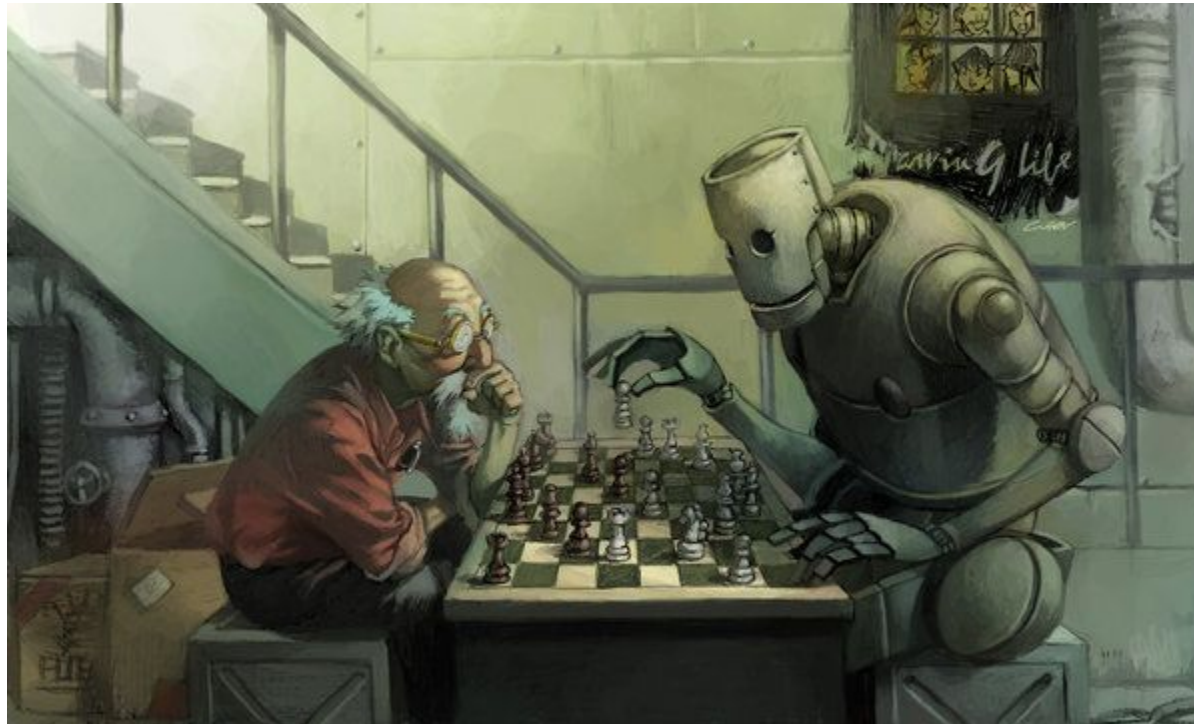


Busca adversária



http://www.zverenish.ru/lj/cuson/play_chess_with_robot_by_cuson.jpg

Competições em IA

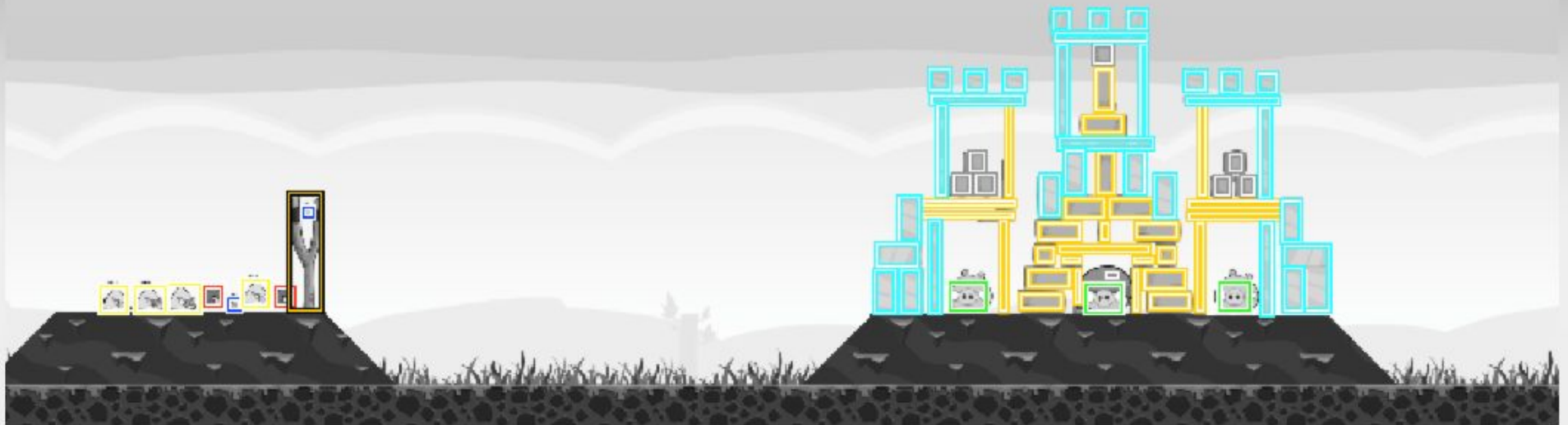
Visual Doom AI Competition @ CIG 2016



Competições em IA

AI Birds.org
Angry Birds AI Competition

Font size [Bigger](#) [Reset](#) [Smaller](#)



You are here: Home

Jogos : considerações gerais

- **Aplicações atrativas para métodos IA desde o início.**
 - Formulação simples do problema (ações bem definidas)
 - Ambiente acessível;
 - Abstração (representação simplificada de problemas reais)
 - Sinônimo de inteligência
 - Primeiro algoritmo para xadrez foi proposto por Claude Shannon na década de 50
- **Porém desafiador:**
 - Tamanho + limitação de tempo (35^{100} nós para xadrez);
 - Incerteza devido ao outro jogador;
 - Problema “contingencial”: agente deve agir antes de completar a busca

Jogos : considerações gerais

- **Tipos**

- Múltiplos jogadores
- Completamente ou parcialmente observáveis
- Determinístico ou não determinístico

	Determinístico	Não-Determinístico
Observável	Xadrez, Damas, Go, Othello	Gamão, Banco Imobiliário
Parcialmente Observável	Jogo da Velha Cego	Poquer

Jogos : considerações gerais

- **Jogo de soma zero**
 - O melhor movimento de A leva ao pior movimento de B.
 - Valores de utilidade (x,y) : $(+1,0)$, $(0,0)$, $(0,+1)$
 - Escreve-se $U=x-y$: $+1, 0, -1$.
 1. A é o jogador MAX
 2. B é o jogador MIN



jogo de soma zero



jogos em geral



Algoritmo min-max

Jogos de tabuleiro: Formulando e resolvendo o problema

- **Formulação**

- **Estado inicial:** posições do tabuleiro + de quem é a vez
- **Estado final:** posições em que o jogo acaba
- **Operadores:** jogadas legais
- **Função de utilidade:** valor numérico do resultado (pontuação)

- **Busca: algoritmo minimax**

- Idéia: maximizar a utilidade (ganho) supondo que o adversário vai tentar minimizá-la
- Minimax faz busca cega em profundidade
- O agente é MAX e o adversário é MIN

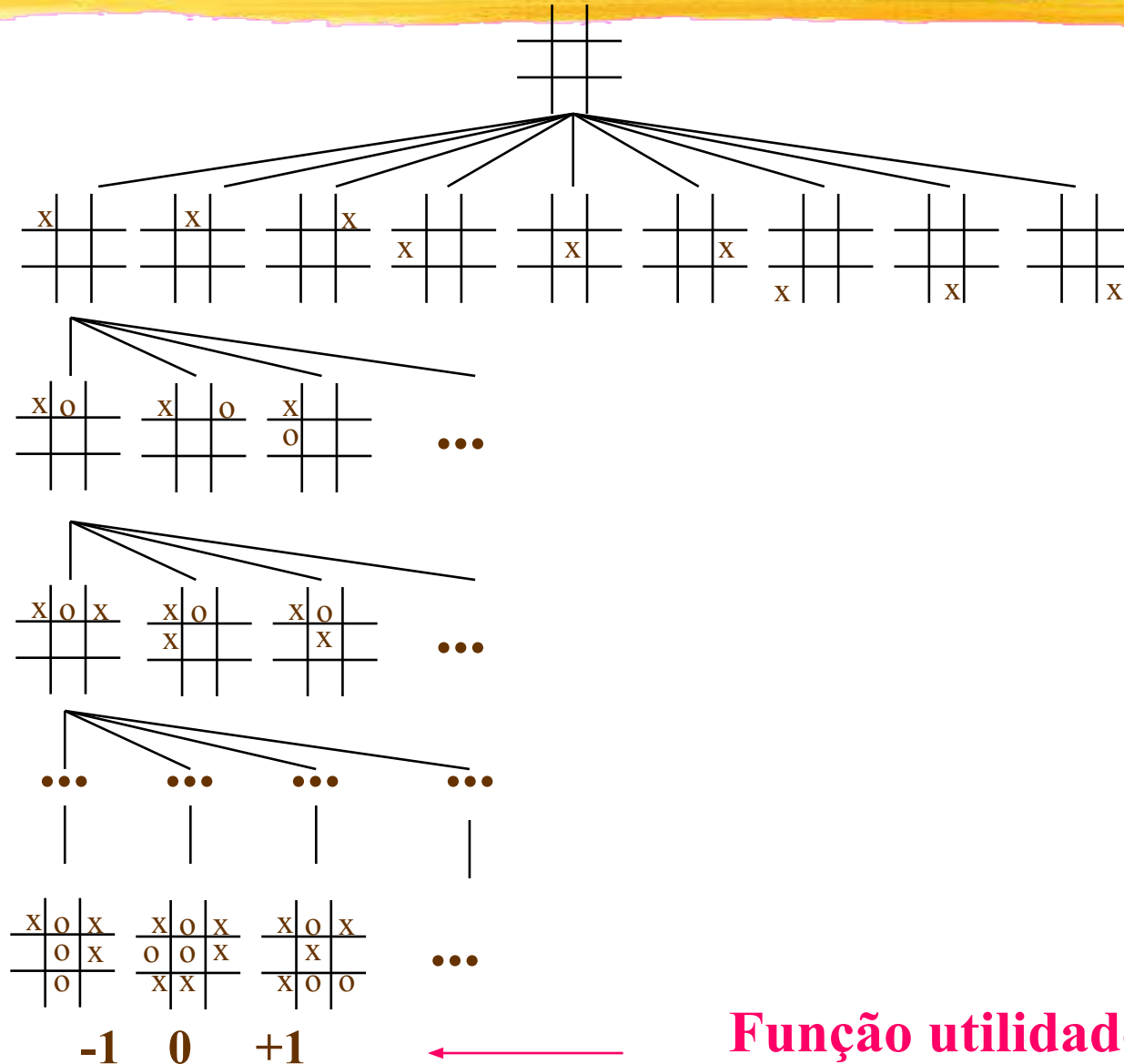
Jogo da velha (min-max)

Max(X)

Min(O)

Max(X)

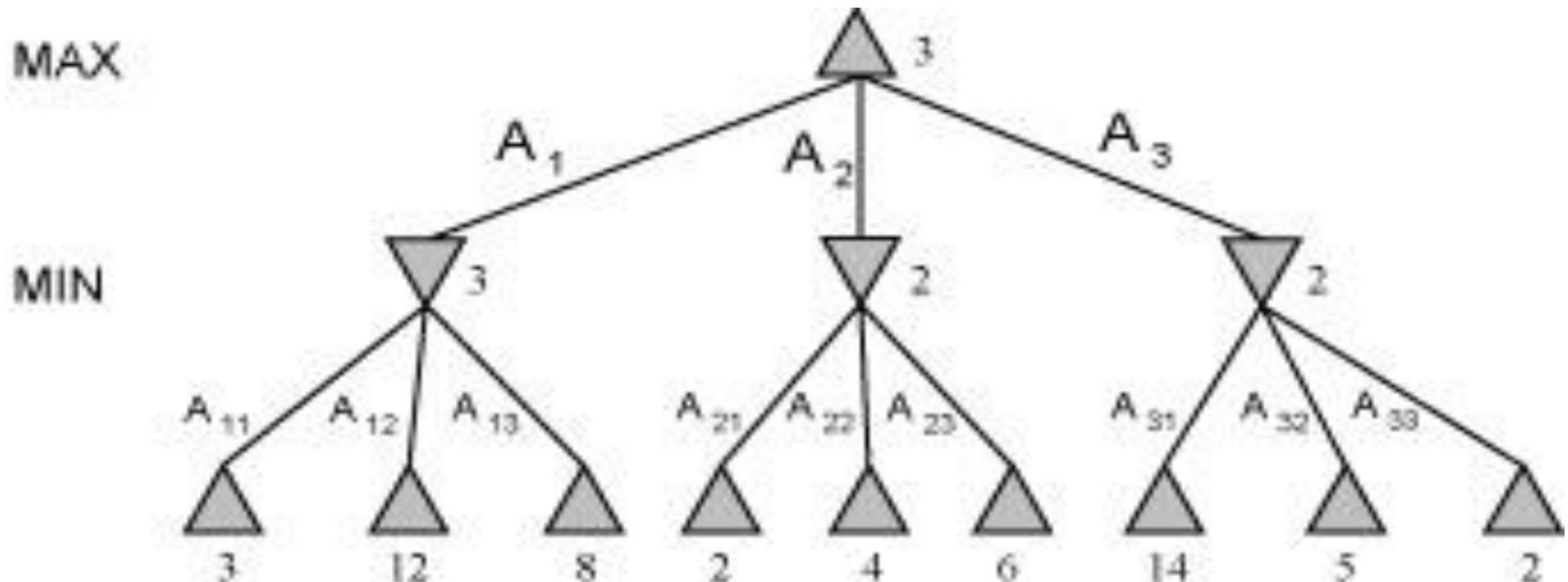
Min(O)



Minimax

Passos

- Gera a árvore inteira até os estados terminais.
- Aplica a função de utilidade nas folhas.
- Propaga os valores dessa função subindo a árvore através do minimax
- Determinar qual o valor que será escolhido por MAX.



Críticas

- **Problemas**

- Tempo gasto é totalmente impraticável, porém o algoritmo serve como base para outros métodos mais realísticos.
- Complexidade: $O(b^m)$.

- **Para melhorar**

1) Substituir a profundidade n de $\text{min-max}(n)$ pela estimativa de $\text{min-max}(n)$: função de avaliação

2) Podar a árvore onde a busca seria irrelevante: poda alfa-beta

Funções de Avaliação

- **Reflete as chances de ganhar:** baseada no **valor material**
 - ex. valor de uma peça independentemente da posição das outras
- **Função Linear de Peso de propriedade do nó:**
 - $w_1f_1 + w_2f_2 + \dots + w_nf_n$
 - Ex. Os pesos (w) no xadrez poderiam ser o tipo de pedra do xadrez (Peão-1, ..., Rainha-9) e os (f) poderiam ser o número de cada peça no tabuleiro.
- **Escolha das propriedades relevantes ainda não pode ser realizada.**
- **Escolha crucial: compromisso entre precisão e eficiência**

Função de avaliação para o jogo da velha

X		
	0	

$$H = 6 - 5 = 1$$

X	0	

$$H = 4 - 6 = -2$$

		0
	X	

$$H = 5 - 4 = 1$$

X		
	X	
		0

X tem 6 possibilidades

X		
		0

0 tem 5 possibilidades

Uso da Funções de Avaliação

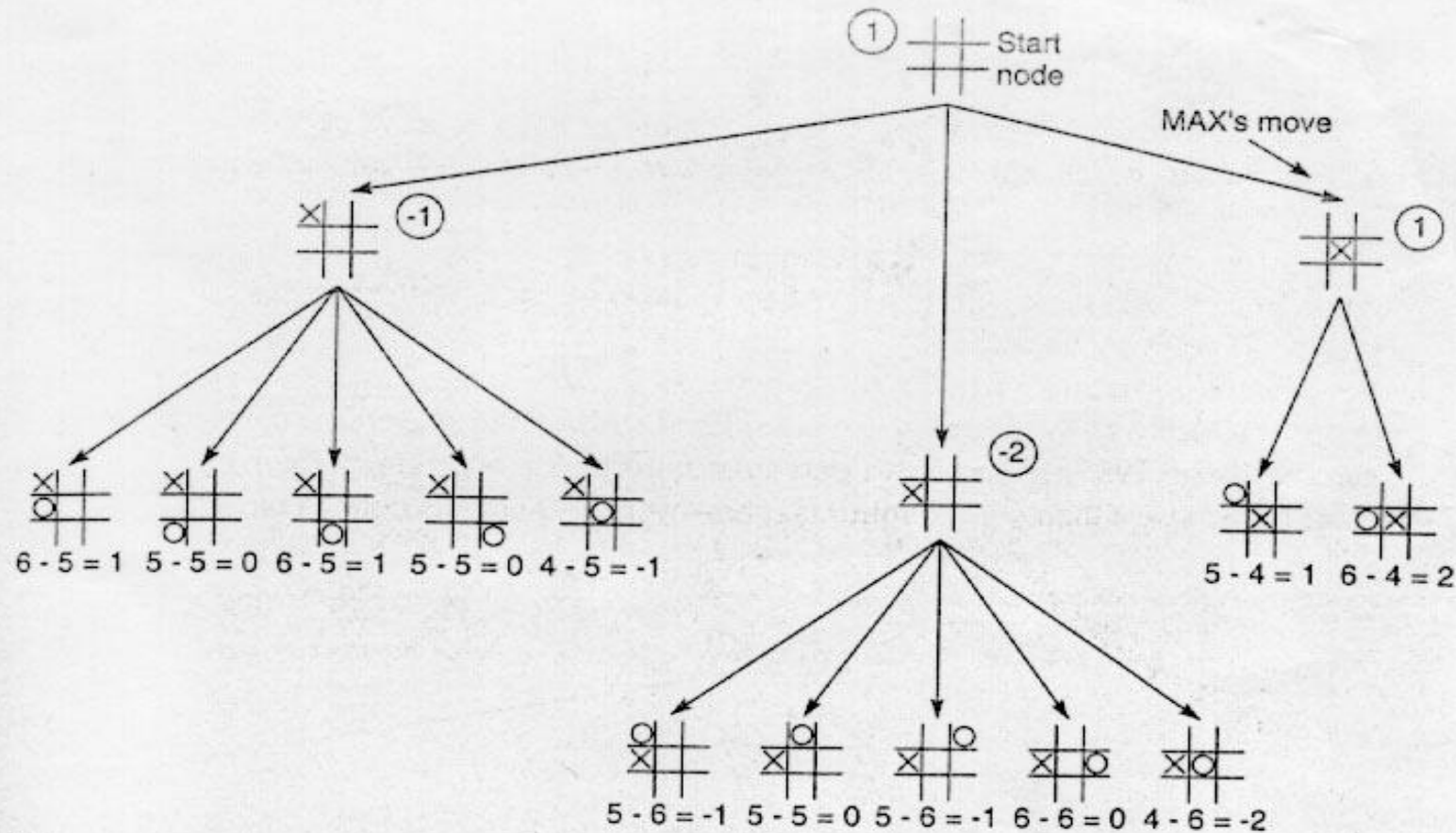


Figure 4.17 Two-ply minimax applied to the opening move of tic-tac-toe.

Quando aplicar a função de avaliação?

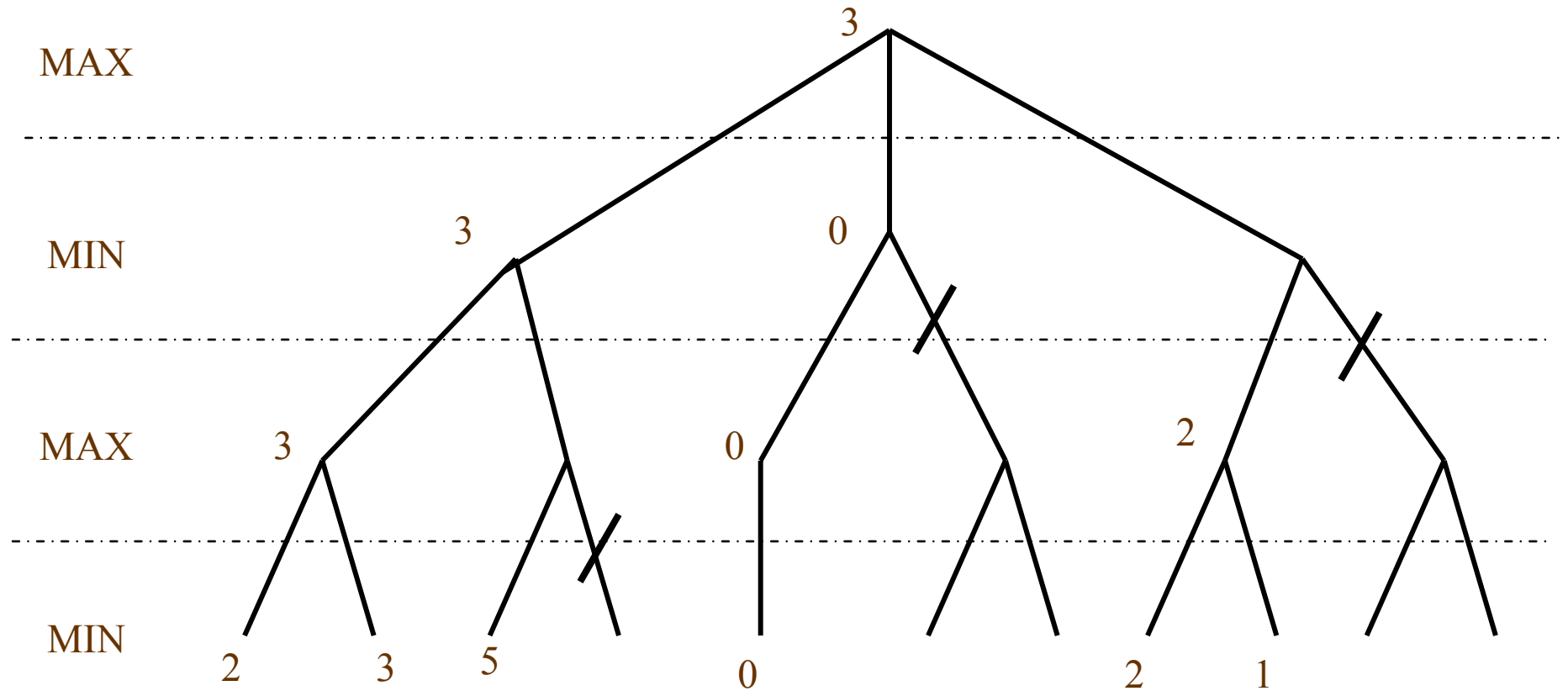
- Definir uma profundidade máxima ou iterativa não funciona devido à incerteza inerente ao problema
- **Solução: Procura Tranquila (*Quiescence search*):**
 - **Idéia:** evitar avaliação em situações a partir das quais pode haver mudanças bruscas
 - No caso do jogo da velha, toda posição é tranquila mas no xadrez não.... (ex. um peça de xadrez a ser comida)
 - **Algoritmo:** Se a situação (nó) é “tranquila”, então aplica a função de avaliação, senão busca até encontrar uma situação “tranquila”

Alpha-Beta Pruning

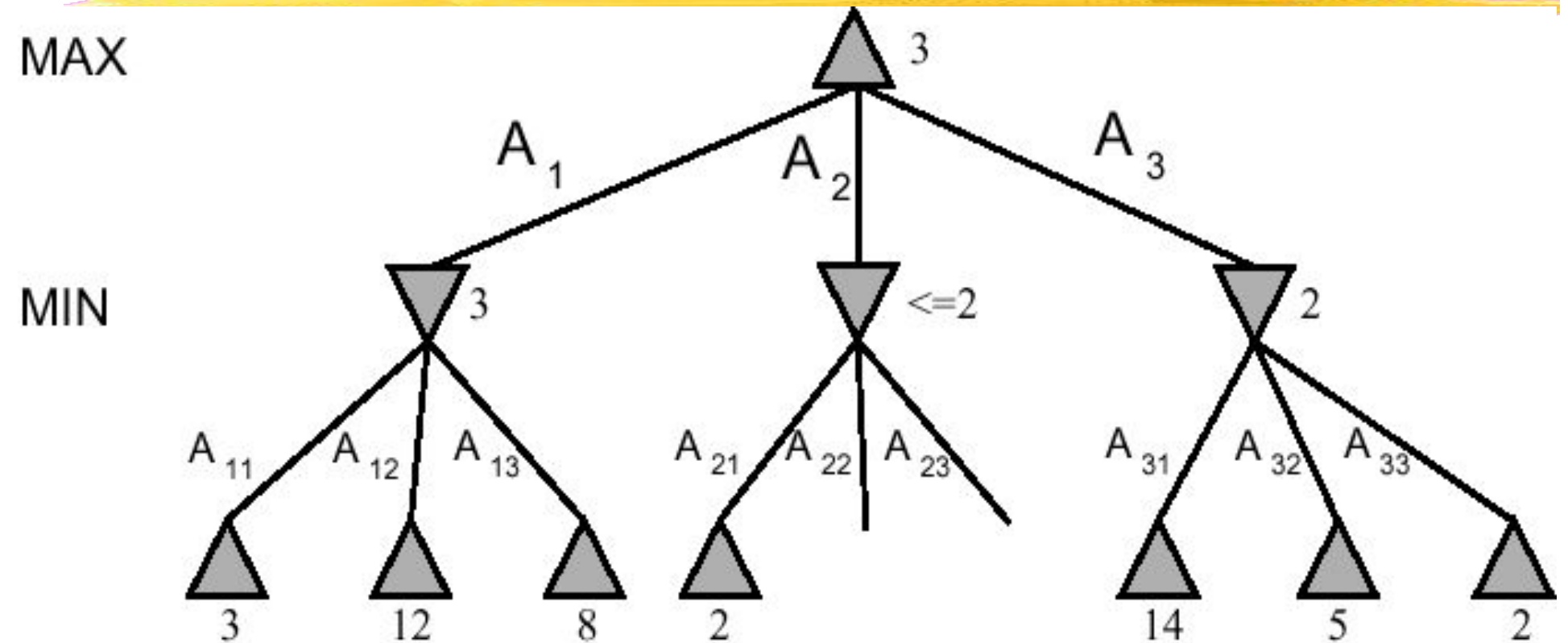


- Função:
 - Não expandir desnecessariamente nós durante o minimax.
- Idéia: não vale a pena piorar, se já achou algo melhor
- Mantém 2 parâmetros:
 - α - melhor valor (no caminho) para MAX
 - β - melhor valor (no caminho) para MIN
- Teste de expansão:
 - α não pode diminuir (não pode ser menor que um ancestral)
 - β não pode aumentar (não pode ser maior que um ancestral)

Alpha-Beta Pruning: exemplo



Alpha-Beta Pruning: exemplo



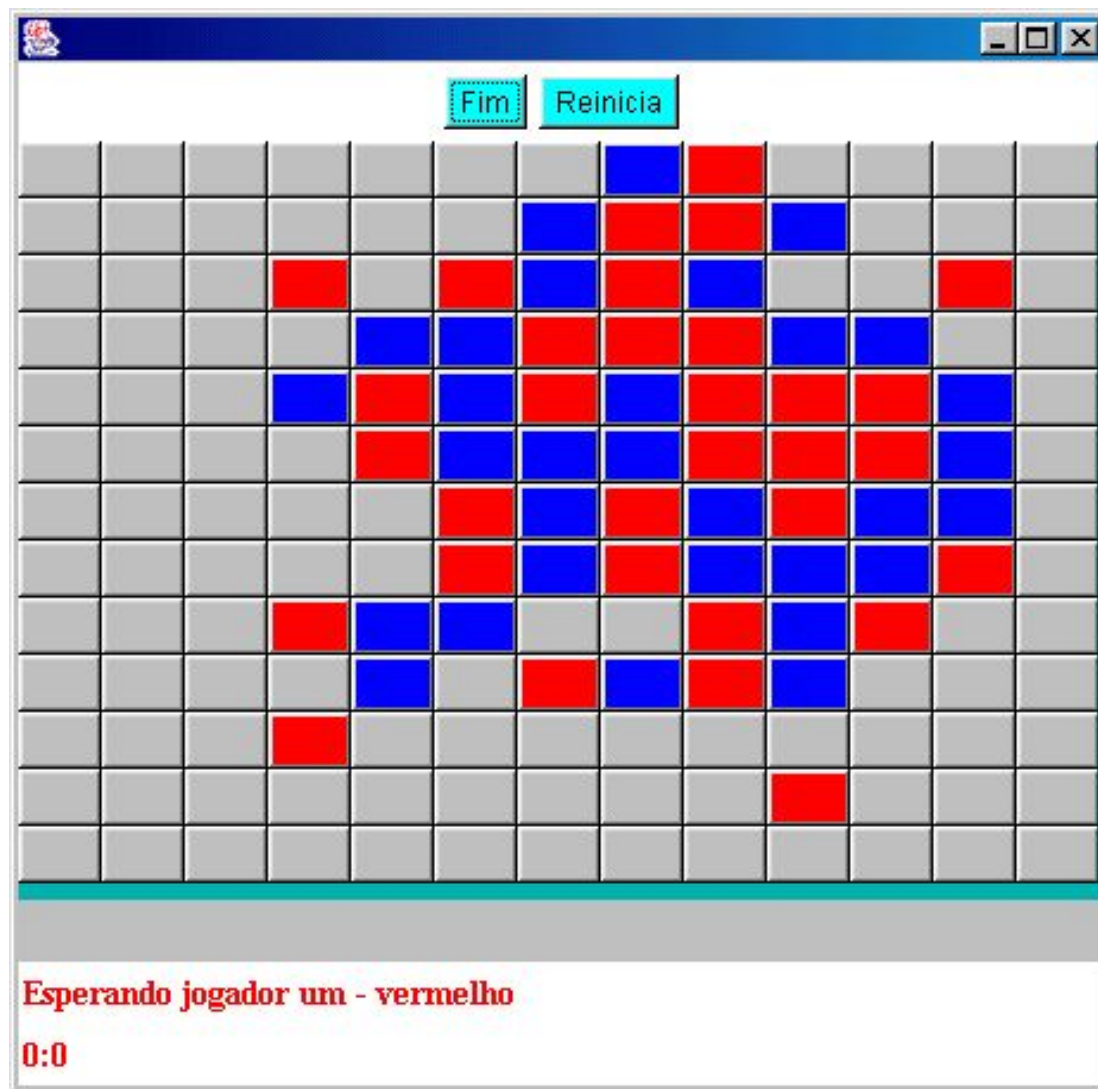
Limitações do Minimax



- O algoritmo Minimax com horizonte limitado depende da função de avaliação.
- Boas funções de avaliação não estão disponíveis para todos os problemas.

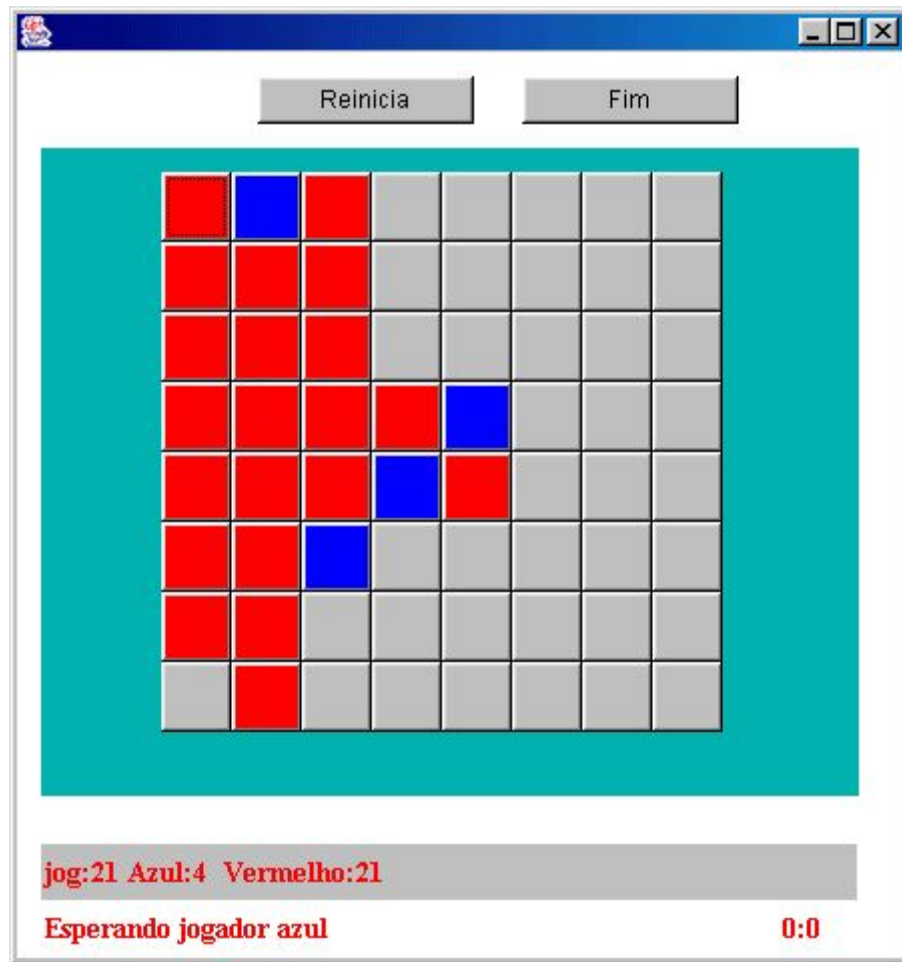
A $\text{C}_{10}\text{H}_{16}\text{O}$ $\text{C}_{10}\text{H}_{16}\text{O}$ $\text{C}_{10}\text{H}_{16}\text{O}$

Ano de 2002 - GoMoku



Jogos de tabuleiro

Ano de 2001 - Reversi (Othelo)



Jogos de tabuleiro



- Foi implementado um servidor de jogos que é responsável pela apresentação do tabuleiro.
- Cada programa representando um jogador comunica-se com o servidor via TCP-IP.
- O jogador é limitado por uma profundidade máxima que a árvore minimax pode expandir.
- A programação do servidor é feita em Java.
- São fornecidos “esqueletos” de programa de jogadores em Java contendo a rotina de comunicação, mas os grupos podem optar por implementar os jogadores em outras linguagens.
- Os três primeiros lugares são premiados com um número extra de pontos.



Método de Monte Carlo para busca em árvore

Método de Monte Carlo para busca em árvore

- Utiliza amostragem aleatória para aproximar valores.
- Parte do Projeto Manhattan nos anos 40.



Stanisław Ulam

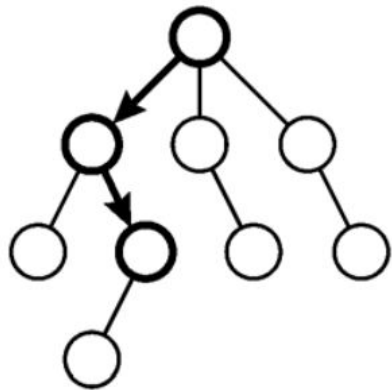
Método de Monte Carlo para busca em árvore

- Utiliza amostragem aleatória para aproximar o valor minimax.
- Revolução recente (~2006) em IA. (vitória no jogo Go)
- Minimax expande toda a árvore até nível d .
- Busca Monte Carlo expande os ramos “mais promissores” até os nós terminais. (diminui a importância das funções de avaliação)

Método de Monte Carlo para busca em árvore

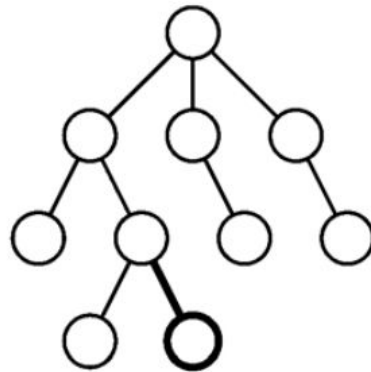
- Repete-se um número de iterações igual ao tempo disponível.
- Escolhe-se a ação com maior valor esperado para a recompensa.

Seleção



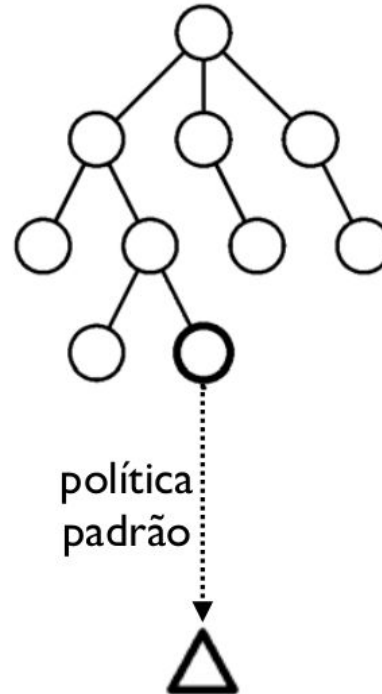
política de
árvore

Expansão



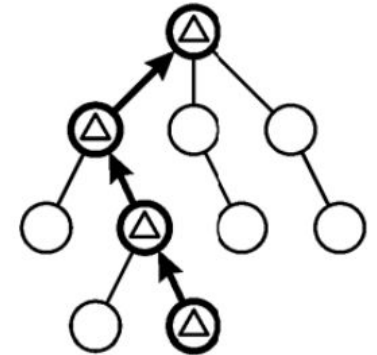
política de
árvore

Simulação

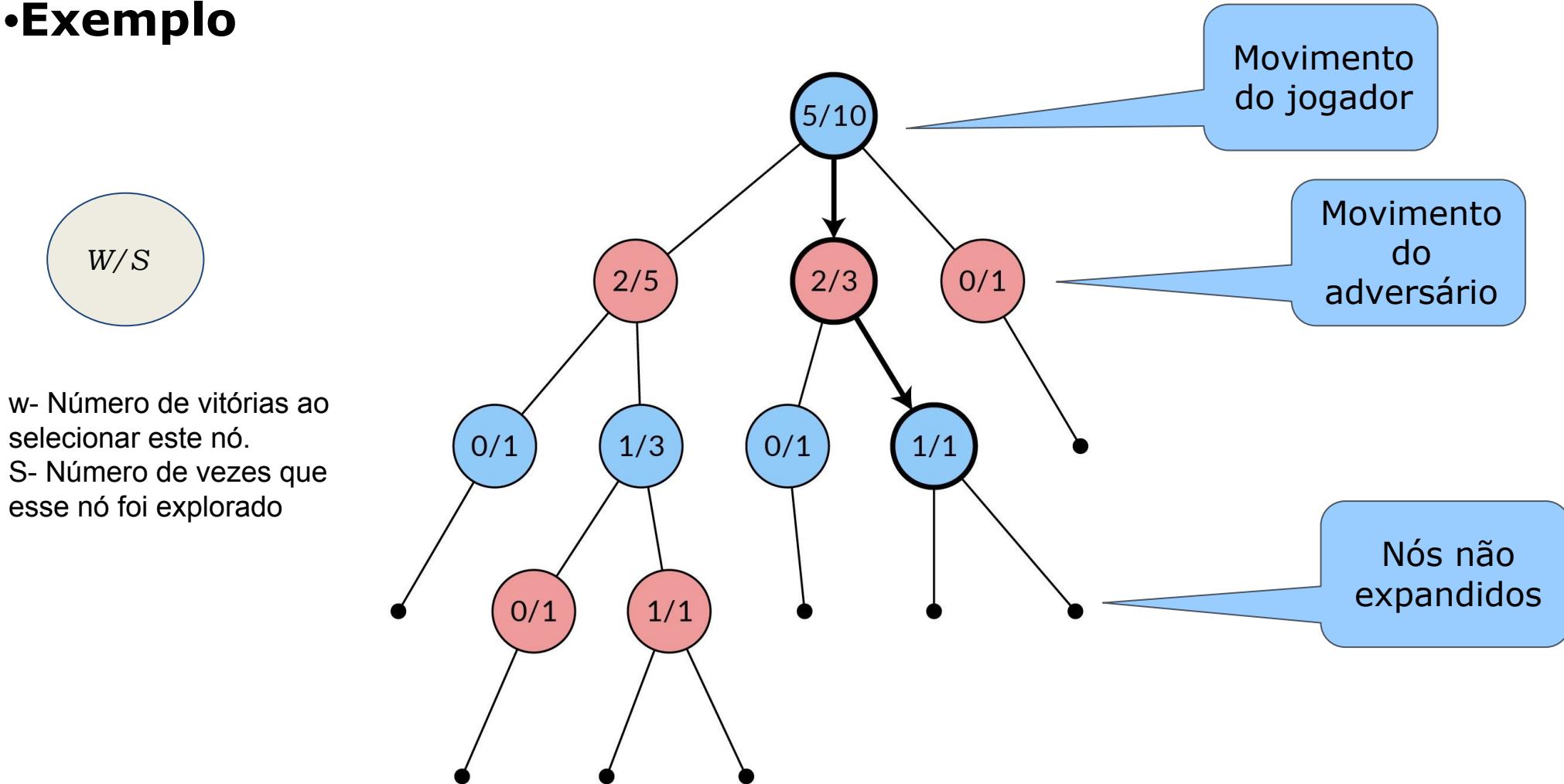


política
padrão

Retro-propagação



•Exemple



Método de Monte Carlo para busca em árvore



•Etapa de seleção

Começando no nó raiz, descemos a árvore repetidamente:

(1) selecionando uma ação; e

(2) avançando para o nó filho correspondente. Se pelo menos uma ação não tiver um nó correspondente na árvore de pesquisa, interrompemos a seleção.

Método de Monte Carlo para busca em árvore



•Etapa de seleção

A seleção de caminhos deve atingir dois objetivos:

Explorar novos caminhos para obter informações
e usar as informações existentes para explorar caminhos que são
bons.

*Balancear a exploração e exploração (aproveitamento)
para minimizar o arrependimento*

Método de Monte Carlo para busca em árvore

•Etapa de seleção

O balanceamento pode ser obtido com a fórmula de **Upper Confidence Bounds (UCB1)**.

$$\frac{w_i}{s_i} + c \sqrt{\frac{\ln s_p}{s_i}}$$

Parâmetro de
exploração (alto
para nós com muitas
vitórias)

Parâmetro de
exploração (alto para nós
pouco explorados)

w_i : número de simulações que resultaram em vitória

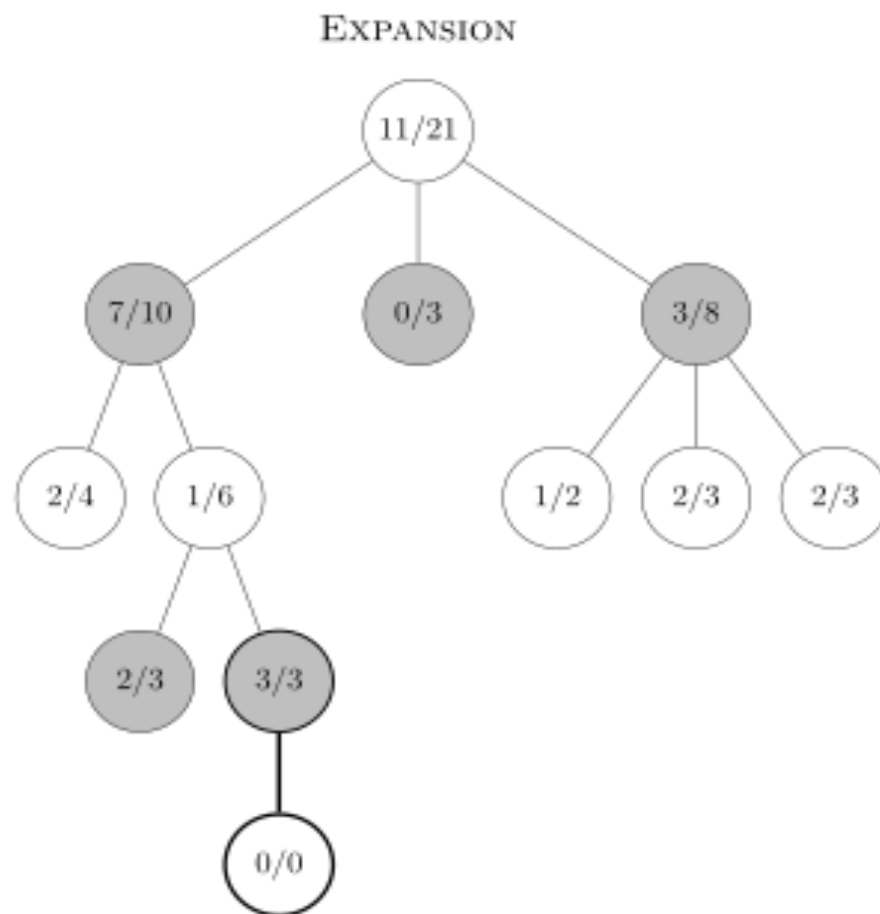
s_i : número total de simulações neste nó

s_p : número total de simulações do nó pai

c : parâmetro de exploração

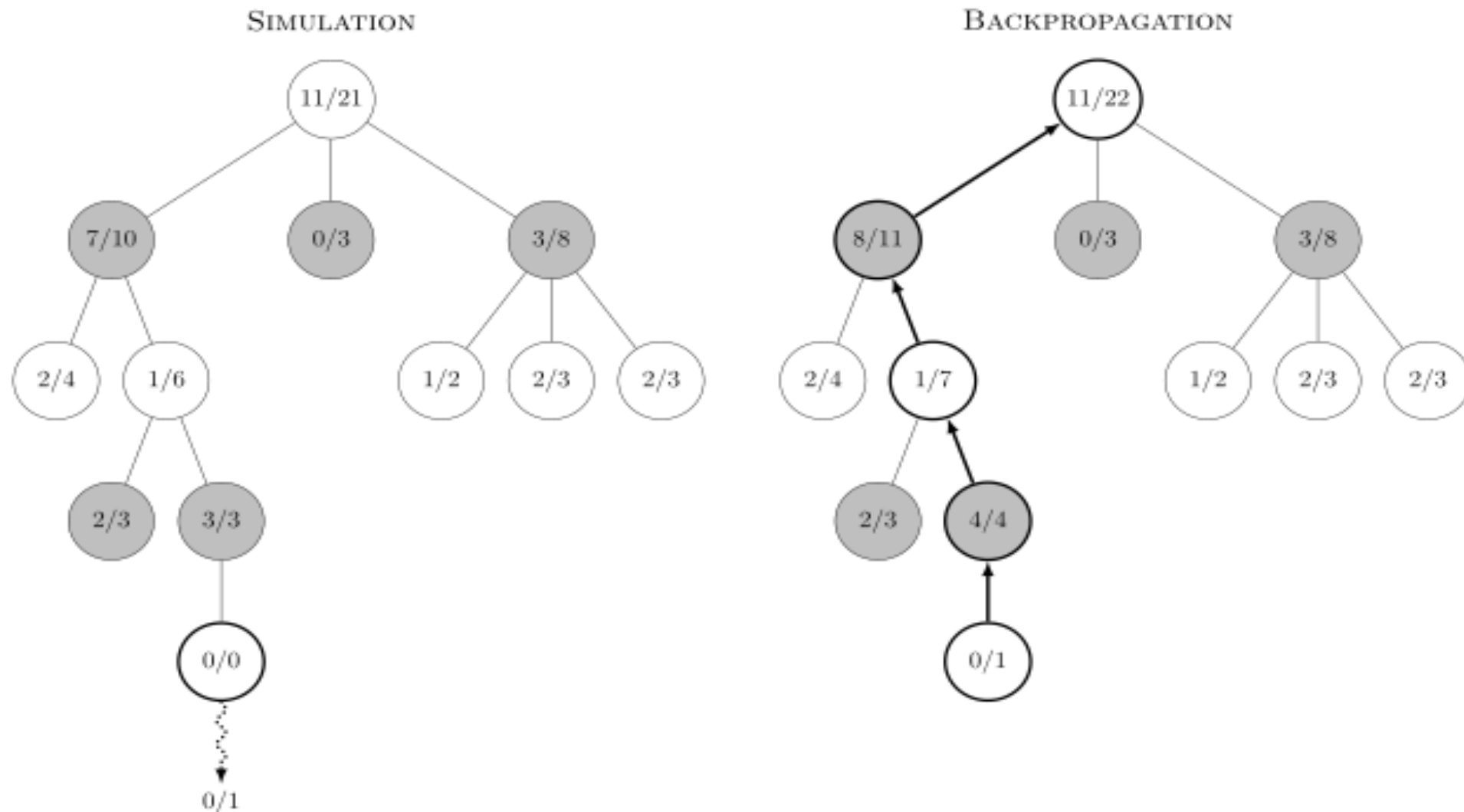
Método de Monte Carlo para busca em árvore

•Etapa de seleção/expansão



Método de Monte Carlo para busca em árvore

•Etapa de simulação/retropropagação





FIM