Inteligência Artificial

Estratégias de Busca Parte 4

Busca com Competição

Prof. Jefferson Morais

Busca Com Competição

- Contexto
  - Estudamos anteriormente os **ambientes multiagentes**
    - cada agente precisa considerar as ações de outros agentes e o modo como essas ações afetam seu próprio bem-estar
  - Nesta aula abordaremos os ambientes **competitivos**
    - os objetivos dos agentes estão em **conflito**, dando origem a problemas de **busca competitiva** (**teoria de jogos**)
  - Teoria de jogos (**ramo da economia**) visualiza qualquer ambiente multiagente como um jogo
    - Em IA, os "jogos" **determinísticos completamente observáveis** em que dois agentes agem **alternadamente** e em que os valores de **utilidade** são simétricos.

Busca com competição

- Teoria dos Jogos (Jogos em IA)
  - Em IA, os jogos são determinísticos, de revezamento de dois jogadores, com informações perfeitas (totalmente observável)
  - A posição (favorável ou desfavorável) de um jogador num determinado instante (estado) do jogo pode ser medida por uma função de utilidade
  - Os valores de utilidade dos agentes no fim do jogo são iguais e opostos (simétricos): +1 (ganha), ou –1 (perde) (Soma zero).
  - O objetivo da busca competitiva é planejar com antecedência num mundo em que outros agentes estão fazendo planos contra nós

Busca com competição

- Teoria dos Jogos (Jogos em IA)
  - Entre os primeiros domínios de aplicação, pois:
    - É fácil representar o estado de um jogo
    - Em geral, os agentes estão restritos a um pequeno número de ações com resultados definidos por regras precisas
    - Constituem uma tarefa estruturada em que é fácil medir o sucesso ou fracasso
    - Supunha-se que os jogos podiam ser solucionados por uma busca direta do estado inicial para a posição vencedora, sem grandes quantidades de conhecimento

  - Exceção aos jogos simulados
    - O futebol de robôs é um jogo físico, com descrições muito mais complicadas envolvendo ações bastante imprecisas

Busca com competição

- Teoria dos Jogos (Jogos em IA)
  - 1950 - Pioneiros da IA nos Jogos
    - Konrad Zuse, Claude Shannon, Norbert Wiener e Alan Turing começaram a estudar jogos como o xadrez
    - Turing escreveu sobre a possibilidade de uma máquina jogar xadrez antes mesmo dos computadores eletrônicos existirem
  - Máquinas Superam Humanos em Jogos Clássicos
    - **Xadrez**: Deep Blue, da IBM, derrotou Garry Kasparov, campeão mundial, em 1997. Embora Kasparov tenha vencido algumas partidas, essa foi uma vitória histórica para a IA.
    - **Damas e Othello**: As máquinas superaram os humanos definitivamente, com damas sendo resolvido (jogadas perfeitas de ambos os lados podem ser previstas)
    - **Gamão**: Programas de IA são competitivos e derrotaram campeões mundiais em várias ocasiões

Busca com competição

- Teoria dos Jogos (Jogos em IA)
  - Damas - Pioneira em Aprendizado de Máquina
    - Arthur Samuel (IBM): Desenvolveu, nos anos 1950, um programa de damas que utilizava **aprendizado de máquina** para melhorar sua função de avaliação
    - 1962: O programa de Samuel derrotou Robert Nealy, campeão humano, devido a um erro de Nealy
    - **Chinook** (Jonathan Schaeffer): Tornou-se campeão mundial em 1994, sendo o primeiro programa a conquistar um título mundial em um jogo competitivo contra humanos.
  - Exceção Notável – **Go**
    - Devido à complexidade do jogo e ao enorme número de combinações possíveis, Go era considerado um desafio para IA.
    - 2016: AlphaGo, da **DeepMind**, derrotou o campeão mundial Lee Sedol, utilizando redes neurais e aprendizado por reforço, revolucionando o campo da IA aplicada a jogos complexos

Busca Com Competição

- Exemplo: jogo de xadrez
  - Se um jogador ganha o outro perder (**soma zero**)
  - Totalmente observável (**informação imperfeita)**
  - Fator médio de ramificação: 35
  - Número Médio de jogadas: 50 para cada jogador
  - Assim, a árvore completa de busca de um jogo terá aproximadamente 35^{100} ou 10^{154} nós
  - Portanto, uma busca cega é inviável, mesmo para realizar o primeiro movimento
  - Se deve fazer o melhor uso possível do tempo disponível para uma jogada: tomar alguma decisão, mesmo que a jogada ótima não seja determinada em tempo.

Busca Com Competição

- Uma restrição comum: **limite de tempo**
  - Técnicas para escolher um bom movimento
    - Poda: ignorar partes da árvore de busca que não fazem diferença para a escolha final
    - Funções heurísticas: aproximar a utilidade de um estado sem realizar uma busca completa

- **Considere um jogo de dois jogadores**
  - Jogador **Max** e Jogador **Min**
  - **Max** faz o primeiro movimento
  - Depois há revezamento até o jogo terminar
  - No fim do jogo, os pontos são dados ao jogador vencedor
  - Penalidades são impostas ao jogador perdedor

- Formalmente, esse jogo pode ser descrito a seguir

Busca Com Competição

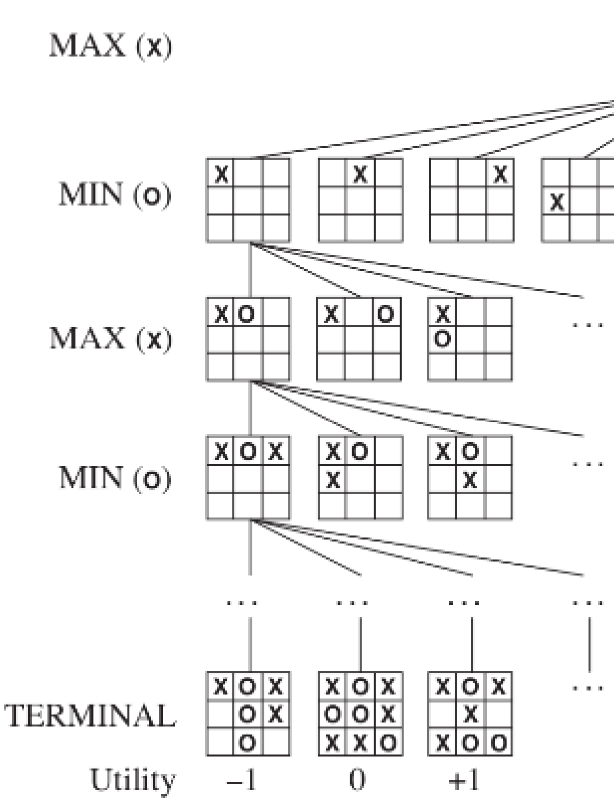
- Modelagem do jogo como um problema de busca
  - **Estado inicial**: S_0 (especifica como o jogo é criado no inicio)
  - **JOGADORES(s)**: indica qual jogador deve se mover
  - **AÇÕES(s)**: retornam o conjunto de movimentos válidos
  - **Modelo de transição RESULTADO(s, a)**: resultado de um movimento
  - **TESTE DE TÉRMINO(s)**: determina quando o jogo termina
  - **UTILIDADE(s, p)**: <u>função objetivo</u> retorna o valor numérico para o jogador p no estado terminal s

- Ex.: **Jogos de soma-zero**
  - Jogo de xadrez: win = 1, loss = 0, draw = ½
  - Jogo da velha: win = +1, loss = -1, draw= 0

- Vamos analisar o jogo da velha seguinte

Busca Com Competição

- **Árvore de jogo**: mostra todas as possibilidades do jogo (início ao fim)
  - 2 jogadores: **MAX = X** (jogador) e **MIN = O** (adversário)



Uma árvore de busca (parcial) para o jogo da velha. O nó superior é o estado inicial, e MAX faz o primeiro movimento colocando um X em um quadrado vazio. Após isso, são realizados movimentos alternados por MIN (O) e MAX(X), até alcançar um dos estados terminais, aos quais podem ser atribuídas utilidades de acordo com as regras do jogo.

Estados terminais: três símbolos em uma linha ou todos os quadrados são preenchidos

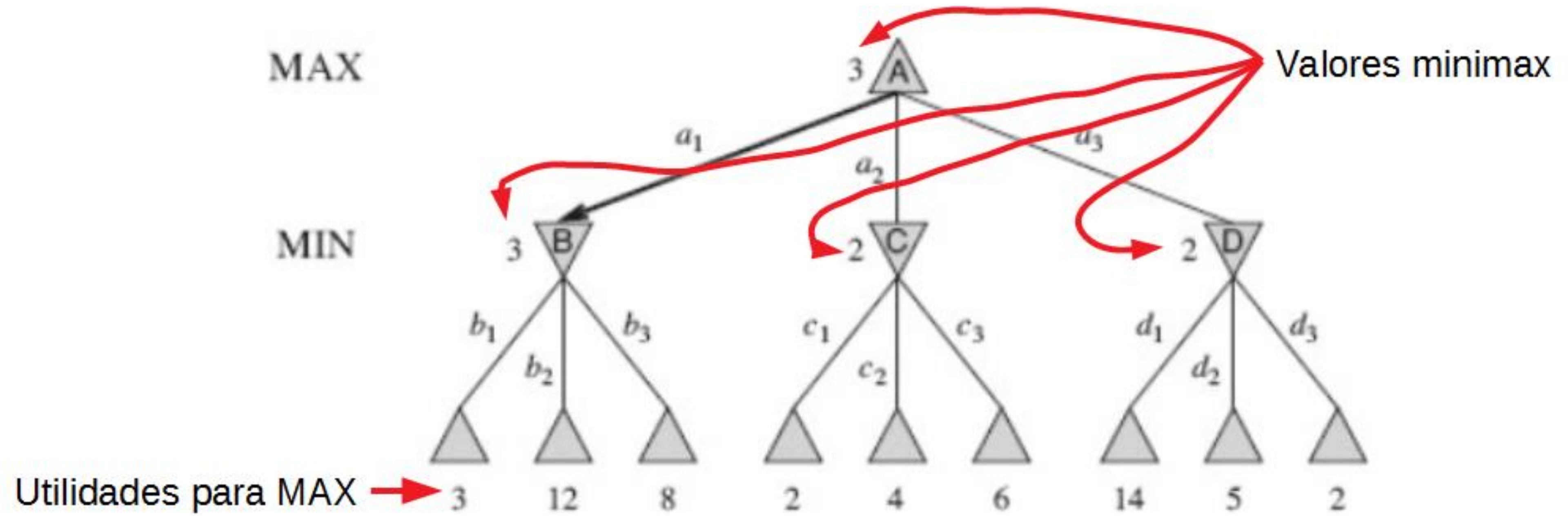
Jogo da velha (*Tic Tac Toe*)

Busca Com Competição

- Jogo da velha tem menos de **9! = 362.880 nós terminais**
- Xadrez tem mais de **10^{40} nós terminais**
- Portanto, a árvore de jogo é apenas uma construção teórica que não podemos perceber no mundo físico
- Usamos o termo árvore de busca para uma árvore que está sobreposta à árvore de jogo completa
- O que interessa de fato: examinar os nós o suficiente para permitir que um jogador determine que lance fazer

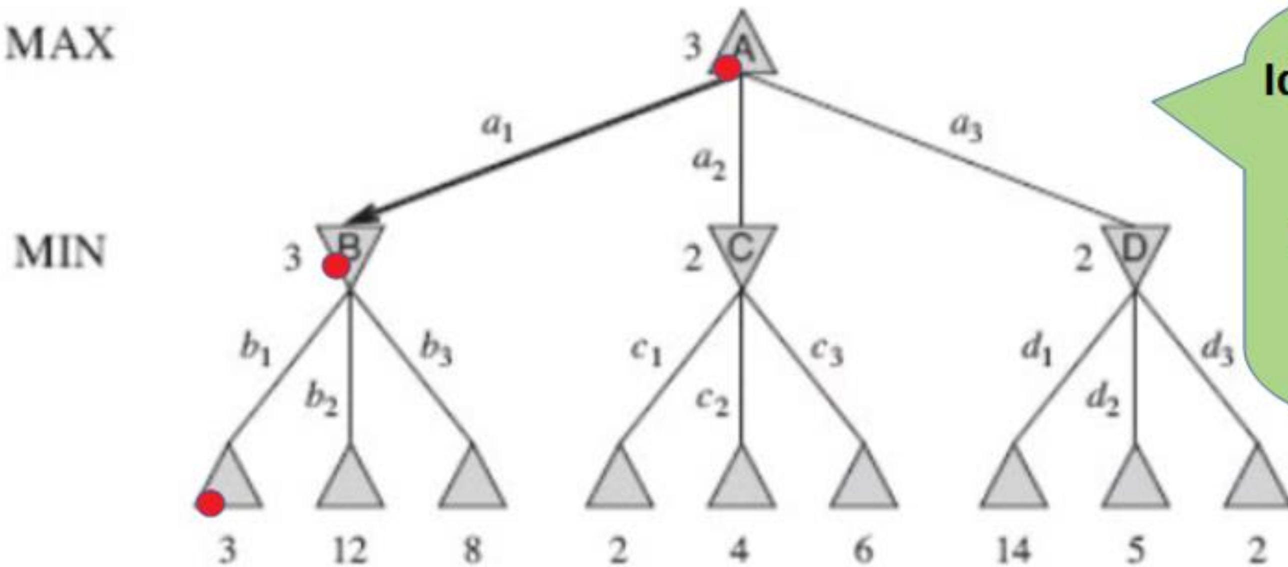
Busca Com Competição - Decisões Ótimas

- O jogo da velha é complexo para traçarmos a árvore jogo inteira, consideraremos um jogo trivial mais simples a seguir
  - Movimentos possíveis para **MAX** na raiz: $\mathbf{a_1}$, $\mathbf{a_2}$ e $\mathbf{a_3}$
  - Movimentos possíveis para **MIN** em resposta $\mathbf{a_1}$: $\mathbf{b_1}$, $\mathbf{b_2}$ e $\mathbf{b_3}$
  - O jogo termina depois de um movimento para cada jogador



Busca Com Competição - Decisões Ótimas

- **Como funciona o algoritmo minimax**
  - **MAX** prefere mover para estado de minimax máximo
  - **MIN** prefere valor minimax 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.

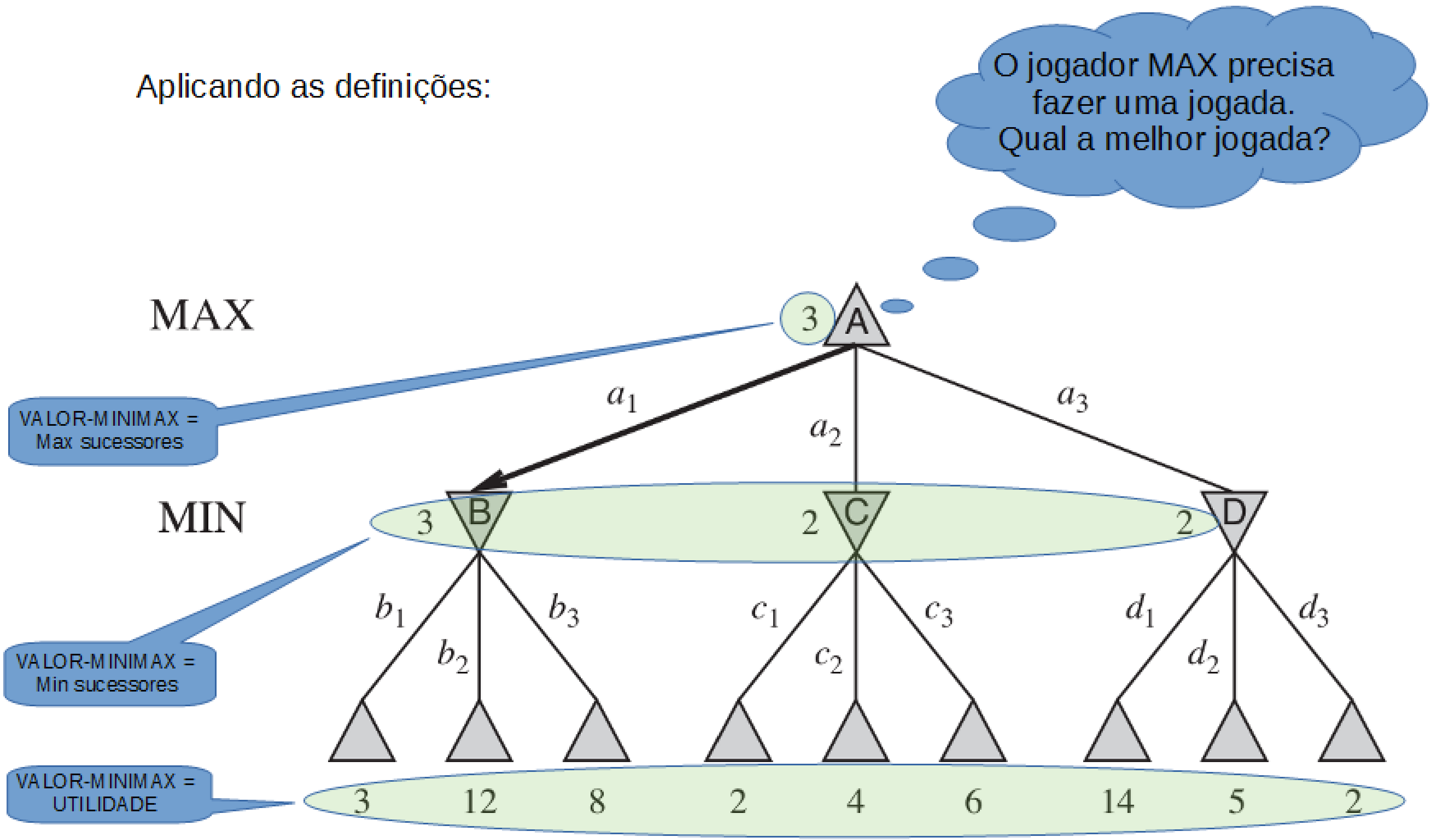
Busca Com Competição - Decisões Ótimas

- O **valor minimax** de um **nó** é a utilidade de se encontrar (para MAX) no estado correspondente
- **MAX** preferirá se mover para um estado de valor máximo, enquanto **MIN** preferirá um estado de valor mínimo
- Essa definição de jogo ótimo para MAX supõe que MIN também jogue de forma ótima
- E se MIN não jogar de forma ótima?
  - Nesse caso, MAX terá um desempenho ainda melhor

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

Busca Com Competição - Decisões Ótimas

Aplicando as definições:



Busca Com Competição - Algoritmo MiniMax

function MINIMAX-DECISION(*state*) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT(*state*, *a*))

---

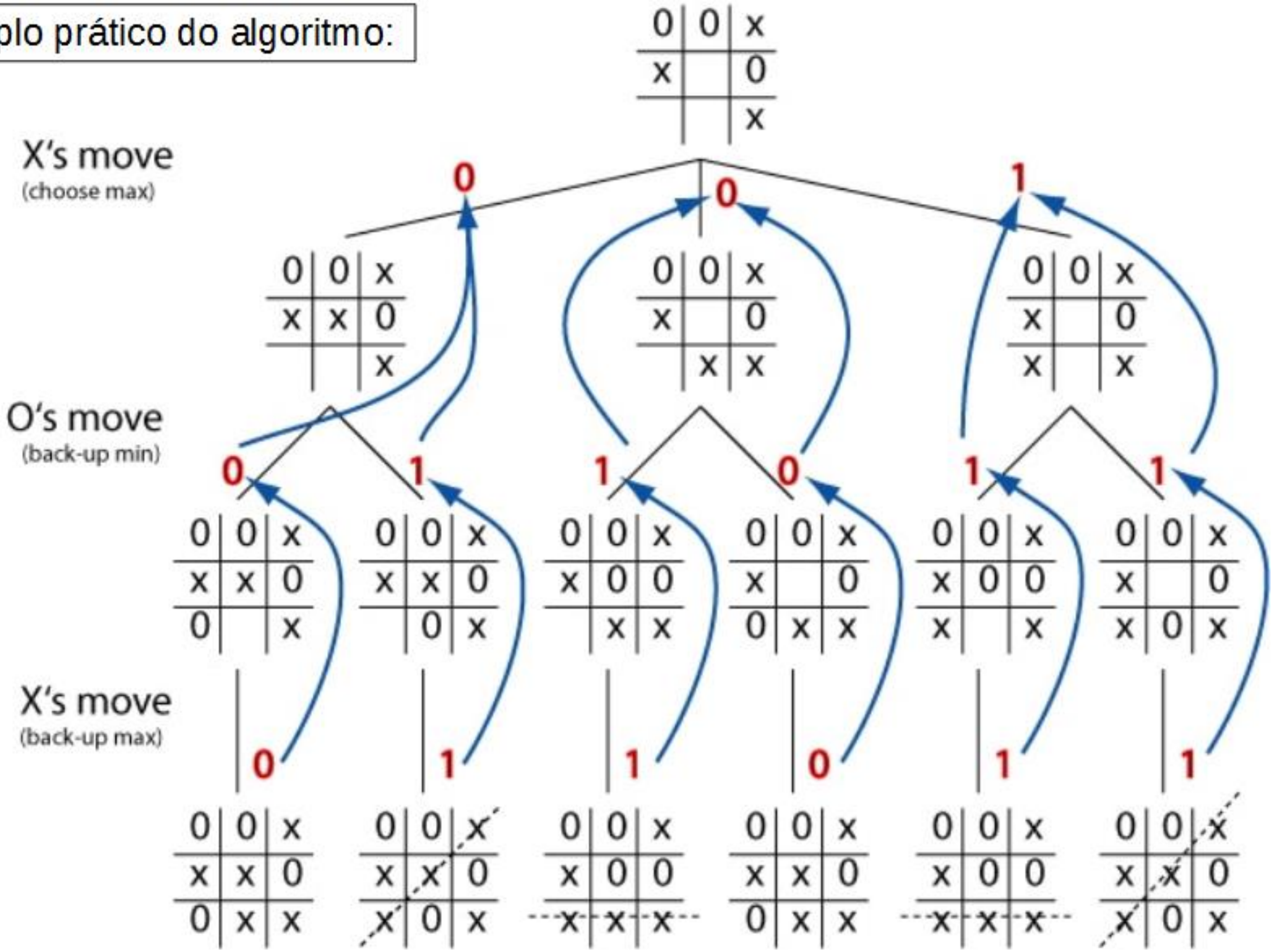
function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for each *a* **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

---

function MIN-VALUE(*state*) **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)))
  return v

Busca Com Competição - Algoritmo MiniMax

Exemplo prático do algoritmo:



Busca Com Competição - Algoritmo MiniMax

- **Desempenho**
  - Completo: sim, se a árvore é finita
  - Ótima: sim, contra um oponente
  - **Complexidade de tempo: O(b^m)**
    - m = profundidade máxima
    - b = movimentos válidos em cada estado
  - **Complexidade de espaço**
    - **O(b^m)** para um algoritmo que gera todos os sucessores de uma vez ou
    - **O(m)** para um algoritmo que gera ações, uma de cada vez

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

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