



Universidade do Minho

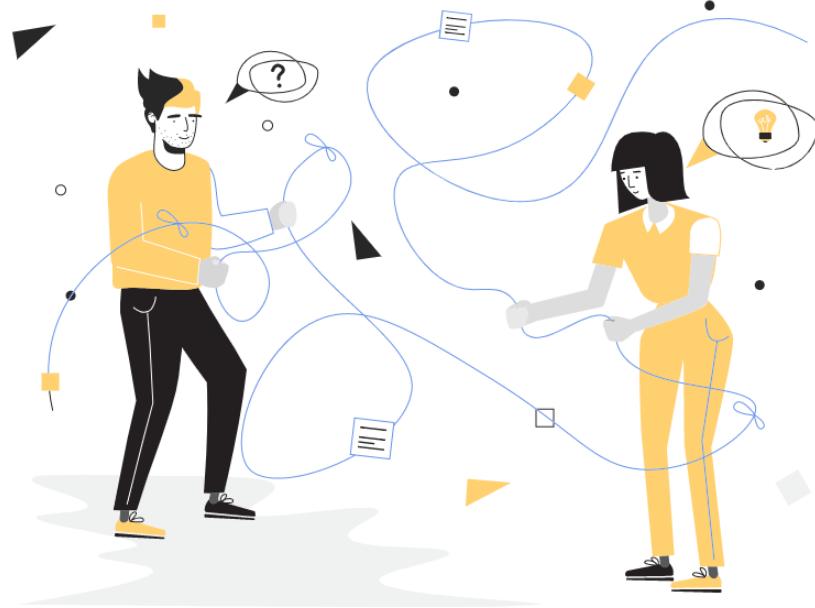
Escola de Engenharia

Departamento de Informática

MÉTODOS DE RESOLUÇÃO DE PROBLEMAS E DE PROCURA

Pesquisa em contextos competitivos (Jogos)

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA
Inteligência Artificial

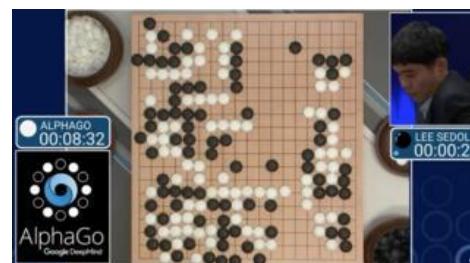


Fonte: <https://milanwittphol.com/projects/adversarial/index.html>

Jogos como Problemas de Pesquisa



Fonte: <https://www.wired.com/2017/05/what-deep-blue-tells-us-about-ai-in-2017/>



Fonte: <https://www.bbc.com/news/technology-35785875>

- Jogos
- Jogos como Problemas de Pesquisa
- Algoritmo Minimax
- Poda Alpha-beta (pruning)
- Decisões Imperfeitas em Tempo Real
- Jogos Determinísticos vs Estocásticos
- Jogos Estocásticos
 - Informação perfeita (Algoritmo Expectiminimax)
 - Informação imperfeita (parcial) (Algoritmo miniminimax e Teoria de jogos)
- Conclusões



Jogos como Problemas de Pesquisa

- Em tópicos anteriores, estudámos que as estratégias de pesquisa estão apenas associadas a uma entidade (agente) que pretende encontrar uma solução (por vezes, expressa numa sequência de ações);
- Porém, podem existir situações em que há mais que um agente na procura de soluções num espaço de procura (o que ocorre habitualmente em jogos);
- Em ambientes multiagentes como estes, cada agente necessita considerar as ações dos outros agentes e como estas afetam-no.

Jogos como Problemas de Pesquisa

- A imprevisibilidade destes agentes pode colocar **contingências** no processo de resolução de problemas por pesquisa do agente.
- Estes ambientes competitivos, nos quais os objetivos dos agentes estão em conflito, dão lugar a problemas de procura com adversários (**adversarial search problems**).
- A Teoria de Jogos considera qualquer ambiente multiagente como um **jogo**, em que o impacto de um agente nos outros agentes é “significativo”, independentemente se os agentes são competitivos ou cooperativos.

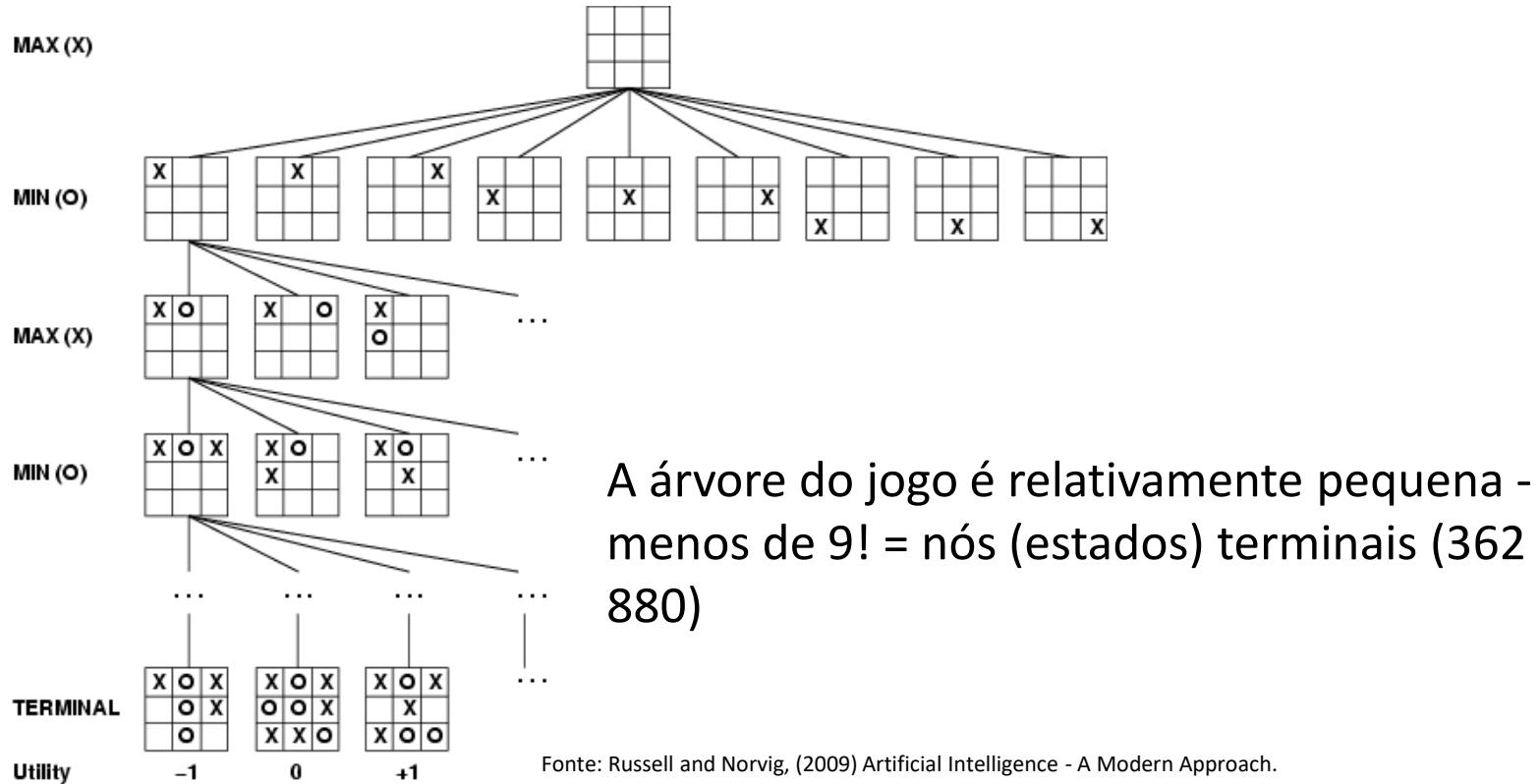
Porquê estudar jogos?

- Os jogos são tradicionalmente uma marca da inteligência;
- Os jogos são (relativamente) fáceis de formalizar;
- Os jogos podem ser um bom modelo de atividades competitivas ou cooperativas do mundo real
 - e.g, confrontações militares, negociações, leilões.

Jogos vs. Pesquisa de agente único

- Não sabemos como o adversário irá agir
 - A solução não é uma sequência fixa de ações do estado inicial para o estado do objetivo, mas uma estratégia ou política (um mapeamento do estado para a melhor jogada nesse estado);
- Eficiência é fundamental para jogar bem
 - O tempo para fazer uma mudança é limitado;
 - O fator de ramificação, a profundidade da pesquisa e o número de configurações do término são enormes.

Um jogo do galo entre dois jogadores, “max” e “min”



Jogos como Problemas de Pesquisa

■ Tipos de Jogos:

- Informação:
 - Perfeita: Xadrez, Damas, Go, Otelo, Gamão, Monopólio
 - Imperfeita: Poker, Scrabble, Bridge, King
- Sorte/Determinístico:
 - Determinístico: Xadrez, Damas, Go, Otelo
 - Jogo de Sorte: Gamão, Monopólio, Poker, Scrabble, Bridge, King

■ Plano de “Ataque”:

- Algoritmo para o jogo perfeito
- Horizonte finito, avaliação aproximada
- Cortes na árvores para reduzir custos (pruning)

Jogos como Problemas de Pesquisa

■ Características:

- Agente Hostil (adversário) incluído no mundo
- Oponente Imprevisível => Solução é um Plano de Contingência
- Tempo Limite => Pouco provável encontrar objetivo. É necessário uma aproximação
- Uma das áreas mais antigas da IA. Em 1950 Shannon e Turing criaram os primeiros programas de Xadrez.
- Xadrez:
 - Todos consideram que é necessário inteligência para jogar
 - Regras simples mas o jogo é complexo
 - Mundo totalmente acessível ao agente
 - Fator de ramificação médio de 35, partida com 50 jogadas => 35^{100} folhas na árvore de pesquisa (embora só existam 10^{40} posições legais)

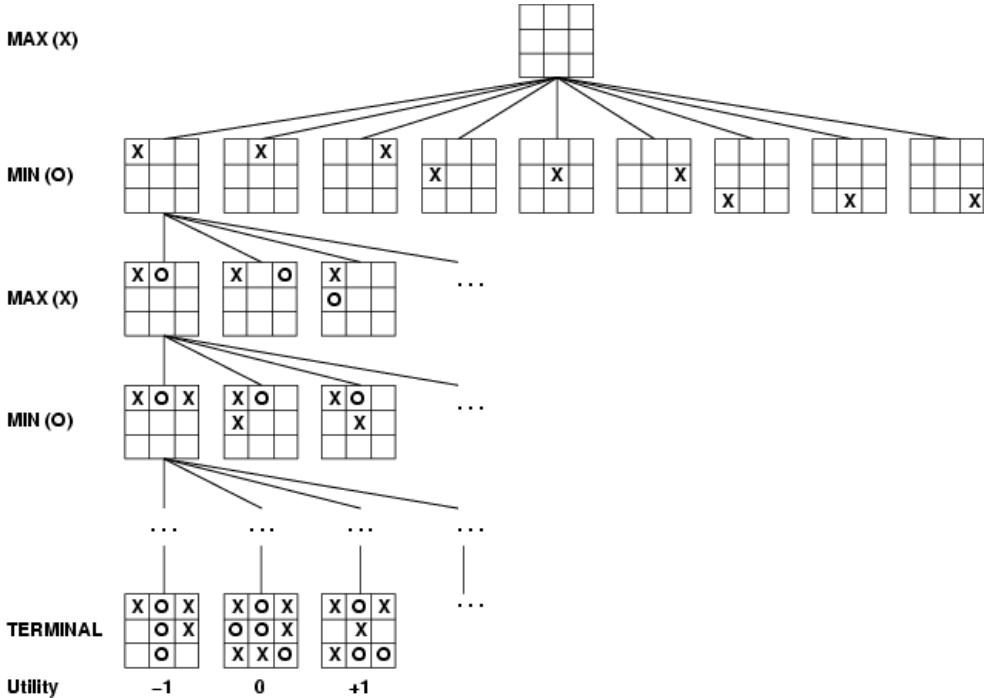
Jogos como Problemas de Pesquisa

- Considerando um jogo de informação perfeita com dois adversários podemos defini-lo formalmente como um problema de pesquisa através dos seguintes elementos:
 - **Estado Inicial** (posição do tabuleiro e qual o próximo jogador a jogar)
 - **Conjunto de Operadores** (que definem os movimentos legais)
 - **Teste Terminal** (que determina se o jogo acabou ou seja está num estado terminal)
 - **Função de Utilidade** (que dá um valor numérico para o resultado do jogo, por exemplo 1 (vitória), 0 (empate), -1 (derrota))
- O algoritmo MiniMax pode ser aplicado como estratégia de resolução neste tipo de jogos (com dois adversários e informação perfeita)

- Minimax: Jogo do Galo

- O nodo do topo é o estado inicial
- MAX joga primeiro, colocando um X no quadrado vazio
- Na imagem ao lado é destacada uma parte da árvore de procura, em que mostra as possíveis jogadas alternativas para MIN(o) e MAX(x) até atingirem os estados terminais (em que pode ser atribuído valores de utilidade mediante as regras do jogo)

Jogos como Problemas de Pesquisa

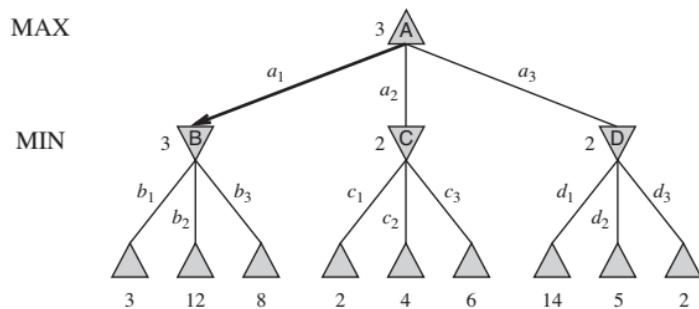


Fonte: Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach.

- Em um problema de pesquisa normal, a solução ideal seria uma sequência de movimentos que levam a um estado de objetivo (um estado terminal que é uma vitória).
- Em um jogo, o MIN tem algo a dizer sobre isso e, portanto, o MAX deve encontrar uma estratégia contingente, que especifique:
 - o movimento do NAX no estado inicial,
 - em seguida, os movimentos do MAX nos estados resultantes de todas as respostas possíveis do MIN,
 - então os movimentos de MAX nos estados resultantes de toda resposta possível de MIN a esses movimentos
- Uma estratégia ideal leva a resultados pelo menos tão bons quanto qualquer outra estratégia quando se está jogando com um oponente infalível.

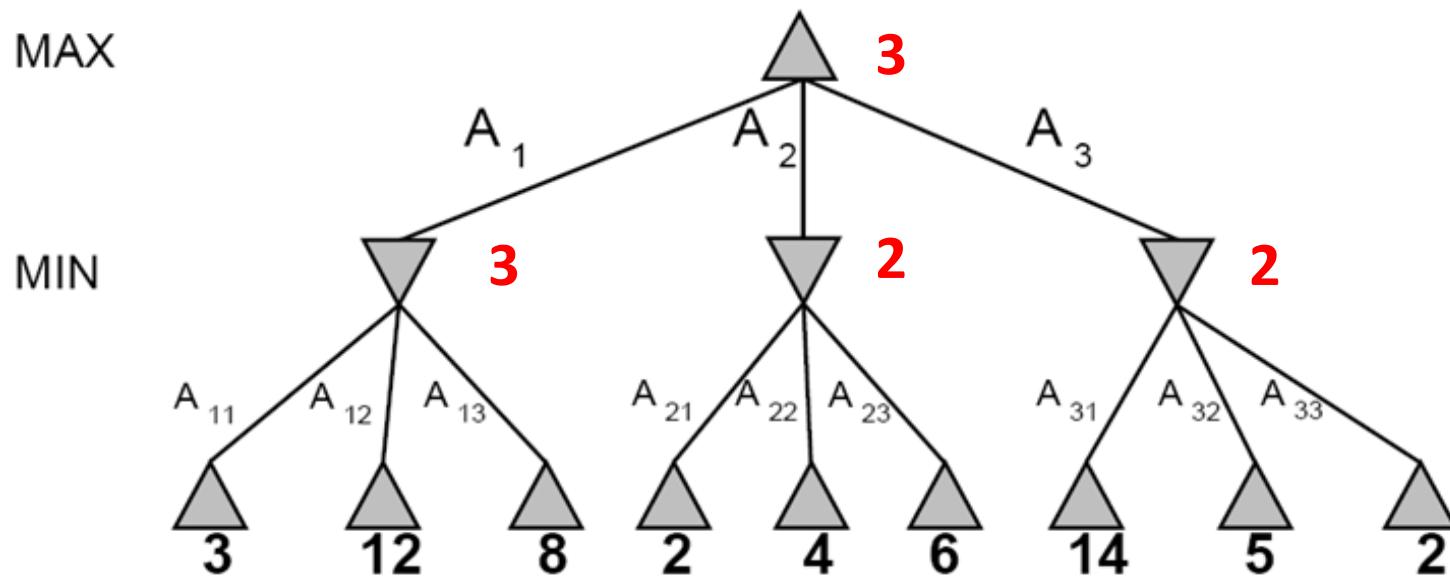
Jogos como Problemas de Pesquisa

- O algoritmo MiniMax pode ser aplicado como estratégia de resolução neste tipo de jogos (determinísticos, com dois adversários e informação perfeita) e consiste em:
 - Gerar a árvore completa até aos estados terminais
 - Aplicar a função utilidade a esses estados
 - Calcular os valores da utilidade até a raiz da árvore, uma camada de cada vez
 - Escolher o movimento com o valor mais elevado.



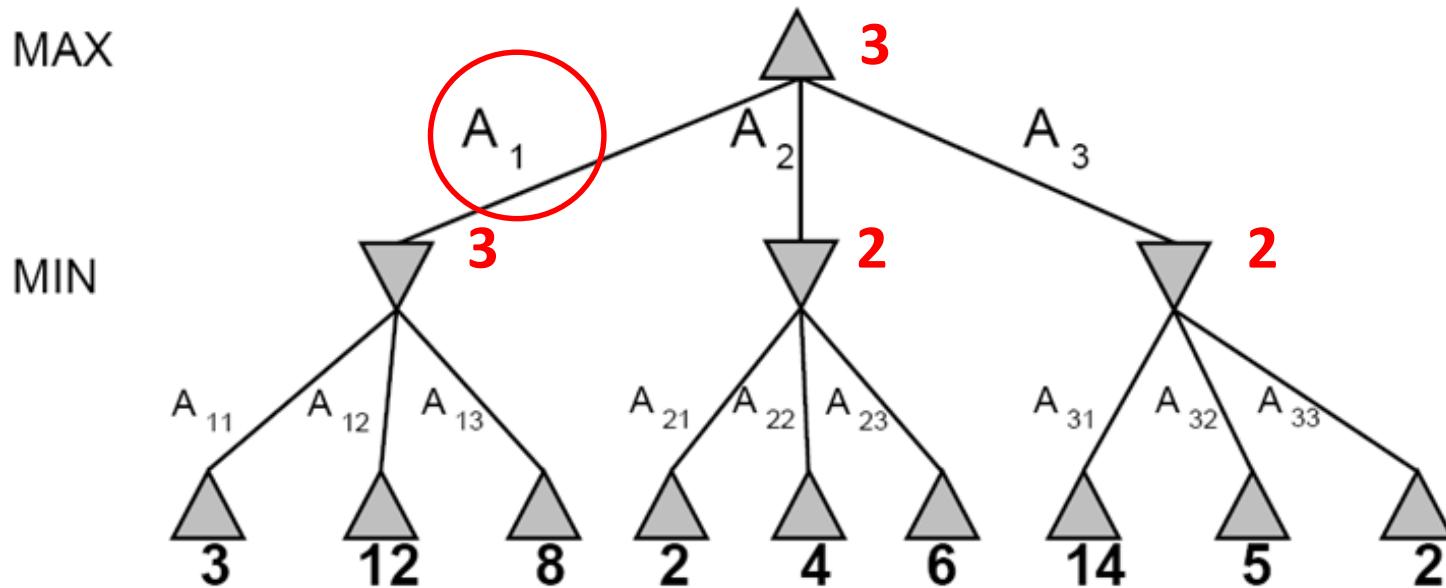
Fonte: Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach.

Árvore do jogo



- **Valor minimax de um nó:** a utilidade (para MAX) de estar no estado correspondente, assumindo o desempenho perfeito dos dois lados
- **Estratégia Minimax:** escolher a jogada que oferece o melhor retorno do pior caso

Calculando o valor minimax de um nó



- Minimax (nó) =
 - utilidade (nó) se o nó for terminal
 - $\text{Max}_{\text{action}} \text{ Minimax} (\text{Sucessor(nó, ação)})$ se player = MAX
 - $\text{min}_{\text{action}} \text{ Minimax} (\text{Sucessor(nó, ação)})$ se player = MIN

- Dada uma árvore de jogo, a estratégia ideal pode ser determinada examinando o valor minimax de cada nó (**MINIMAX-VALUE (n)**)
- O valor minimax de um nó é a utilidade de estar no estado correspondente, supondo que os dois jogadores joguem da melhor maneira dali para o final do jogo.
- Com uma opção, MAX prefere passar para um estado de valor máximo, enquanto MIN prefere um estado de valor mínimo.

```
function MINIMAX-DECISION(state) returns an action
    v  $\leftarrow$  MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v



---


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



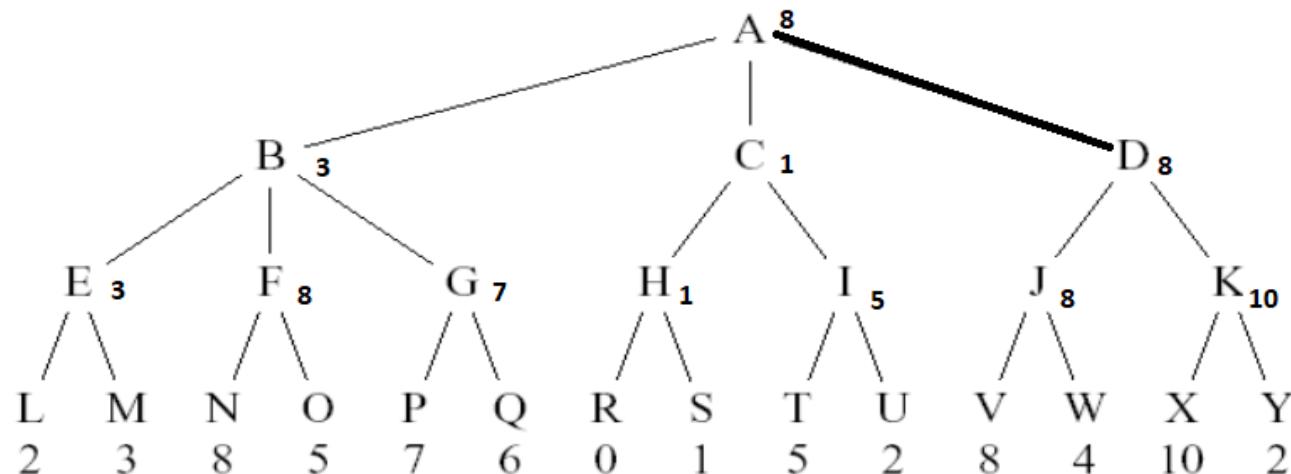
---


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

Exercício – MINIMAX

Solução:

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax à seguinte árvore, indicando o movimento selecionado pelo algoritmo e o respetivo valor estimado.



■ Propriedades:

- Completo? Sim se a árvore for finita!
- Ótimo? Sim contra um adversário ótimo! Senão?
- Complexidade no Tempo? $O(b^m)$
- Complexidade no Espaço? $O(bm)$ (exploração primeiro em profundidade)

■ Problema:

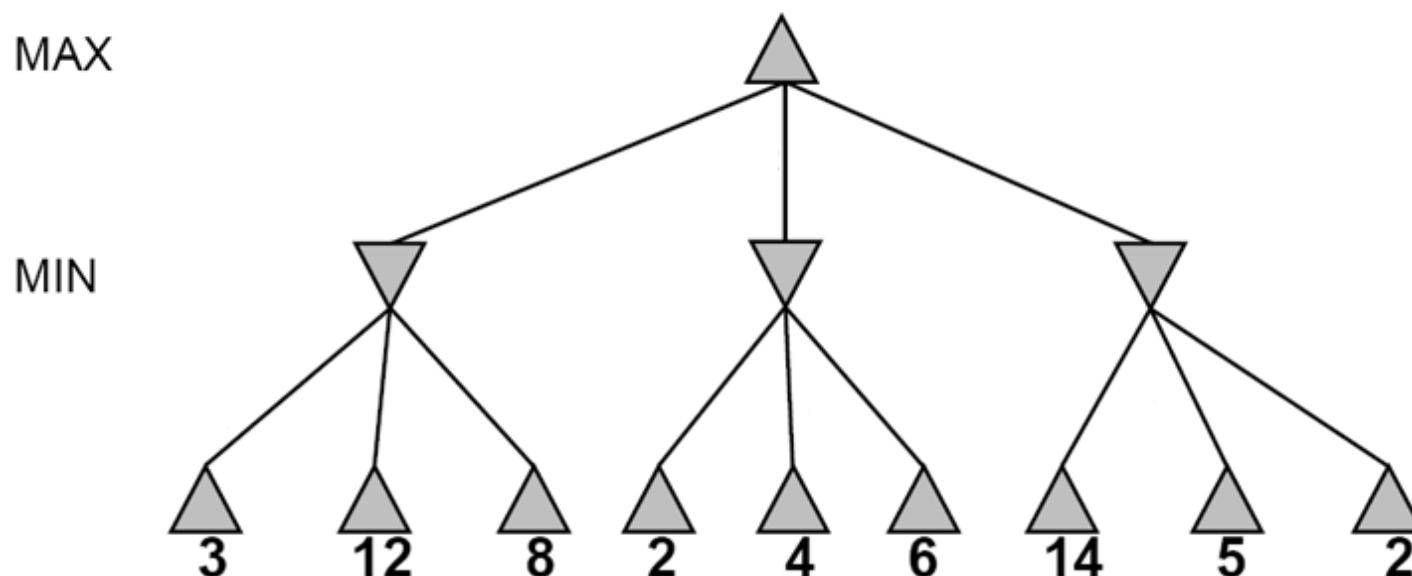
- Inviável para qualquer jogo minimamente complexo

- b: o máximo fator de ramificação (o número máximo de sucessores de um nó) da árvore de pesquisa
- d: a profundidade da melhor solução
- m: a máxima profundidade do espaço de estados

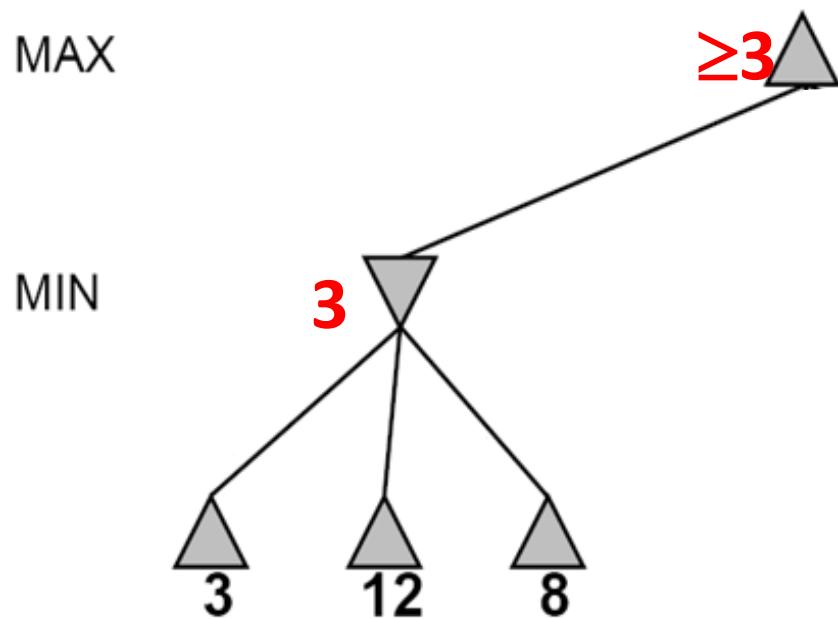
■ Exemplo:

- Para o xadrez ($b=35$, $m=100$), $b^m = 35^{100} = 2.5 \cdot 10^{154}$
- Supondo que são analisadas 450 milhões de hipóteses por segundo $\Rightarrow 2 \cdot 10^{138}$ anos para chegar à solução!

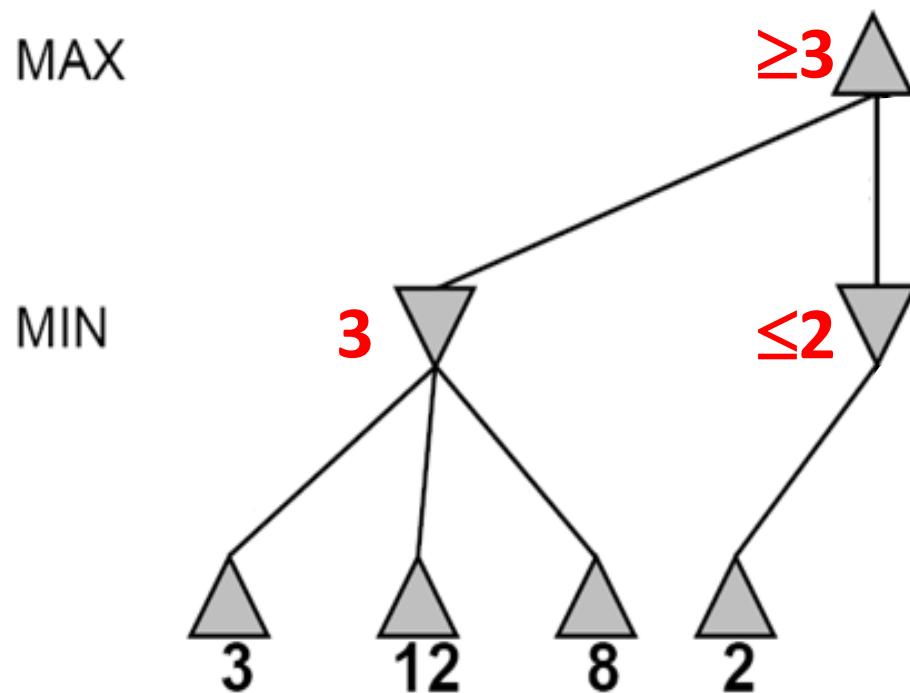
- É possível calcular a decisão minimax exata sem expandir todos os nós na árvore do jogo...



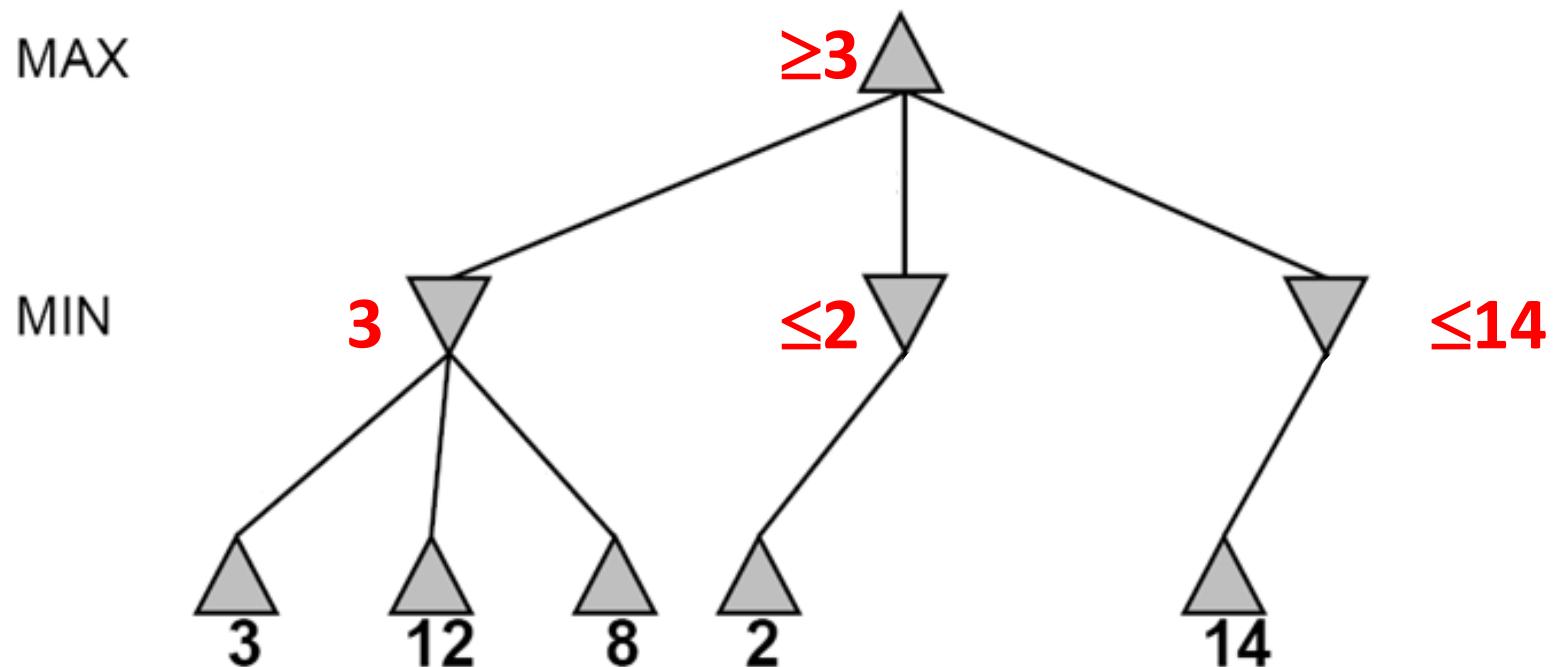
Alpha-beta pruning



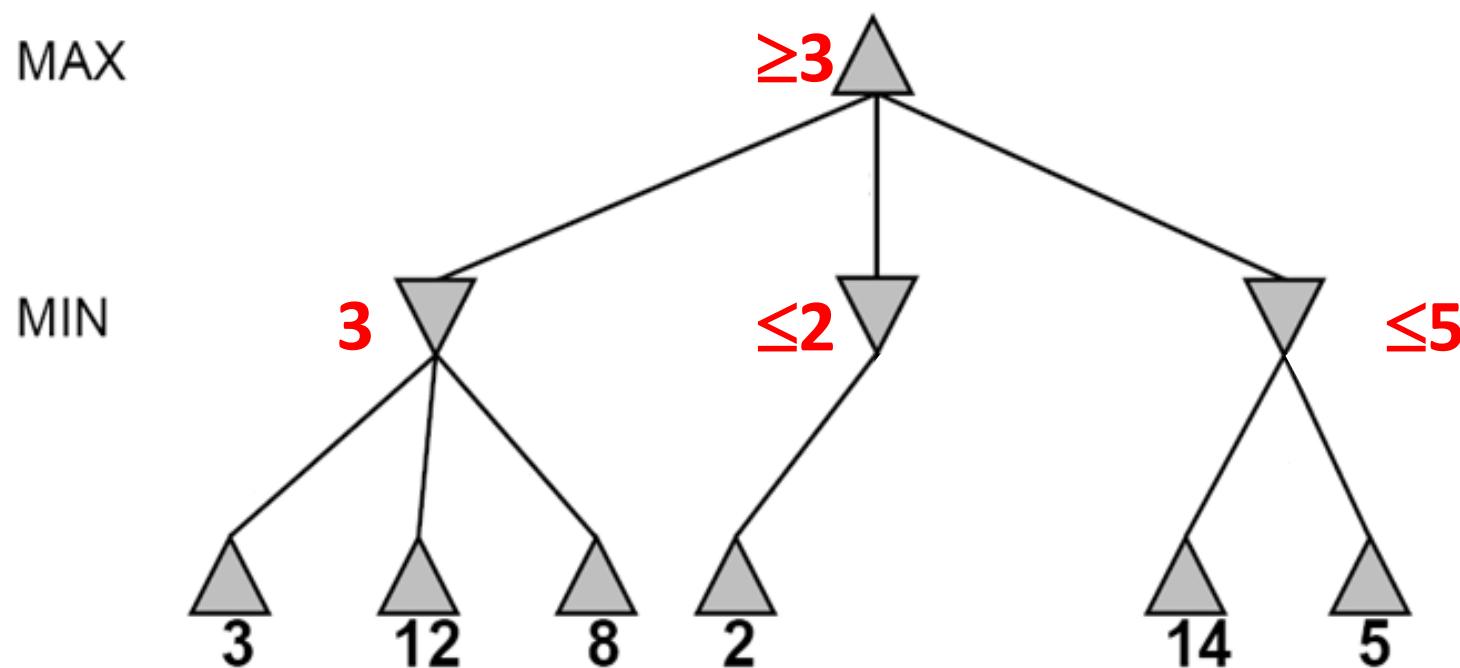
Alpha-beta pruning



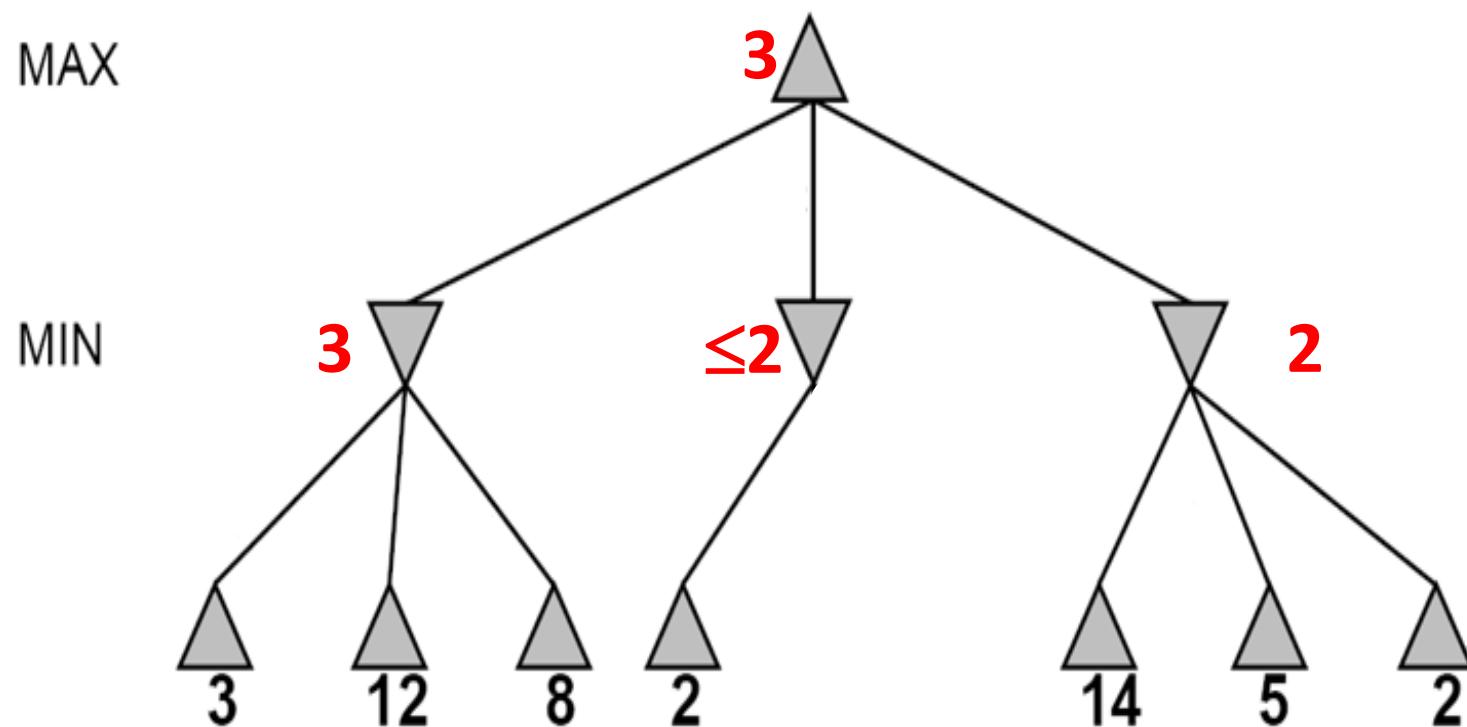
Alpha-beta pruning



Alpha-beta pruning



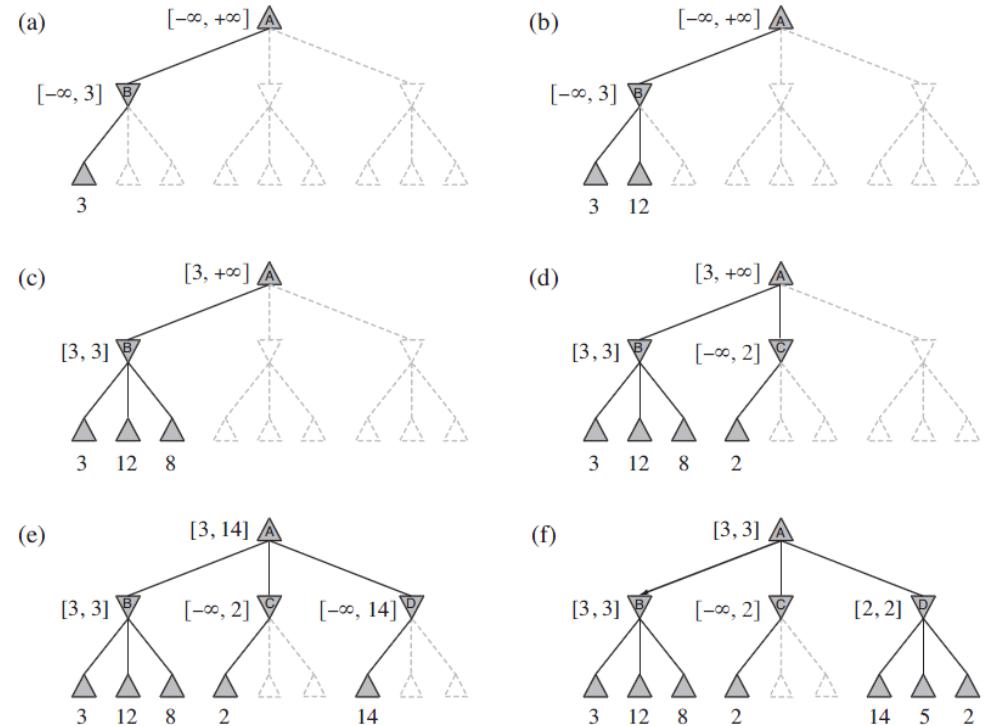
Alpha-beta pruning



- α é o melhor valor (para Max) encontrado até agora no caminho correto
- Se V for pior do que α , Max deve evitá-lo => cortar o ramo
- β é definido da mesma forma para Min

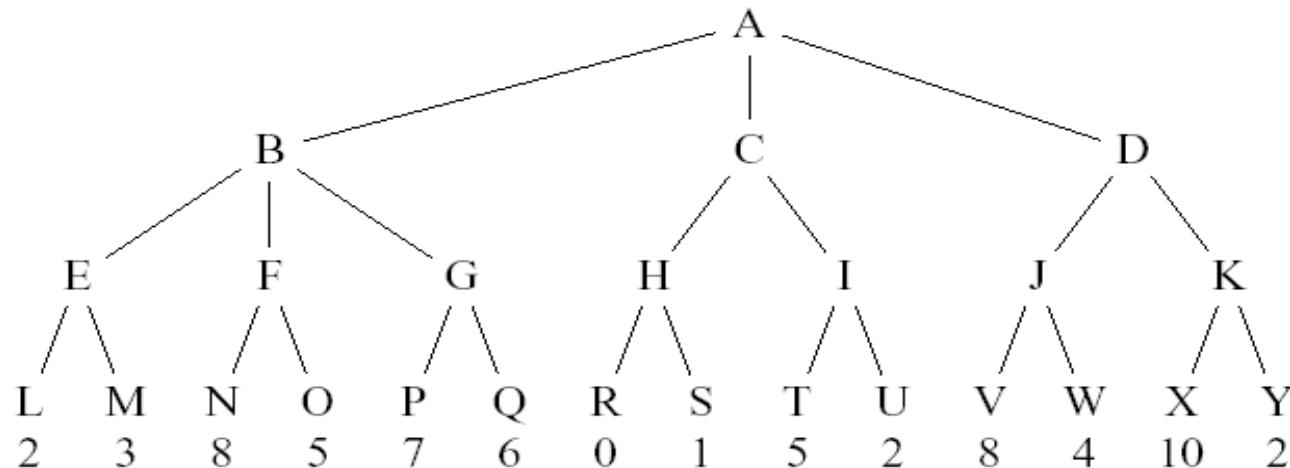
Cortes Alfa-Beta não afetam o resultado final
Boa ordenação melhora a eficiência dos cortes

Alpha-beta pruning Cortes Alfa-Beta



Exercício – MINIMAX com Cortes

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax com cortes Alfa-Beta à seguinte árvore, indicando o movimento selecionado pelo algoritmo. Indique graficamente e justifique todos os cortes que efetuar na aplicação do algoritmo Minimax

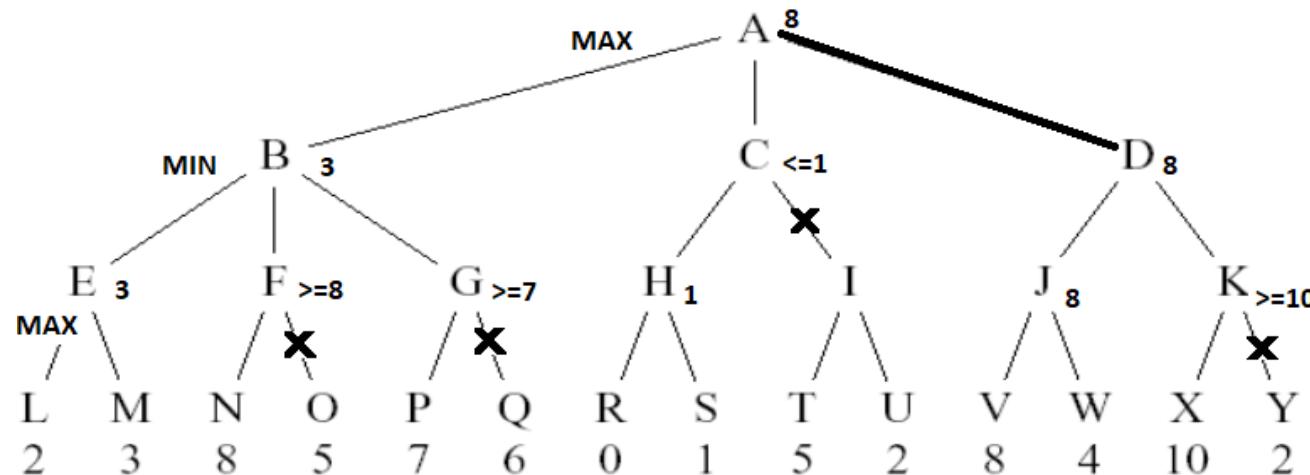


Fonte: Luís Paulo Reis, Artificial Intelligence (2019), Universidade do Porto

Exercício – MINIMAX com Cortes

Solução:

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax com cortes Alfa-Beta à seguinte árvore, indicando o movimento selecionado pelo algoritmo. Indique graficamente e justifique todos os cortes que efetuar na aplicação do algoritmo Minimax



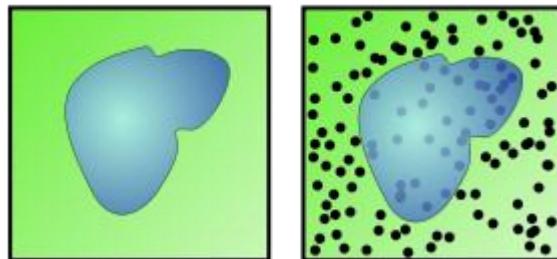
Problema **Decisões Imperfeitas em Tempo Real**

- O algoritmo minimax gera todo o espaço de pesquisa do jogo, enquanto o algoritmo alfa – beta nos permite remover grande parte dele. No entanto, o alfa – beta ainda precisa pesquisar até os estados terminais em pelo menos uma parte do espaço de pesquisa;
- Essa profundidade geralmente não é prática, porque as mudanças devem ser feitas em um período de tempo razoável - normalmente alguns minutos no máximo;
 - Função de Avaliação: Utilidade (interesse) estimada para a posição
 - Teste de Corte: Profundidade Limite
- Surge eventualmente aqui outro problema: Problema do horizonte!
 - Tabela de transposição para armazenar estados expandidos anteriormente;
 - Poda (*pruning*) direta para evitar considerar todos os movimentos possíveis;
 - Tabelas de pesquisa para movimentos de abertura e jogos finais

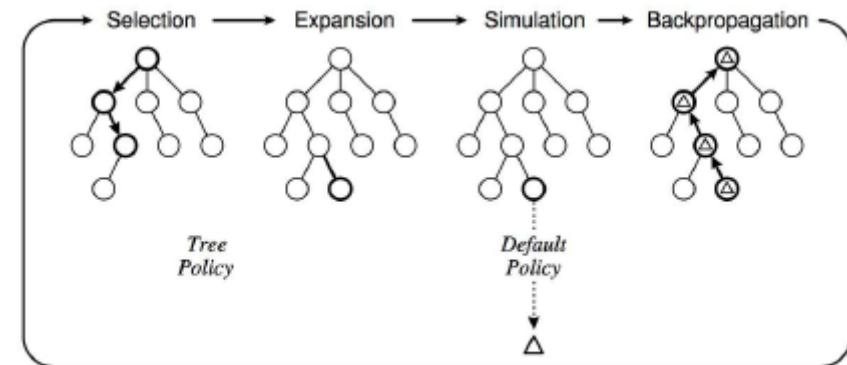
- Na base: 200 milhões de avaliações de nós por movimento (3 min), minimax com uma função de avaliação decente e procura por quiescência
 - 5 jogadas ≈ humano novato
- Adicione poda alfa-beta
 - 10 jogadas, jogador experiente
- Deep Blue: 30 bilhões de avaliações por jogada, função de avaliação com 8000 recursos, grandes bases de dados de movimentos de abertura e final de jogo
 - 14 jogadas ≈ Garry Kasparov
- Estado da arte atual (Hydra, et al., 2006): 36 bilhões de avaliações por segundo, técnicas avançadas de poda
 - 18 jogadas - melhor do que qualquer ser humano vivo?

Monte Carlo Tree Search

- E em os jogos com árvores profundas, grande fator de ramificação e sem boas heurísticas - como o Go?
- Em vez de pesquisa com profundidade limitada com uma função de avaliação, use simulações aleatórias
- Começando no estado atual (raiz da árvore de pesquisa), itere:
 - Selecione um nó folha para expansão usando uma política de árvore (conflito de descoberta e exploração)
 - Execute uma simulação usando uma política padrão (por exemplo, movimentos aleatórios) até que um estado terminal seja alcançado
 - Propagar novamente o resultado para atualizar as estimativas de valor dos nós internos da árvore



Fonte: https://pt.wikipedia.org/wiki/M%C3%A9todo_de_Monte_Carlo



- Damas:

- **Chinook** acabou com o reinado de 40 anos do campeão humano Marion Tinsley em 1994. Usava uma base de dados para finais de partida definindo a forma perfeita de vencer para todas as posições envolvendo 8 ou menos peças (no total de 443748401247 posições). Hoje em dia é um **problema resolvido**.

- Xadrez:

- **Deep Blue derrotou** o campeão do mundo humano **Gary Kasparov** num jogo com 6 partidas em 1997. Deep Blue pesquisava 200 milhões de posições por segundo e usa uma função de avaliação extremamente sofisticada e métodos (não revelados) para estender algumas linhas de pesquisa para além da profundidade 40!

- Go (2015):

- Campeões humanos recusam-se a competir com computadores pois as **máquinas não conseguem jogar razoavelmente ($b>300$)**

- Go (2017):

- AlphaGo (Fan, Lee), AlphaGo Master, AlphaGo Zero and AlphaZero! Máquinas vencem 100-0 campeões humanos e máquinas anteriores.

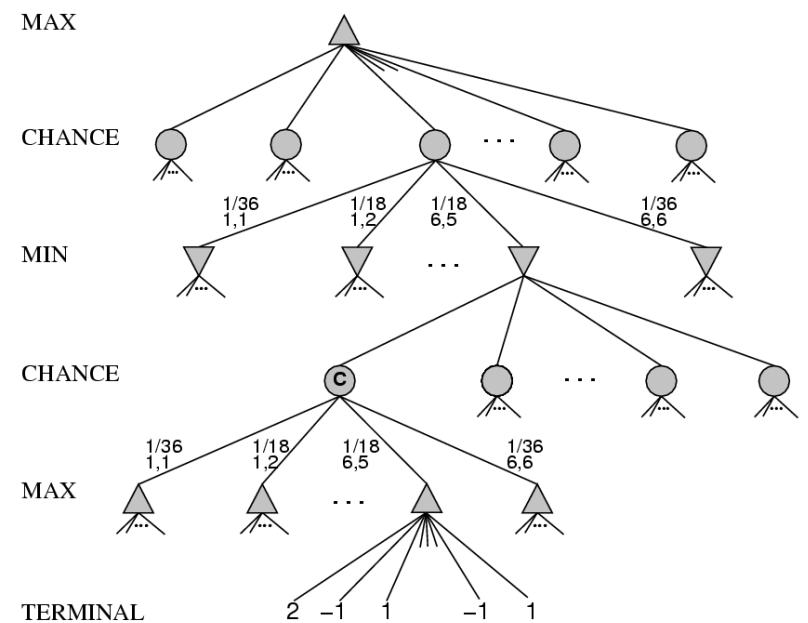
Jogos Estocásticos

- Na vida real, muitos eventos externos imprevisíveis podem nos colocar em situações imprevistas. Muitos jogos refletem essa imprevisibilidade ao incluir um elemento aleatório (e.g., lançamento de dados).

	Determinístico	Estocástico
Informação perfeita (totalmente observável)	Xadrez Go, Damas	Gamão, Monopólio
Informação imperfeita (parcialmente observável)	Batalha Naval	Scrabble, Poker, Bridge

Jogos Estocásticos

- Jogos Estocásticos são jogos que tipicamente combinam habilidade e sorte;
- Árvore de pesquisa deve incluir nós de probabilidade;
- Decisão é efetuada com base no valor esperado;
- *Algoritmo ExpectiMiniMax.*



Fonte: Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach.

Minimax vs. Expectiminimax

- **Minimax:**

- **Maximize** (over all possible moves I can make) the
- **Minimum** (over all possible moves Min can make) of the
- Reward

$$Value(node) = \max_{\text{my moves}} \left(\min_{\text{Min's moves}} (Reward) \right)$$

- **Expectiminimax:**

- **Maximize** (over all possible moves I can make) the
- **Minimum** (over all possible moves Min can make) of the
- **Expected** reward

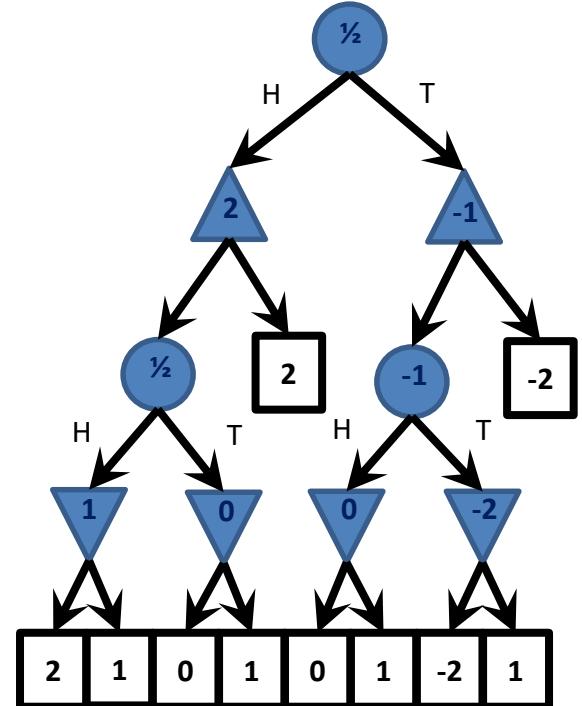
$$Value(node) = \max_{\text{my moves}} \left(\min_{\text{Min's moves}} (\mathbb{E}[Reward]) \right)$$

$$\mathbb{E}[Reward] = \sum_{outcomes} \text{Probability}(outcome) \times \text{Reward}(outcome)$$

- **Expectimax:** para nós sorte (chance) somar os valores dos estados sucessores ponderados pela probabilidade de cada sucessor.
- **Value(node)** =
 - Utility(*node*) if *node* is terminal
 - $\max_{action} \text{Value}(\text{Succ}(node, action))$ if *type* = MAX
 - $\min_{action} \text{Value}(\text{Succ}(node, action))$ if *type* = MIN
 - $\sum_{action} P(\text{Succ}(node, action)) * \text{Value}(\text{Succ}(node, action))$ if *type* = CHANCE

Jogos Estocásticos Exemplo

- ALEATÓRIO: Max joga uma moeda. Ou é Cara (H) ou coroa (T).
 - Para nas Cara: o jogo termina, Max vence (valor = € 2).
 - Para no Coroa: o jogo termina, Max perde (valor = -€ 2).
 - Continuar: o jogo continua
- ALEATÓRIO: Min joga uma moeda.
 - CARA-CARA: valor = € 2
 - Coroa-Coroa: valor = -€ 2
 - Cara-Coroa or Coroa-Cara: valor = 0
- MIN: Min decide se deve manter o resultado atual (valor acima) ou pagar uma penalidade (valor = € 1).



- Todos os métodos anteriores são úteis:
 - Poda (pruning) alfa-beta
 - Função de avaliação
 - etc
- A complexidade computacional é muito complicada
 - O fator de ramificação da escolha aleatória pode ser alto
 - O dobro dos "níveis" na árvore
- No Expectiminimax: como vimos para nós sorte (Chance), ele soma os valores dos estados sucessores ponderados pela probabilidade de cada sucessor.
 - Fator de ramificação pode ser muito desagradável, definindo funções de avaliação e algoritmos de poda mais difíceis
- Eventualmente usar a Simulação de Monte Carlo: quando se chega a um nó sorte, simula-se um grande número de jogos com jogadas aleatórias de dados e use a percentagem de ganhos como uma função de avaliação.
 - Pode funcionar bem para jogos como o Gamão.

Jogos Estocásticos com informação imperfeita (parcial)

Os jogos de cartas fornecem um dos melhores exemplos de jogos estocástica com informação imperfeita (parcial), onde as informações não conhecidas (imperfeitas) advêm de aleatoriedade.

Exemplo: em muitos jogos, as cartas são distribuídas aleatoriamente no início do jogo, com cada jogador recebendo uma mão que não é nem visível nem conhecida pelos os outros jogadores.

Jogos: Bridge, Copas e Poker.



▪ **Minimax:**

- **Maximize** (over all possible moves I can make) the
- **Minimum**
 - (over all possible states of the information I don't know,
 - ... over all possible moves Min can make) the
- Reward.

$$Value(node) = \max_{my\ moves} \left(\min_{\substack{missing\ info, \\ Min's\ moves}} (Reward) \right)$$

Jogos Estocásticos com informação imperfeita Técnicas

- Se conhecemos as probabilidades de diferentes configurações e desejamos maximizar os ganhos médios (por exemplo, se podemos jogar o jogo muitas vezes): **expectimax**
- Se não temos ideia das probabilidades de diferentes configurações; ou, se pudemos apenas jogar uma vez e não pudemos (nem queremos) perder: **minimax**
- Se a informação desconhecida foi selecionada intencionalmente pelo oponente: usamos **teoria dos jogos**

- Trabalhar com jogos é extremamente interessante
 - Fácil testar novas ideias
 - Fácil comparar agentes com outros agentes
 - Fácil comparar agentes com humanos
- Jogos ilustram diversos pontos interessantes da IA
 - Perfeição é inatingível => é necessário aproximar!
 - É boa ideia pensar sobre o que pensar
 - Incerteza restringe a atribuição de valores aos estados
- Jogos funcionam para a IA como a Formula 1 para a construção de automóveis...

Bibliografia Recomendada

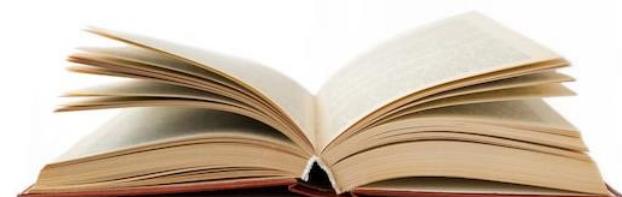
- Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach, 3rd edition, ISBN-13: 9780136042594, Chapter 5.

Outro material

- Svetlana Lazebnik, Lecture notes Fall 2017 Artificial Intelligence, University of Illinois.
- Luís Paulo Reis, Lecture notes Artificial Intelligence (2019), Universidade do Porto

Para quem quer ir mais longe

AlphaGo: Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489 (2016). <https://doi.org/10.1038/nature16961>



Universidade do Minho

Escola de Engenharia

Departamento de Informática

MÉTODOS DE RESOLUÇÃO DE PROBLEMAS E DE PROCURA

Pesquisa em contextos competitivos (Jogos)

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA
Inteligência Artificial