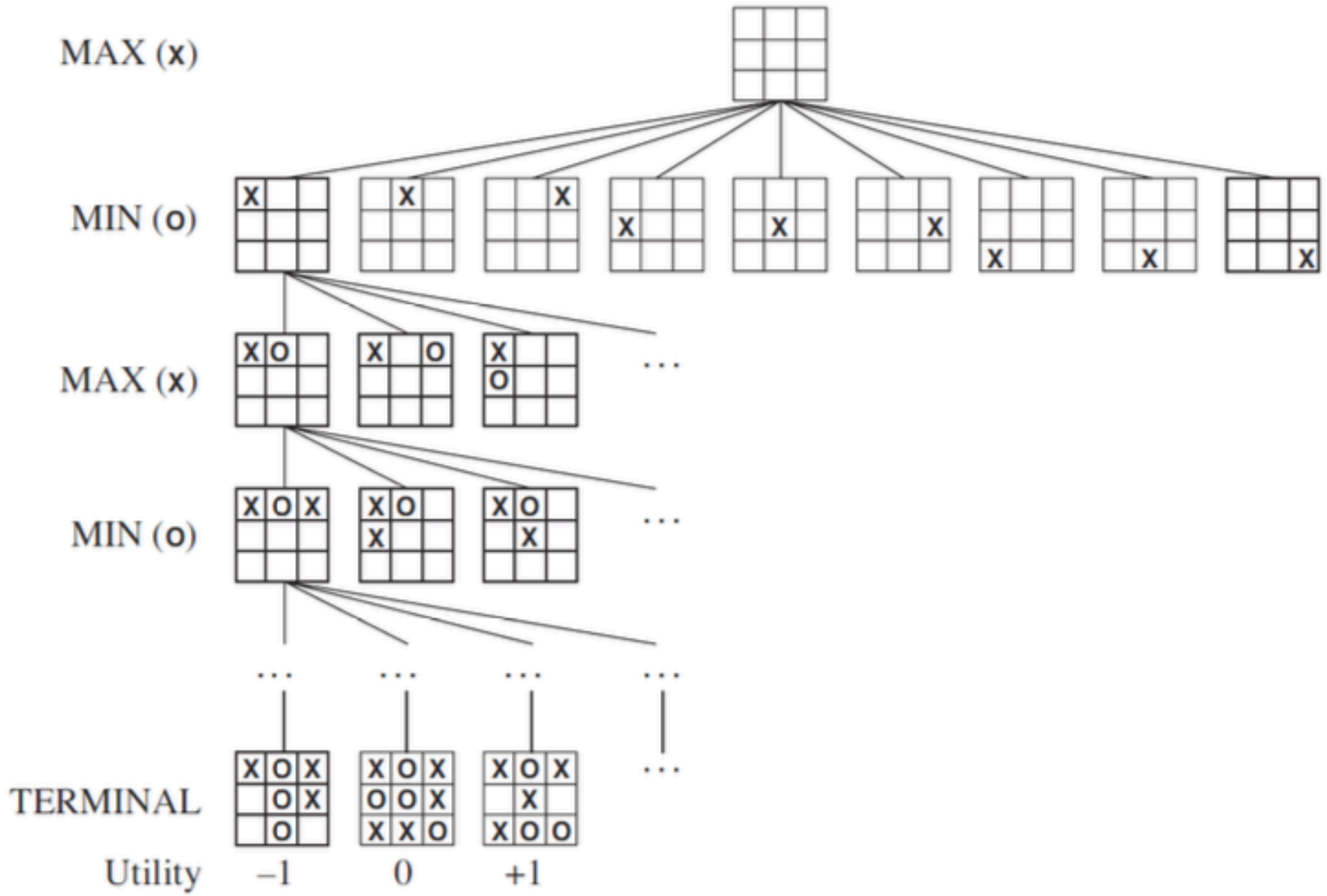
Problema de busca em jogos

- **Estado inicial**
  - Posição do tabuleiro e identifica o jogador que fará o movimento
- **Modelo de transição**
  - Indica um movimento válido e o estado resultante
- **Teste de objetivo**
  - verifica se o jogo terminou
- **Função de avaliação**
  - atribui um valor numérico aos estados terminais
  - Ex: Xadrez: (1, -1 ou 0); Gamão: [-192 a 192]

Problema de busca em jogos

- **Soma zero**
  - uma única função de utilidade é utilizada para determinar a qualidade de uma posição para ambos os jogadores
- Um jogador tentará **MAXIMIZAR** a função de utilidade
- O oponente tentará **MINIMIZAR** a função de utilidade

Jogo da velha



Como jogar

- Uma maneira de jogar consite em:
  - Considerar todos os movimentos legais que podem ser realizados
  - Computar a nova posição resultante de cada movimento legal
  - Avaliar cada posição resultante, decidir e executar o melhor movimento
  - Esperar pelo movimento do oponente e repetir o processo
- MAX precisa de uma estratégia que especifica:
  - Movimento inicial (assumimos que MAX inicia o jogo)
  - Movimentos possíveis de MAX para cada movimento de MIN
- Por sua vez MIN precisa de:
  - Movimentos possíveis para cada possível movimento de MAX

Algoritmo Minmax

Minimax

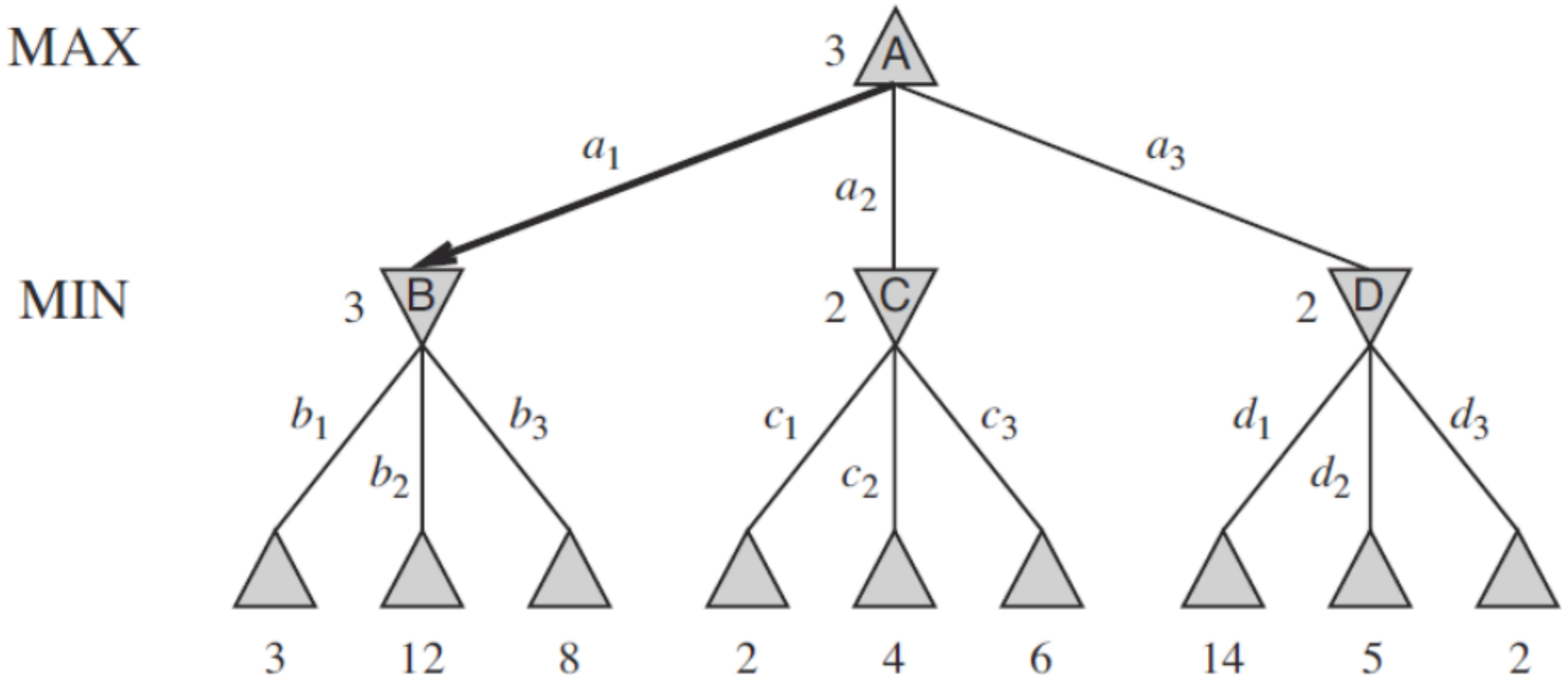
- 1944 - John von Neumann propõe um método de busca (Minimax) para jogos de soma zero que **maximiza** a sua posição enquanto **minimiza** a de seu oponente.
- **Função de utilidade**
  - Mede o quanto é boa a nossa posição
- Inicialmente, será um valor que descreve exatamente a nossa posição

Minimax

- Dada uma árvore de jogo
  - a estratégia ótima pode ser determinada a partir do valor minimax de cada nó
- O valor minimax para MAX
  - utilidade de MAX para cada estado, assumindo que MIN escolhe os estados mais vantajosos para ele mesmo

$$VALOR\_MINIMAX = \begin{cases} UTILIDADE(n), & \text{se } n \text{ for terminal} \\ max_{x \in succ(n)} VALOR\_MINIMAX(x), & \text{se } n \text{ é um nó de MAX} \\ min_{x \in succ(n)} VALOR\_MINIMAX(x), & \text{se } n \text{ é um nó de MIN} \end{cases}$$

Minimax



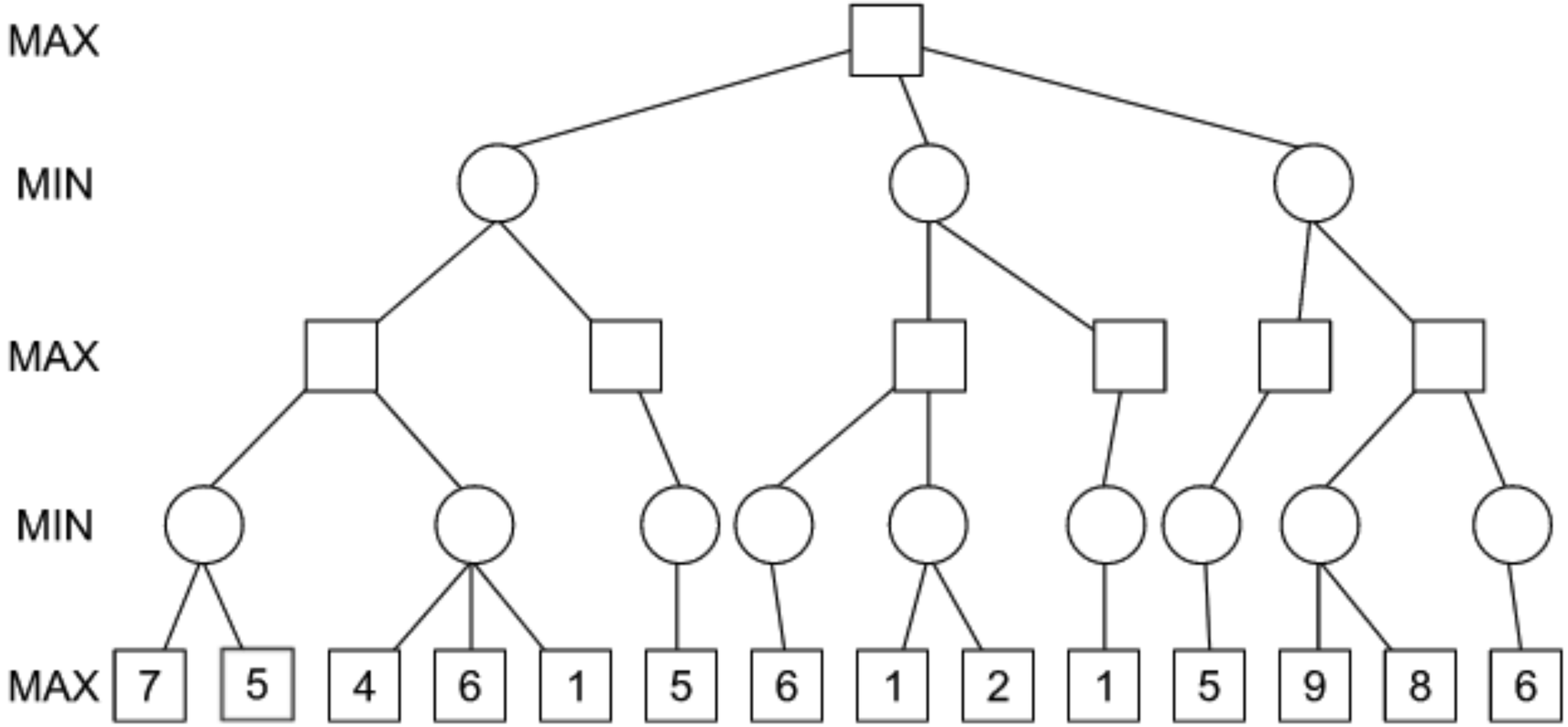
Minimax

- Valor Minimax de cada nó assume que ambos os jogadores jogam de forma ótima
- Mas assumir que MIN joga otimamente é uma boa estratégia?
  - É uma análise de pior caso
  - Se MIN não joga otimamente, MAX vai ter resultados ainda melhores
  - Outras estratégias contra oponentes sub-ótimos pode dar melhores resultados
  - Essas estratégias desempenham pior contra oponentes ótimos

Minimax

1. Gerar a árvore do jogo
2. Calcular a função de utilidade de cada estado terminal
3. Propagar a utilidade dos nós terminais para níveis superiores:
   - se no nível superior é a vez de MIN jogar, escolher o menor valor
   - se no nível superior é a vez de MAX jogar, escolher o maior valor
4. No nodo raiz, MAX escolhe o movimento que leva ao maior valor

Exercício

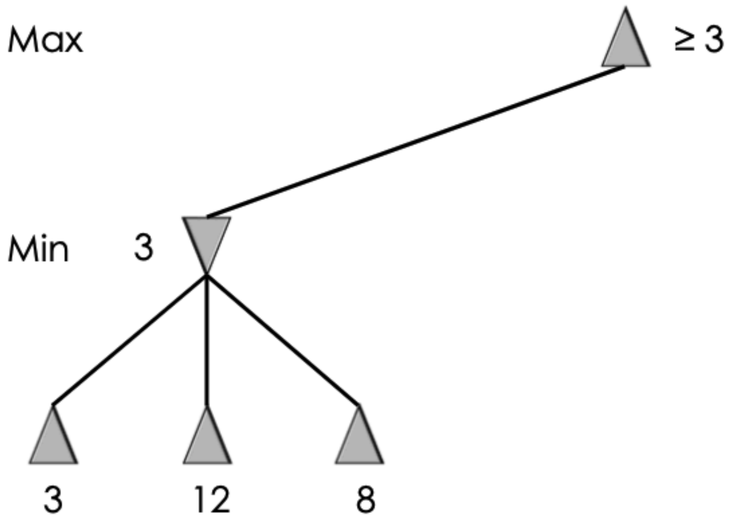


Poda $\alpha\beta$

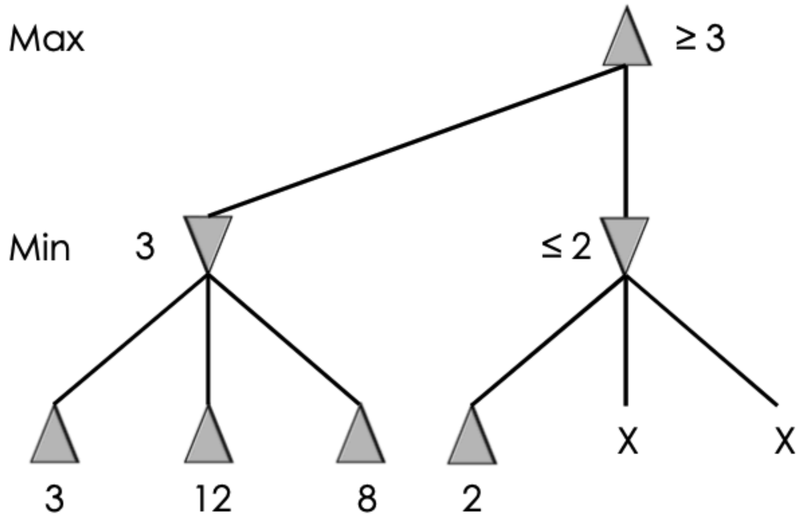
Poda $\alpha\beta$

- O algoritmo Minimax é feito em dois passos
  - Descida em profundidade e aplicação da heurística
  - Retropropagação dos valores
- O número de estados do jogo é exponencial em relação do número de movimentos
- Alguns ramos não precisam ser analisados, podendo ser podados

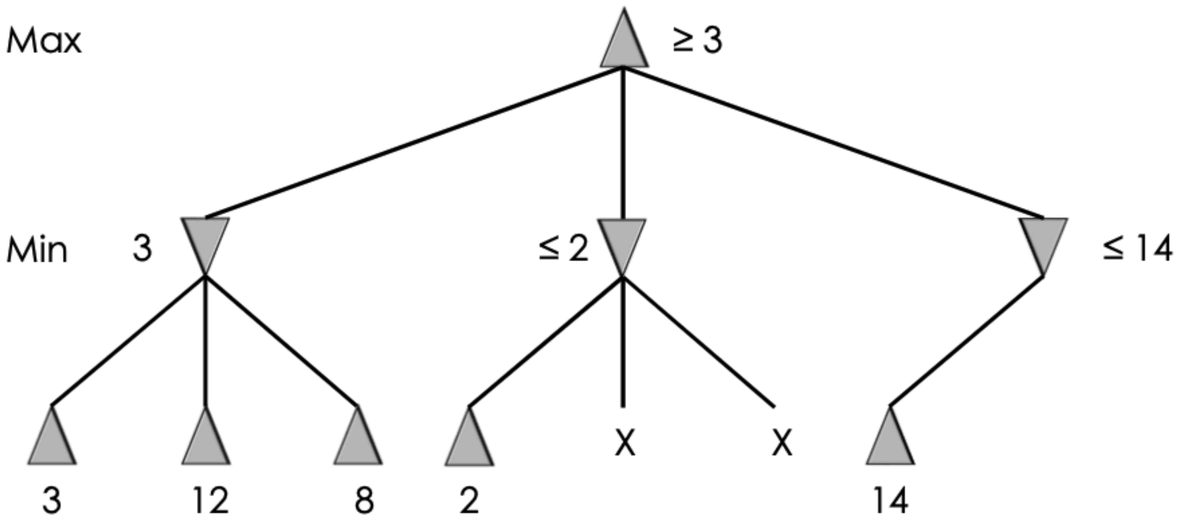
Poda $\alpha\beta$



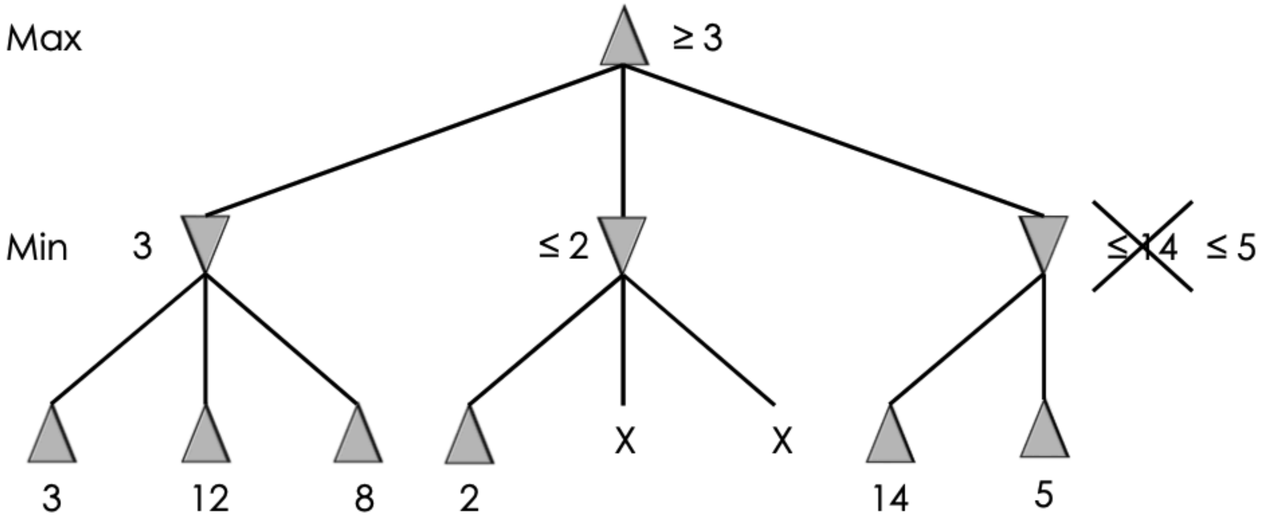
Poda $\alpha\beta$



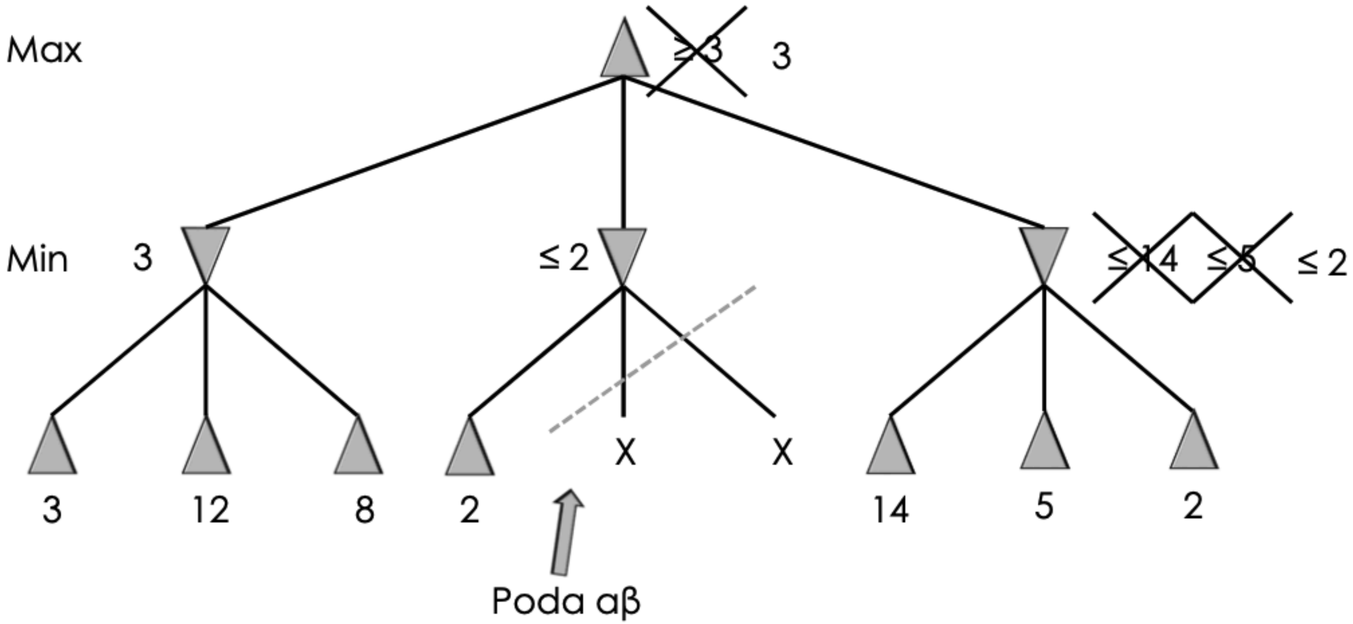
Poda $\alpha\beta$



Poda $\alpha\beta$



Poda $\alpha\beta$



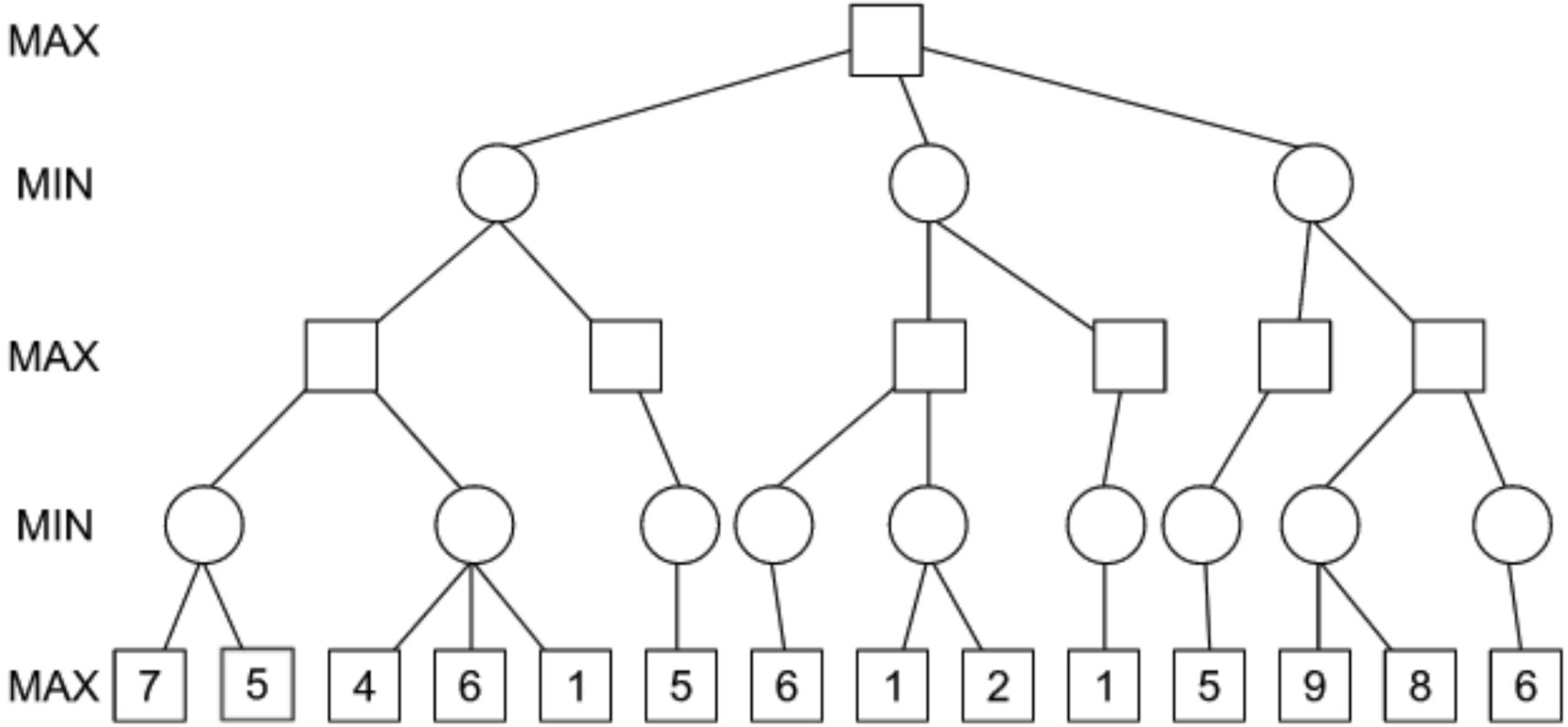
Poda $\alpha\beta$

- **Ideia principal:** não processar sub-árvores que não afetam o resultado
- Valor α
  - Valor da melhor escolha encontrada até agora no caminho para MAX (maior valor) e não pode decrescer
  - É usado nos nós MIN para decisão de poda
- Valor β
  - Valor da melhor escolha encontrada até agora no caminho para MIN (menor valor) e não pode aumentar
  - É usado nos nós MAX para decisão de poda

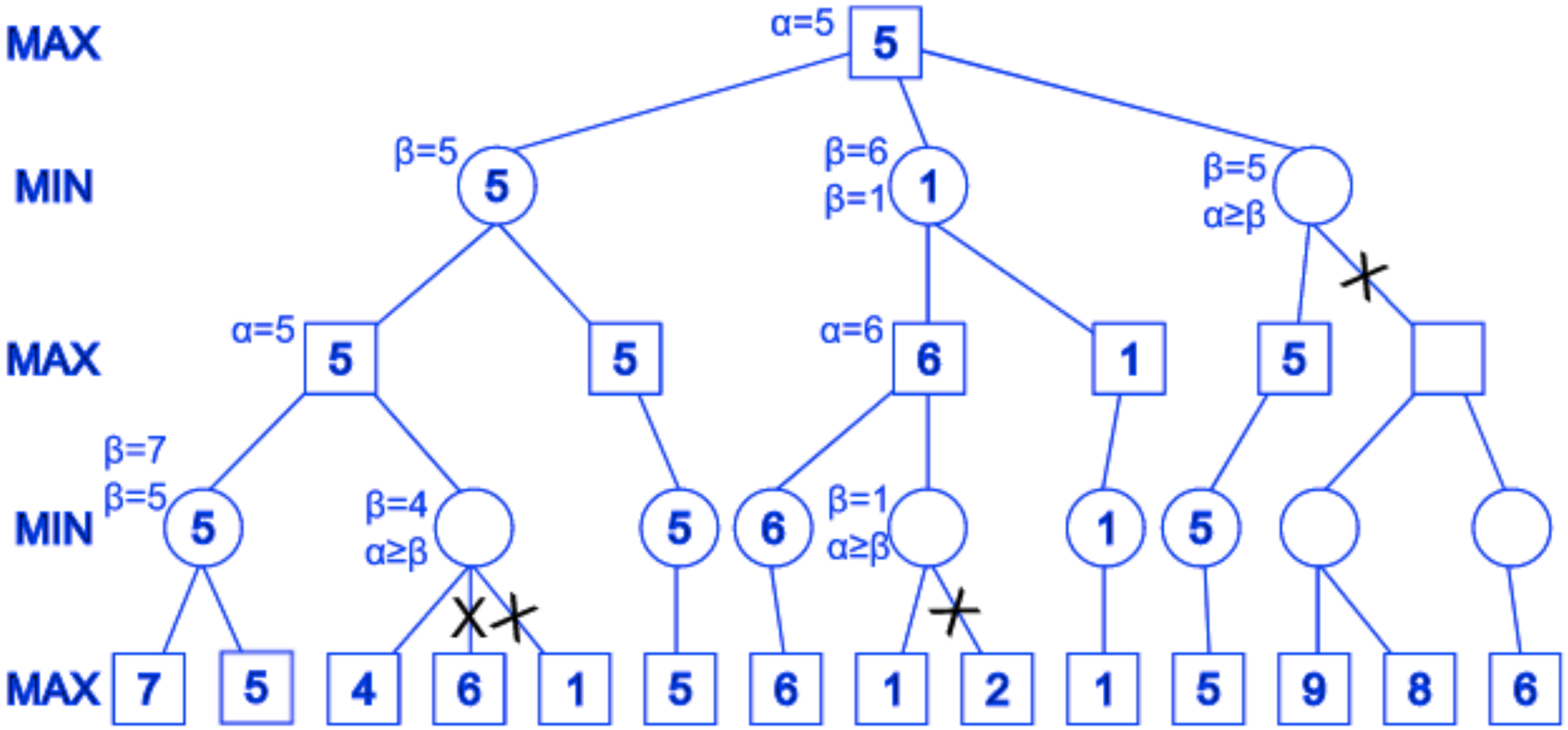
Poda $\alpha\beta$

1. Visite a árvore de busca em profundidade
2. Para cada nó n MAX, α(n) = máximo valor até agora
3. Para cada nó n MIN, β(n) = mínimo valor até agora
   - O valor de α começa em −∞ e somente incrementa
   - O valor de β começa em +∞ e somente decresce
4. Corte β: dado um nó n, corte a busca após n MAX se α(n) >= β(i) para algum nó i MIN ancestral de n
5. Corte α: corte a busca abaixo de um nó n MIN se β(n) <= α(i) para algum nó i MAX ancestral de n

Exercício



Exercício


MAX
MIN
MAX
MIN
MAX
α=5
5
β=5
5
β=6
β=1
1
β=5
α≥β
α=5
5
5
α=6
6
1
5
β=7
β=5
5
β=4
α≥β
5
6
β=1
α≥β
1
5
7
5
4
6
1
5
6
1
2
1
5
9
8
6

Referências Bibliográficas

Referências Bibliográficas

- S. J. Russell & P. Norvig. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 3rd edition, 2010.