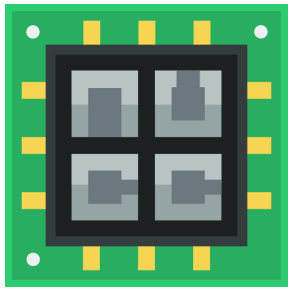
AI aplicada a jogos adversariais

Afonso Fernandes, Gonçalo Paredes, Ricardo Pereira



16 de Abril de 2019

Conteúdo

Link

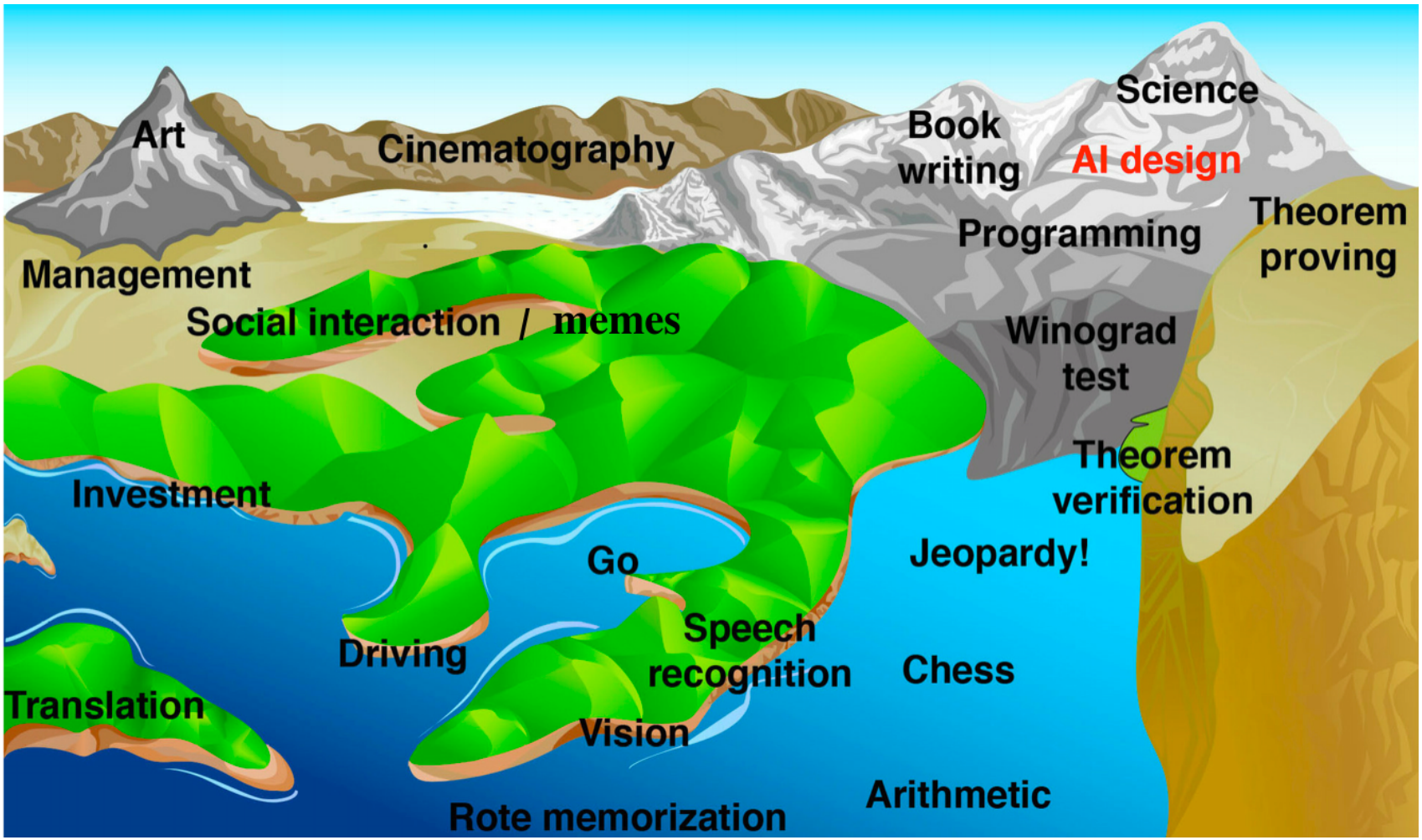
https://afonsonf.github.io/ia-enemath-web/

Conteúdo

Introdução - AI

- Inteligência das máquinas
- Estuda agentes inteligentes
- Raciocínio Automático, Representação de Conhecimento, Processamento de Linguagem Natural, Planeamento, Machine Learning

Introdução - AI



Landscape of Human Intelligence - Hans Moravec

Introdução - AI

Introdução - Jogo

Introdução - Jogo

- Estado inicial (posição + jogador)

Introdução - Jogo

- Estado inicial (posição + jogador)
- Função para gerar sucessores

Introdução - Jogo

- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais

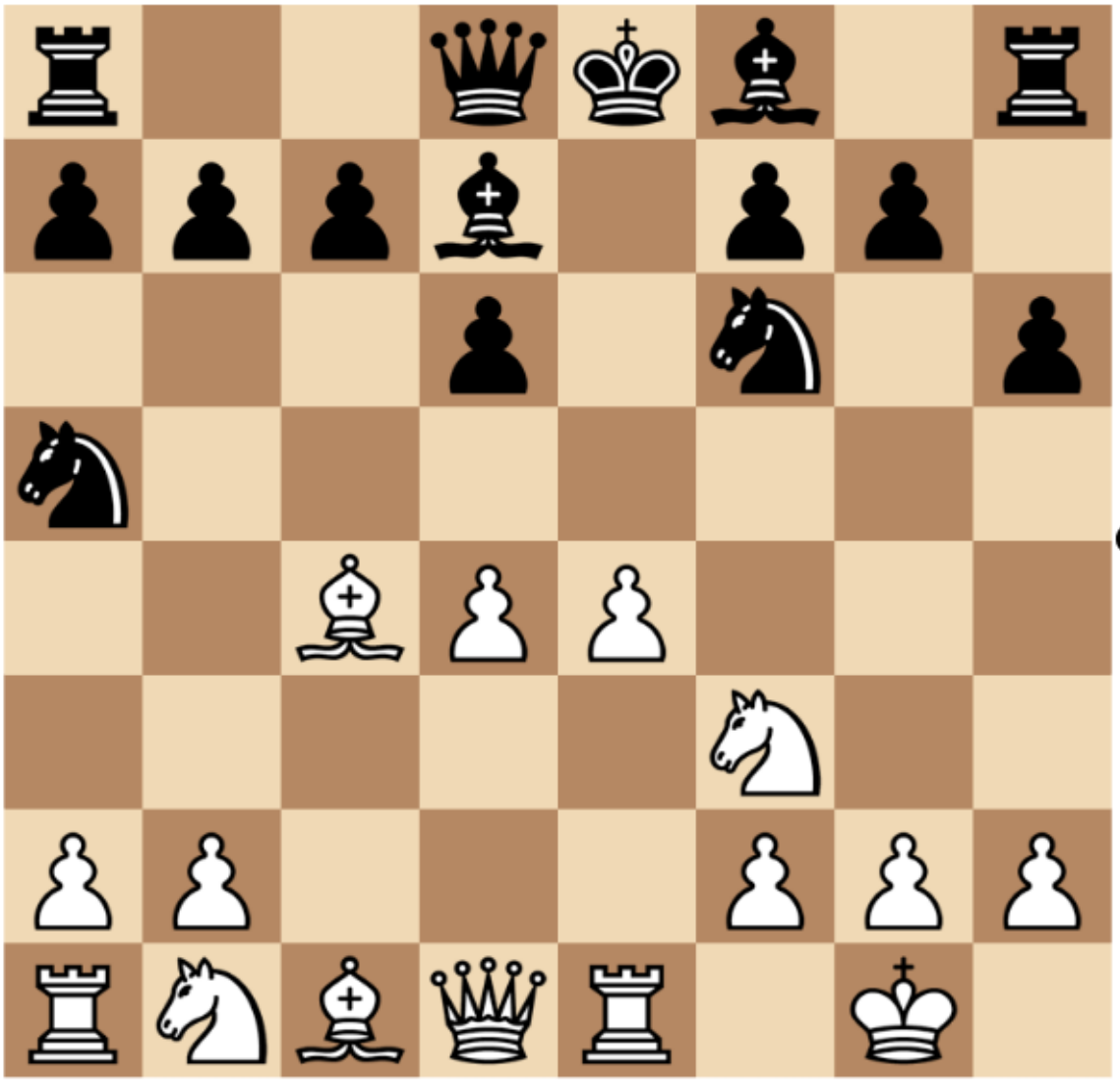
Introdução - Jogo

- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais
- Sequencial

Introdução - Jogo

- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais
- Sequencial
- Informação perfeita

Introdução - Heurística



→ 73

Conteúdo

Minimax

Minimax

1. Método mais simples de decisão em inteligência artificial;

Minimax

1. Método mais simples de decisão em inteligência artificial;
2. Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;

Minimax

1. Método mais simples de decisão em inteligência artificial;
2. Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;
3. Originalmente desenvolvido para jogos de 2 jogadores de soma nula;

Minimax

1. Método mais simples de decisão em inteligência artificial;
2. Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;
3. Originalmente desenvolvido para jogos de 2 jogadores de soma nula;
4. Tem versões mais complexas, mas vamos apenas explorar a mais simples.

Minimax

Minimax

1. Explorar todas as jogadas possíveis;

Minimax

1. Explorar todas as jogadas possíveis;
2. Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.

Minimax

1. Explorar todas as jogadas possíveis;
2. Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.
3. No caso de uma jogada terminar o jogo atribuir um valor C ou $-C$ consoante quem ganha;

Minimax

1. Explorar todas as jogadas possíveis;
2. Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.
3. No caso de uma jogada terminar o jogo atribuir um valor C ou $-C$ consoante quem ganha;
4. Para evitar cálculos que demorem anos, a partir de certa profundidade usa uma heurística para atribuir valor.

Minimax - Pseudo-Código

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Minimax - Exemplo

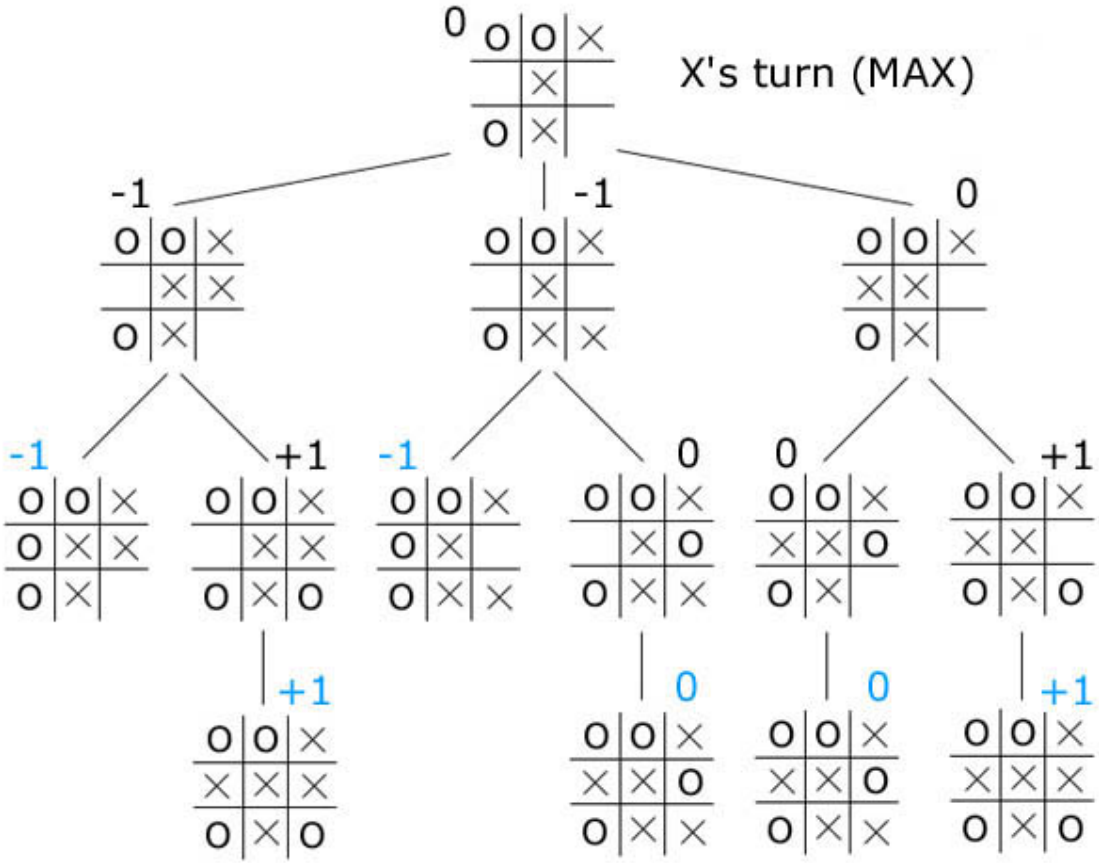


Figura: Exemplo de uma árvore de aplicação do Minimax no jogo do galo.

Minimax - Problemas

Minimax - Problemas

1. Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;

Minimax - Problemas

1. Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;
2. Para valores pequenos de profundidade máxima joga fortemente de acordo com a heurística escolhida;
3. Para valores maiores demora muito a determinar a próxima jogada;

Minimax - Problemas

1. Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;
2. Para valores pequenos de profundidade máxima joga fortemente de acordo com a heurística escolhida;
3. Para valores maiores demora muito a determinar a próxima jogada;
4. É um método muito próximo da força bruta, testando as possibilidades todas.

Conteúdo

Alpha-beta pruning

Alpha-beta pruning

- Baseia-se no minimax

Alpha-beta pruning

- Baseia-se no minimax
- Melhora-o cortando nós que não é necessário explorar

Alpha-beta pruning

- Baseia-se no minimax
- Melhora-o cortando nós que não é necessário explorar
- O ganho em tempo depende de vários fatores, mas geralmente é melhor que o minimax

Alpha-beta - Pseudo-Código

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if α ≥ β then
                break (* α cut-off *)
        return value
```

Alpha-beta - Exemplo

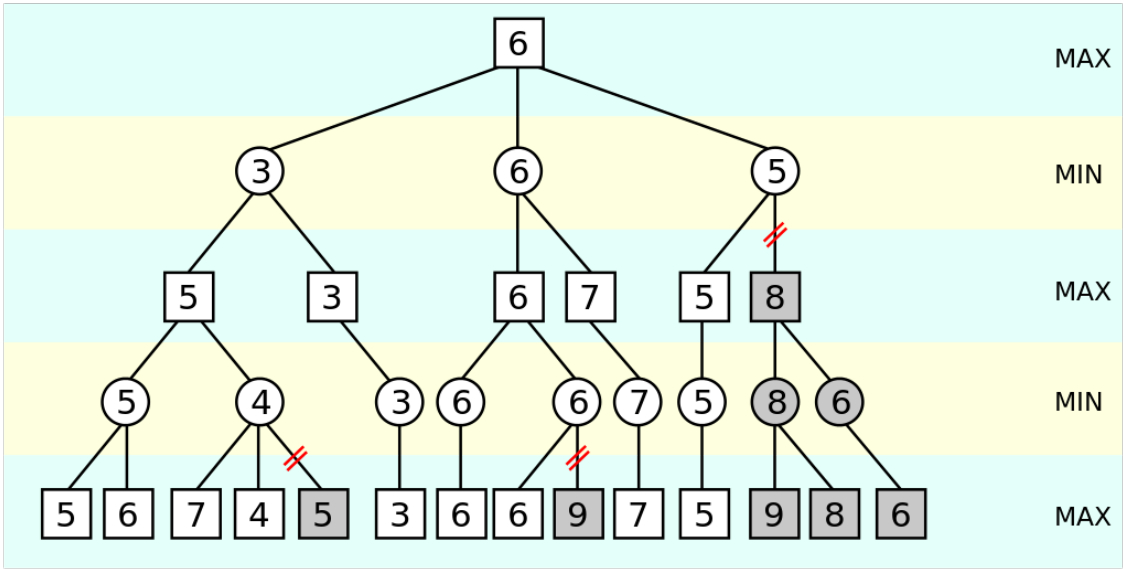
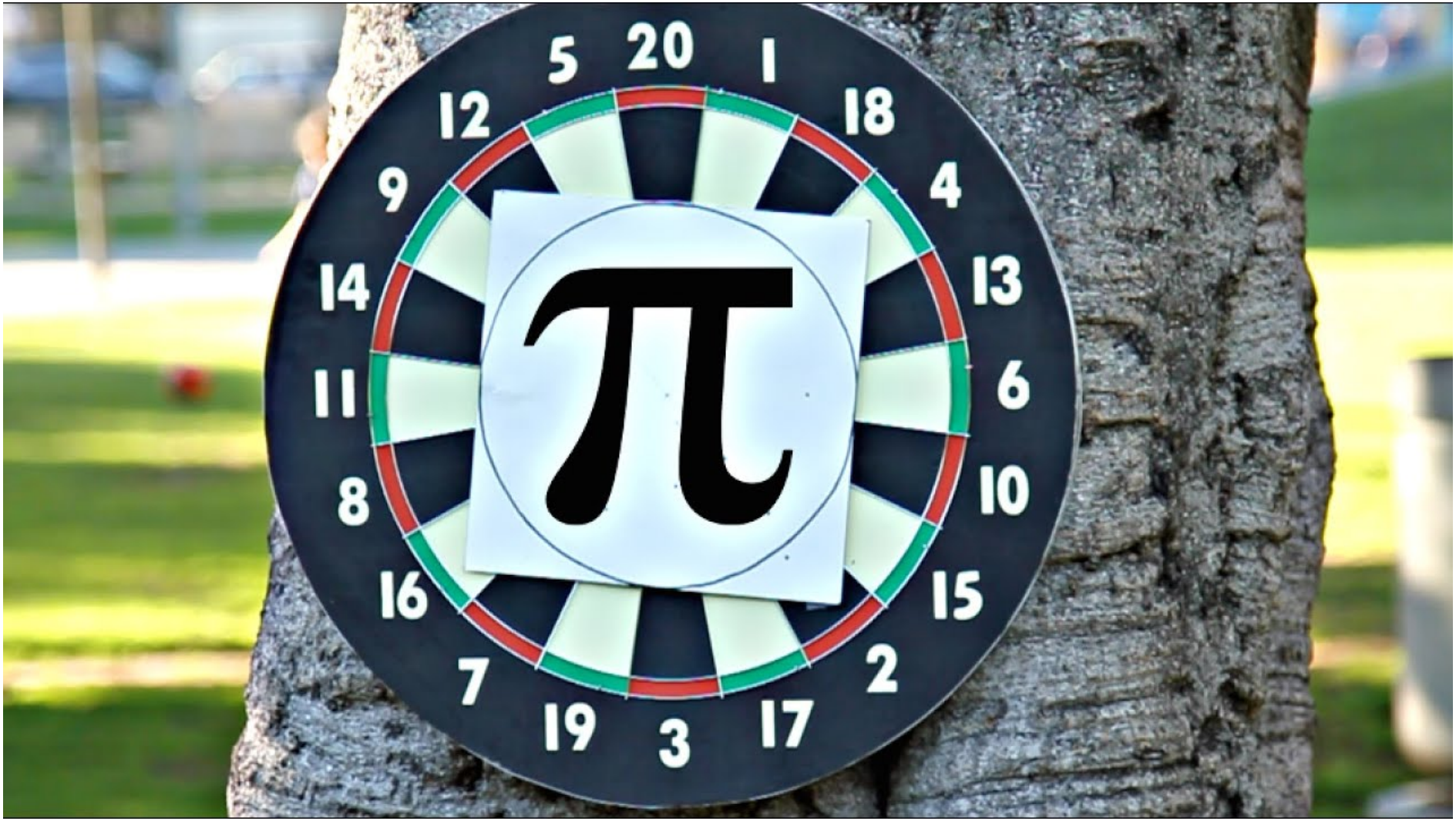


Figura: Exemplo de uma árvore de aplicação de *Alpha-beta pruning*.

Implementação

Conteúdo

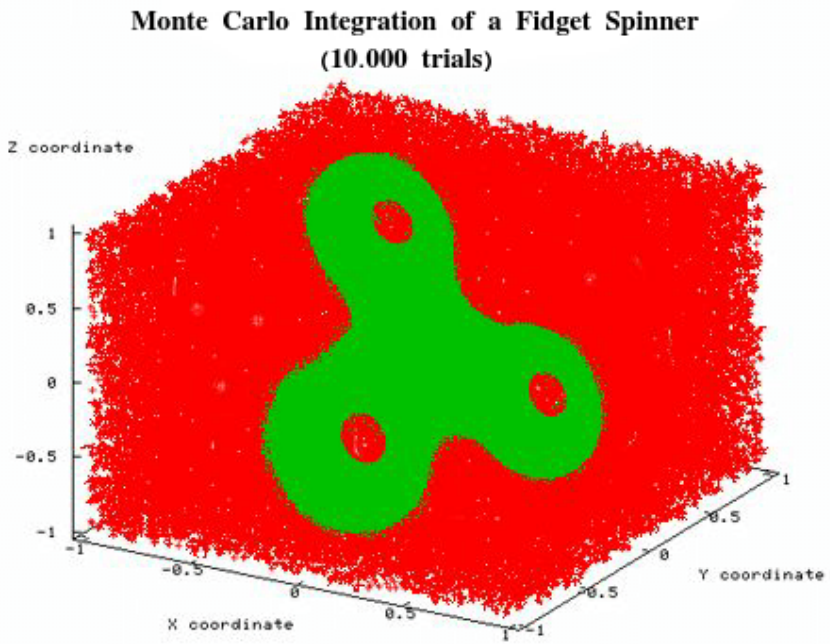
MCTS - Métodos de Monte Carlo



MCTS - Métodos de Monte Carlo

1985: I bet they'll have flying cars in the future

2017:

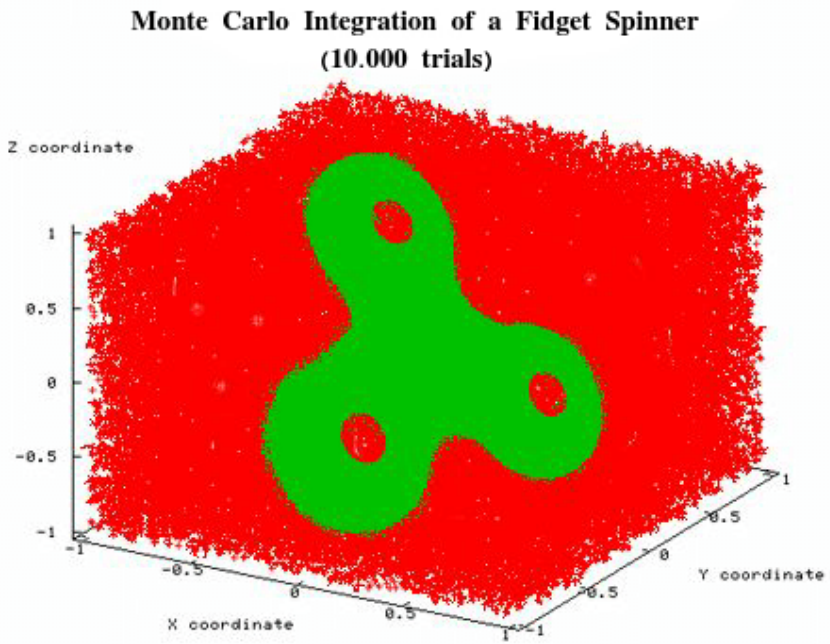


Integração

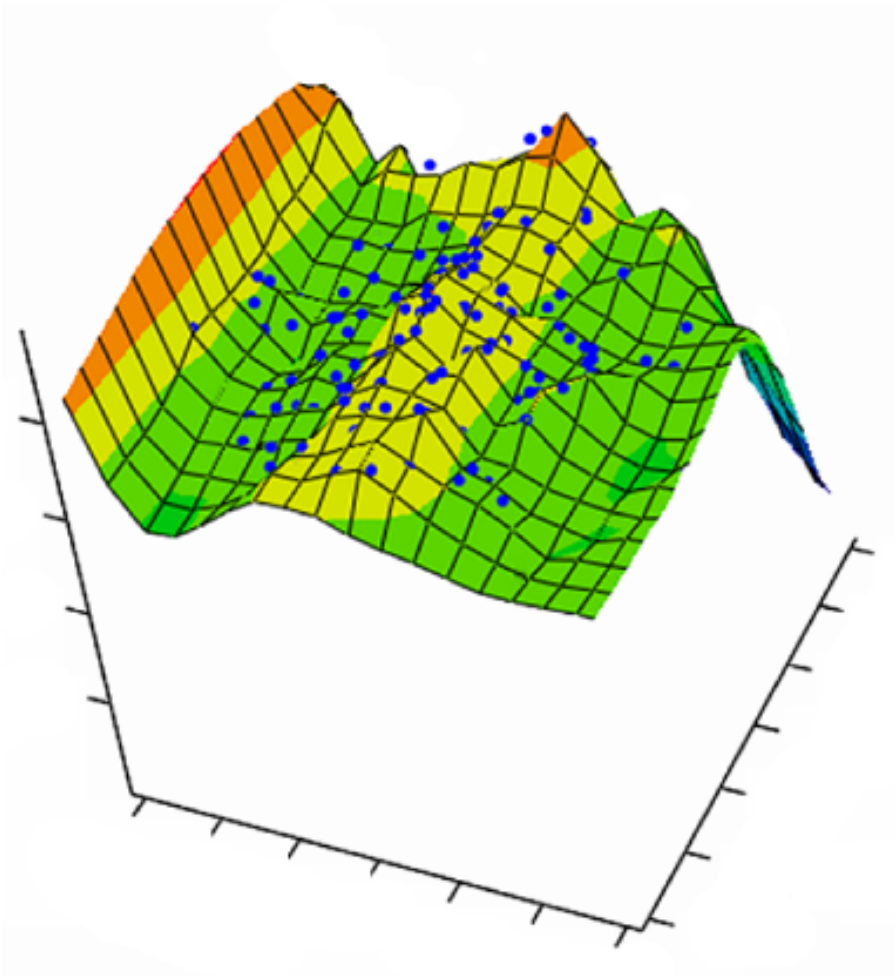
MCTS - Métodos de Monte Carlo

1985: I bet they'll have flying cars in the future

2017:

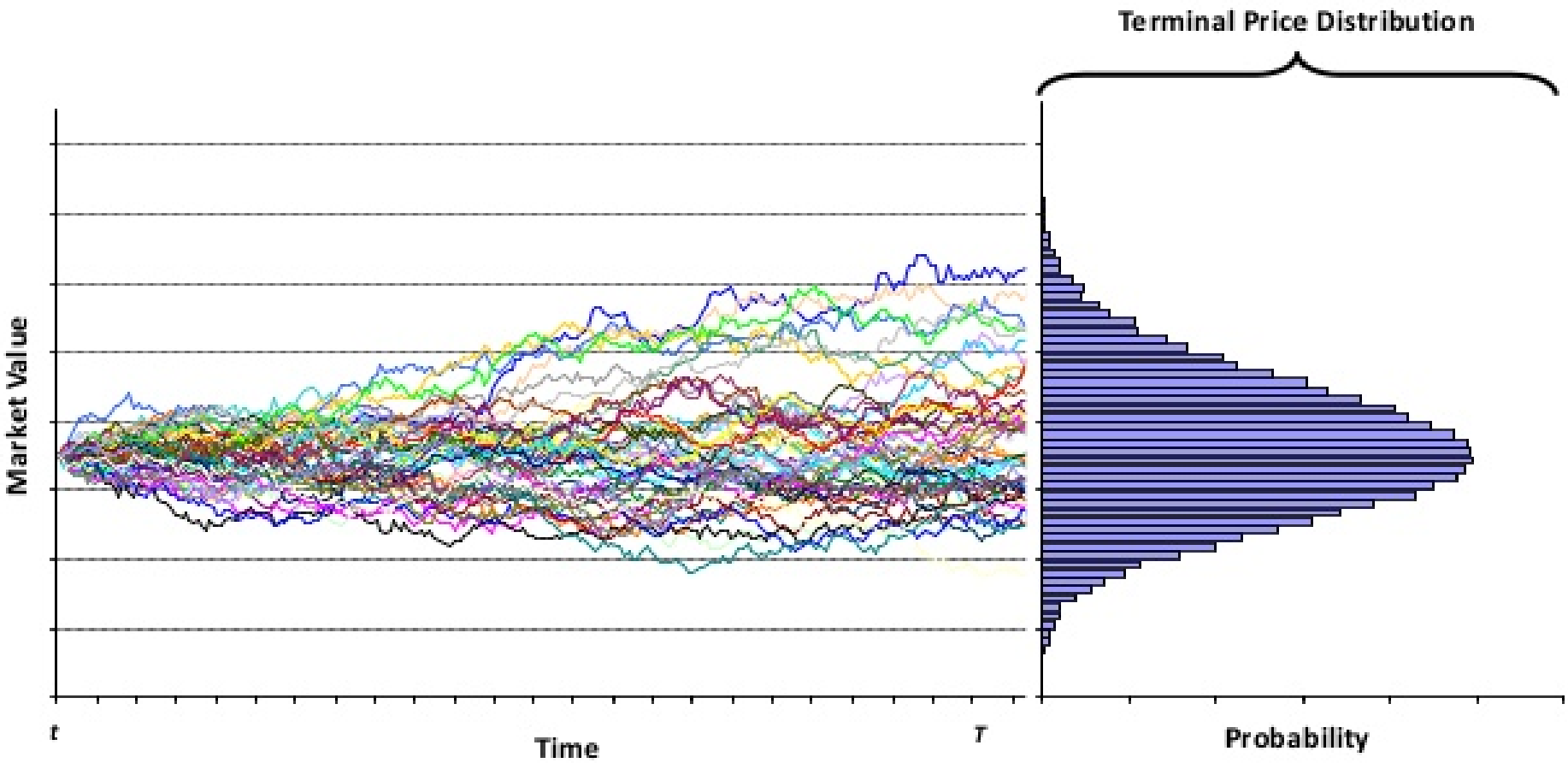


Integração



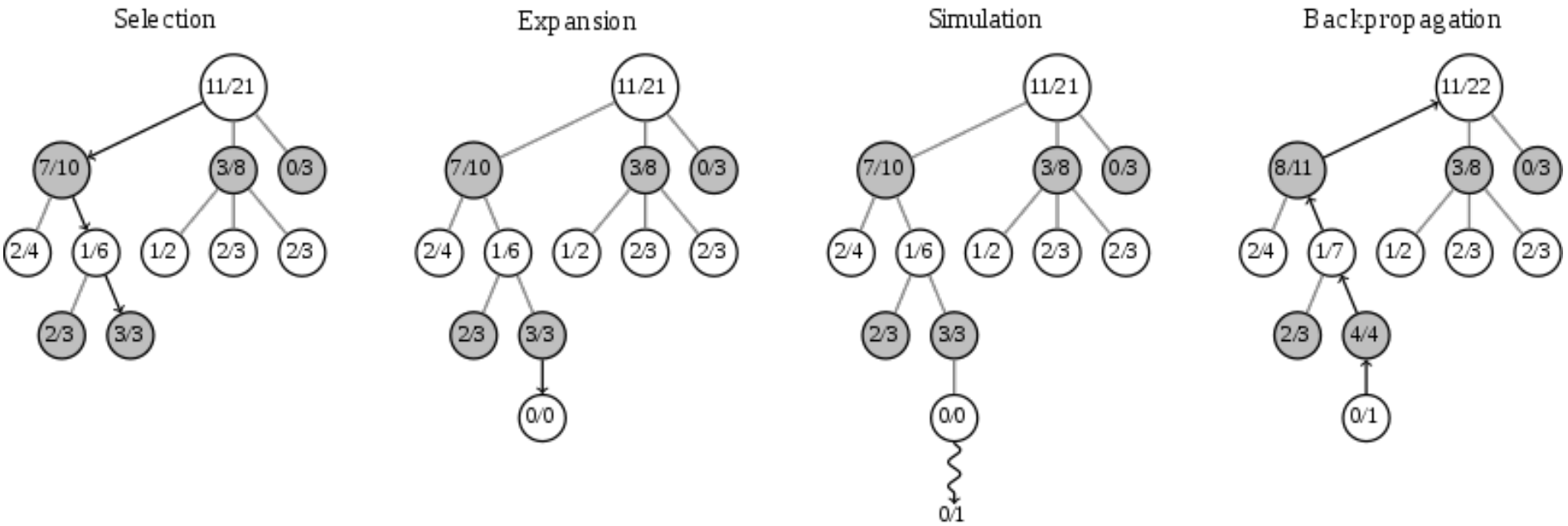
Otimização

MCTS - Métodos de Monte Carlo

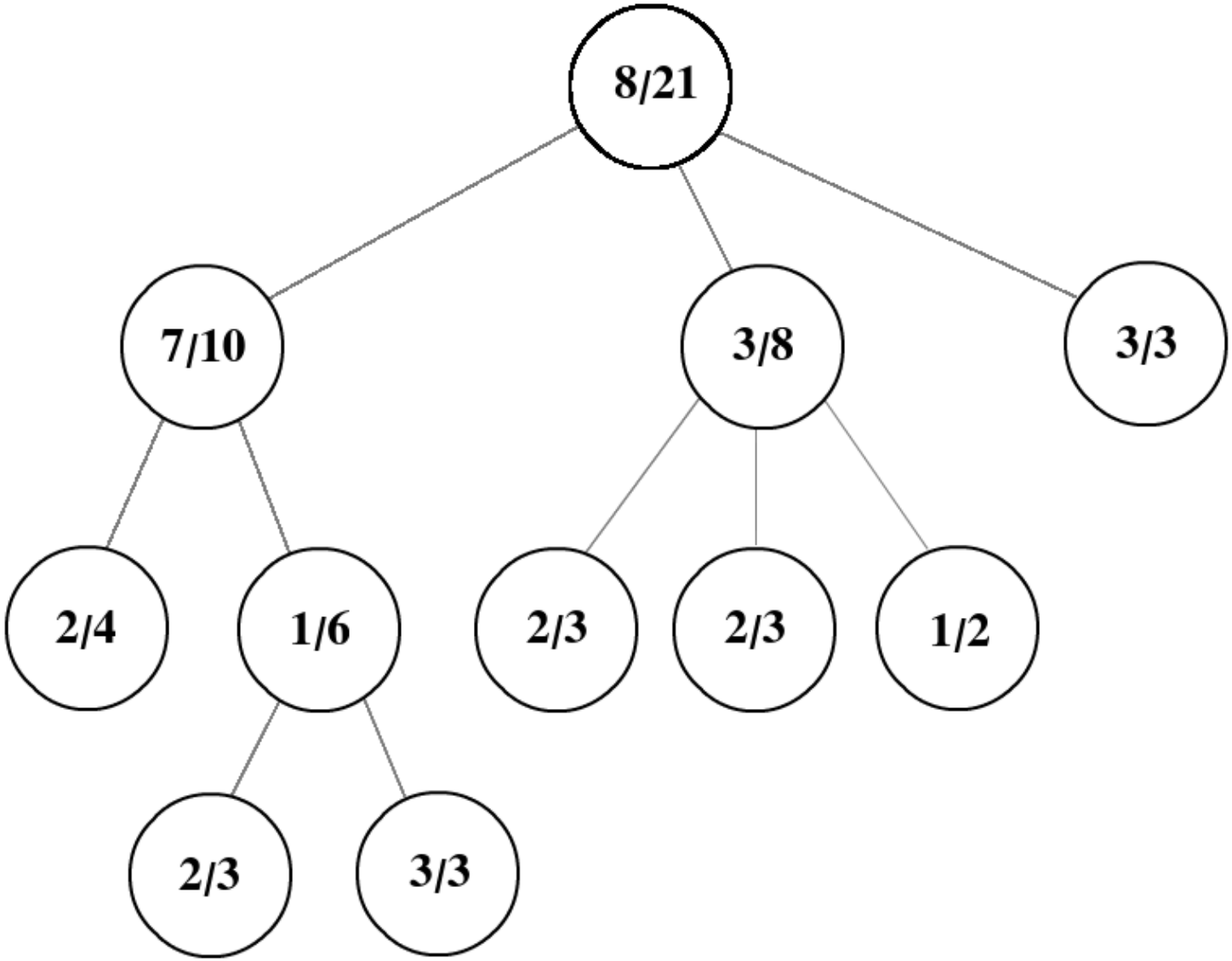


Mercado

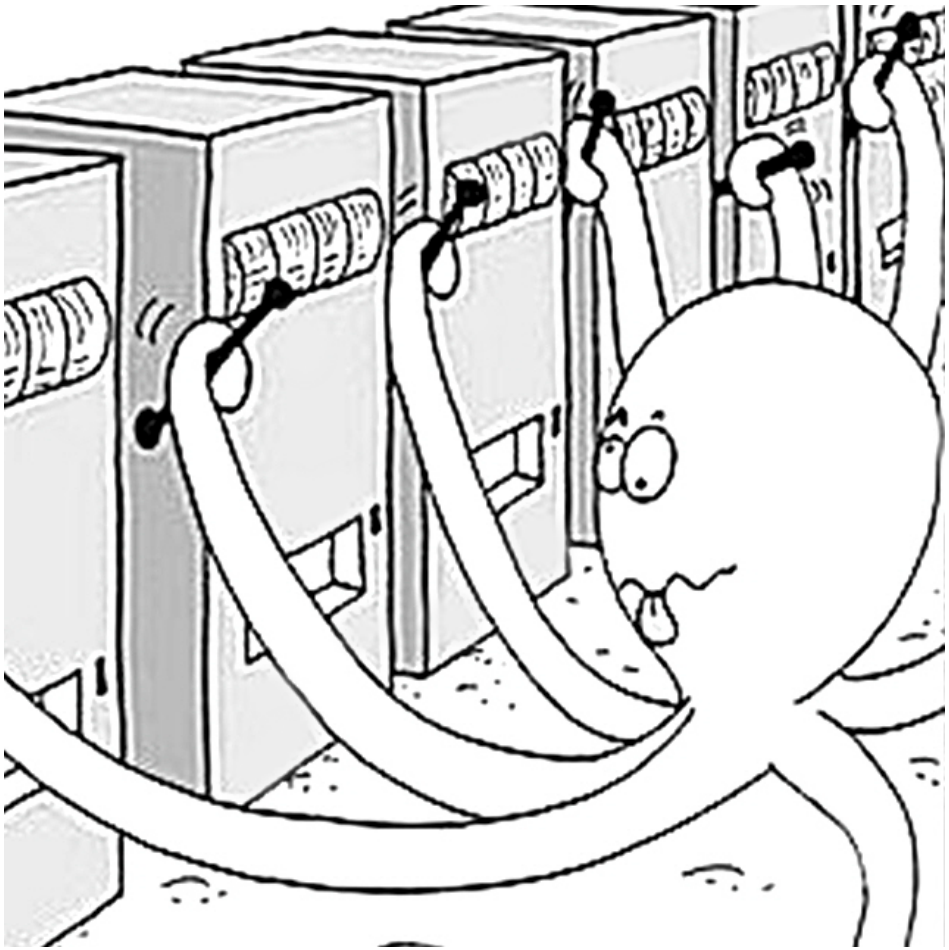
MCTS - Overview



MCTS - Estado Atual

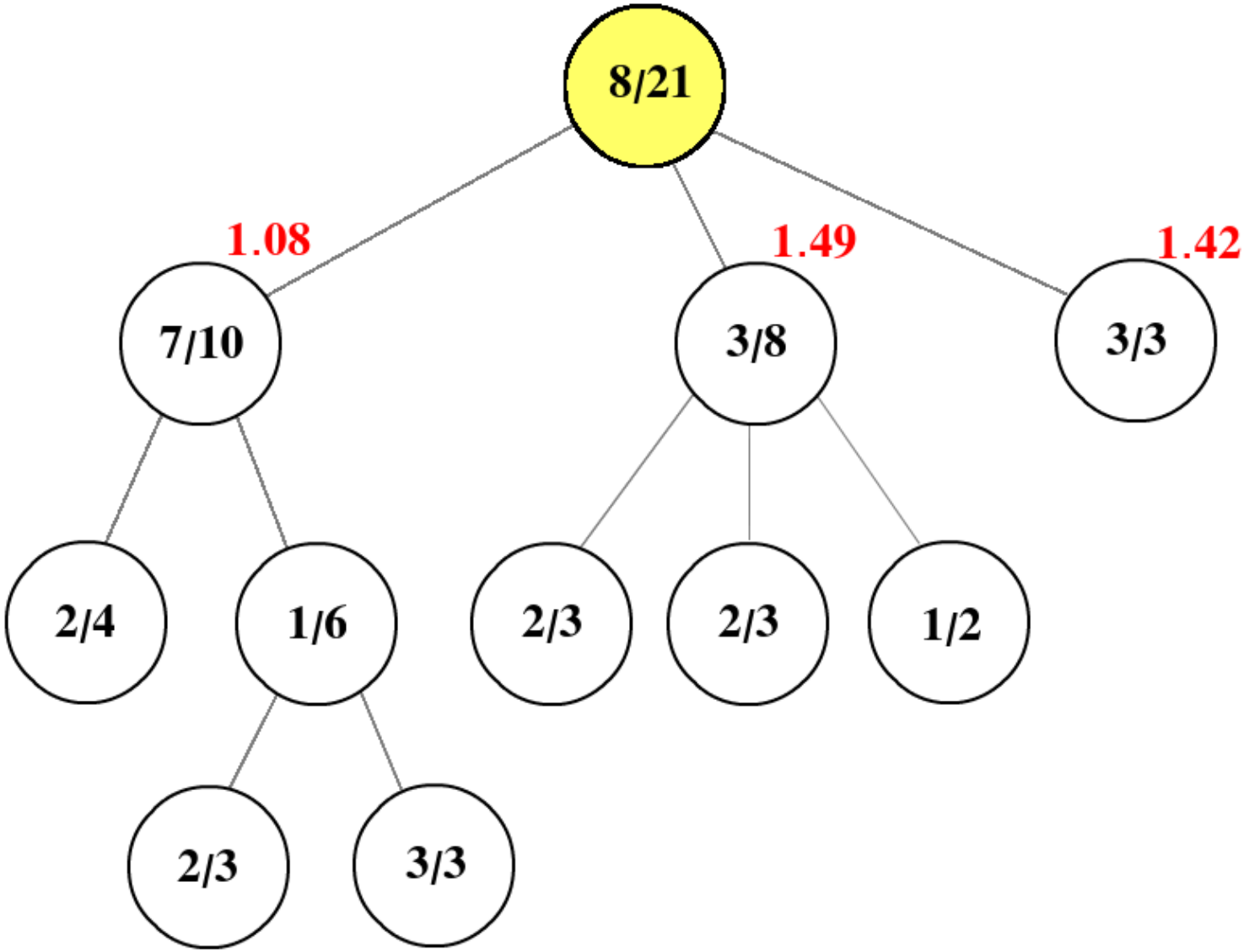


MCTS - Multi Armed Bandit

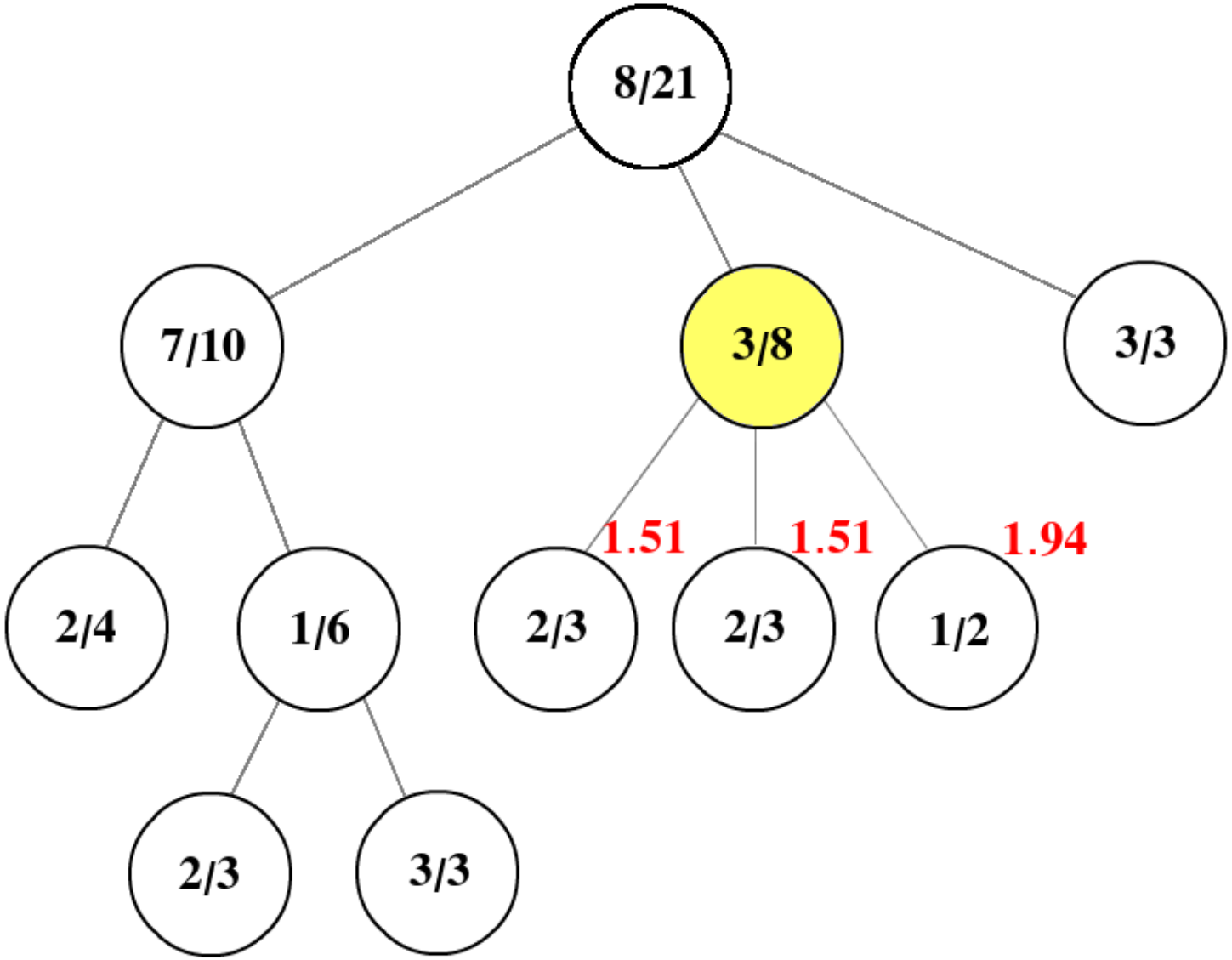


$$\frac{w}{n} + c\sqrt{\frac{lnN}{n}}$$

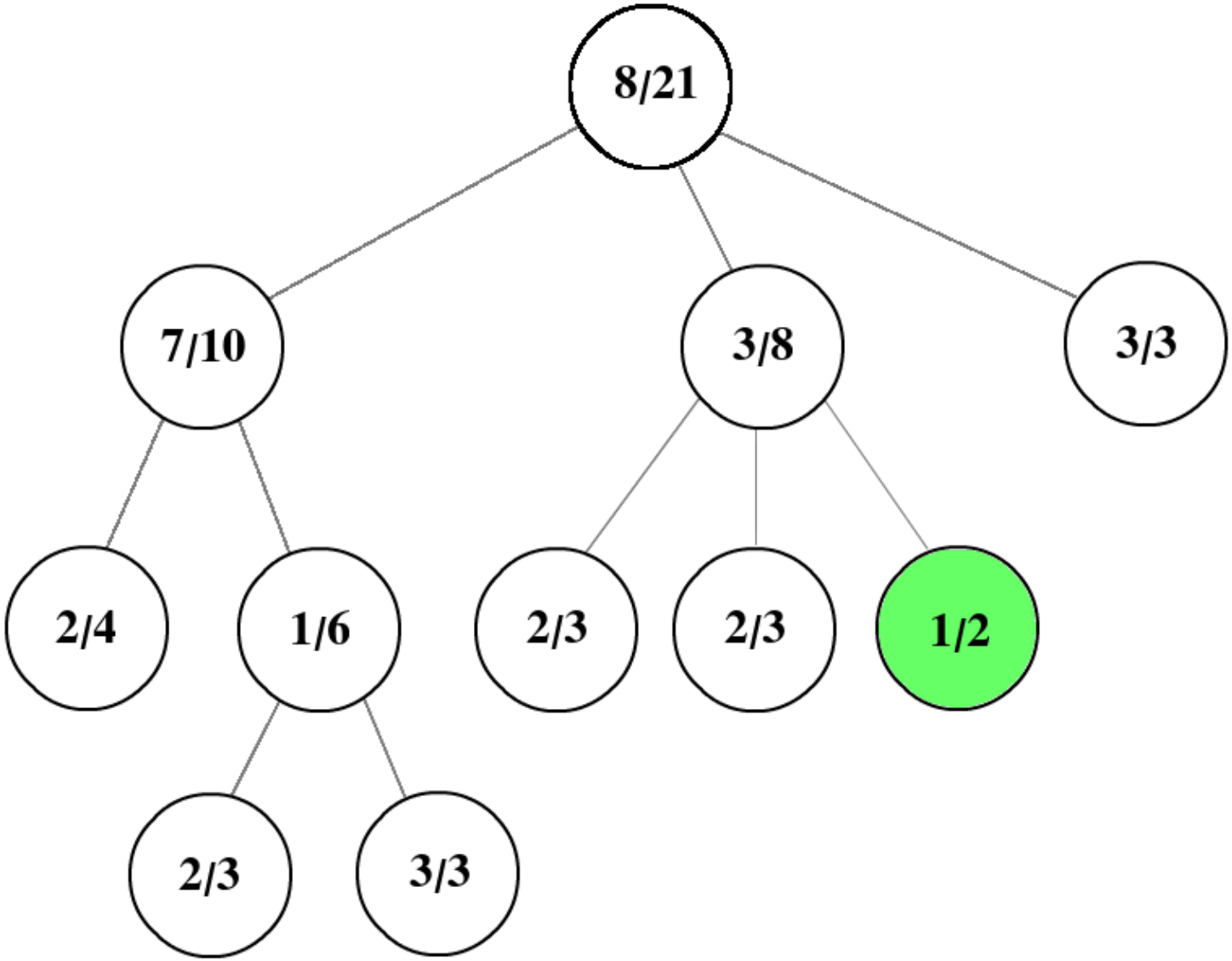
MCTS - Seleção



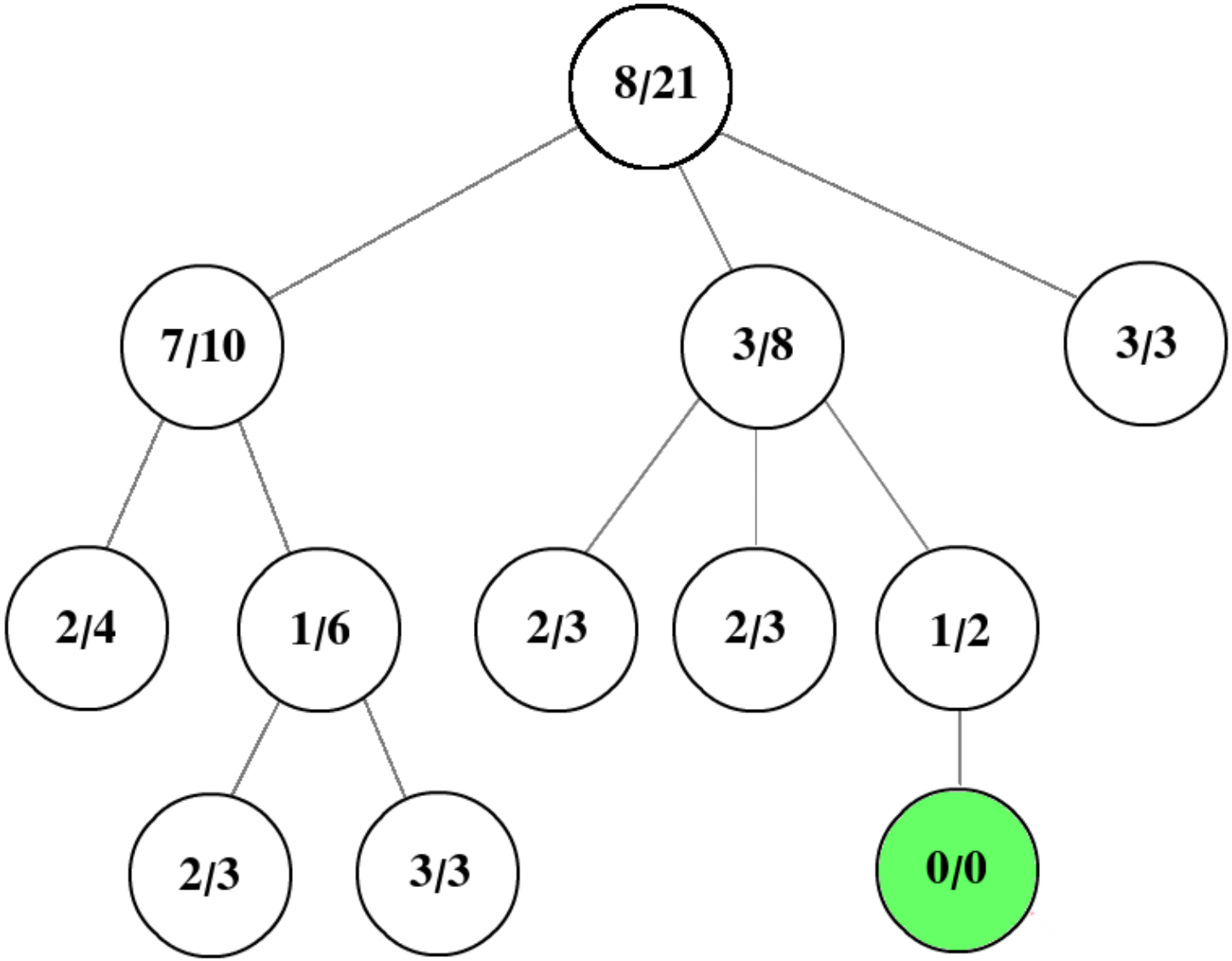
MCTS - Seleção



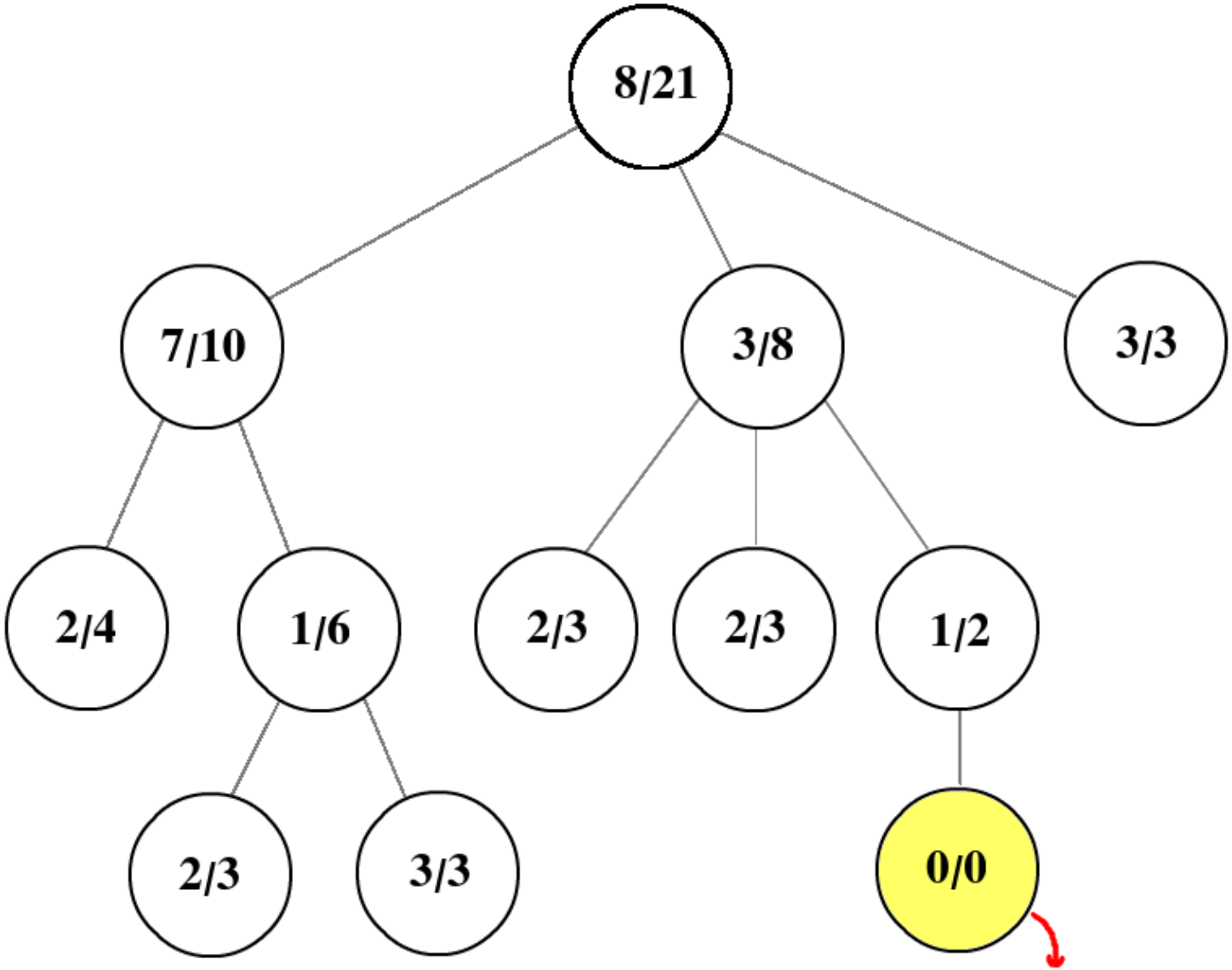
MCTS - Seleção



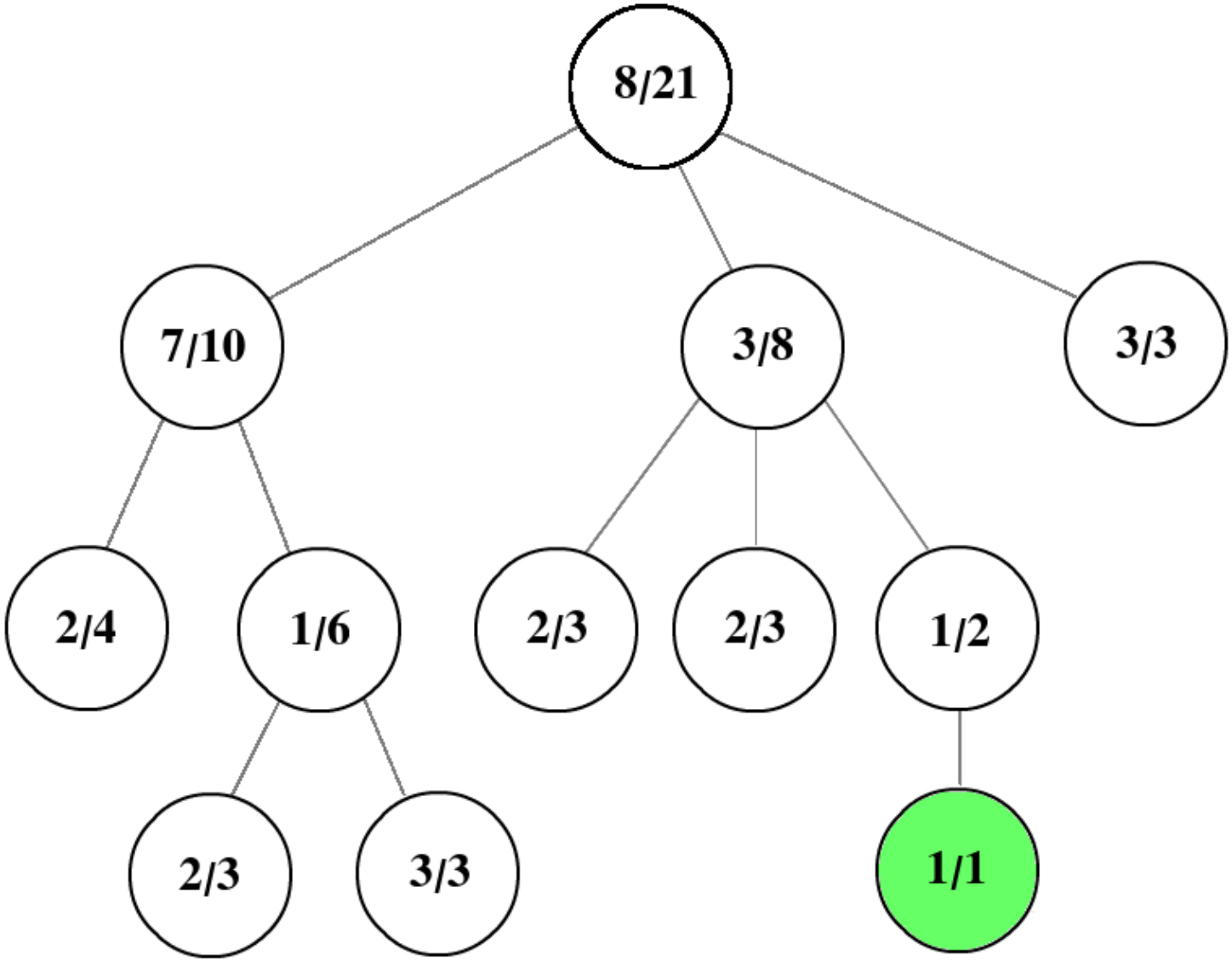
MCTS - Expansão



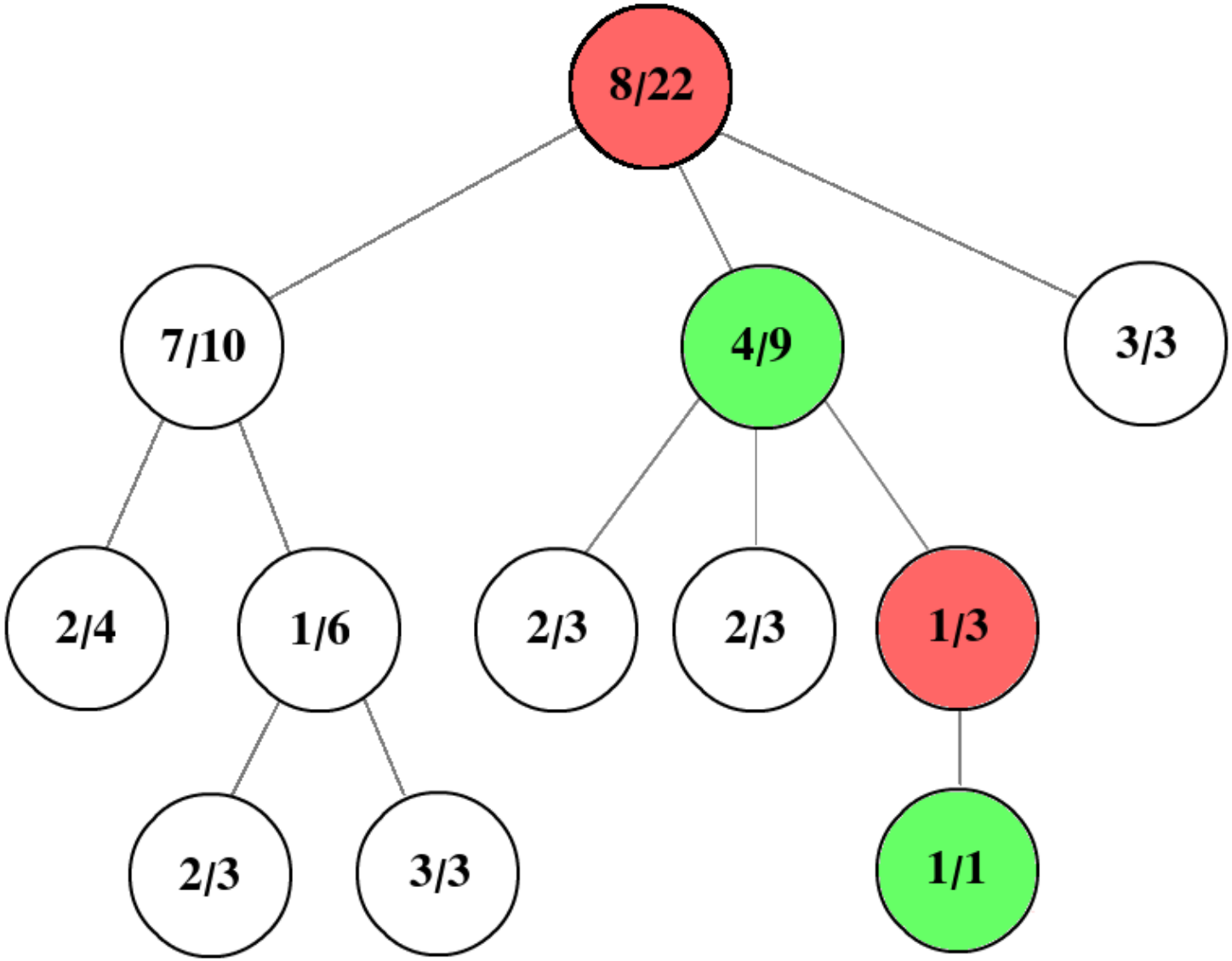
MCTS - Simulação



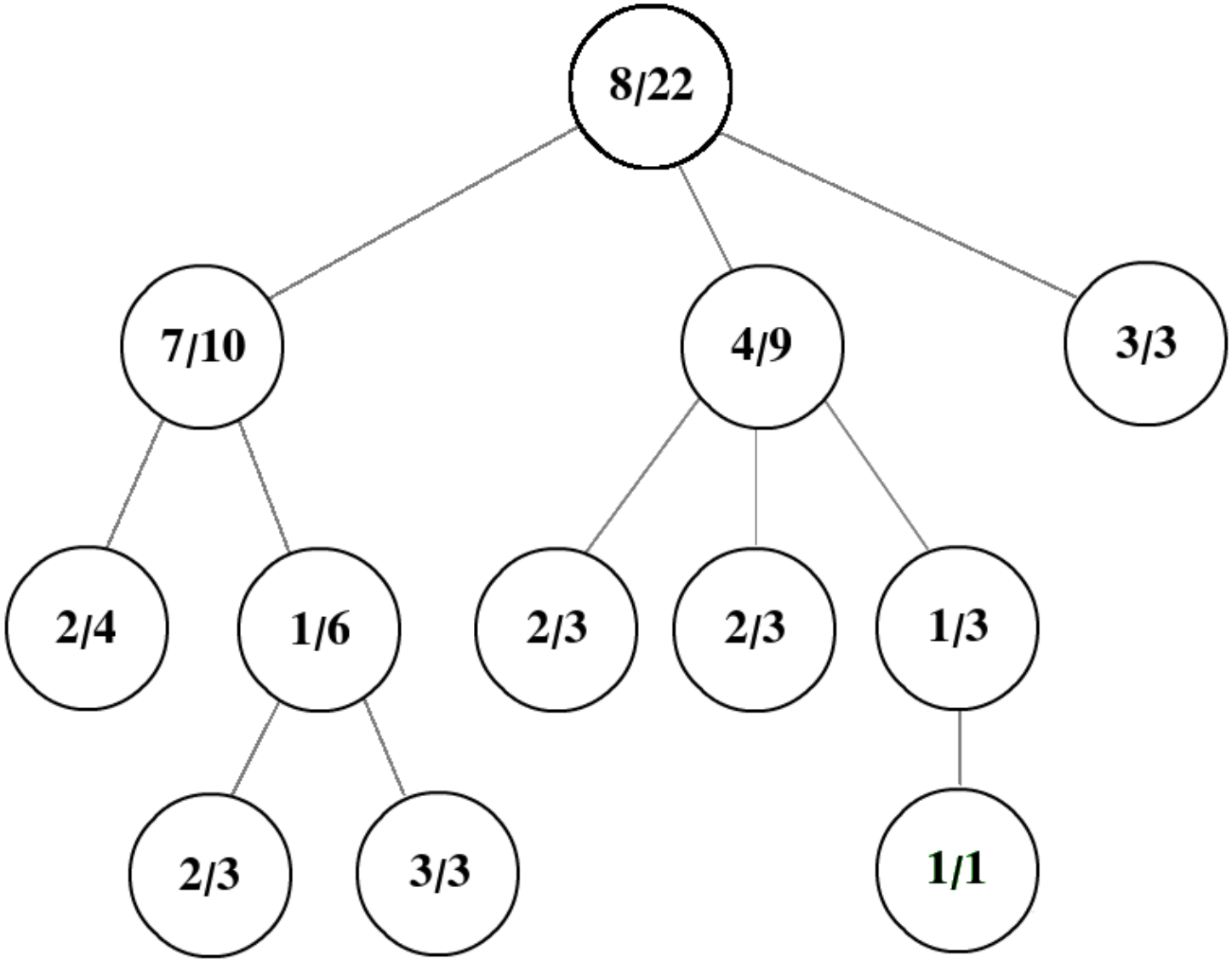
MCTS - Simulação



MCTS - Backpropagation



MCTS - Estado Novo



MCTS - Vantagens

MCTS - Vantagens

- Não precisa de uma heurística decente

MCTS - Vantagens

- Não precisa de uma heurística decente
- Crescimento assimétrico

MCTS - Vantagens

- Não precisa de uma heurística decente
- Crescimento assimétrico
- Resposta "on demand"

Implementação

Confronto Final

https://afonsonf.github.io/ia-enemath-web/

AI aplicada a jogos adversariais

Afonso Fernandes, Gonçalo Paredes, Ricardo Pereira

16 de Abril de 2019