Algoritmo MINIMAX

Até aqui...

- Problemas sem interação com outro agente.
- O agente possui total controle sobre suas ações e sobre o efeito de suas ações.
- Muitas vezes encontrar a solução ótima é factível.

Jogos vs. busca

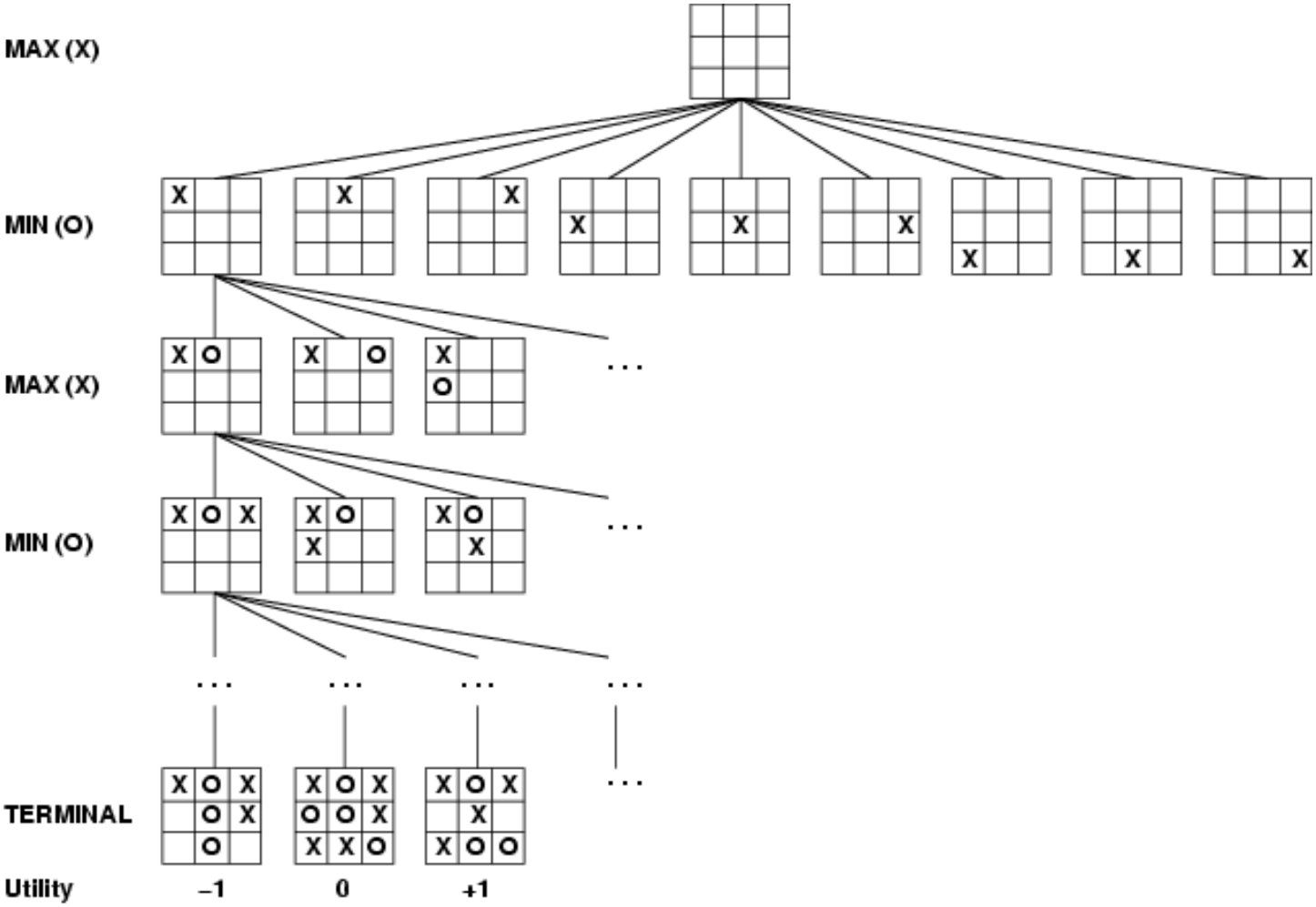
- O oponente é “imprevisível”
  - O agente tem que levar em consideração todos os movimentos possíveis de oponente.
- Limite de tempo
  - O agente tem que tomar uma decisão, mesmo que não seja ótima.

Decisões ótimas em jogos

- Consideraremos jogos com dois jogadores:
  - MAX e MIN
  - MAX faz o primeiro movimento e depois eles se revezam até o jogo terminar.

- Um jogo pode ser definido como um problema de busca com:
  - estado inicial
  - função sucessor (-> movimento, estado)
  - teste de término
  - função utilidade: dá um valor numérico para os estados terminais

Exemplo: Árvore de jogo (2 jogadores)



Do ponto de vista de MAX, valores altos de utilidade são bons.

Estratégias ótimas

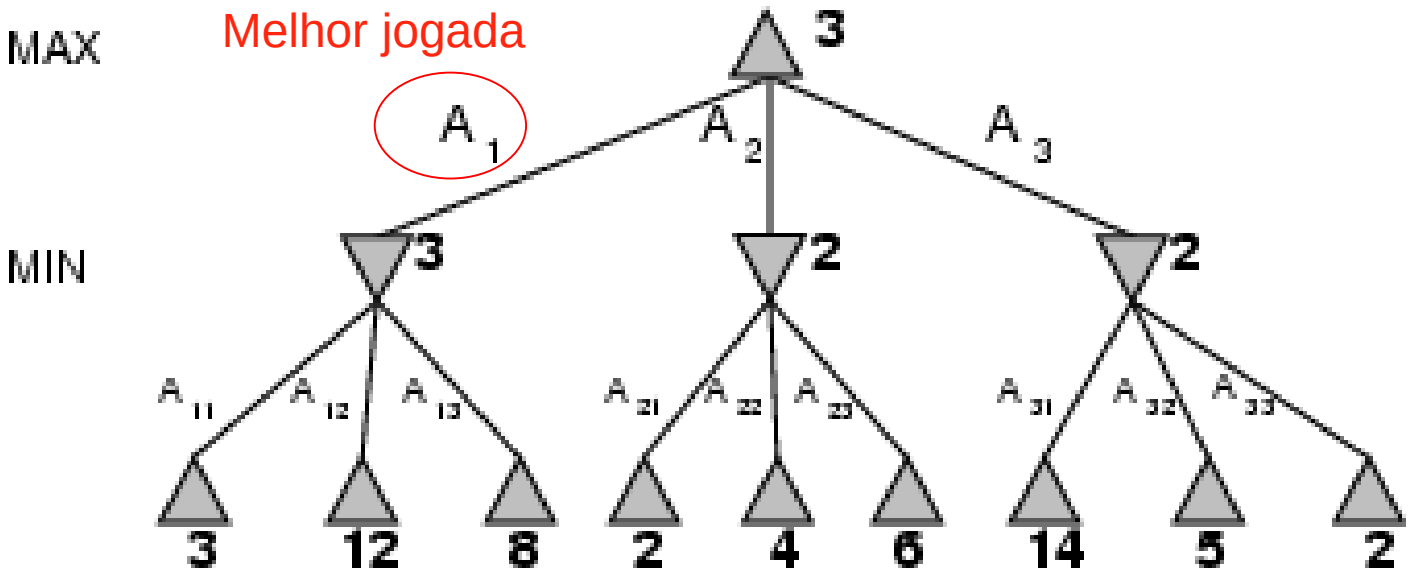
- A solução ótima para MAX depende dos movimentos de MIN, logo:
  - MAX deve encontrar uma *estratégia de contingência* que especifique o movimento de MAX no estado inicial, e depois o movimento de MAX nos estados resultantes de cada movimento de MIN e assim por diante.

Estratégias ótimas

- Dada uma árvore de jogo, a estratégia ótima pode ser determinada a partir do valor ***minimax*** de cada nó.
- O valor minimax (para MAX) é a utilidade de MAX para cada estado, ***assumindo que MIN escolhe os estados mais vantajosos*** para ele mesmo (i.e. os estado com menor valor utilidade para MAX).

Minimax

- Melhor estratégia para jogos determinísticos
- Ideia: escolher a jogada com o melhor retorno possível supondo que o oponente também vai fazer a melhor jogada possível
- Ex: Jogo simples, cada jogador faz um movimento



Minimax

Minimax (ou minmax) é um método usado na Teoria da Decisão, Teoria dos Jogos, Estatística e Filosofia.

Em teoria da decisão, o minimax (ou minmax) é um método para minimizar a possível perda máxima. Pode ser considerado como a maximização do ganho mínimo (maxmin).

Teorema Minimax
Von Neumann foi um brilhante matemático nascido em Budapeste em 1903.
Devido à demonstração do teorema minimax, Von Neumann foi considerado o pai da teoria dos jogos em 1926.

Minimax

Passos:

• Gera a árvore inteira até os estados terminais (ganha, perde ou empata).

• Aplica a função de utilidade nas folhas.

• Propaga os valores dessa função subindo a árvore através do minimax.

• Determinar qual a ação que será escolhida por MAX.

Algoritmo minimax

VALOR-MINIMAX(n)=

$$\begin{cases} \text{UTILIDADE(n) se n é terminal} \\ \max_{x \in Succ(n)} \text{Valor Minimax(s) se } n \text{ é um nó de MAX} \\ \min_{x \in Succ(n)} \text{Valor Minimax(s) se } n \text{ é um nó de MIN} \end{cases}$$

Algoritmo minimax

Algoritmo (Pseudo-código)
Determinar
SE {
profundidade limite atingida
OU Nível é Minimizador
OU Nível é Maximizador }
ENTÃO
SE profundidade limite
Calcular valor do estado corrente
Retornar resultado
SE Nível Minimizador
Aplicar minimax aos sucessores
Retornar Mínimo
SE Nível Maximizador
Aplicar minimax aos sucessores
Retornar Máximo

Propriedades do algoritmo minimax

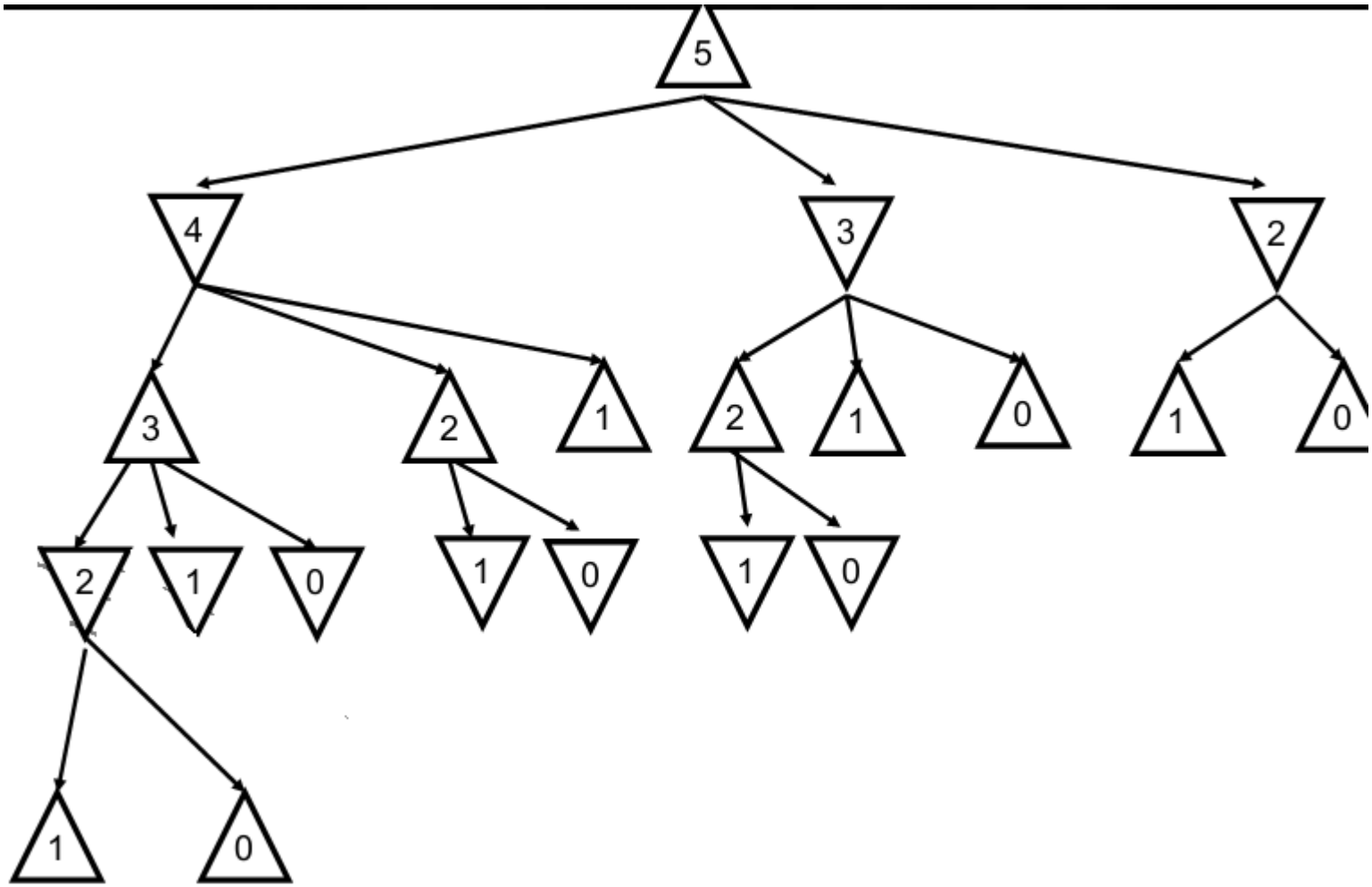
- Equivale a uma busca completa em profundidade na árvore do jogo.
  - m: profundidade máxima da árvore
  - b: movimentos válidos em cada estado

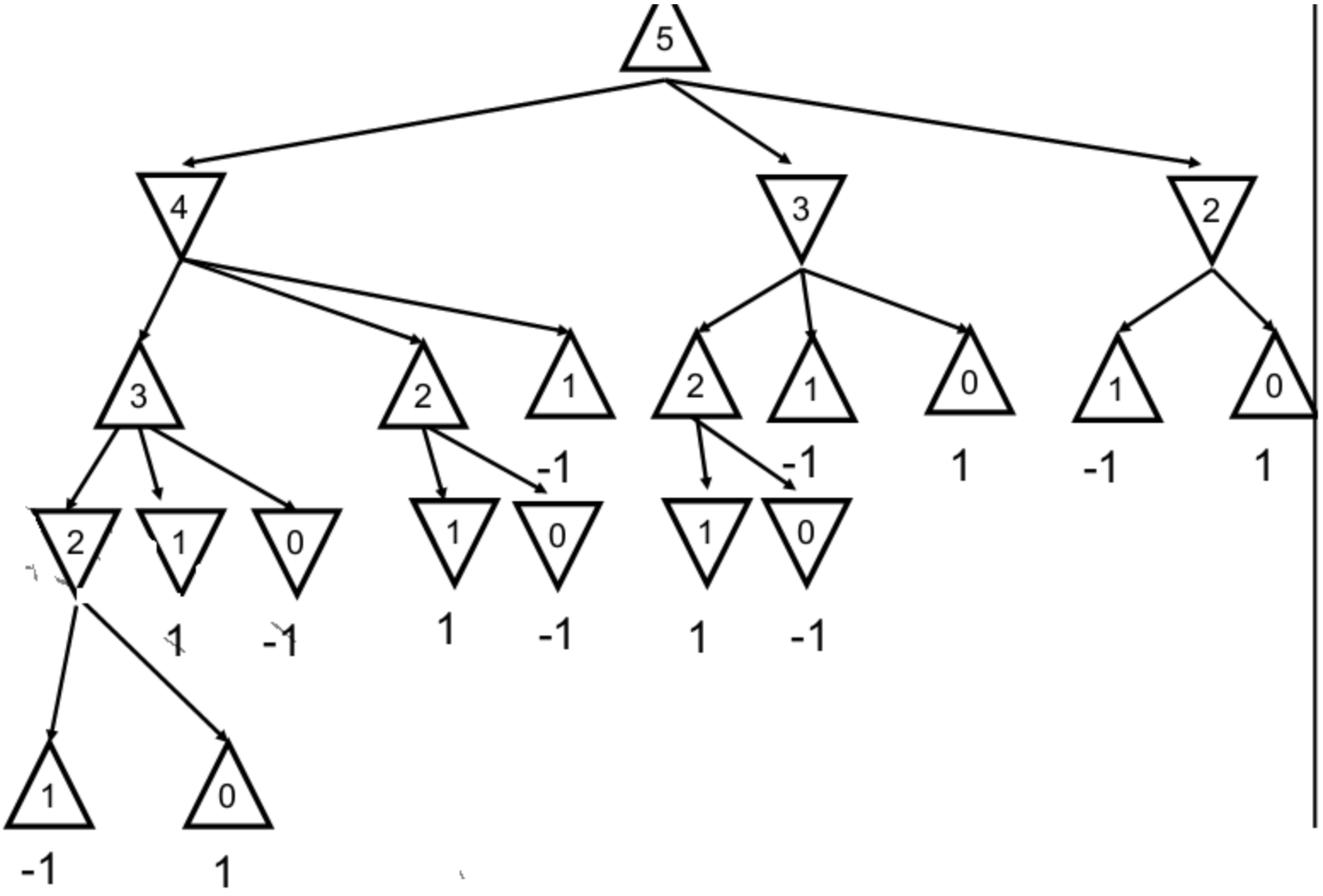
- Completo? Sim (Se a árvore é finita)
- Ótimo? Sim (contra um oponente ótimo)
- Complexidade de tempo? $O(b^m)$
- Complexidade de espaço? O(bm)

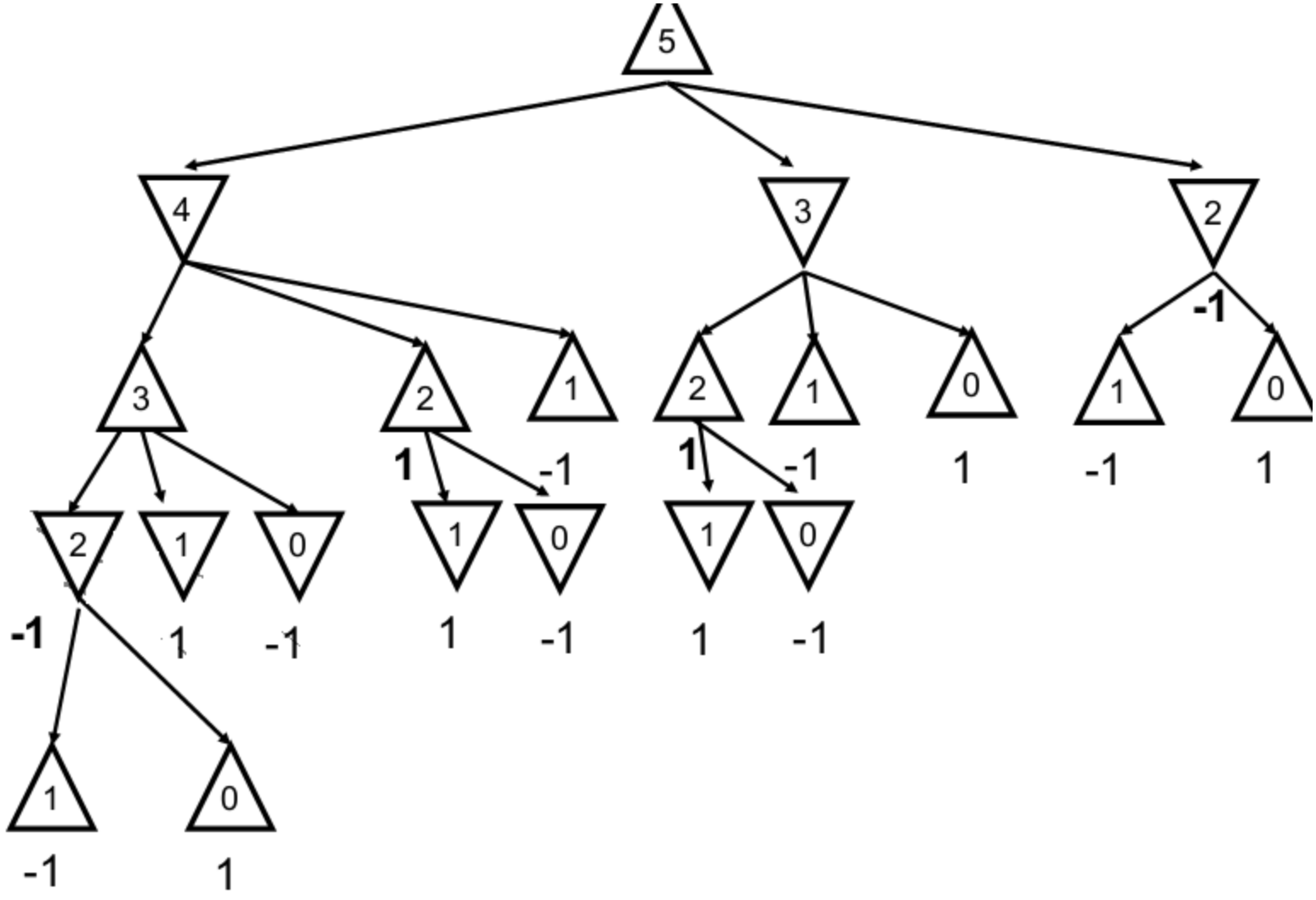
- Para xadrez, b ≈ 35, m ≈100 para jogos “razoáveis”
  → solução exata não é possível

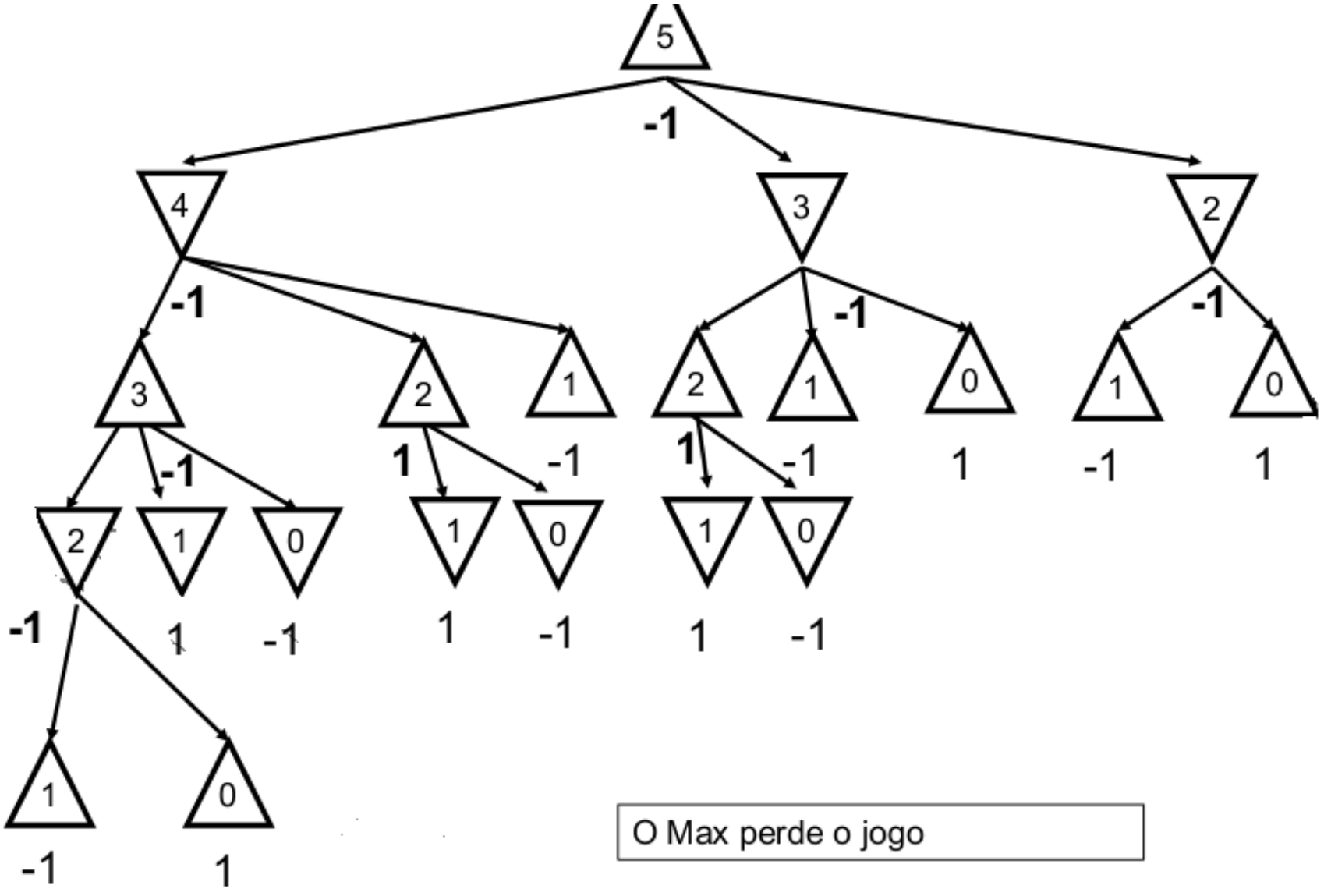
Exercício

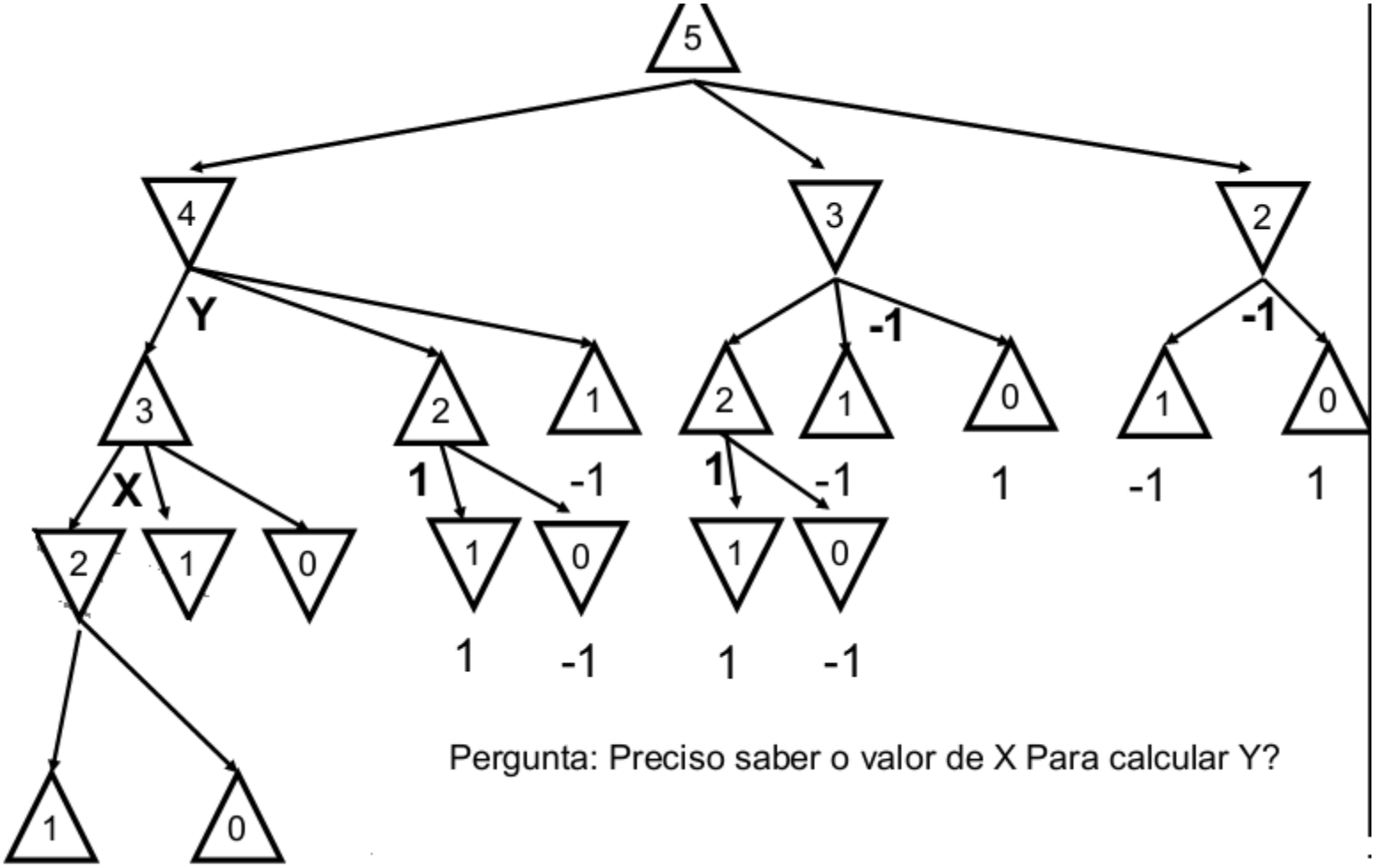
- Dados 5 palitos cada jogador pode retirar 1, 2 ou 3 por turno.
- Perde o jogador que retira o último palito.
- Será que max pode ganhar o jogo?


5
4
3
2
3
2
1
2
1
0
1
0
2
1
0
1
0
1
0
1
0


5
4
3
2
3
2
1
2
1
0
1
0
2
1
0
1
0
1
0
1
1
0
-1
1
-1
1
-1
1
-1
1
-1
1
-1
1


5
4
3
2
3
2
1
2
1
0
1
0
2
1
0
2
1
0
1
0
1
0
1
0
-1
1
-1
1
-1
1
-1
1
-1
1
-1
1
-1
1
-1
1


5
-1
4
3
2
-1
-1
-1
3
2
1
2
1
0
1
0
-1
1
-1
1
-1
1
-1
2
1
0
1
0
1
0
-1
1
-1
1
-1
1
-1
1
0
-1
1
O Max perde o jogo



Pergunta: Preciso saber o valor de X Para calcular Y?

Críticas

Problemas

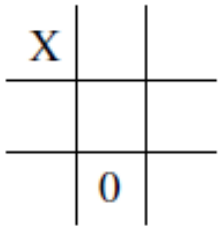
- Tempo gasto para determinar a decisão ótima é totalmente impraticável (ir até as folhas), porém o algoritmo serve como base para outros métodos mais realísticos.
- Complexidade: $O(b^m)$ – idem Busca em Profundidade.

Para melhorar

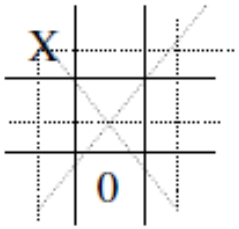
1) Limitar a profundidade da busca e substituir função de utilidade por função de avaliação (heurística);

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

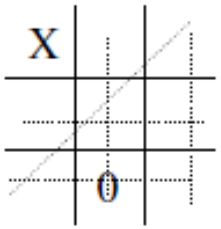
Função heurística para o jogo da velha



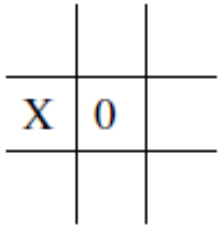
H = 6 - 5 = 1



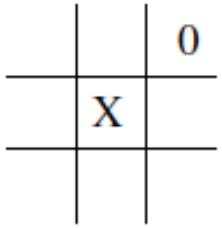
X tem 6 possibilidades



0 tem 5 possibilidades



H = 4 - 6 = = -2



H = 5 - 4 = 1

Poda Alpha-Beta

Objetivo: não expandir desnecessariamente nós durante o minimax.

Ideia: 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)

Poda Alpha-Beta

Ou seja, quais seriam os passos deste algoritmo?

- PASSOS (MINIMAX + passos abaixo)

Na descida, devemos verificar se o nó que estamos olhando tem valor. Se tiver, verificar que tipo de nó é:

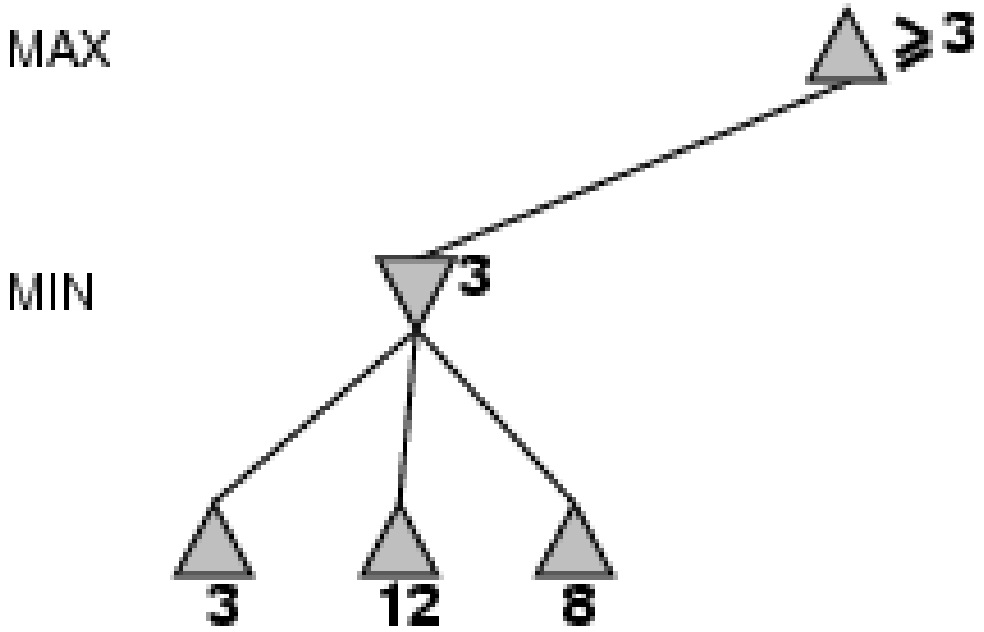
- Se for um nó MIN(beta) e há ancestrais alfa(MAX), então pode-se fazer a seguinte pergunta: qualquer ancestral alfa é maior ou igual a beta desse nó MIN? Se sim, poda os demais ramos do nó MIN (**poda alfa**).

- Se for um nó MAX e há ancestrais beta, faz-se a mesma pergunta, porém de forma inversa. Qualquer ancestral beta é menor ou igual a alfa desse nó MAX? Se sim, poda os demais ramos do nó MIN (**poda beta**).

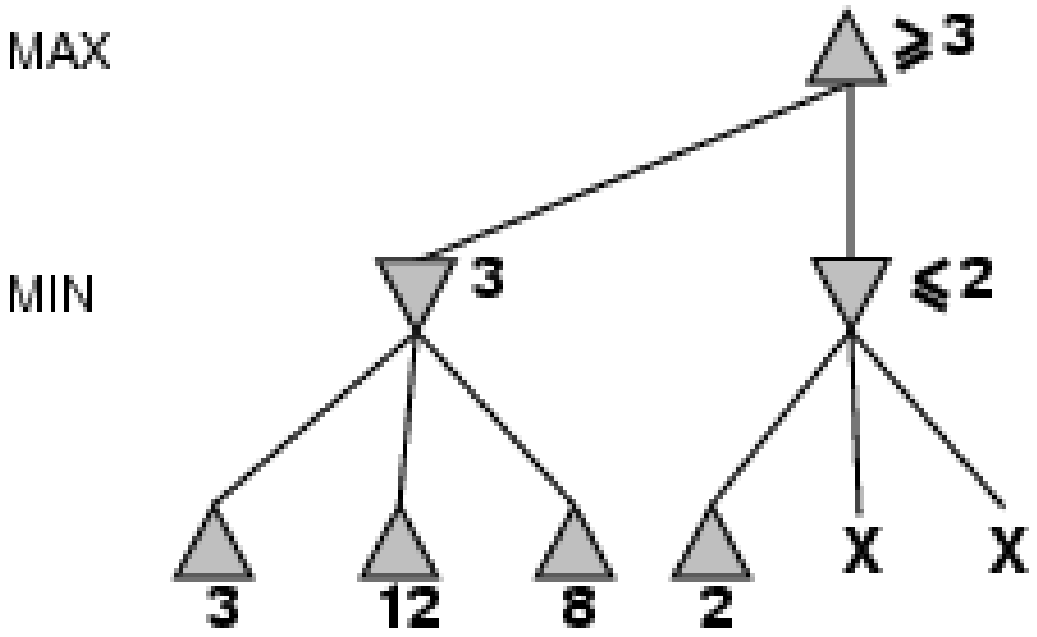
Poda α-β

- Algoritmo minimax: nº de estados do jogo é exponencial em relação ao nº de movimentos
- Poda α-β:
  - calcular a decisão correta sem examinar todos os nós da árvore,
  - retorna o mesmo que minimax, porém sem percorrer todos os estados.

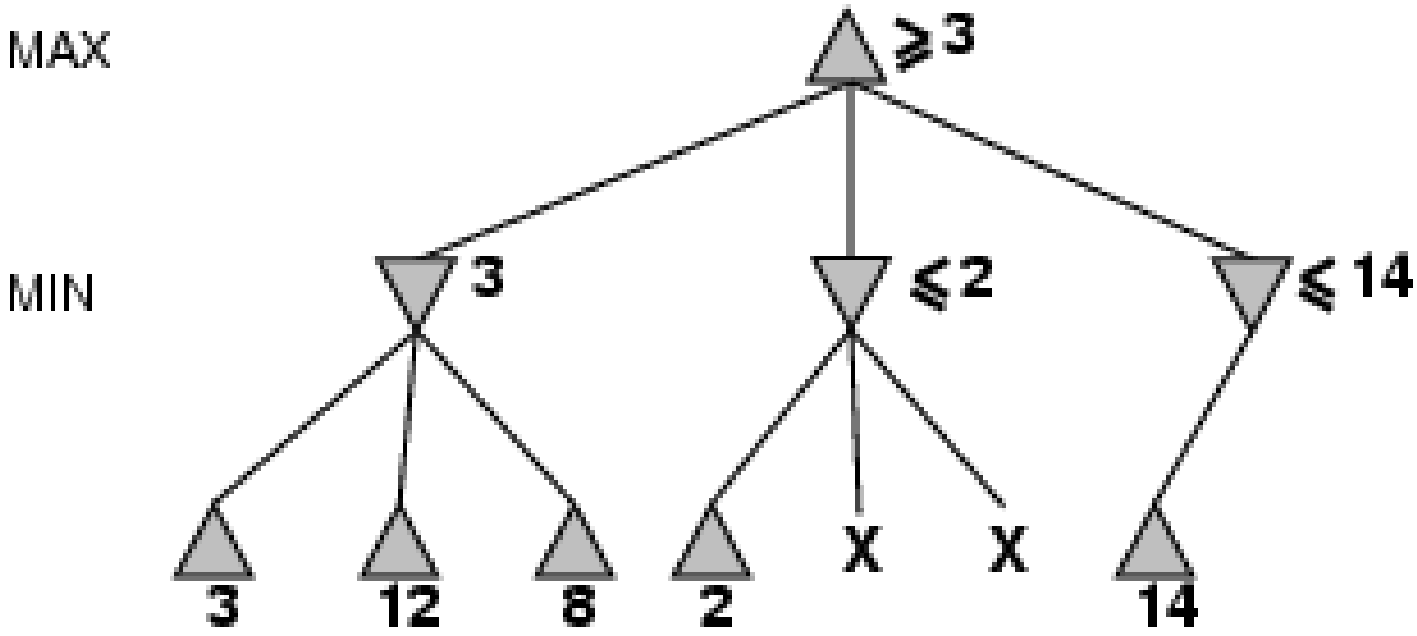
Poda α-β



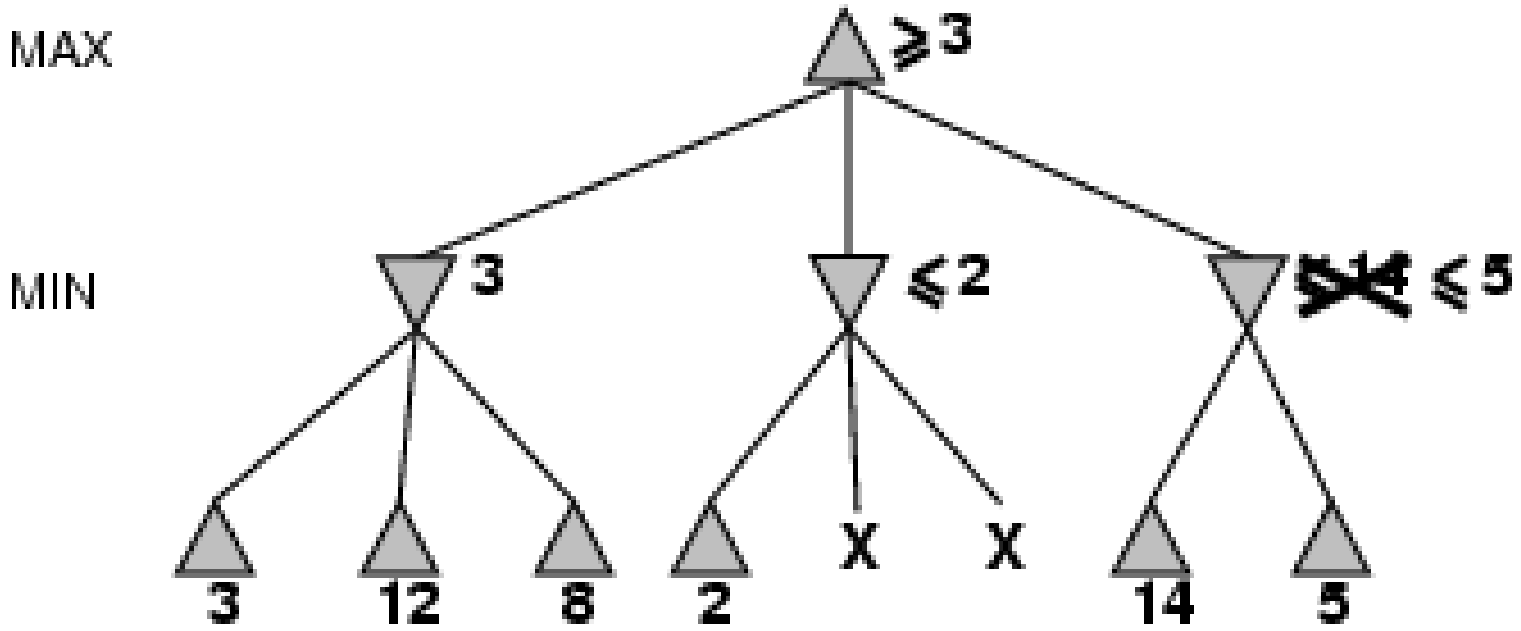
Poda α-β



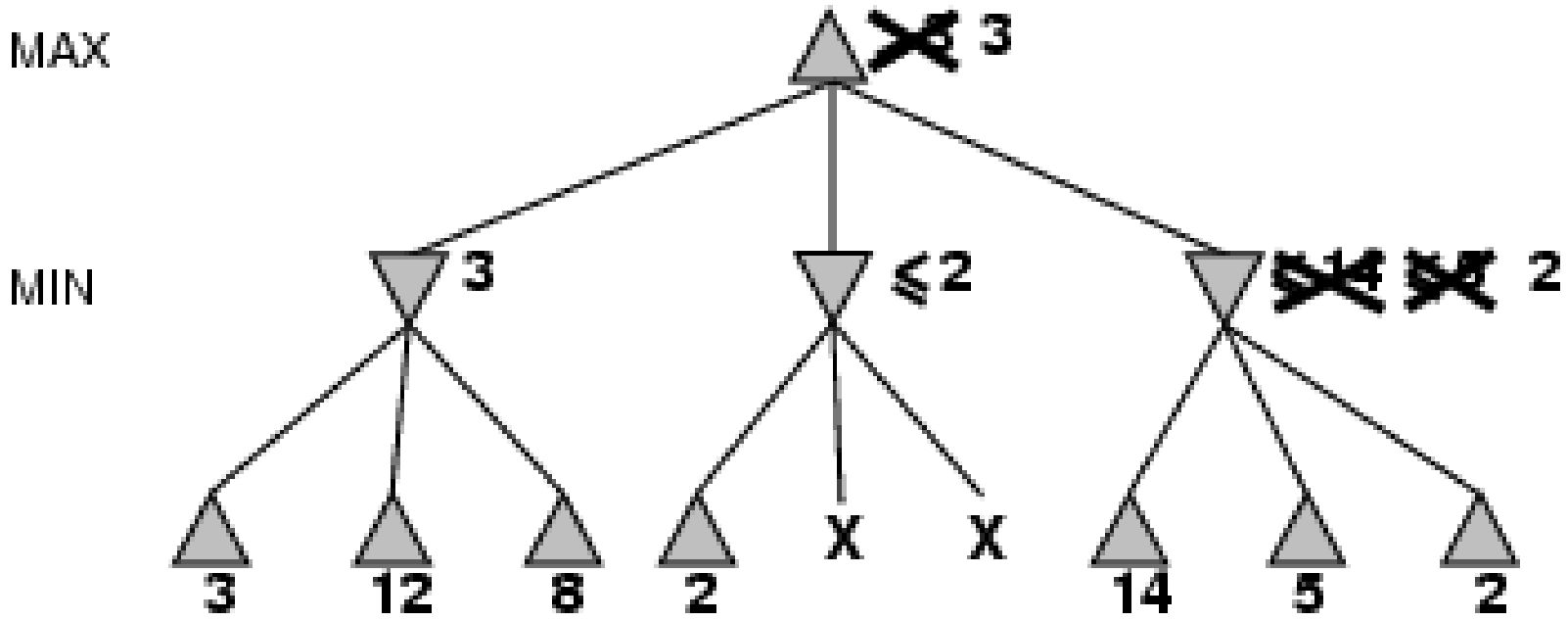
Poda α-β



Poda α-β



Poda α-β



Poda α-β

- A efetividade da poda α-β depende da ordem em que os sucessores são examinados.
- Com a melhor ordem possível a complexidade de tempo = $O(b^{m/2})$

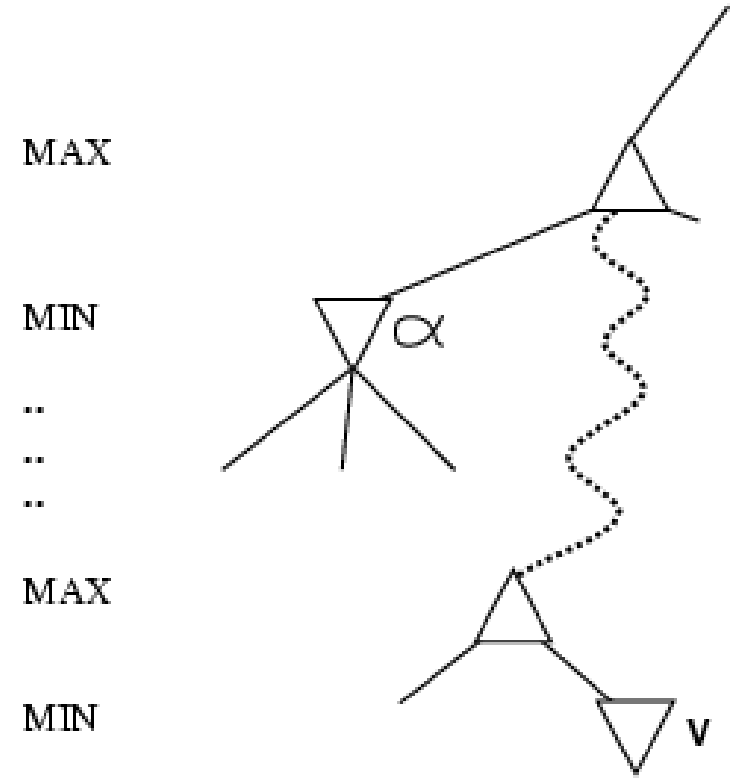
  → dobra a profundidade da busca que conseguimos fazer

NEGAMAX

- Variante do **miniMAX** que se baseia na observação de que max(a, b) = -min(-a, -b), ou seja as heurísticas podem ser as mesmas para ambos os jogadores

Por que "α-β" ?

- α é o valor da melhor escolha (valor mais alto) encontrado até então para qualquer ponto de escolha de MAX;
- Se *v* é pior do que α, MAX não percorrerá este caminho (irá podar este ramo de busca)
- β é definido de maneira análoga.



Referências

http://www.youtube.com/watch?v=wjkNvzsgtbY&feature=related

http://www.youtube.com/watch?v=r0kx3uuvIKk&feature=related

http://www.youtube.com/watch?v=Ep4PUVYTNiU&feature=related

Applet Java

http://ksquared.de/gamevisual/launch.php

http://homepage.ufp.pt/jtorres/ensino/ia/alfabeta.html

Comparando valores de execução:

http://elemarjr.net/2011/12/11/tictactoe-minimax-negamax-alpha-beta-prunning-whatever/

Referências

http://pt.wikipedia.org/wiki/Teoria_dos_jogos#Soma_zero_e_soma_diferente_zero

MININAX

http://www.youtube.com/watch?v=bYMA0y2j9ZM – parte 1 de 3

Corte Alfa-beta

http://www.youtube.com/watch?v=gs6bS25hBsA – parte 2 de 3
http://www.youtube.com/watch?v=Xu0RGLV_r7w – parte 3 de 3

http://ksquared.de/gamevisual/launch.php

NegaMax
http://www.frayn.net/beowulf/theory.html#negamax