PROCURA COM ADVERSÁRIOS (JOGOS) CAP 5

Parcialmente adaptado de

http://aima.eecs.berkeley.edu

Resumo

- Estratégias Óptimas
- Corte α-β
- Recursos Limitados
- Jogos com factor sorte
- Jogos com informação imperfeita

Jogos

- Os Jogos são uma forma de ambiente multi-agente
  - O que fazem os outros agentes e como é que afectam o nosso sucesso
  - Ambientes multi-agente cooperativos vs. competitivos
  - Ambientes multi-agente competitivos levam ao surgimento de problemas adversariais a.k.a. jogos
- Porque estudar Jogos?
  - Entretenimento
  - Porque são difíceis
  - Fáceis de representar e agentes estão limitados a um número reduzido de ações.

Jogos vs. Procura

- Procura – não há adversários
  - Solução é um método (heurístico) para encontrar o objectivo
  - Métodos existentes (frequentemente) permitem encontrar solução óptima
  - Função de avaliação: estima o custo de ir do início ao objectivo através de um dado nó
  - Exemplos: planeamento de percursos; escalonamento de actividades, …
- Jogos – há adversários
  - Solução é uma estratégia (especificar uma jogada para cada resposta possível do adversário) para ganhar o jogo
  - Limites temporais forçam uma solução aproximada
  - Função de avaliação: estima a qualidade de uma posição do jogo
  - Exemplos: Xadrez, Damas, Othello, Gamão,…

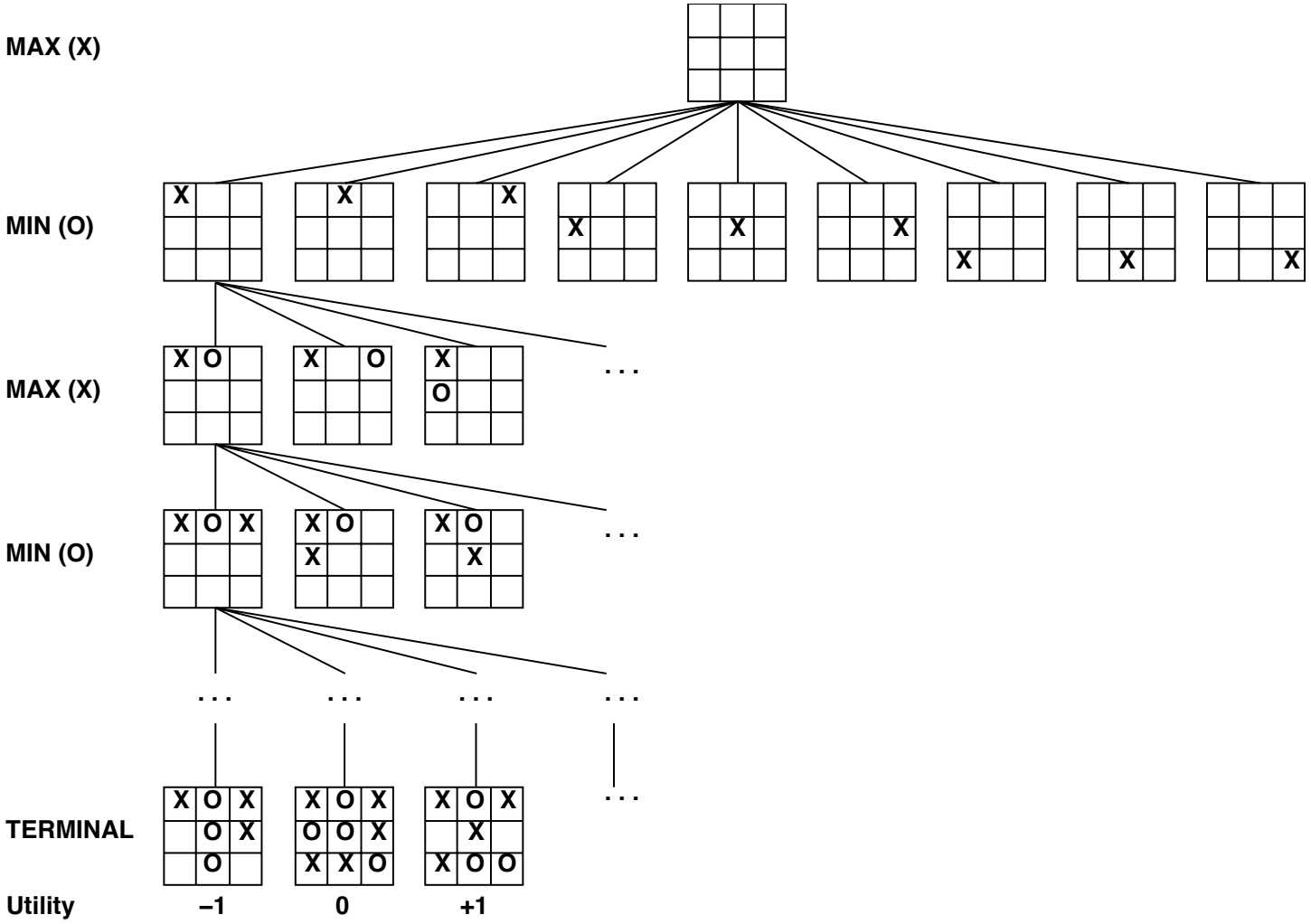
Tipos de Jogos

	Deterministicos	Sorte
Informação Perfeita	Xadrez, Damas, Go, Otello, Jogo do Galo	Gamão, Monopólio
Informação Imperfeita	Batalha Naval, Jogo do Galo Cego	Poker, Bridge, Scrabble

Definição de Jogo

- 2 Jogadores: MAX e MIN
- MAX joga primeiro. Jogadores alternam até ao fim do jogo. Jogador que ganha recebe um prémio e o que perde uma penalização.
- Jogos como um tipo de problema de pesquisa com:
  - Estado inicial (S_0): e.g. configuração de um tabuleiro de xadrez
  - Player(s): define quem deve jogar num dado estado
  - Actions(s): define as jogadas possíveis (legais) num dado estado
  - Result(s,a): Modelo de transição que define o resultado de uma jogada
  - Terminal-test(s): Define quais os estados terminais do jogo i.e. quando é que o jogo acaba
  - Utility(s,p): função de utilidade (aka. Função objectivo, função de pagamento) que define o valor de um dado estado terminal s para o jogador p.
- Define uma árvore de pesquisa (árvore de jogo)
  - Assume-se que MAX joga primeiro.
  - Números nas folhas indicam utilidade do ponto de vista de MAX, assumindo que valores elevados são bons para MAX e maus para MIN.

Árvore de pesquisa parcial para o Jogo do Galo

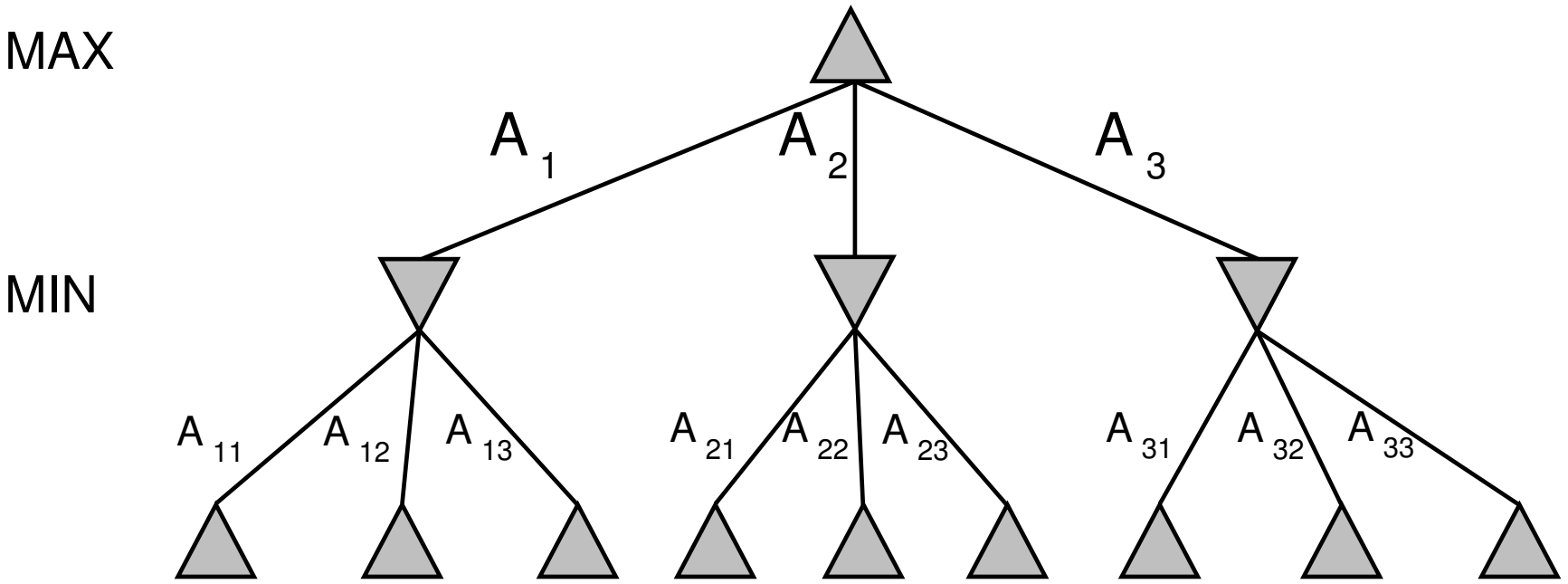


Estratégia Óptima

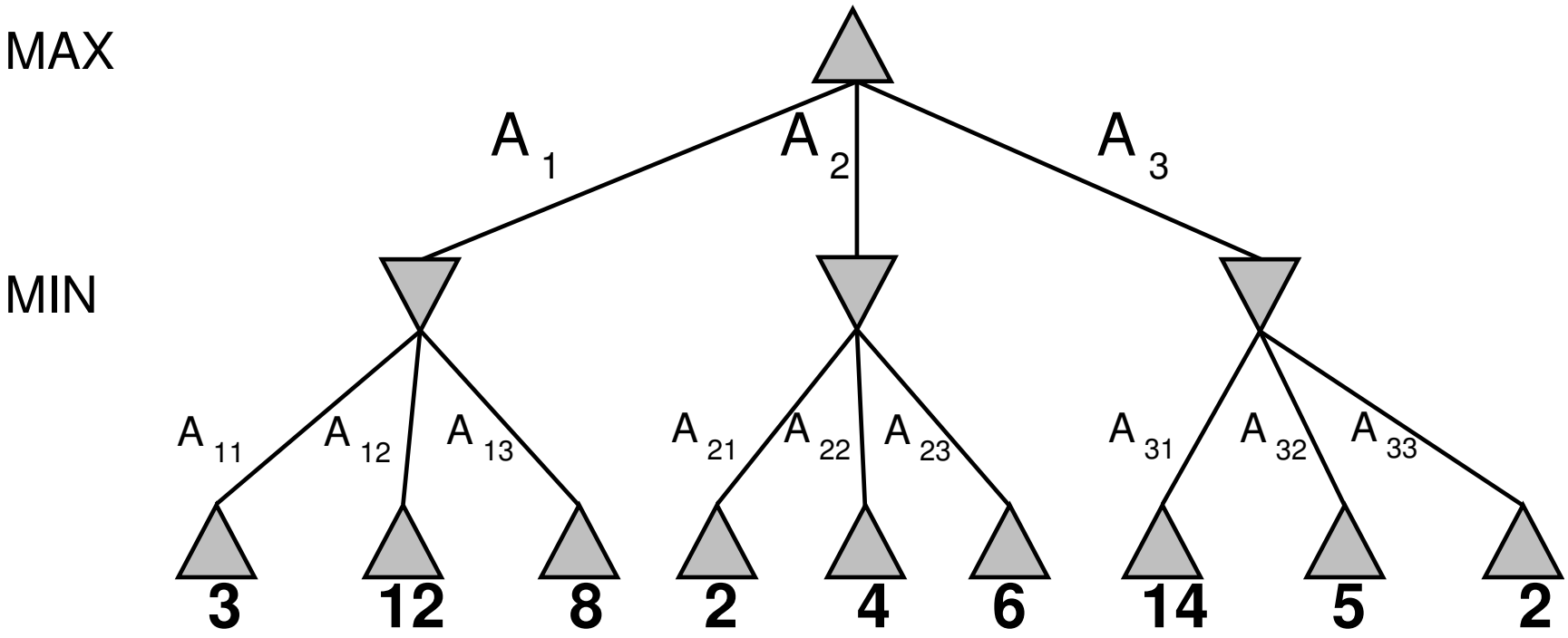
- Encontrar uma estratégia de contingência para o jogador MAX, assumindo um jogador MIN infalível.
- Dada uma árvore de jogo, a estratégia óptima pode ser determinada usando o valor minimax de cada nó.
- O valor minimax de cada nó corresponde à utilidade (de MAX) de estar no estado correspondente, assumindo que ambos os jogador jogadores são óptimos.

$$\text{MINIMAX-VALUE}(n) = \begin{cases} \text{UTILITY}(n) & \text{Se } n \text{ é terminal} \\ \max_{s \in successors(n)} \text{MINIMAX-VALUE}(s) & \text{Se } n \text{ é um nó de MAX} \\ \min_{s \in successors(n)} \text{MINIMAX-VALUE}(s) & \text{Se } n \text{ é um nó de MIN} \end{cases}$$

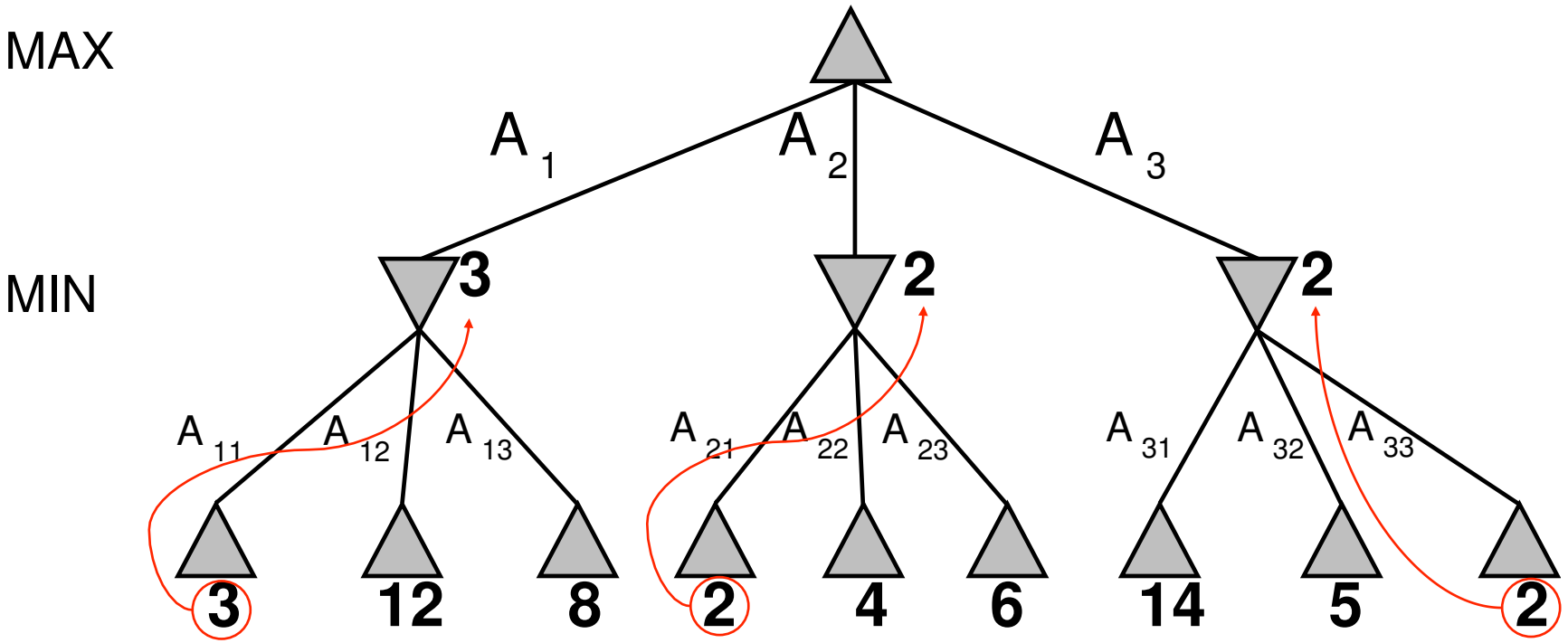
Árvore de Jogo com 2 jogadores



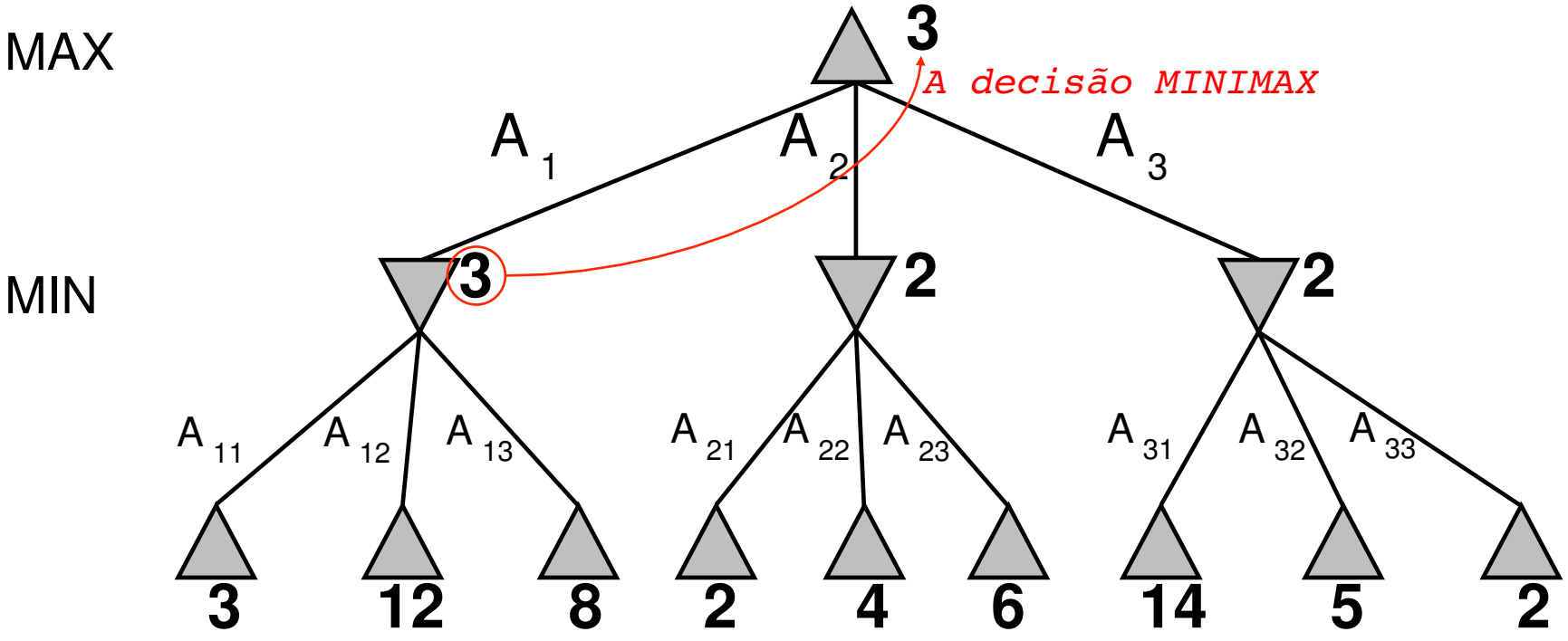
Árvore de Jogo com 2 jogadores



Árvore de Jogo com 2 jogadores



Árvore de Jogo com 2 jogadores



- MINIMAX maximiza o resultado de MAX na pior situação.

Algoritmo Minimax

```
function MINIMAX-DECISION(state) returns an action
  inputs: state, current state in game
  v←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 ← -∞
  for a,s in SUCCESSORS(state) do
    v ← MAX(v,MIN-VALUE(s))
  return v
```

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

Propriedades do Minimax

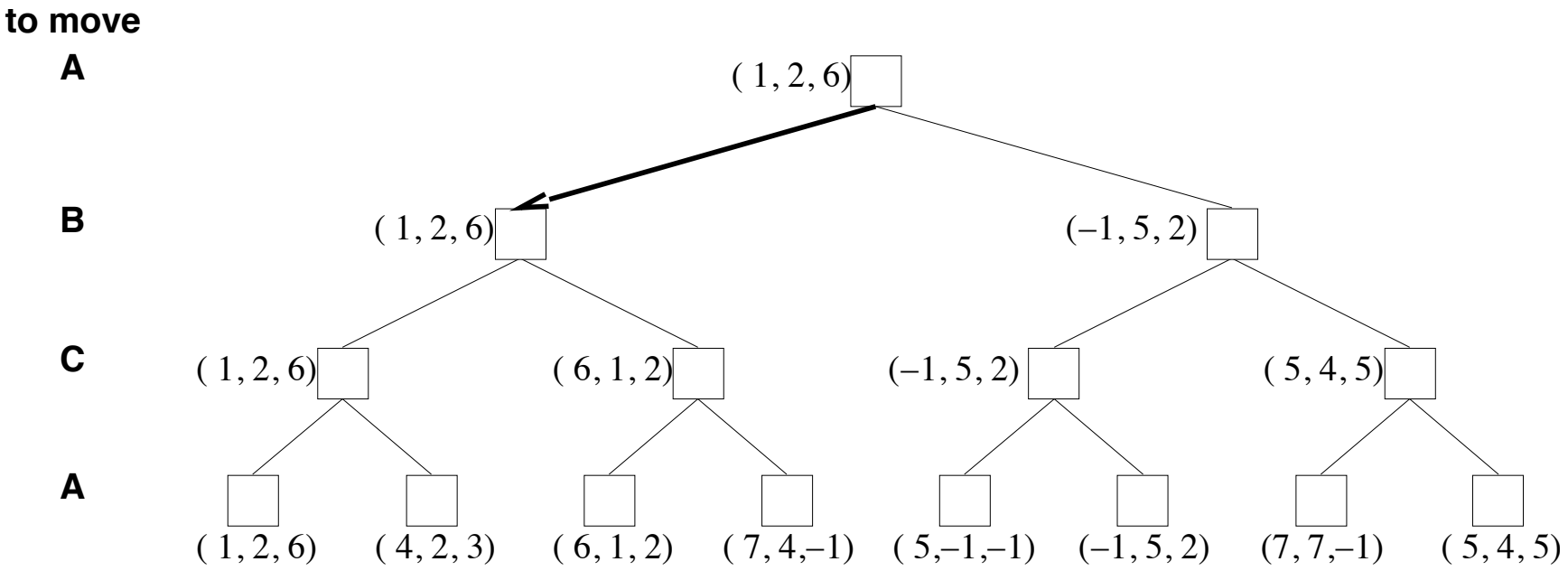
Completo?	Sim, se a árvore é finita. NB: uma estratégia finita pode existir mesmo em árvores infinitas!
Óptimal?	Sim, contra um adversário perfeito.
Complexidade Temporal	$O(b^m)$
Complexidade Espacial	O(bm) (exploração pelo melhor primeiro)

- Para o Xadrez, b≈35, m≈100 (para jogos “razoáveis”)
  - ⇒ solução exacta é completamente impossível.
- Na prática, o Minimax serve como base matemática para analisar jogos, e como base para algoritmos mais eficientes

E se MIN não jogar de forma óptima?

- A definição de jogada óptima para MAX assume que MIN joga de forma óptima: maximiza o resultado de MAX na pior situação.
- Se MIN não jogar de forma óptima, pode-se demonstrar que MAX terá um resultado ainda melhor.

Jogos com vários jogadores

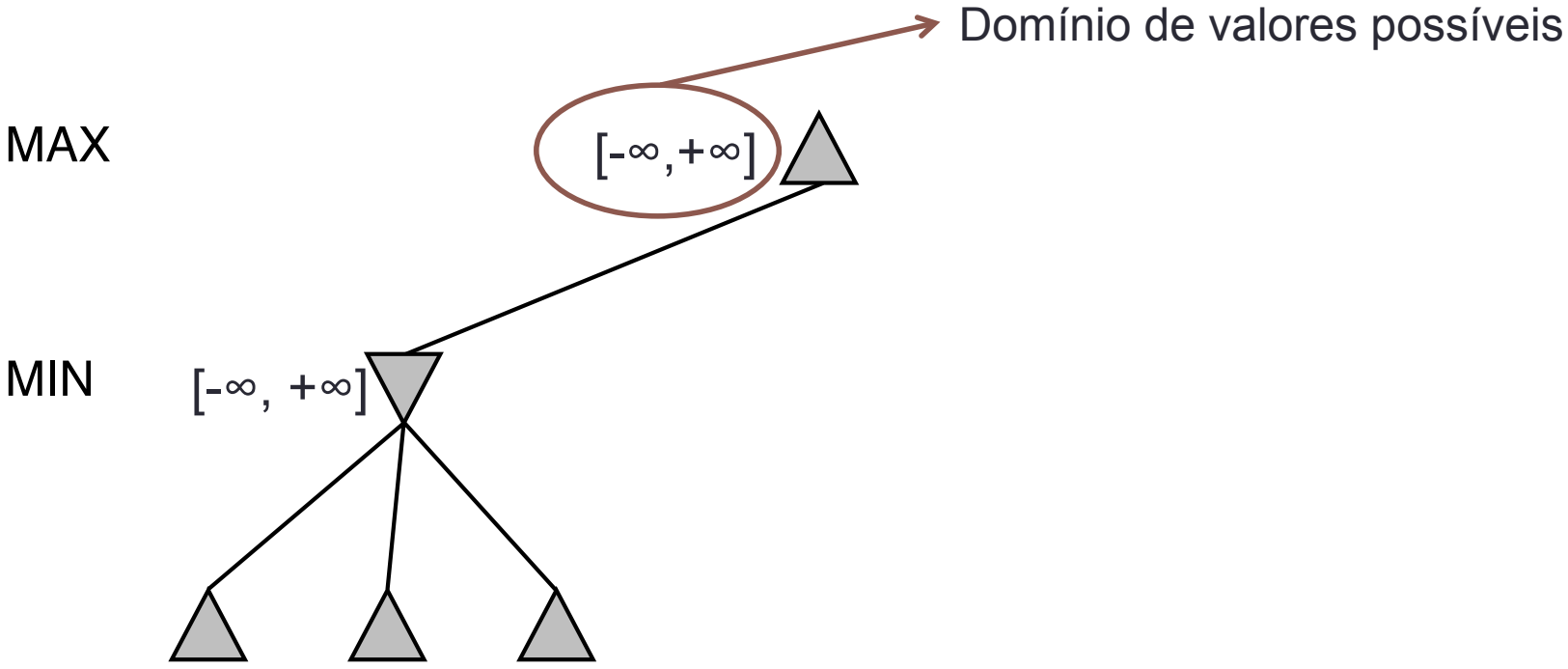


- Jogos podem permitir mais do que 2 jogadores (3 nesta figura)
- Valores minimax passam a ser vectores. Cada jogador tenta maximizar o seu valor no vector.

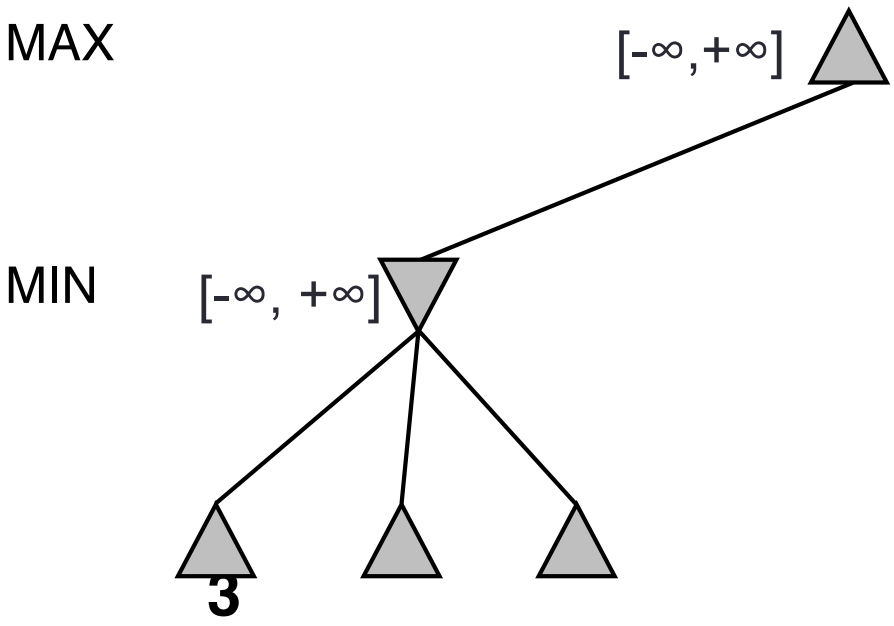
Problema com o Minimax

- O número de estados do jogo é exponencial no número de jogadas.
- Solução: tentar não examinar todos os nós.
- Corte alfa-beta (α-β)
  - α = valor da melhor escolha encontrada até ao momento num ponto de escolha ao longo de um caminho de MAX
  - β = valor da melhor escolha encontrada até ao momento num ponto de escolha ao longo de um caminho de MIN
- Voltando ao exemplo…

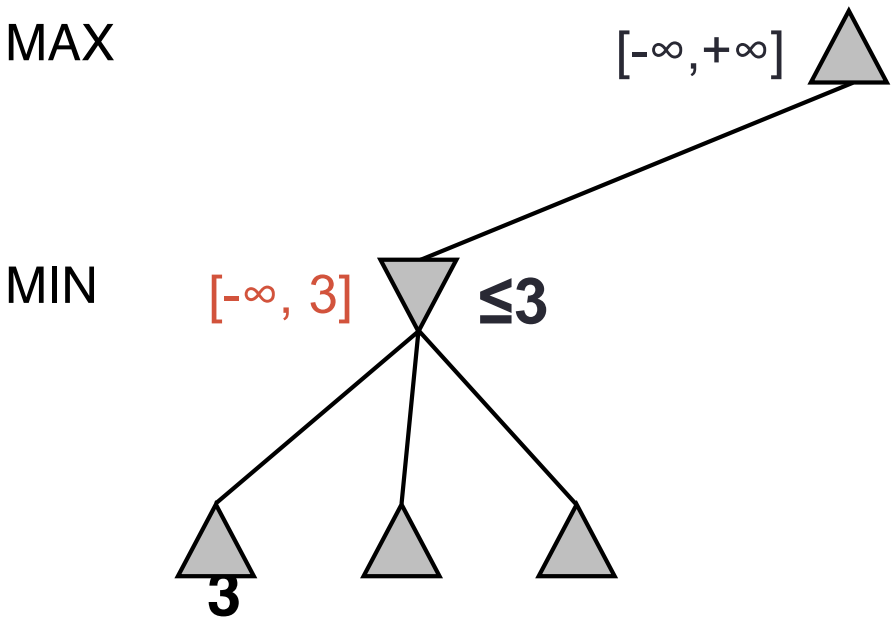
Exemplo de corte α-β



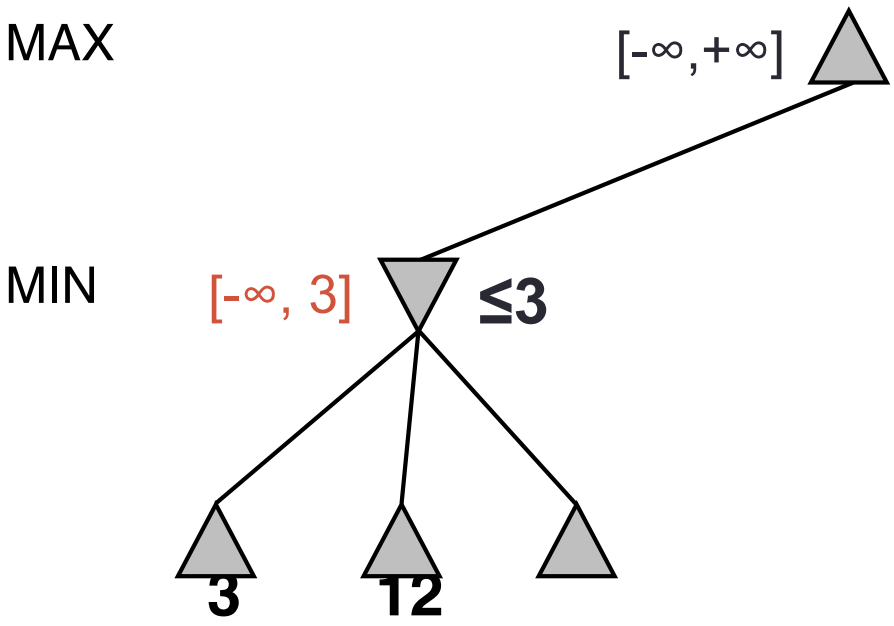
Exemplo de corte α-β



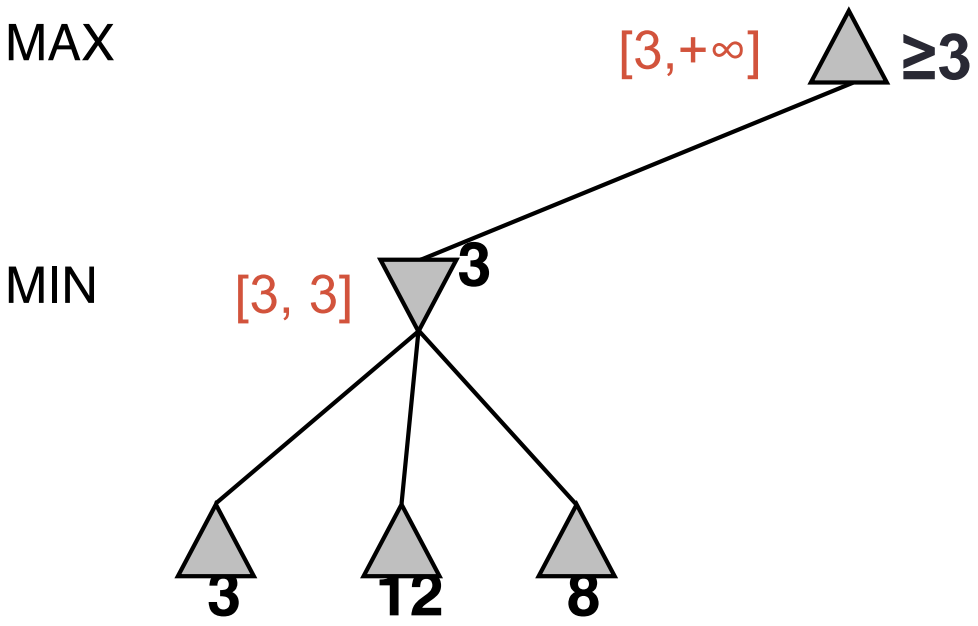
Exemplo de corte α-β



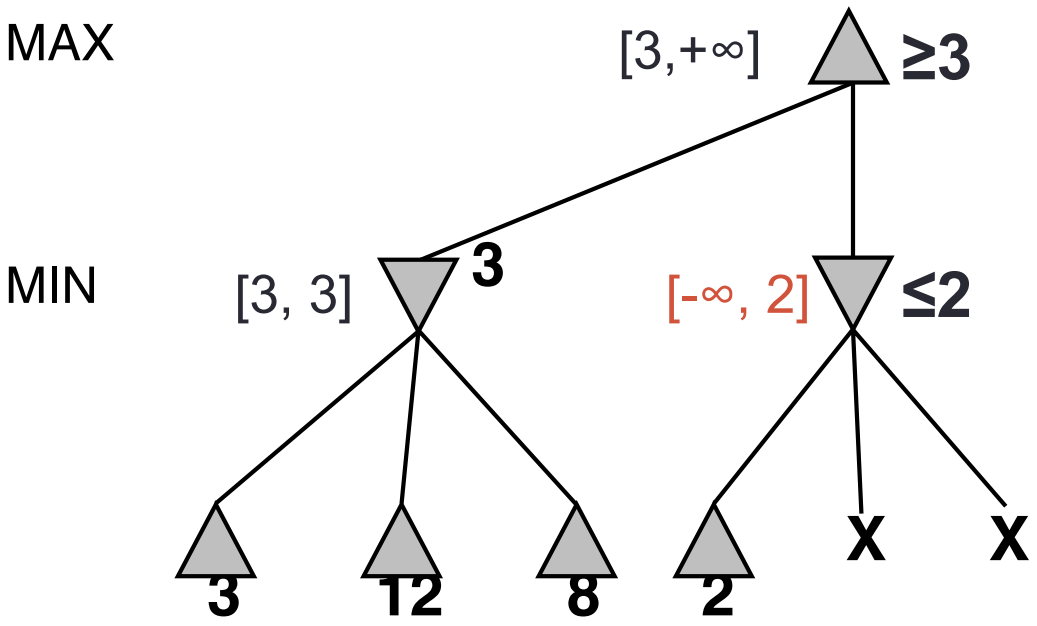
Exemplo de corte α-β



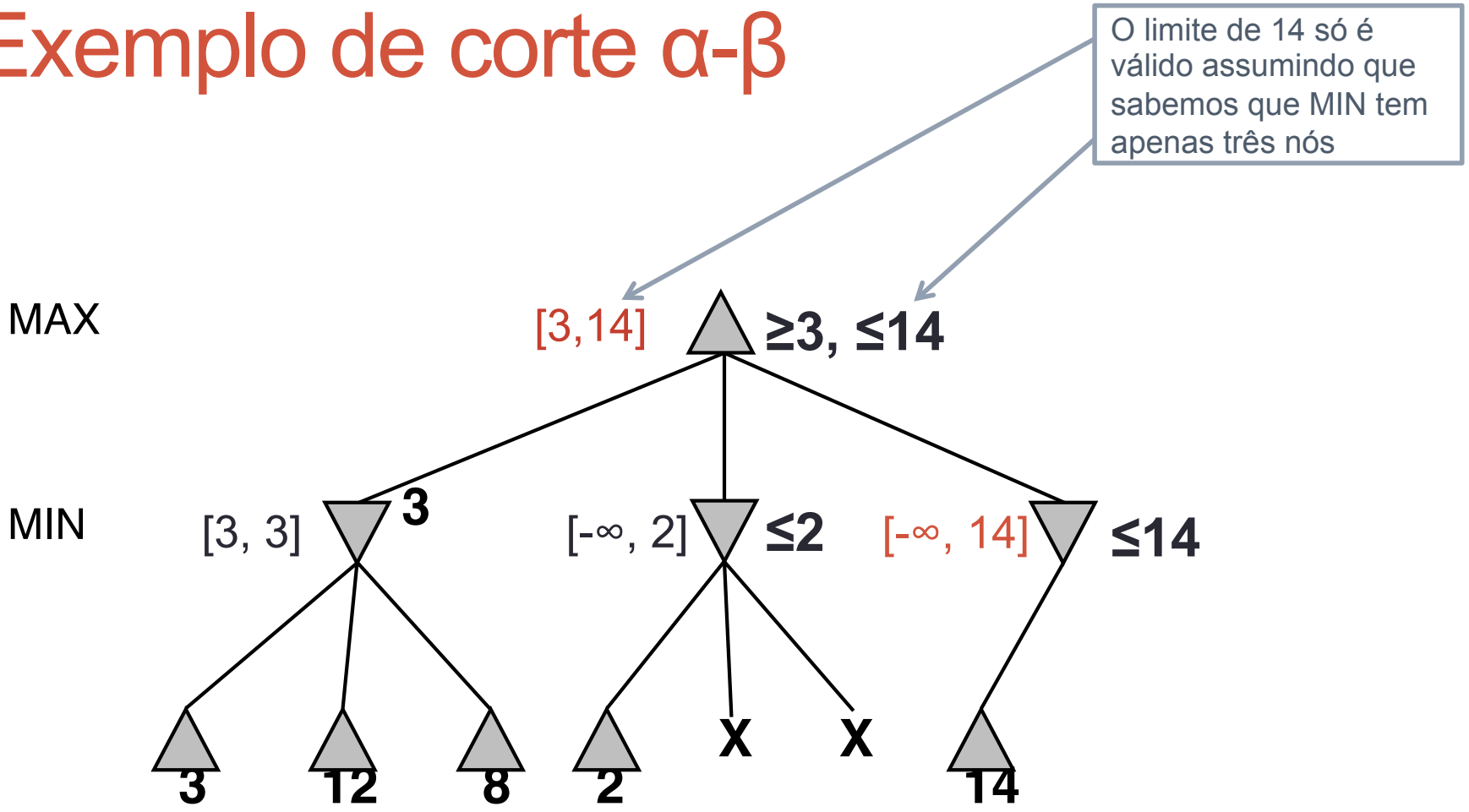
Exemplo de corte α-β



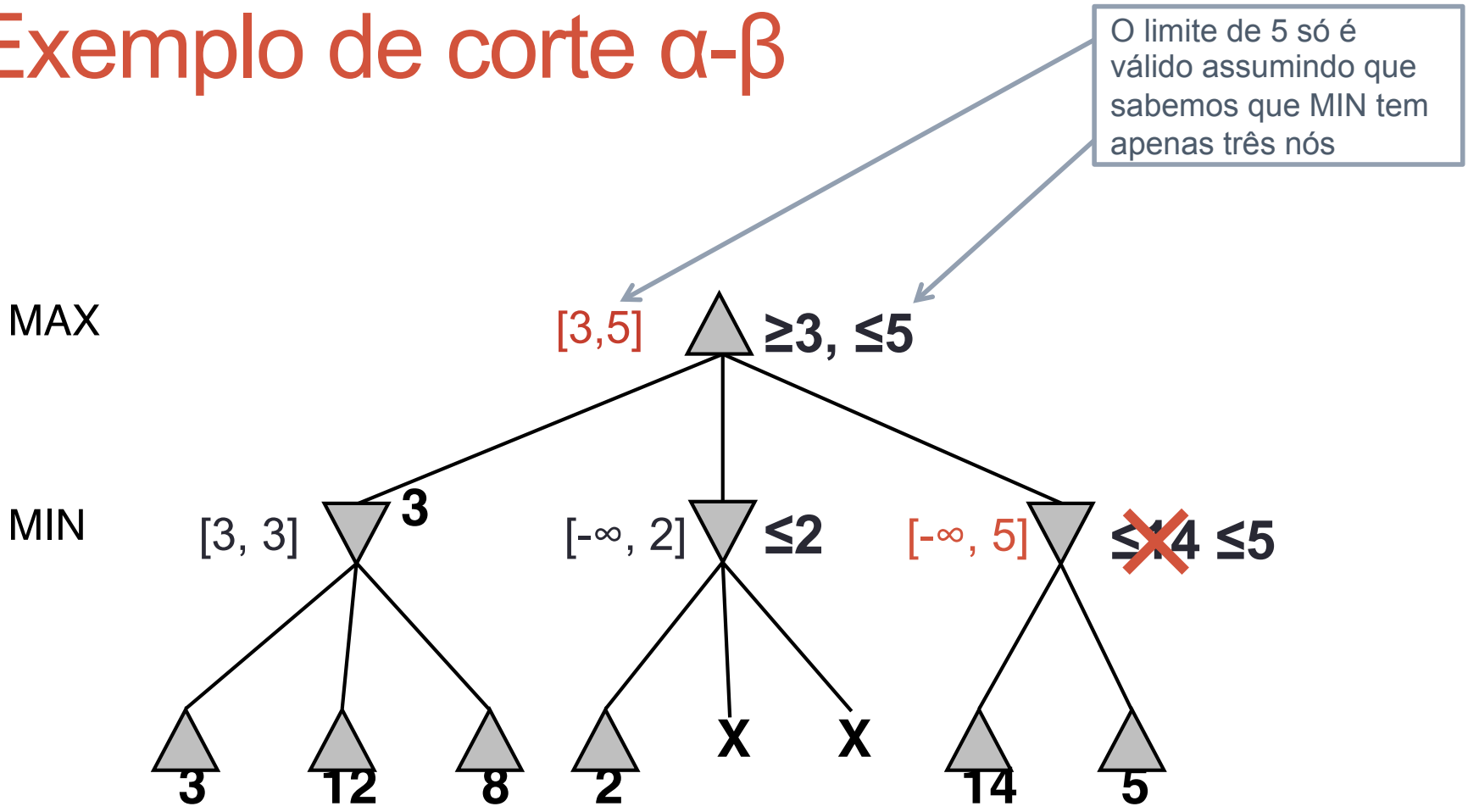
Exemplo de corte α-β



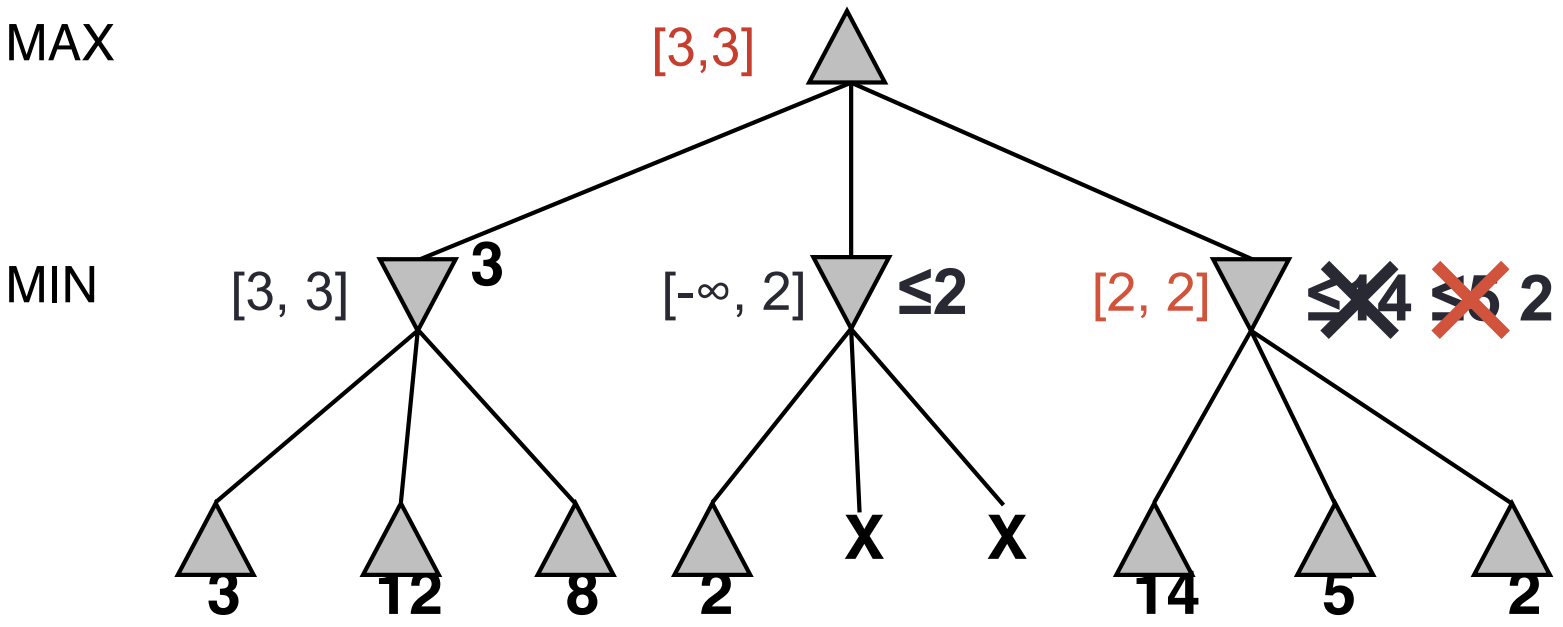
Exemplo de corte α-β



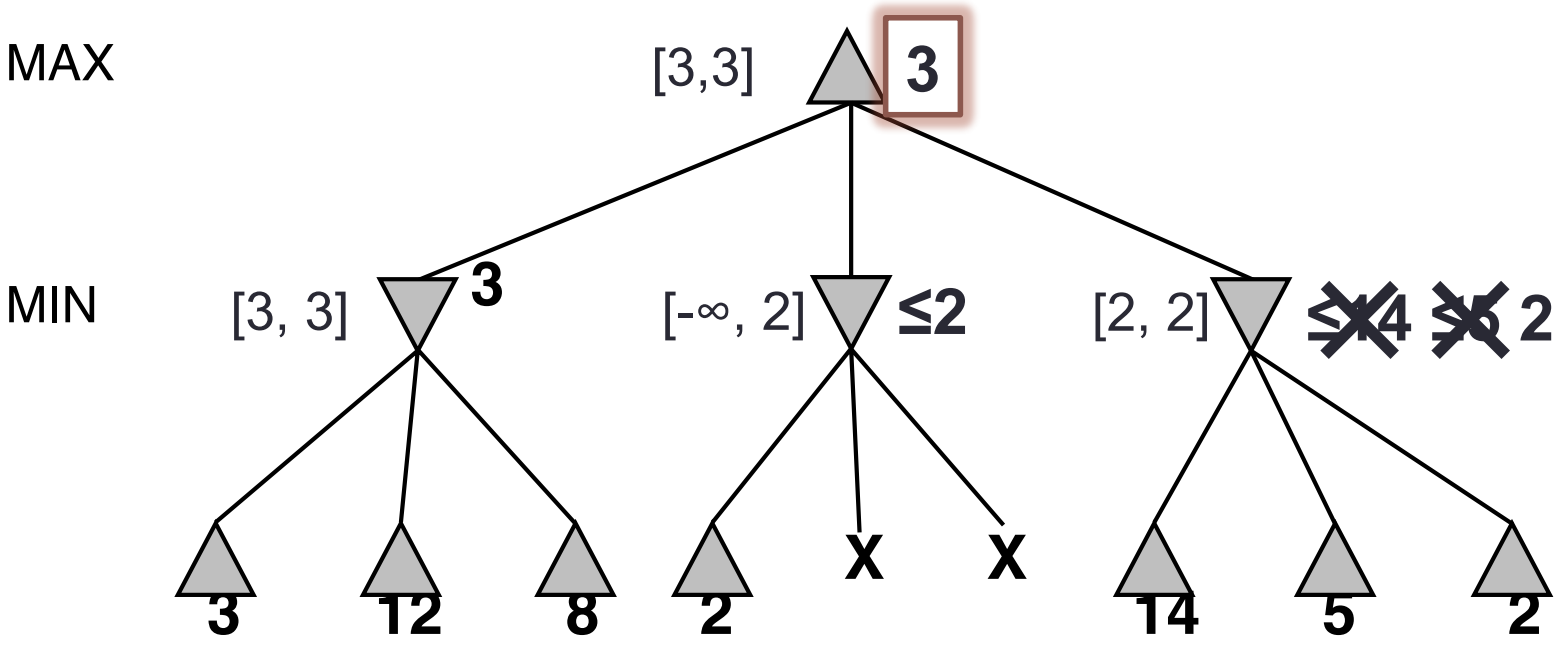
Exemplo de corte α-β



Exemplo de corte α-β



Exemplo de corte α-β



Algoritmo α-β

function ALPHA-BETA-SEARCH(*state*) returns *an action*
  inputs: *state*, current state in game
  $v \leftarrow$ MAX-VALUE(*state*, $-\infty$, $+\infty$)
  return the *action* in SUCCESSORS(*state*) with value *v*

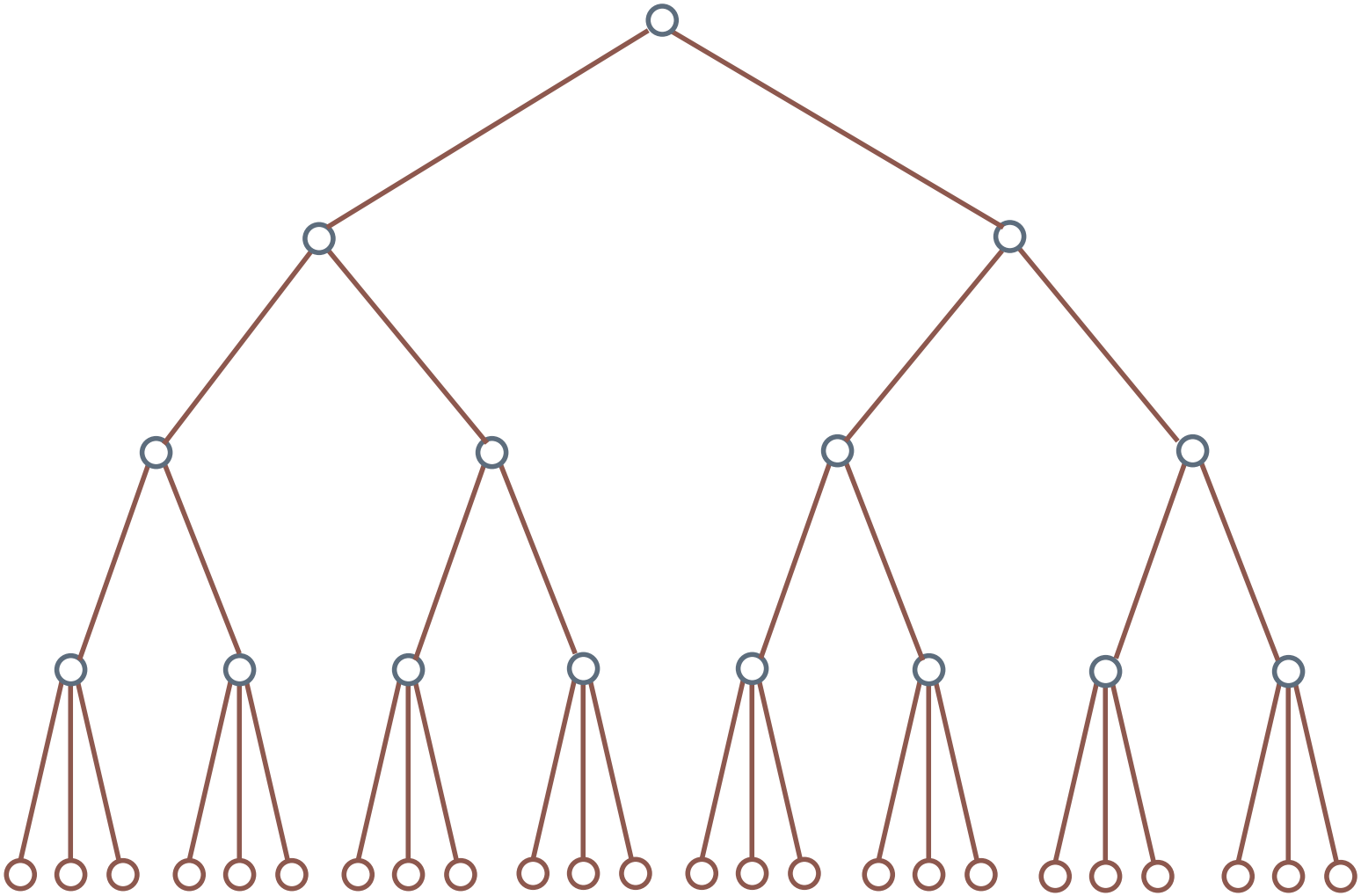
---

function MAX-VALUE(*state*, α , β) 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, α , β))
    if $v \geq \beta$ then return v
    $\alpha \leftarrow$ MAX(α ,v)
  return *v*

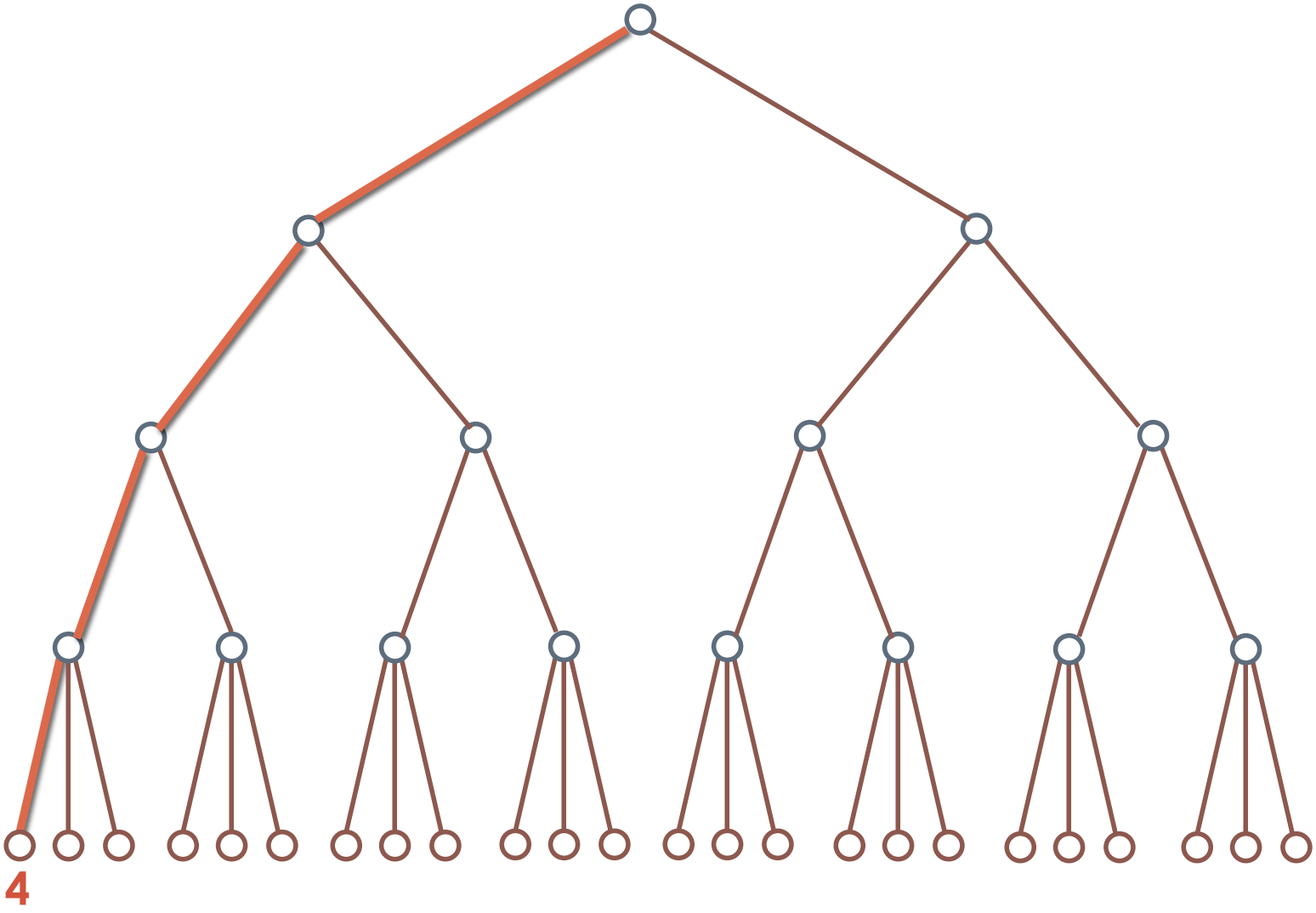
Algoritmo α-β

function MIN-VALUE(*state*, α , β) 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, α , β))
    if $v \leq \alpha$ then return v
    $\beta \leftarrow$ MIN(β ,v)
  return v

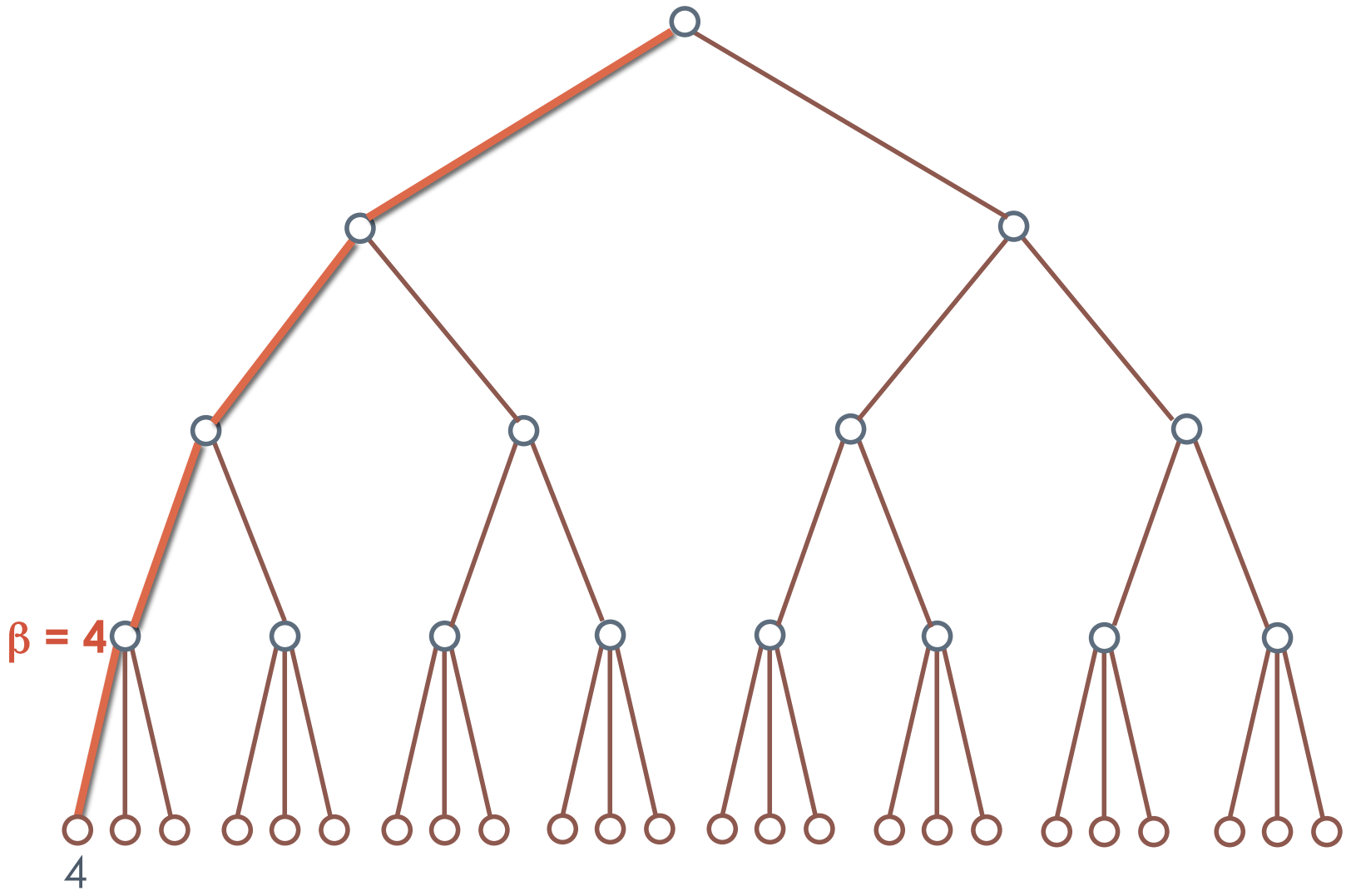
Algoritmo de corte α-β



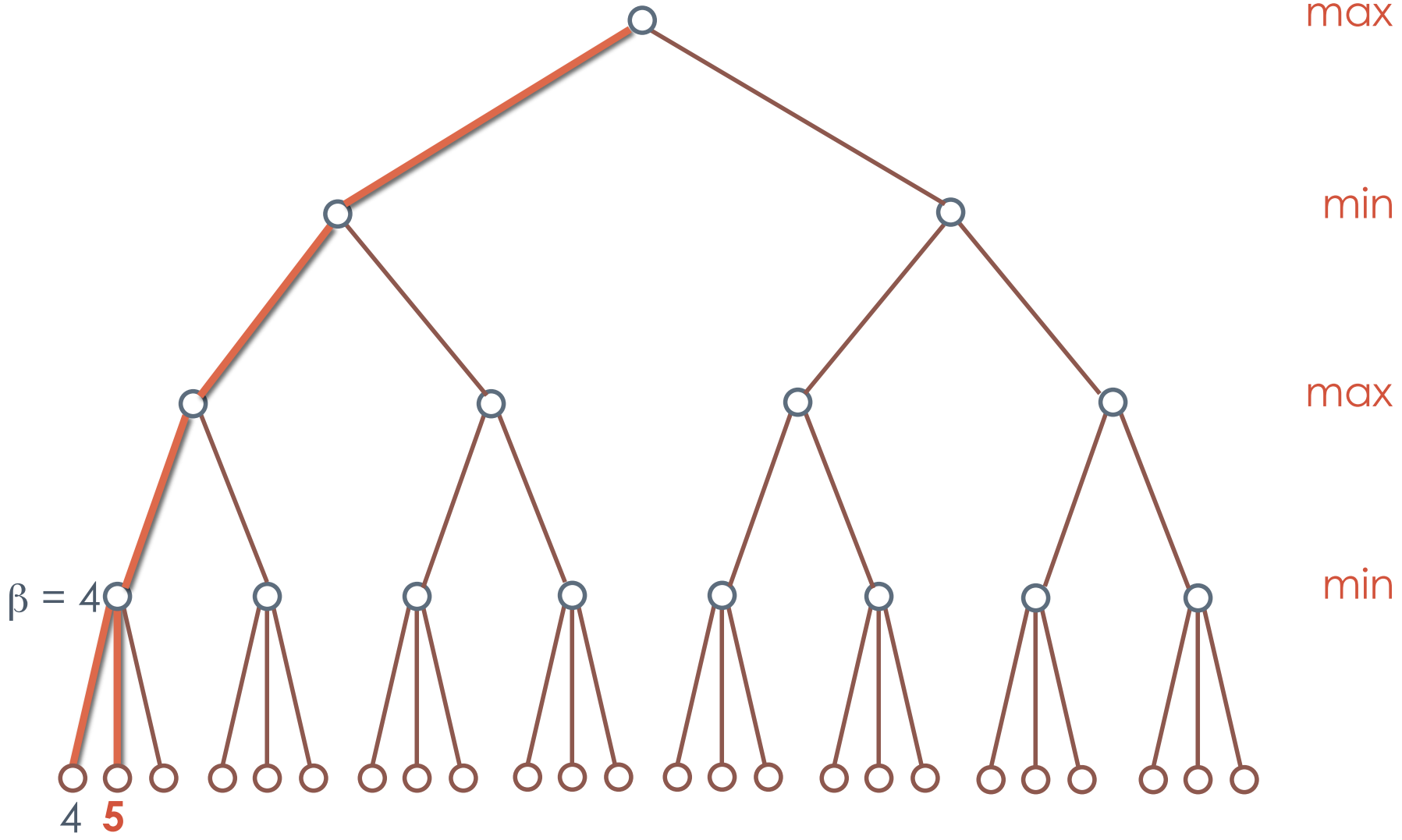
Algoritmo de corte α-β



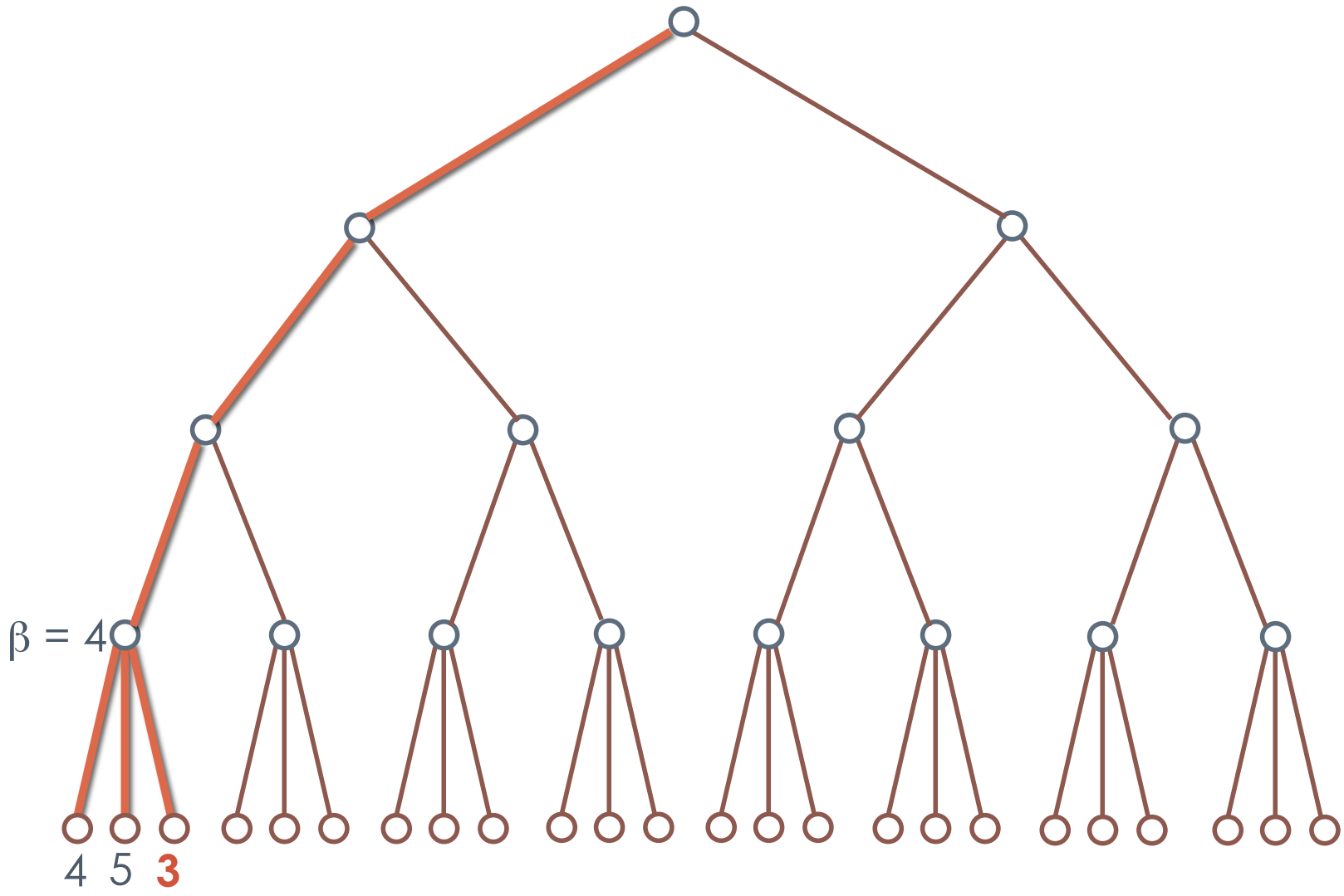
Algoritmo de corte α-β



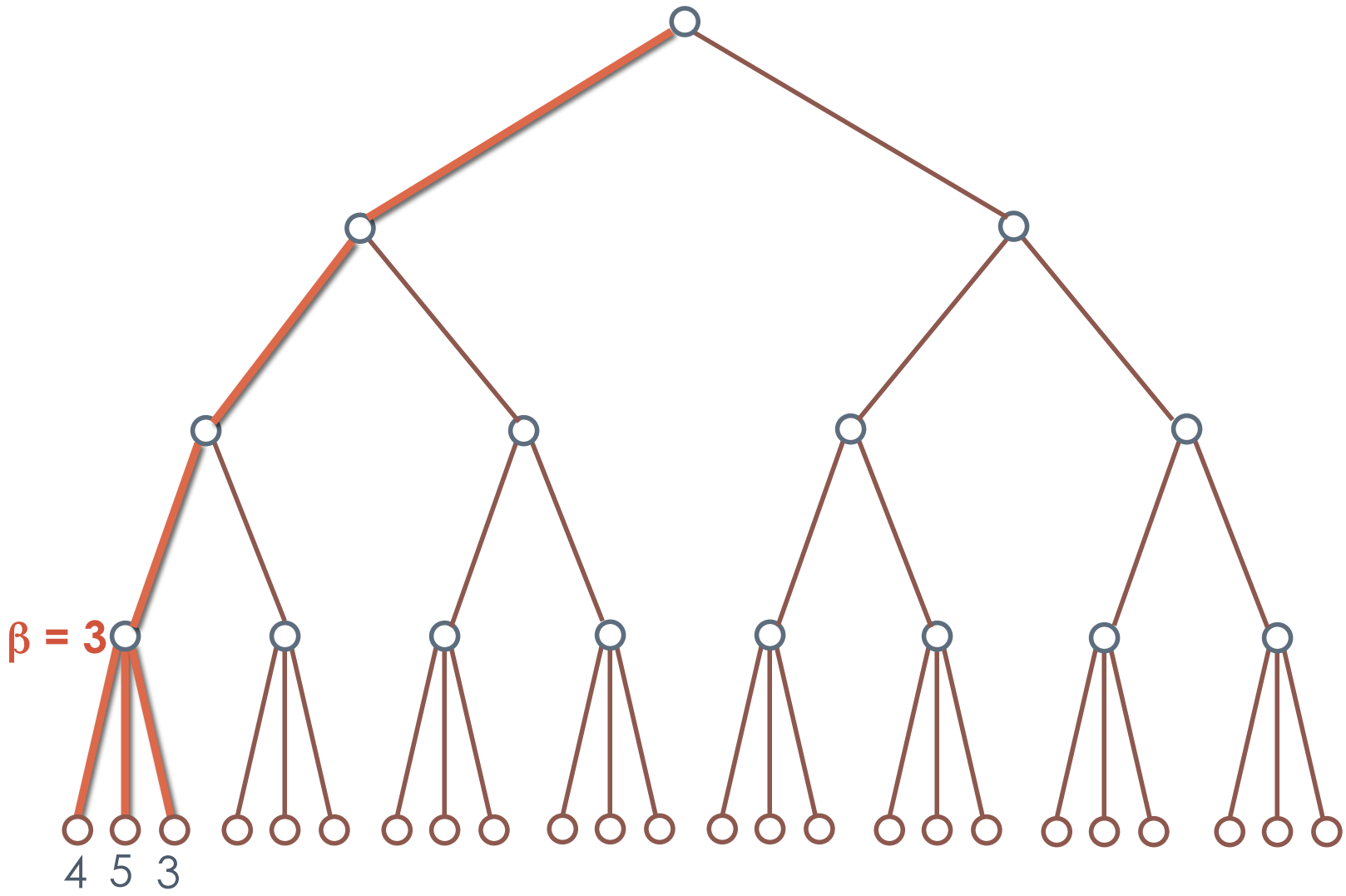
Algoritmo de corte α-β



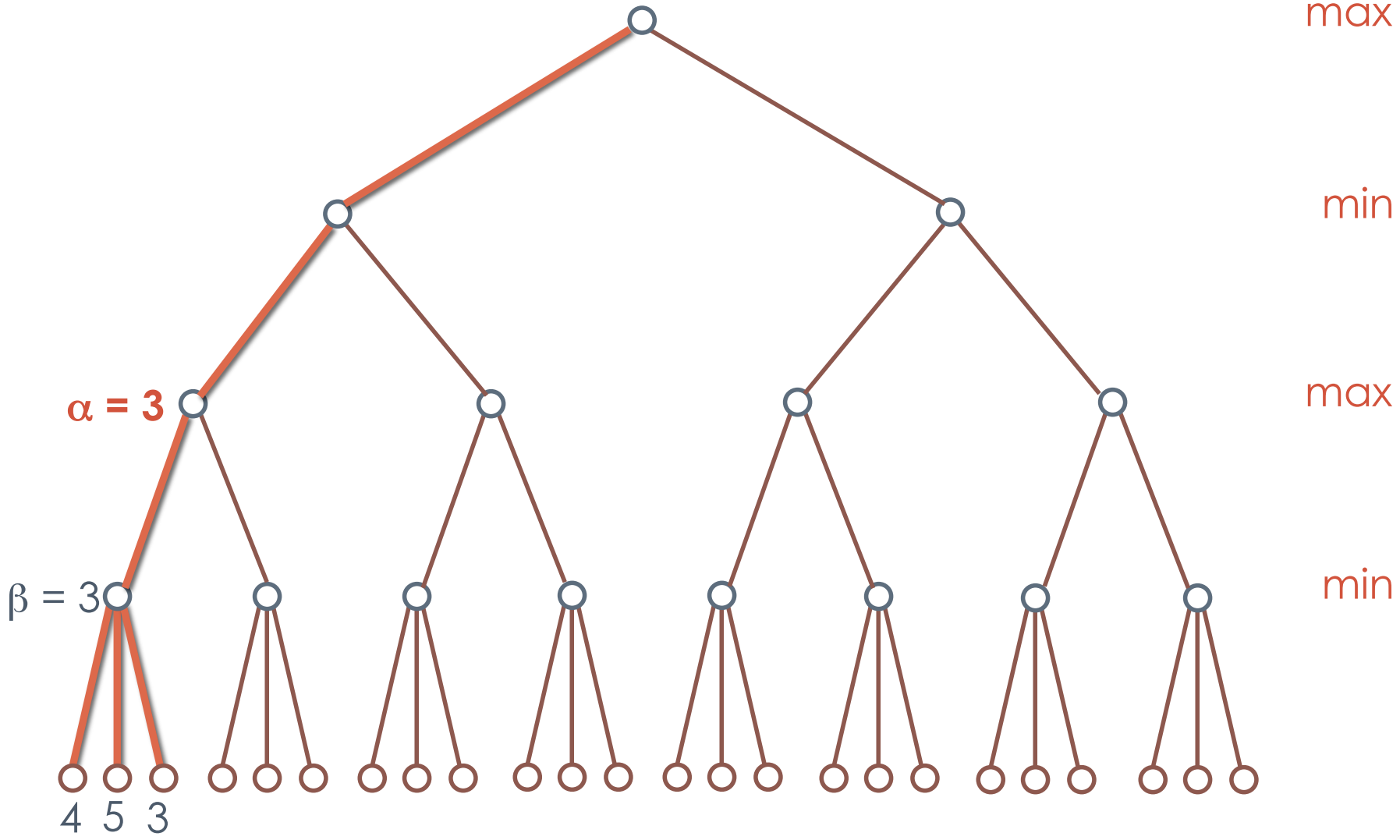
Algoritmo de corte α-β



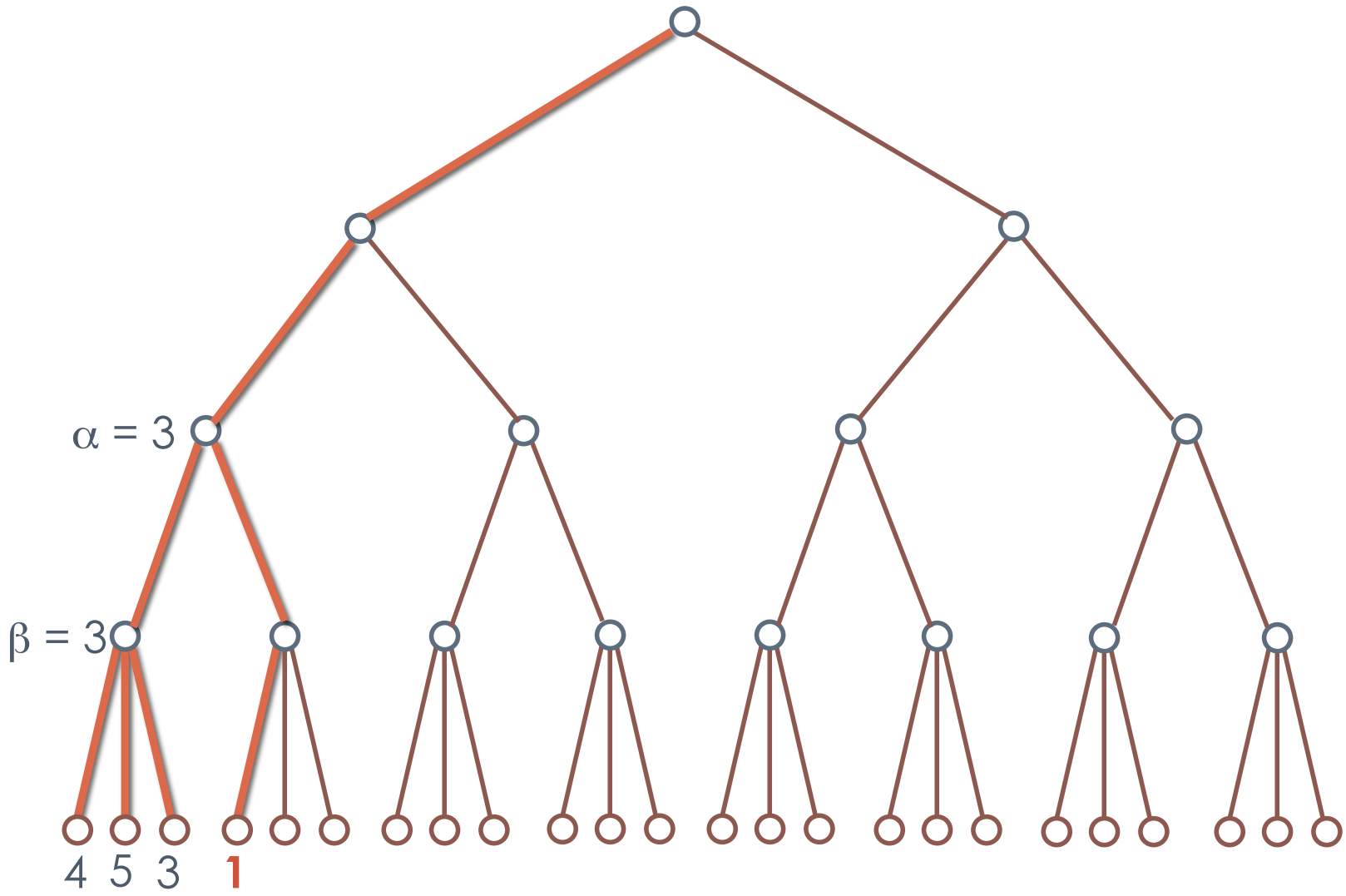
Algoritmo de corte α-β



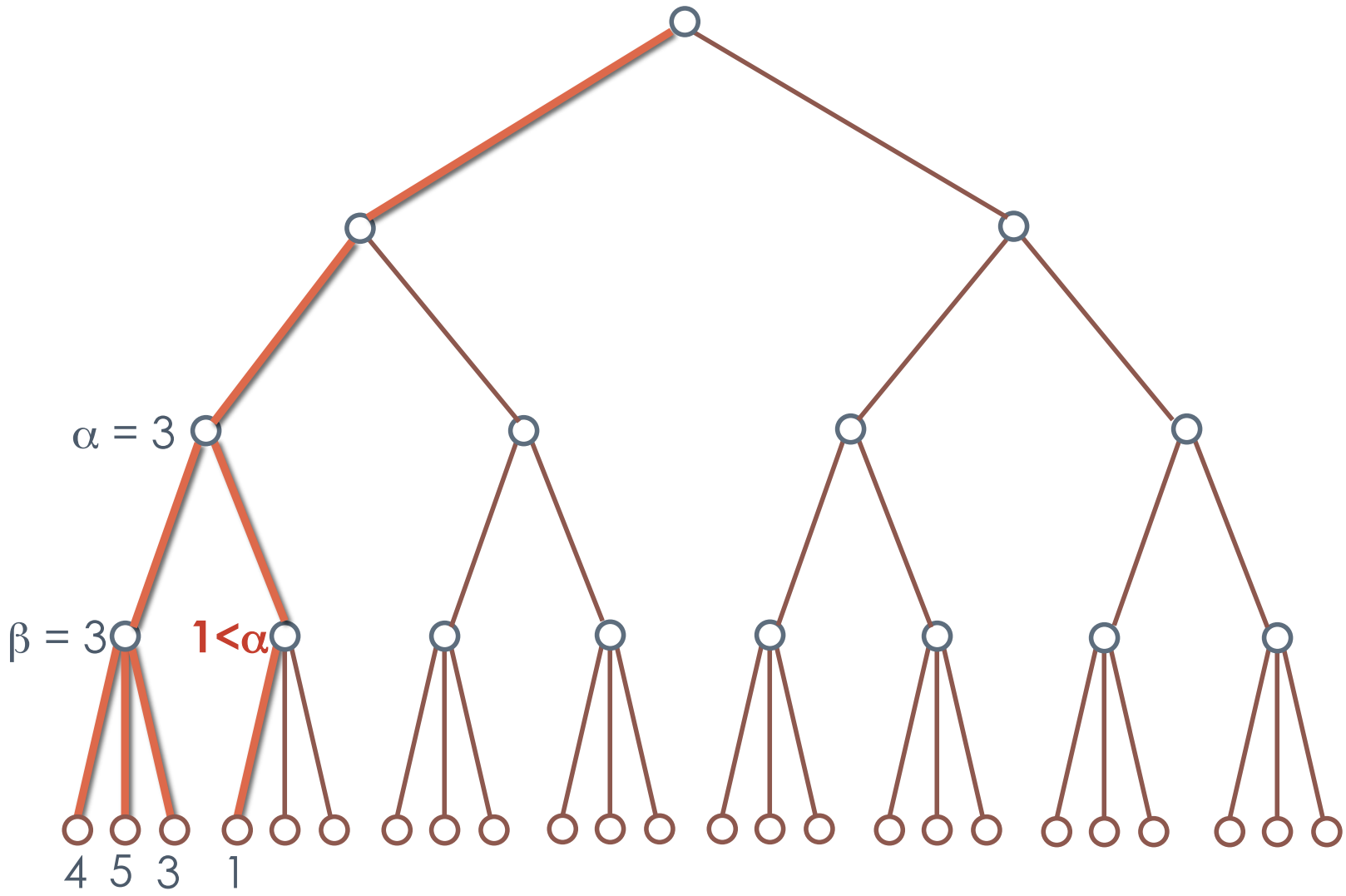
Algoritmo de corte α-β



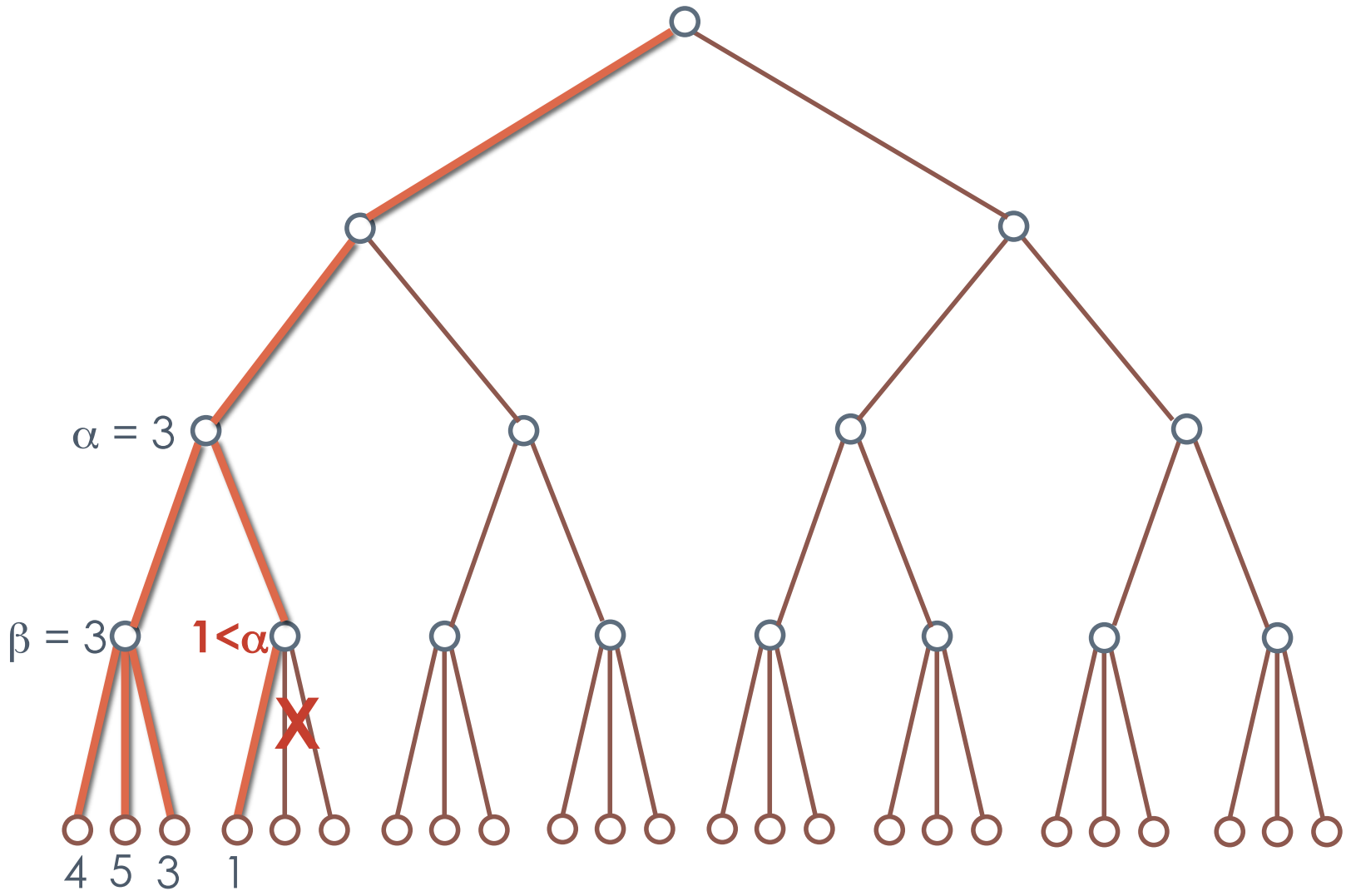
Algoritmo de corte α-β



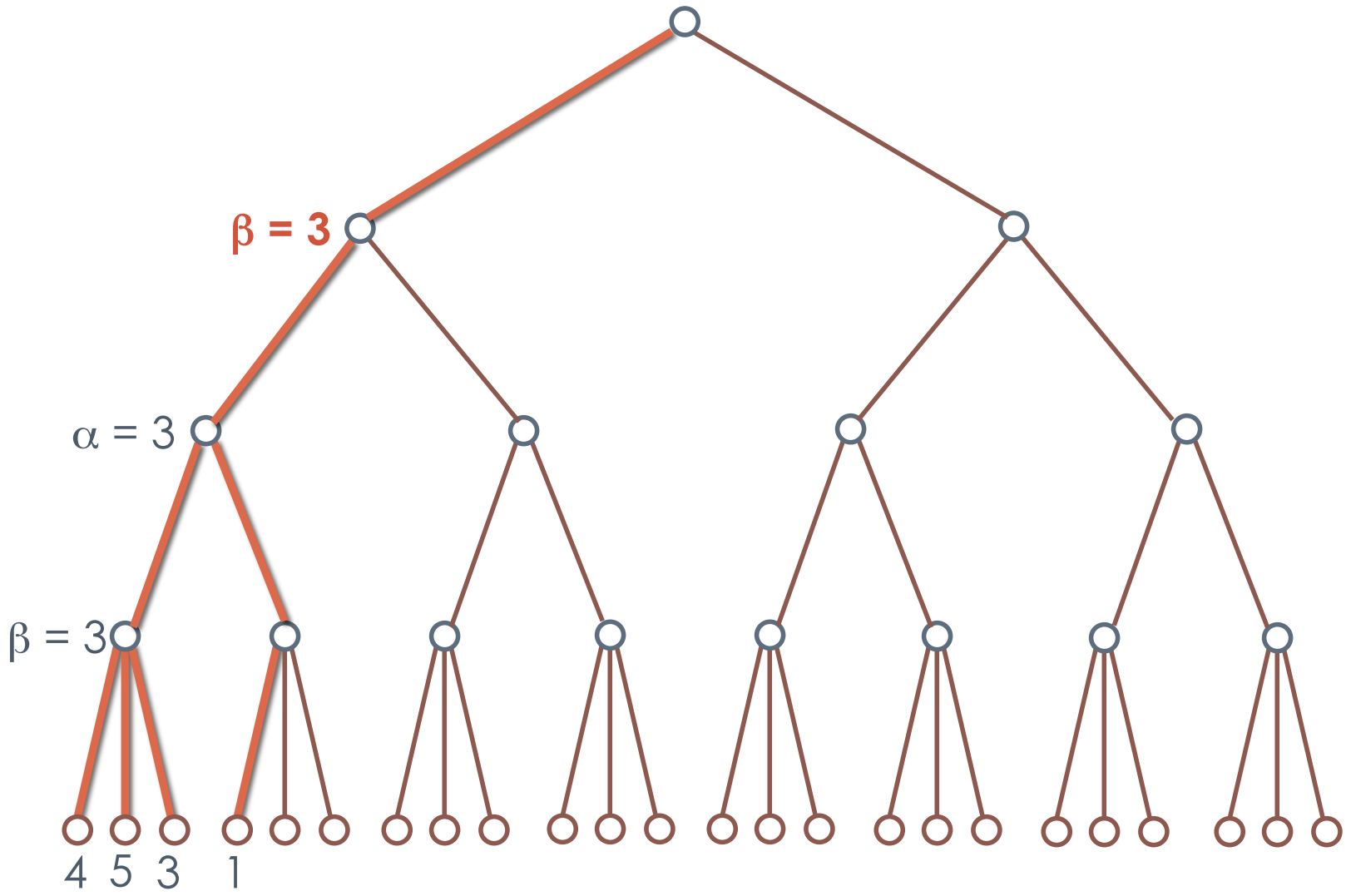
Algoritmo de corte α-β



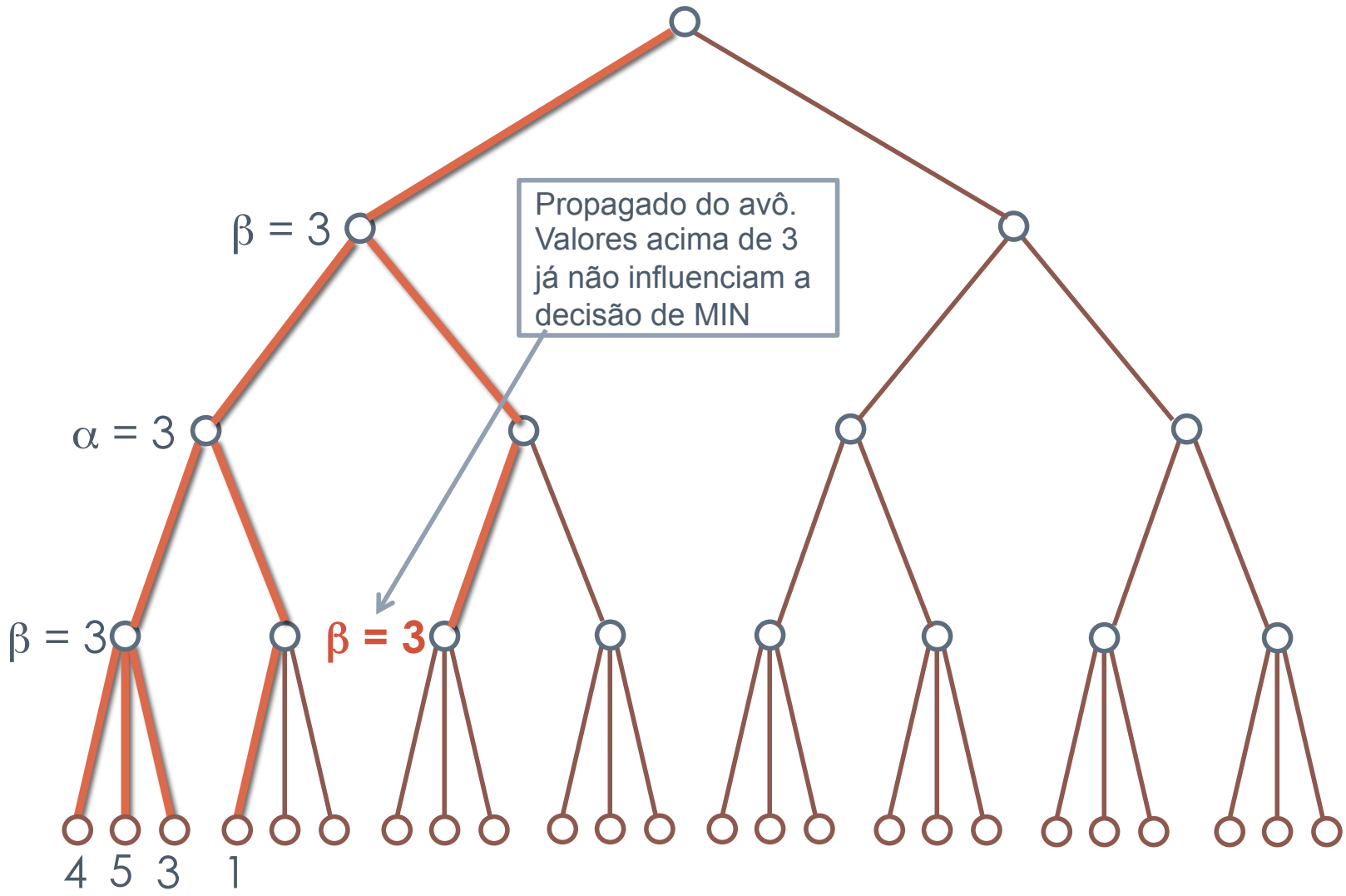
Algoritmo de corte α-β



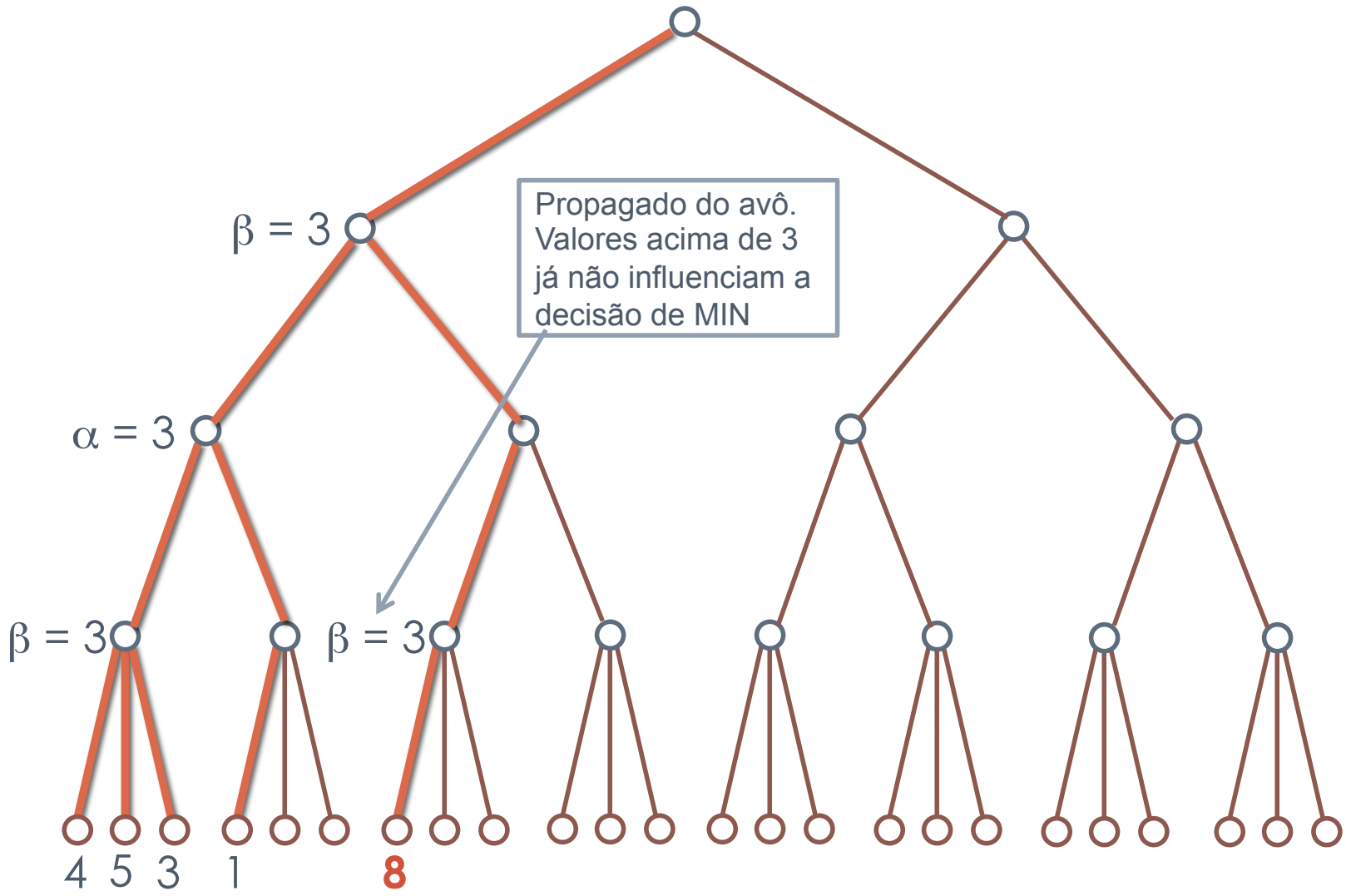
Algoritmo de corte α-β



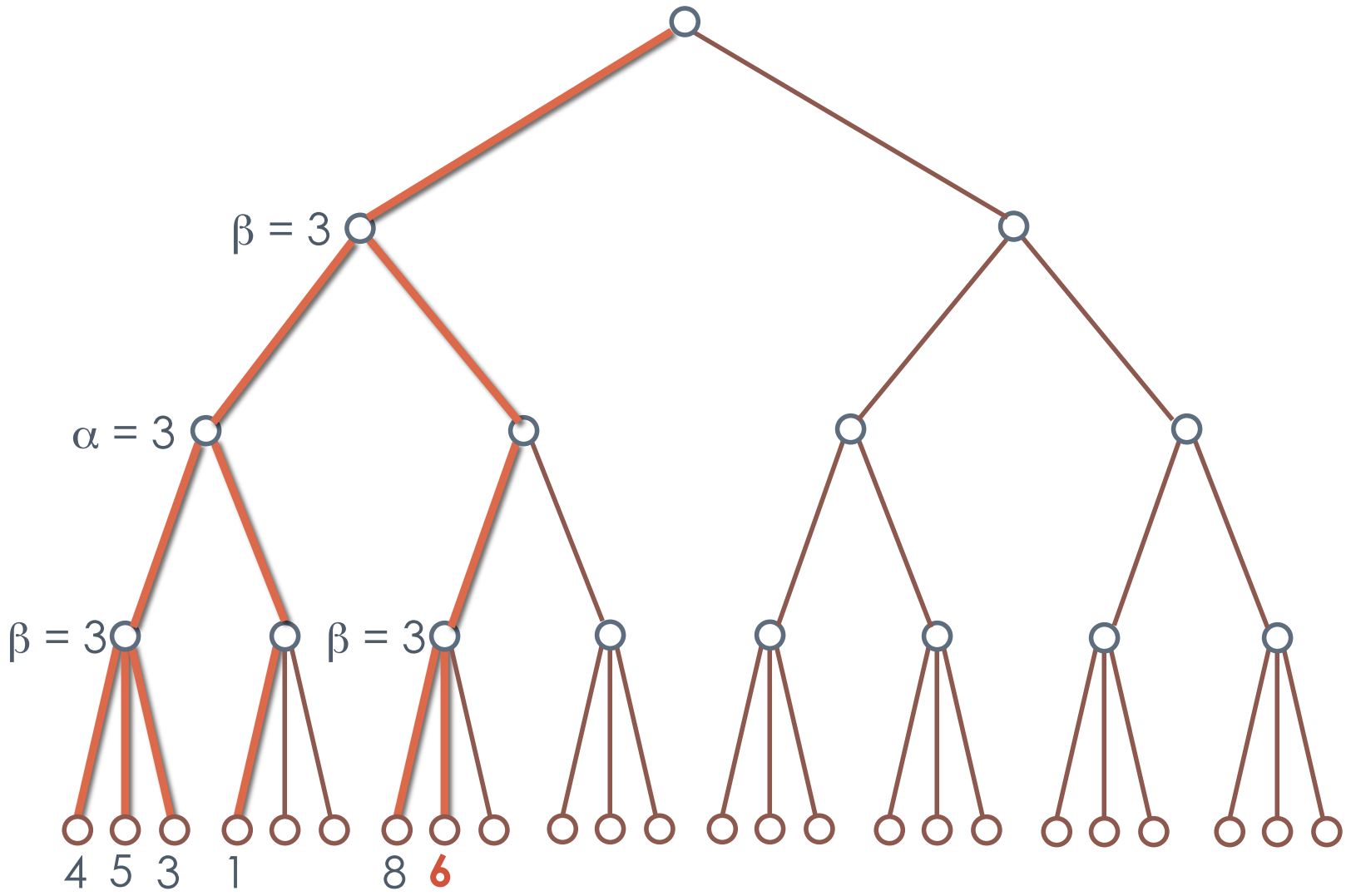
Algoritmo de corte α-β



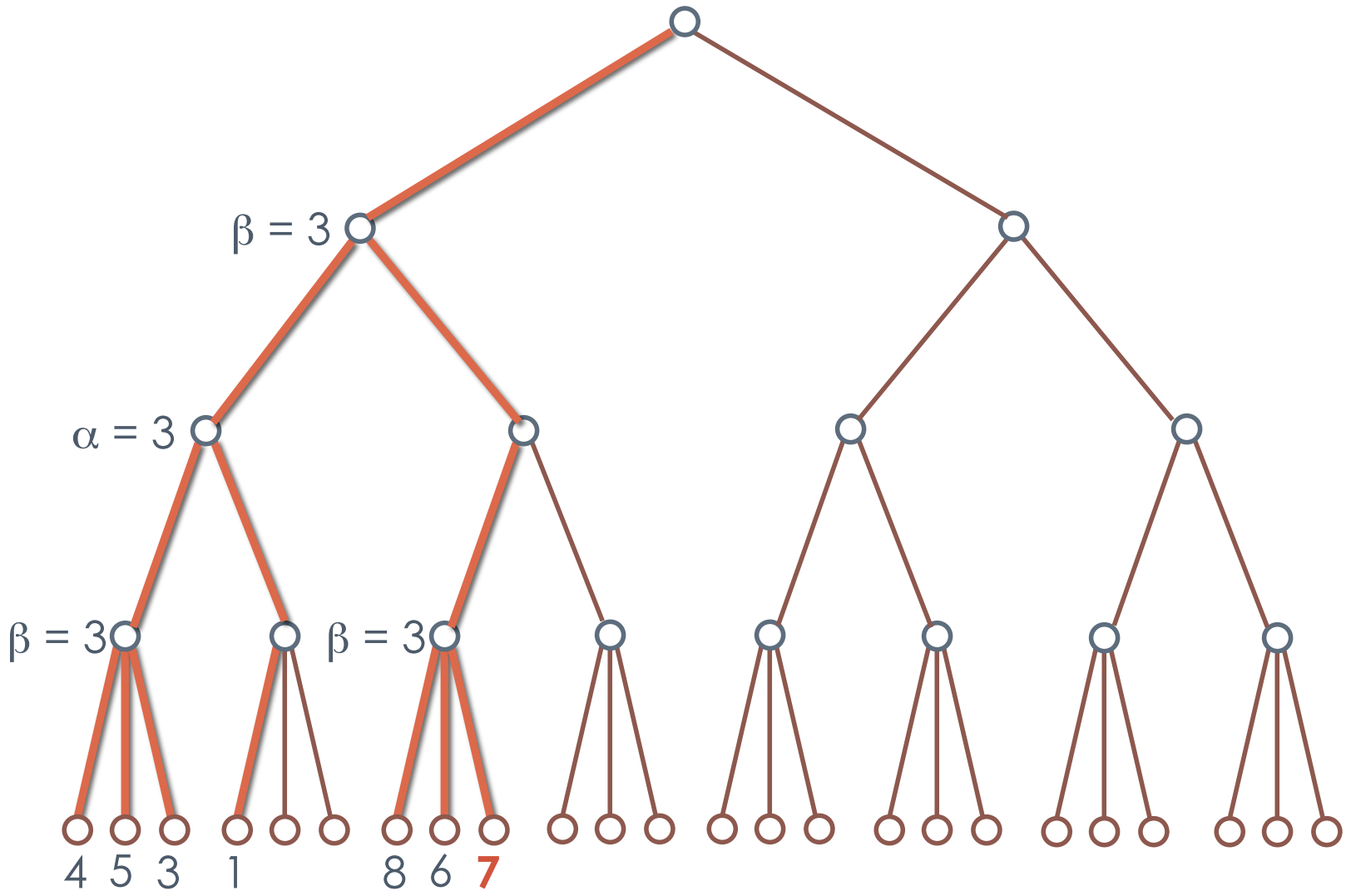
Algoritmo de corte α-β



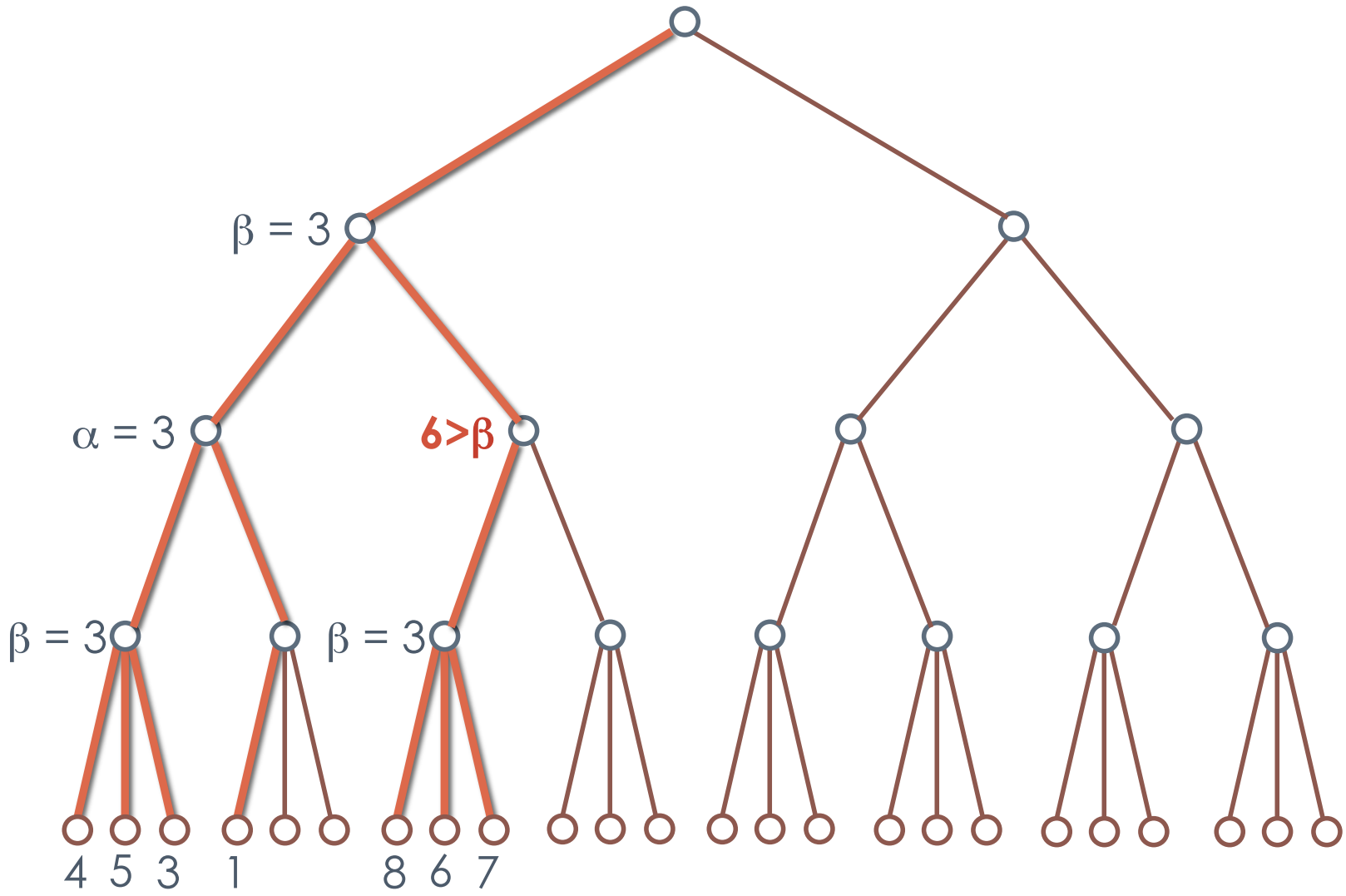
Algoritmo de corte α-β



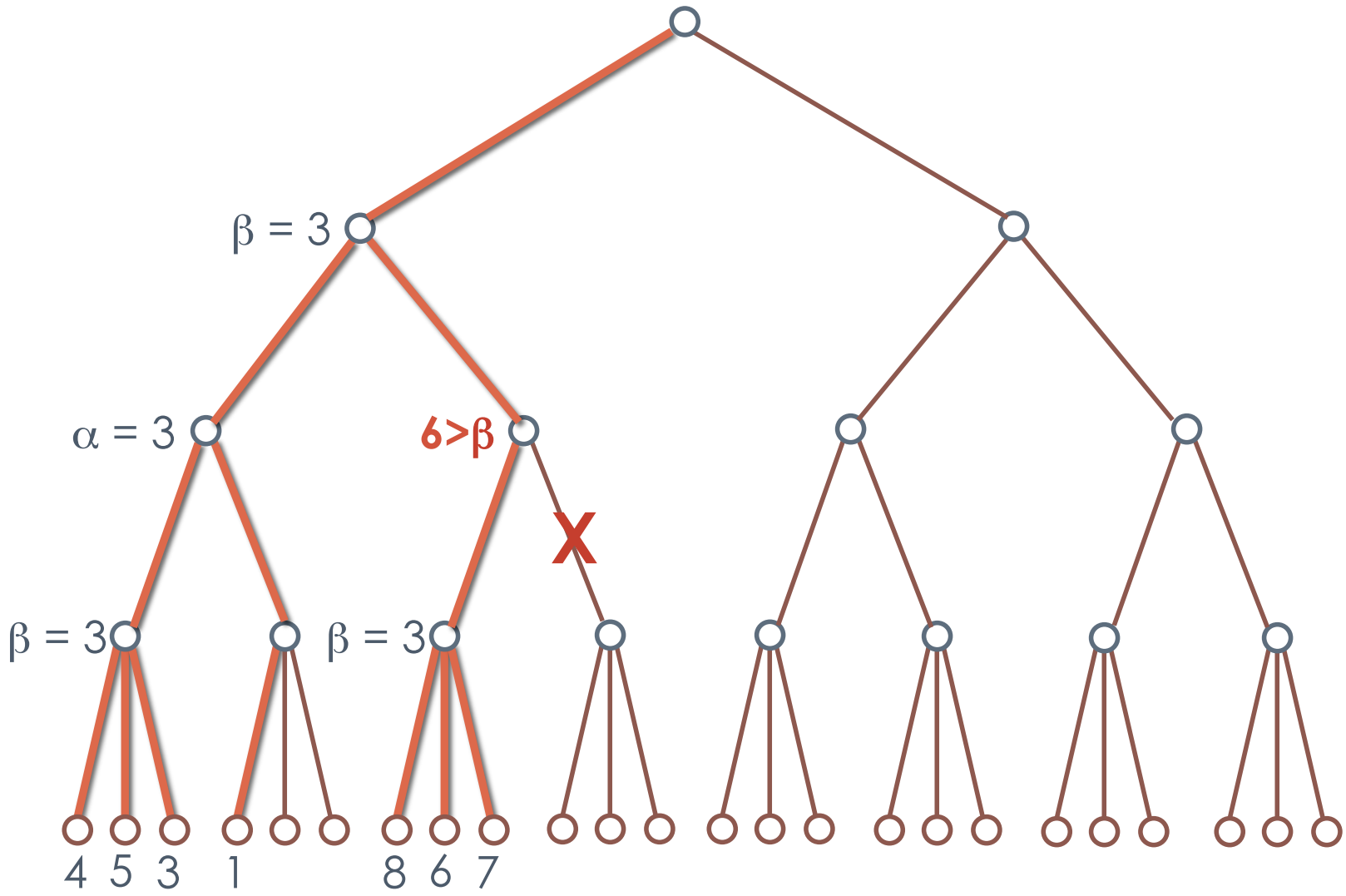
Algoritmo de corte α-β



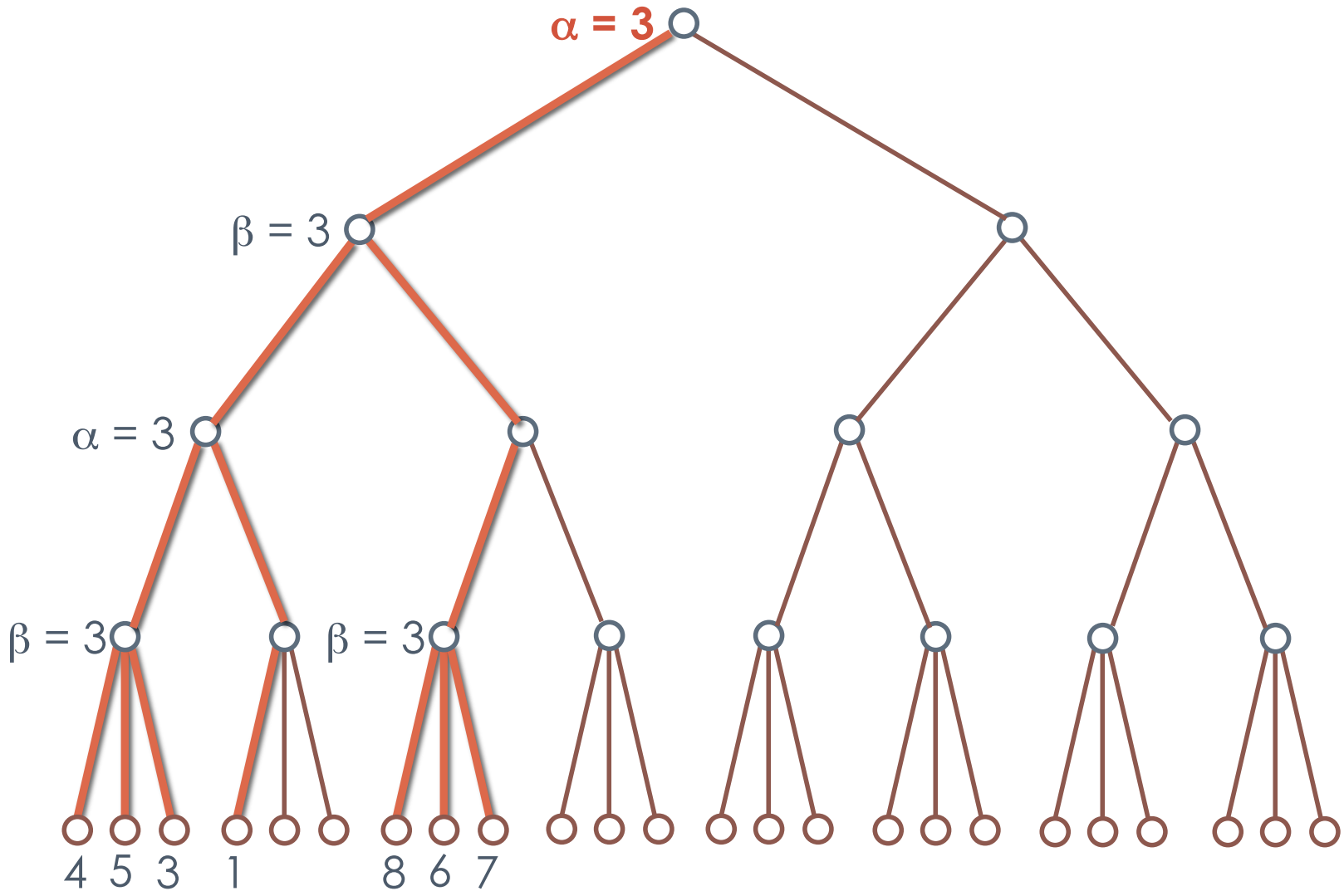
Algoritmo de corte α-β



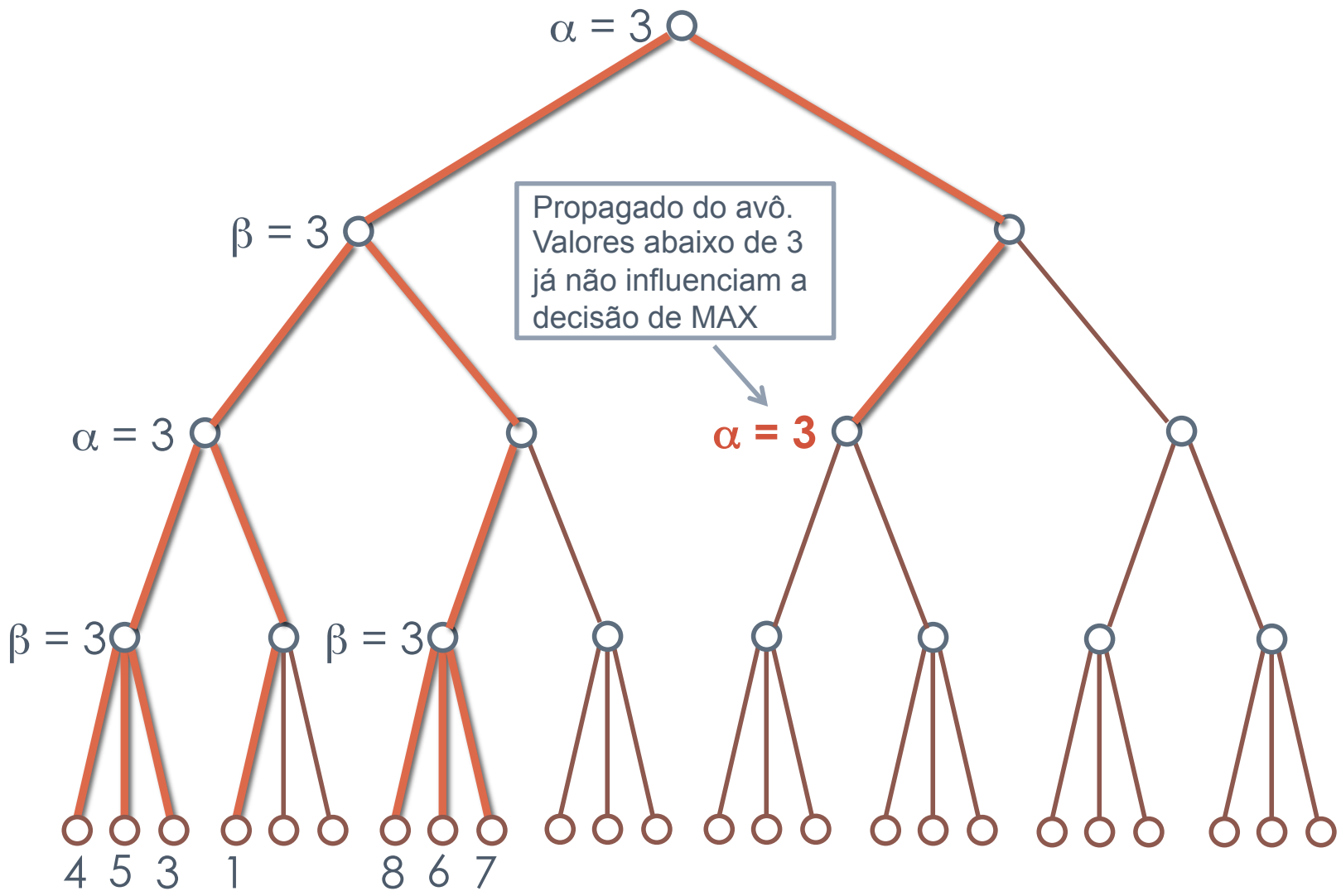
Algoritmo de corte α-β



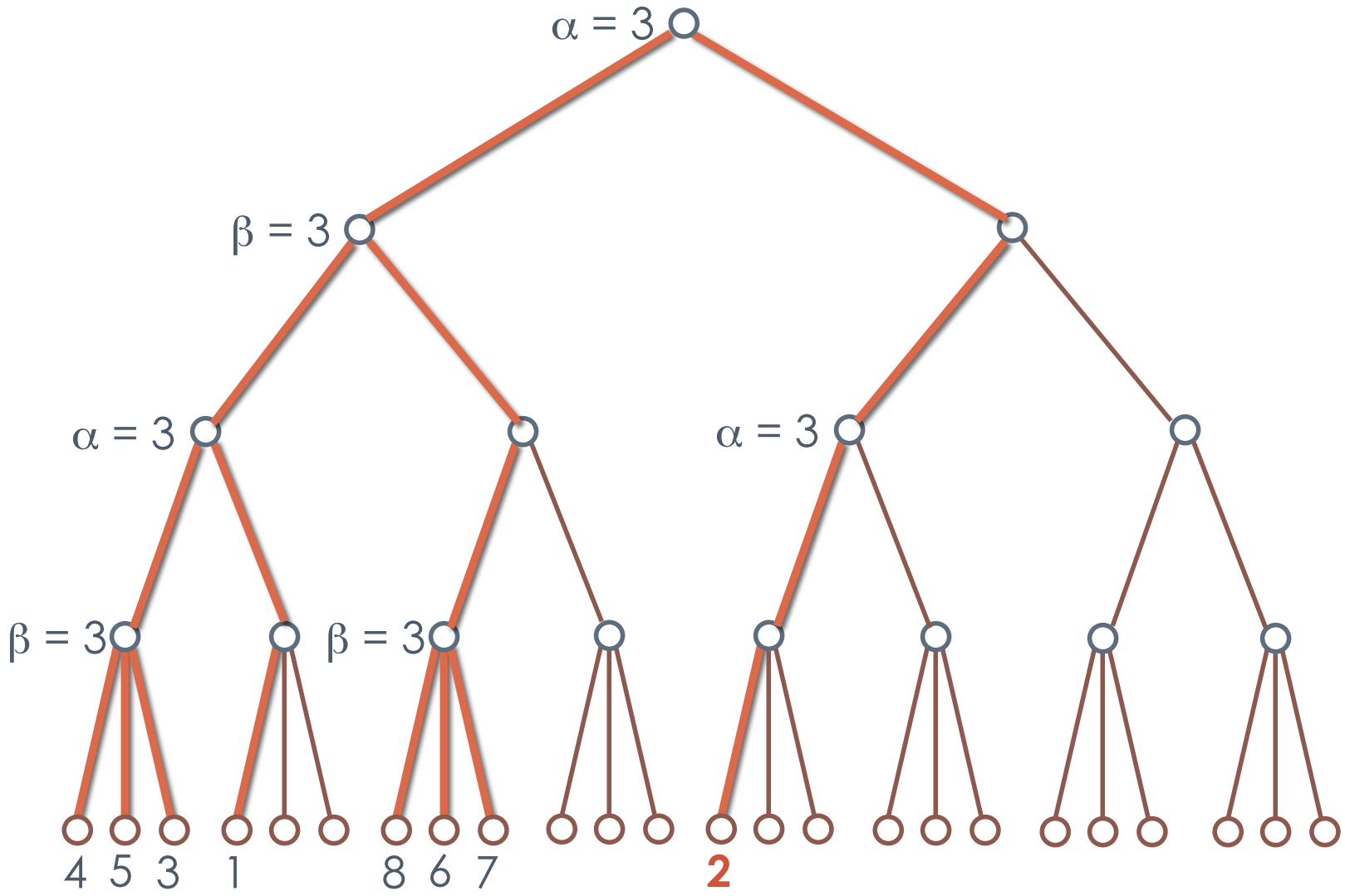
Algoritmo de corte α-β



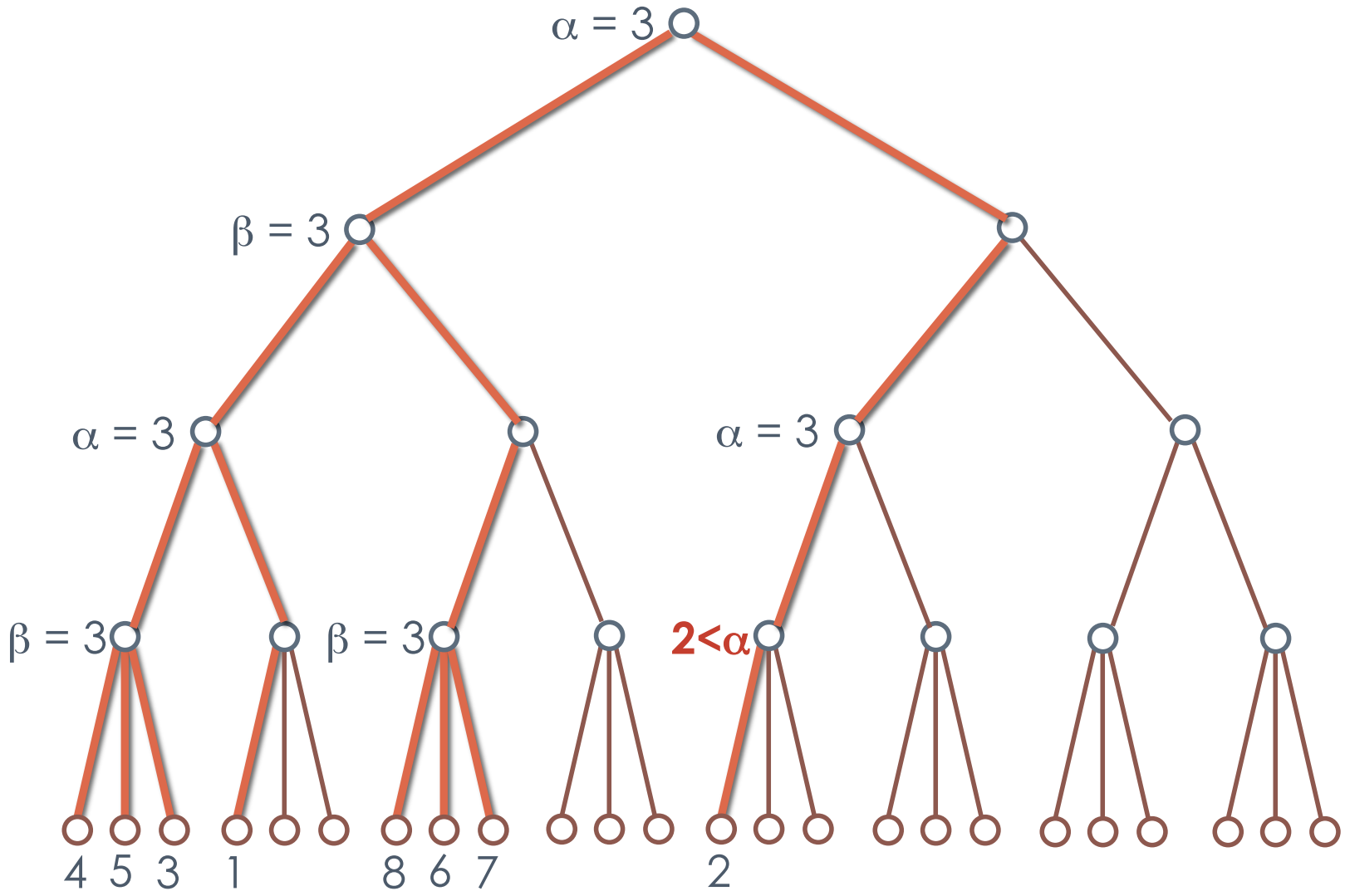
Algoritmo de corte α-β



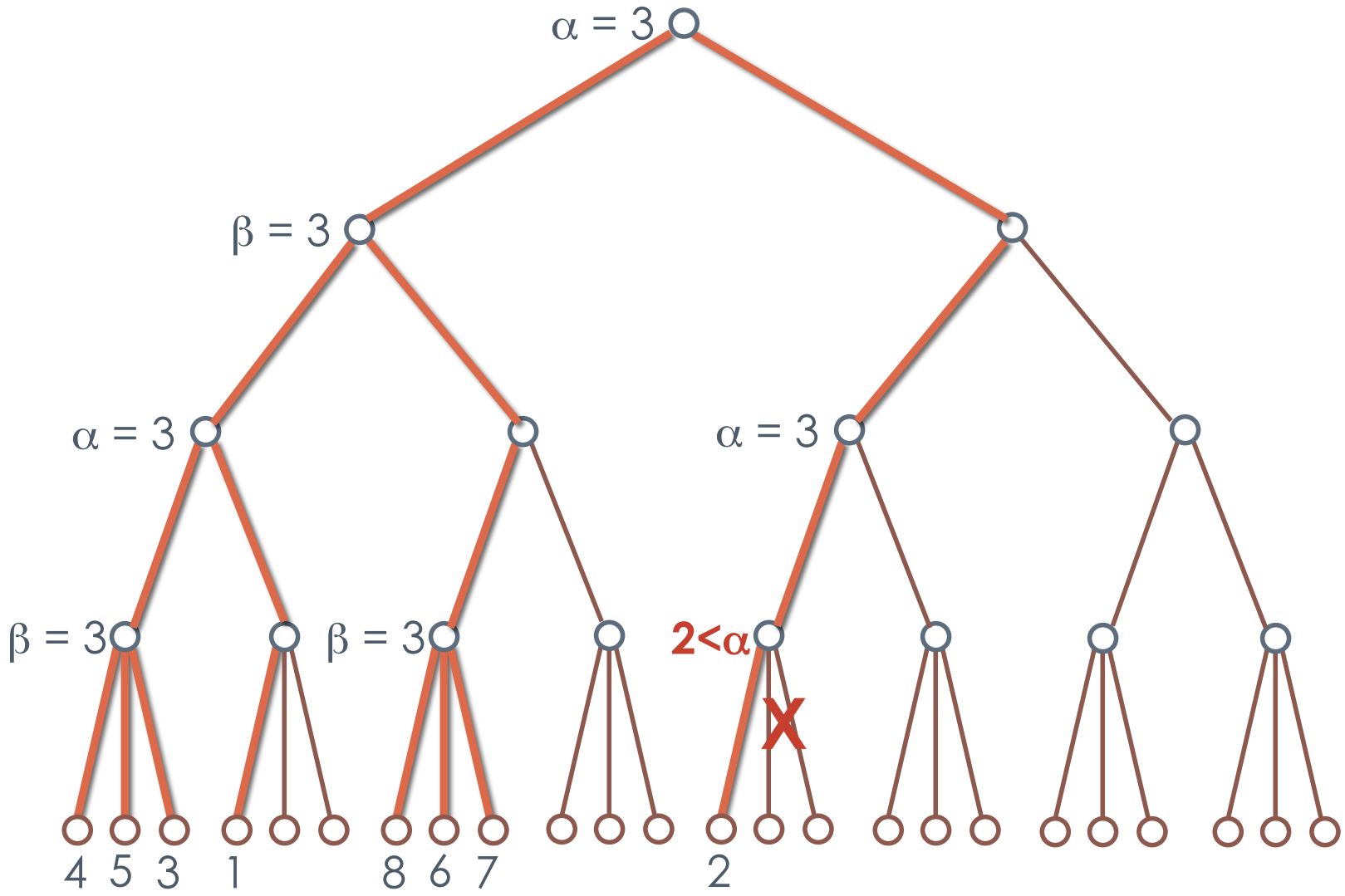
Algoritmo de corte α-β



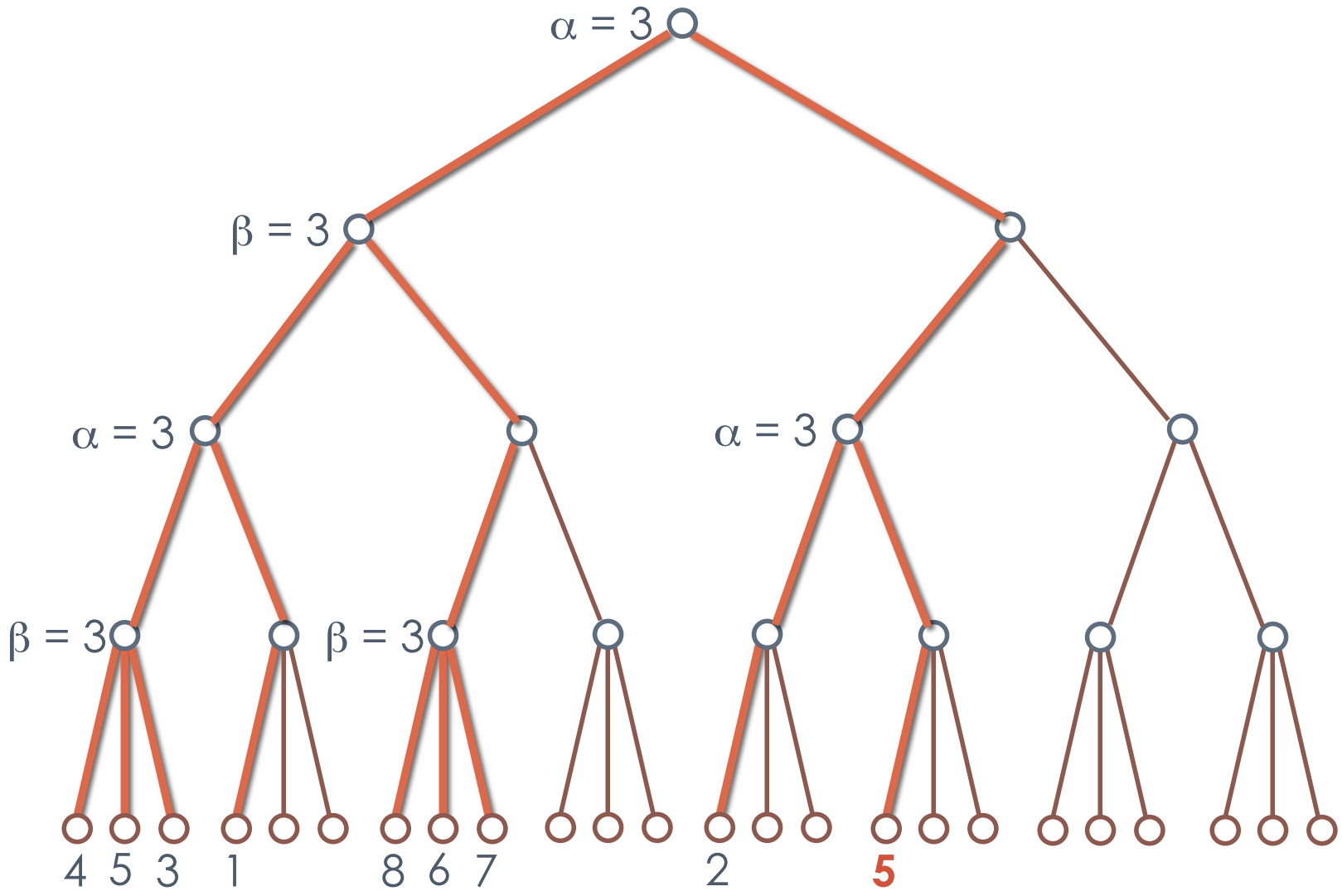
Algoritmo de corte α-β



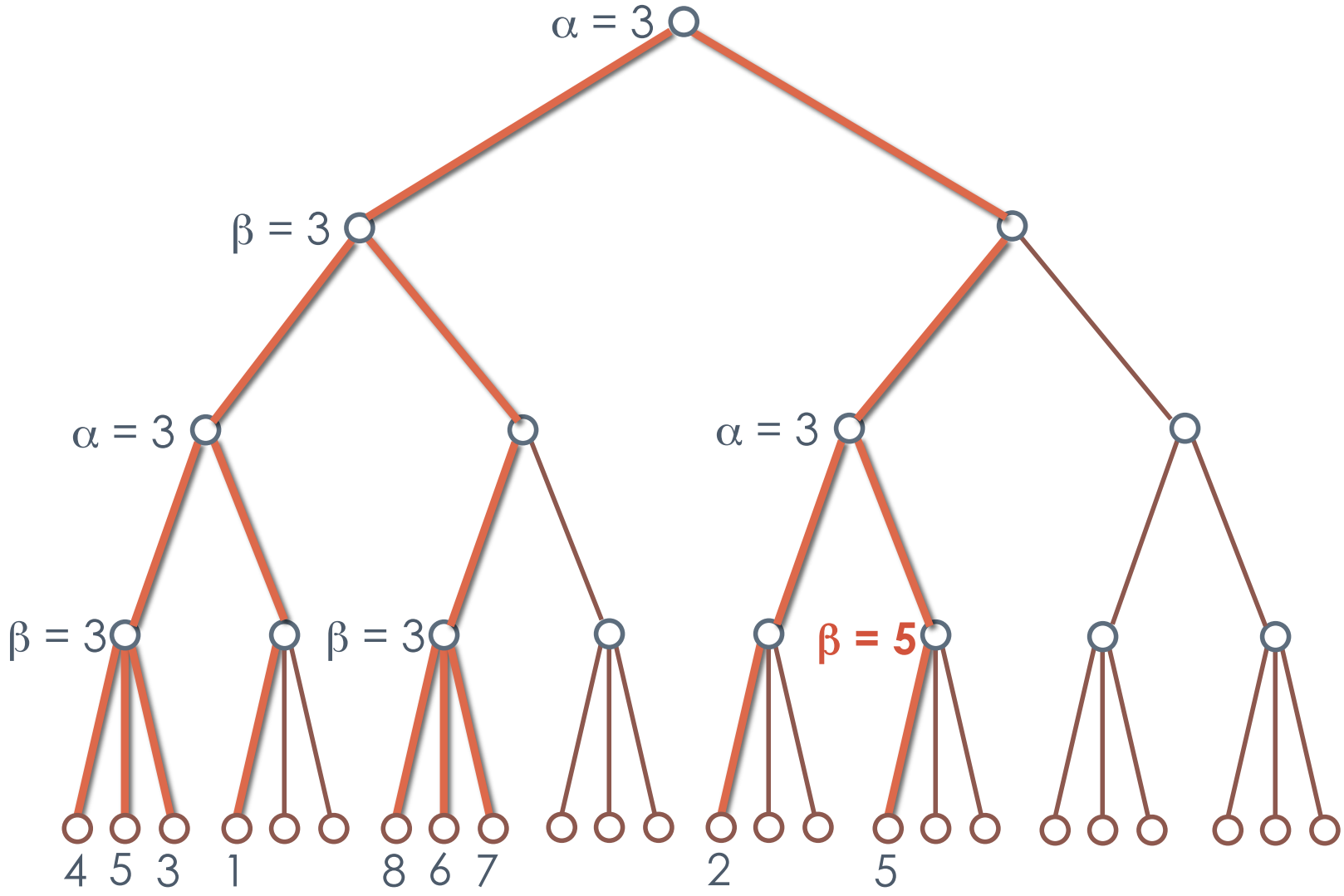
Algoritmo de corte α-β



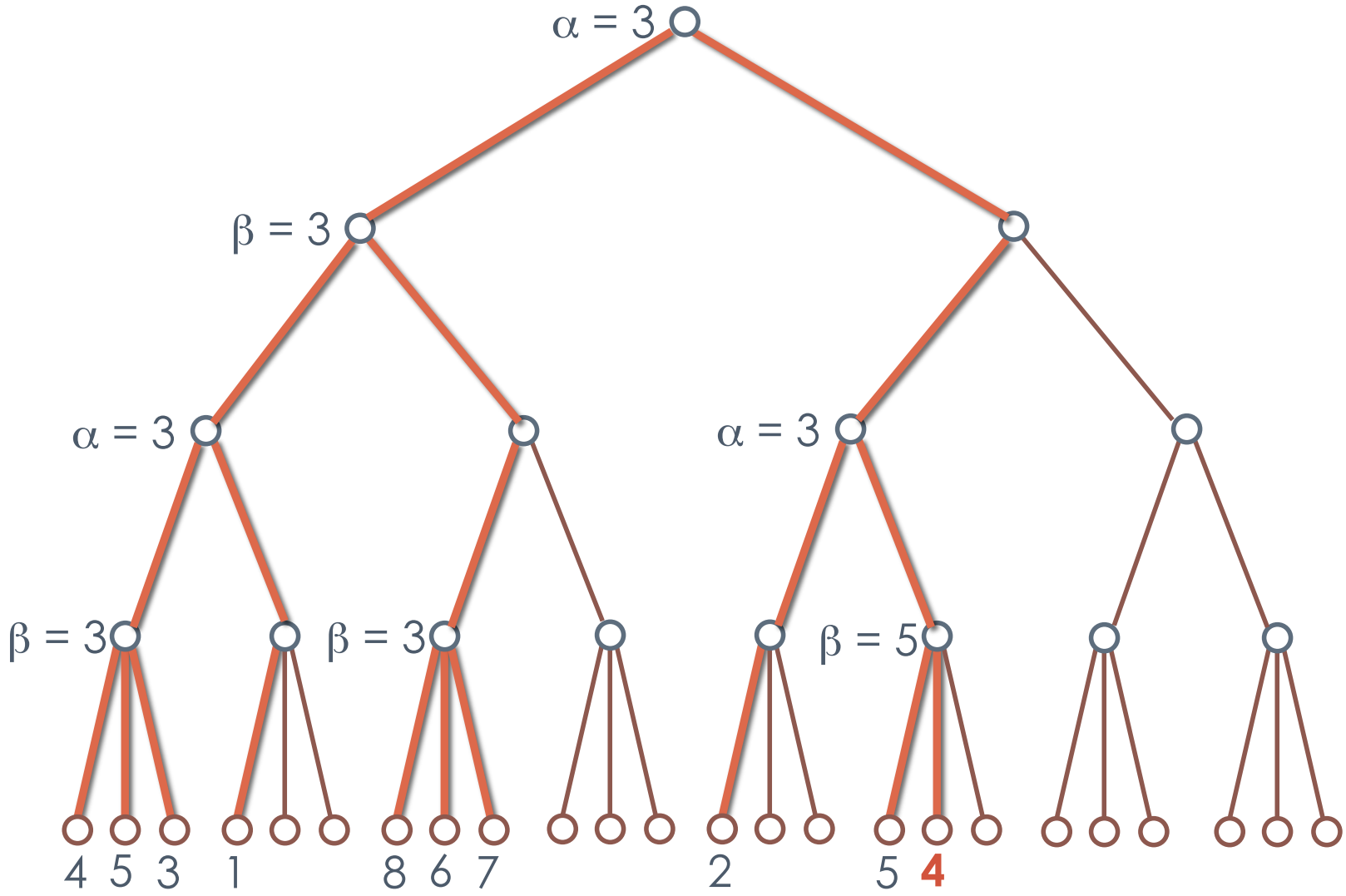
Algoritmo de corte α-β



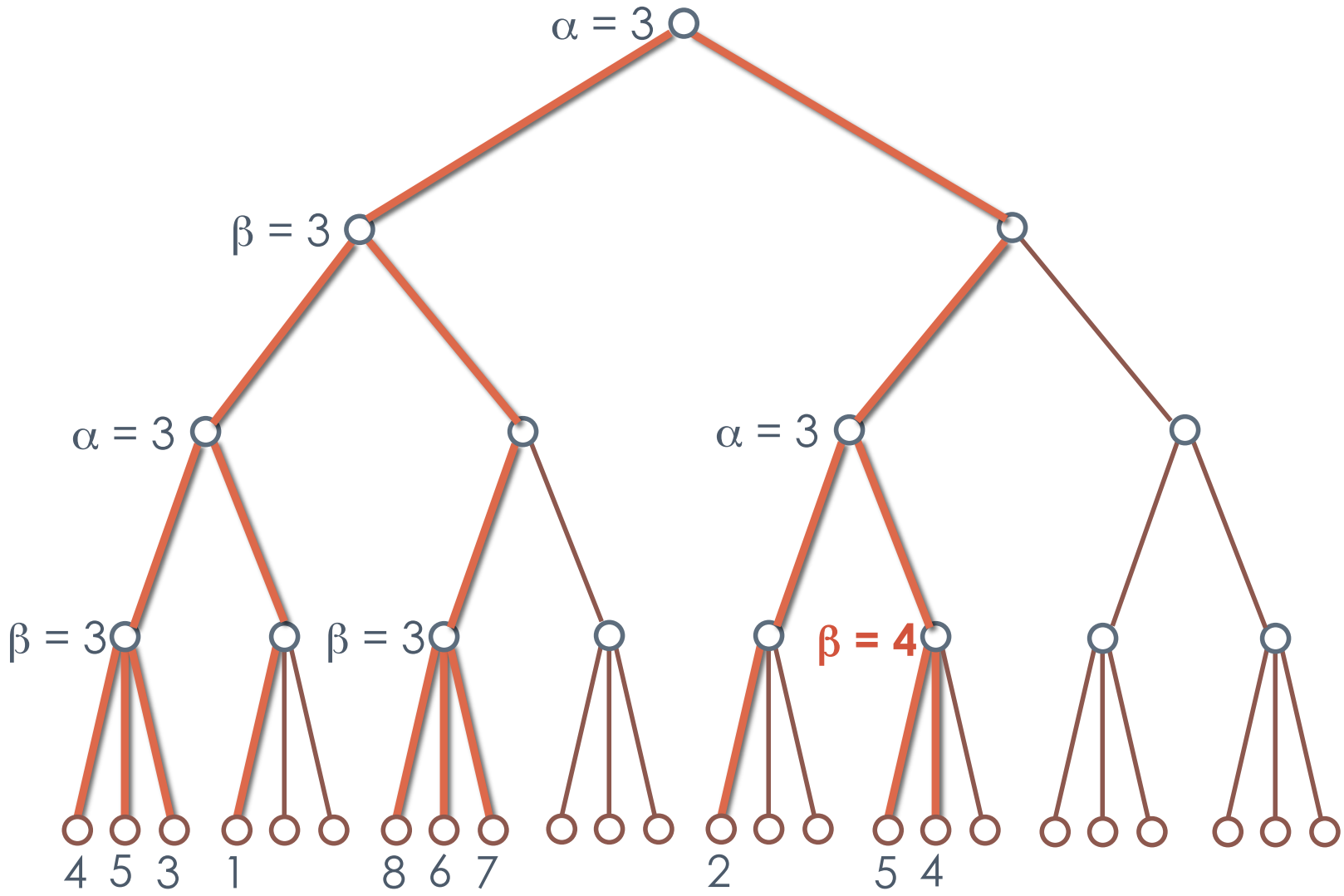
Algoritmo de corte α-β



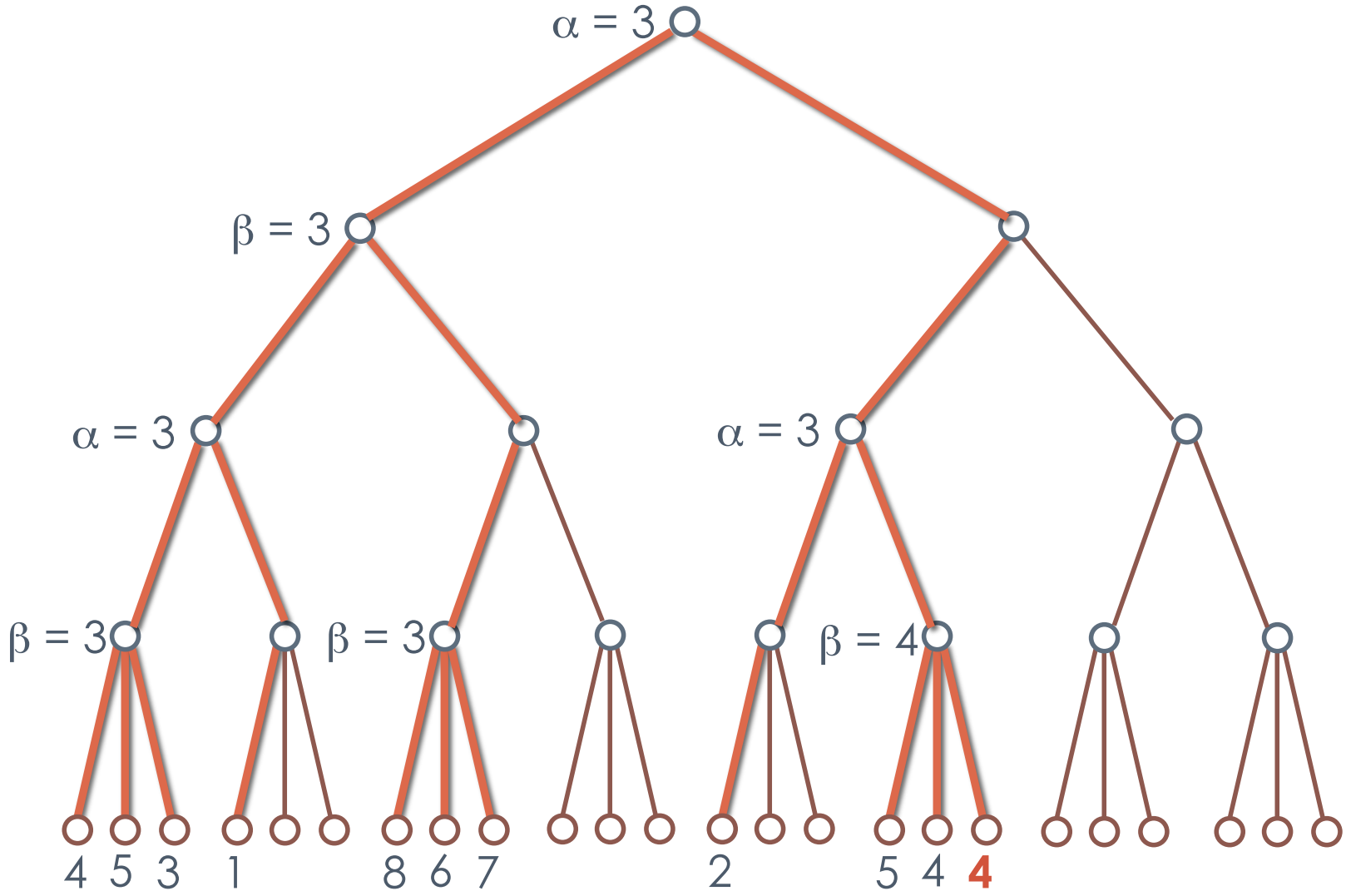
Algoritmo de corte α-β



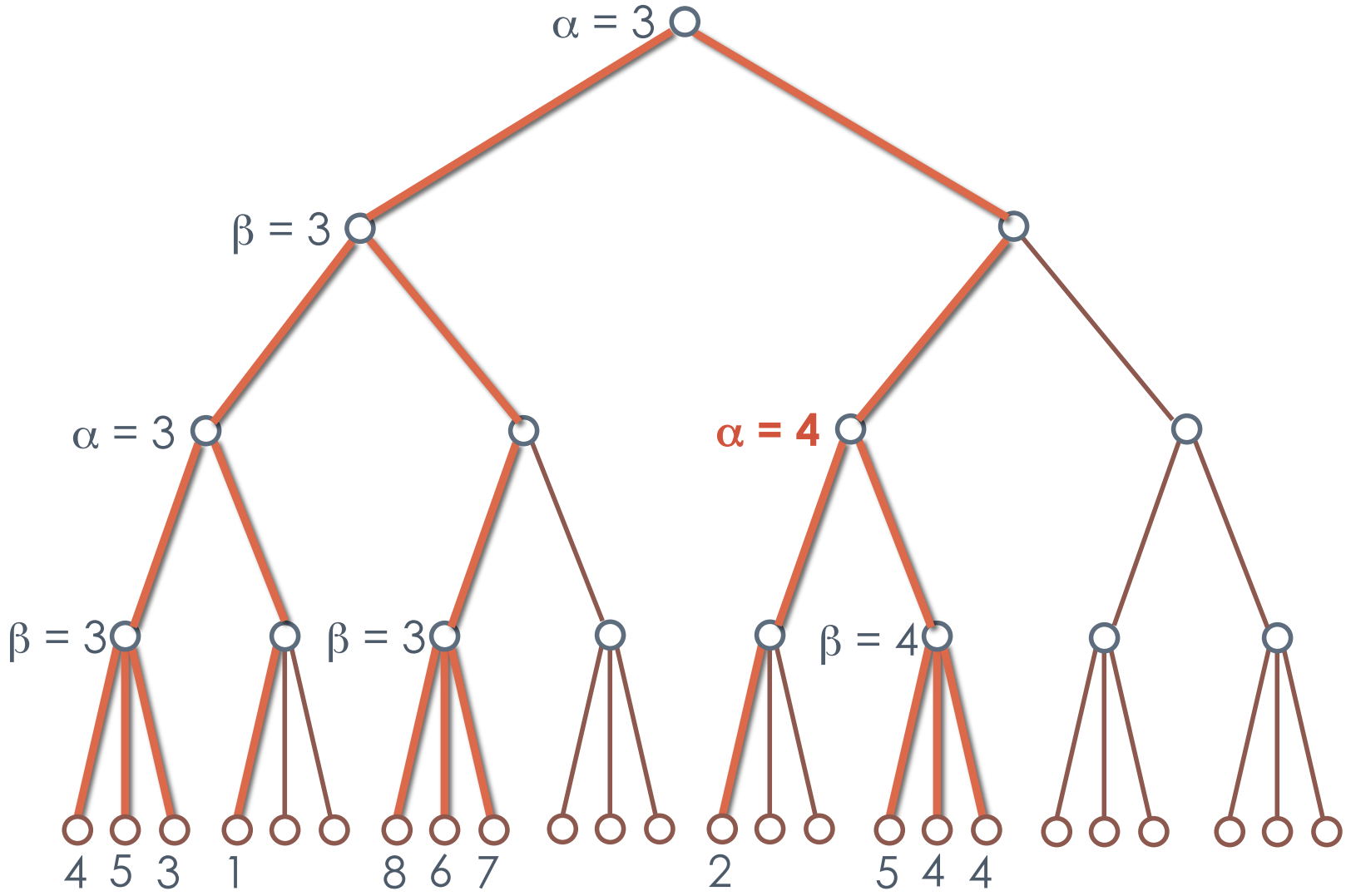
Algoritmo de corte α-β



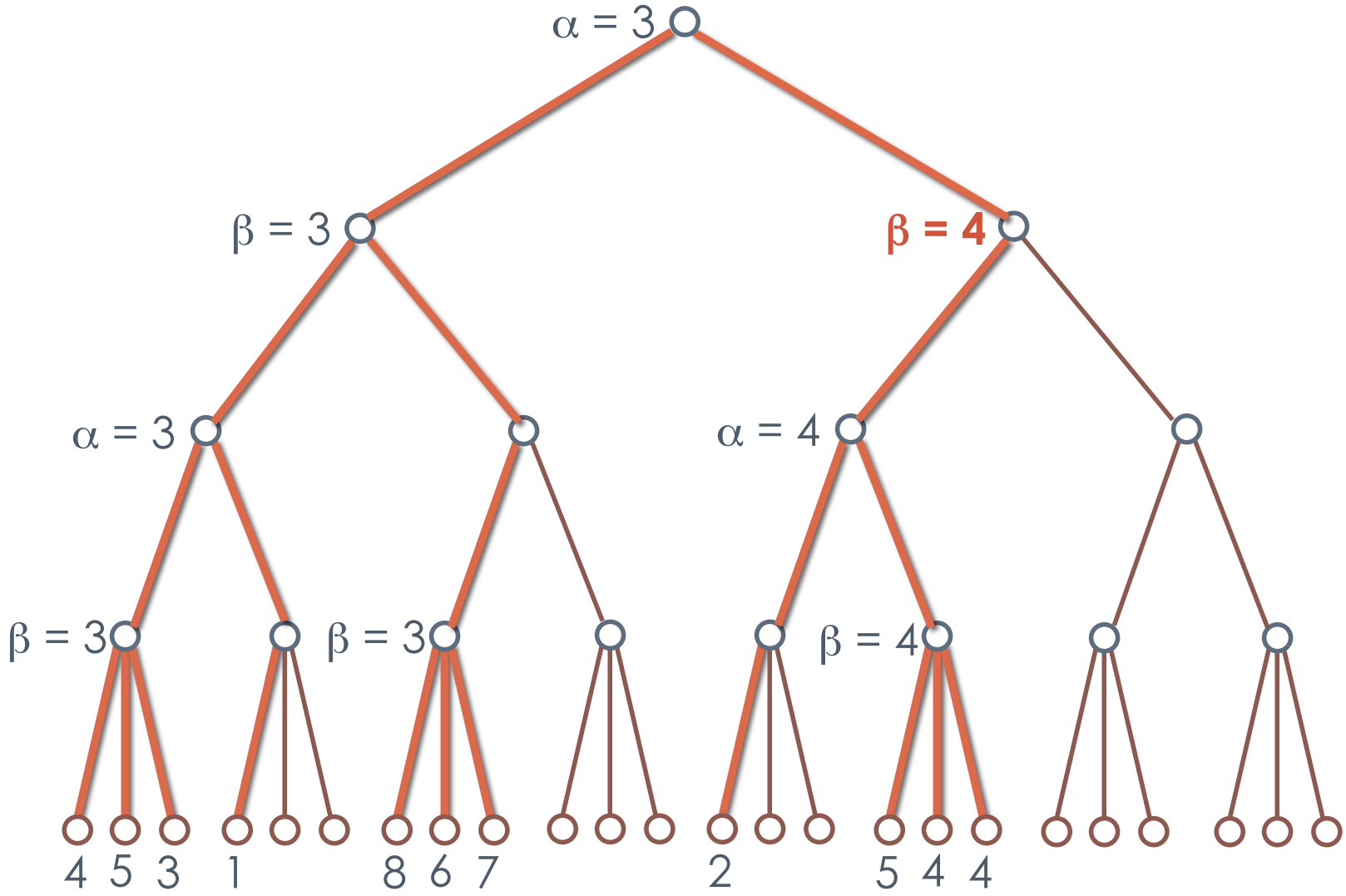
Algoritmo de corte α-β



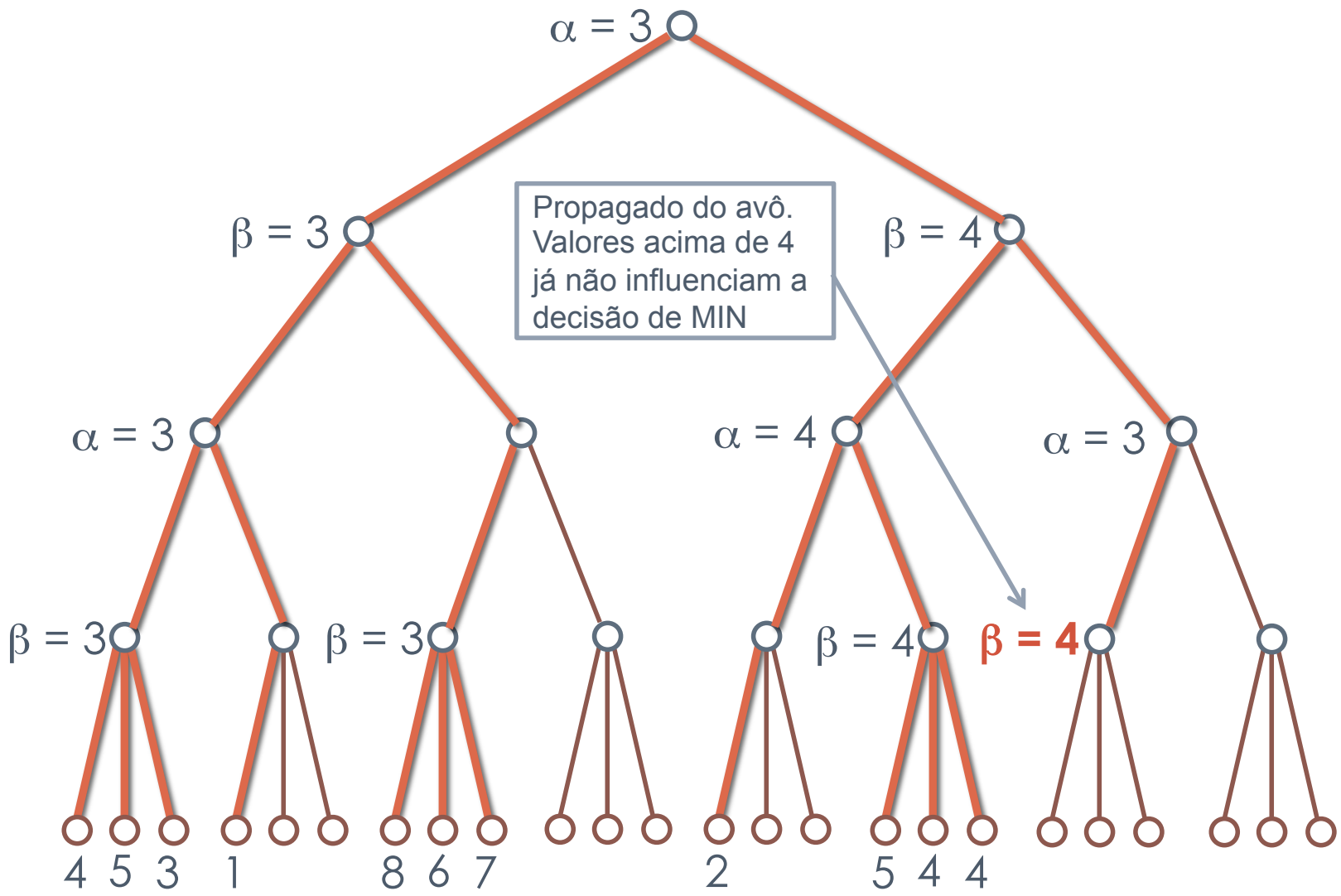
Algoritmo de corte α-β



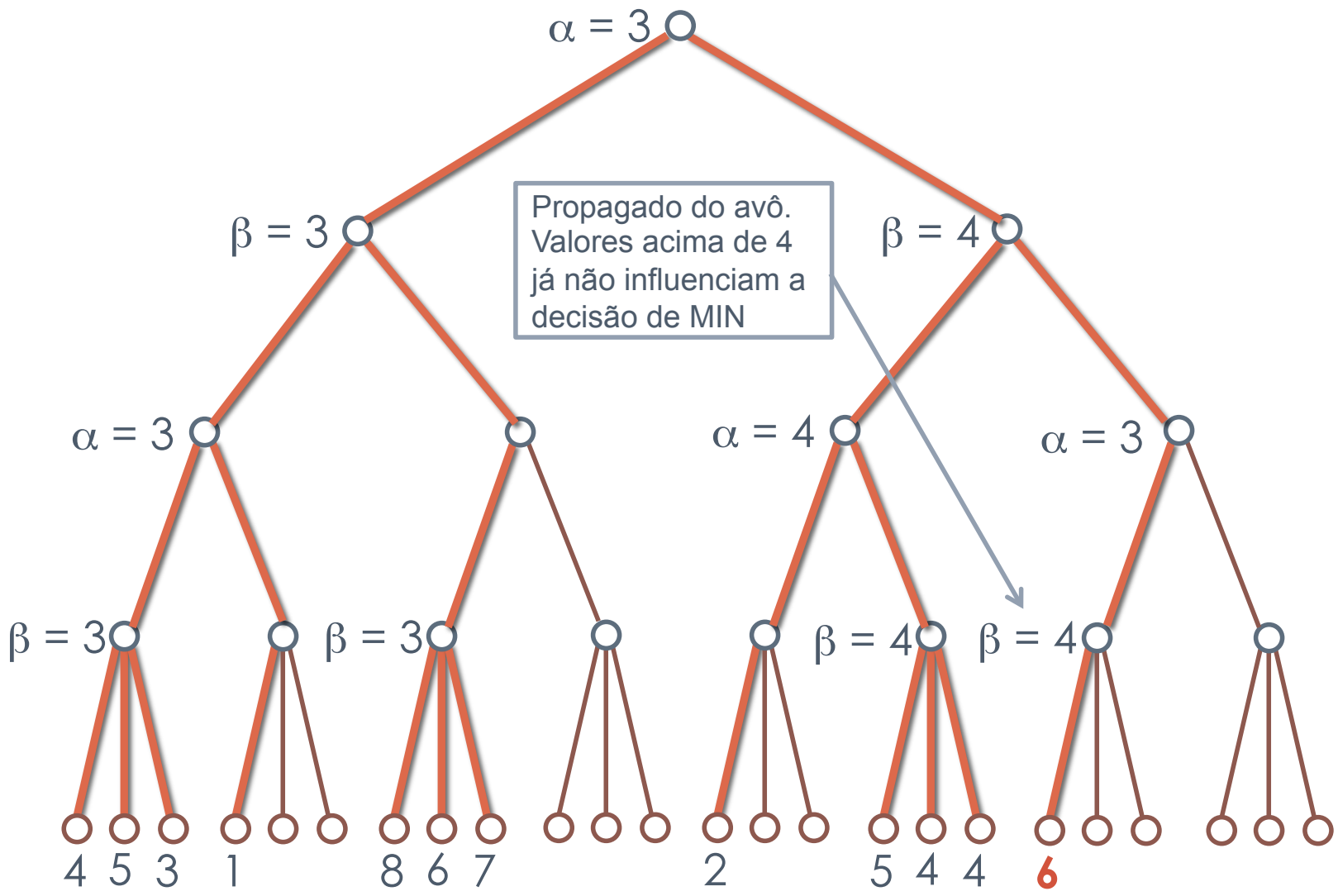
Algoritmo de corte α-β



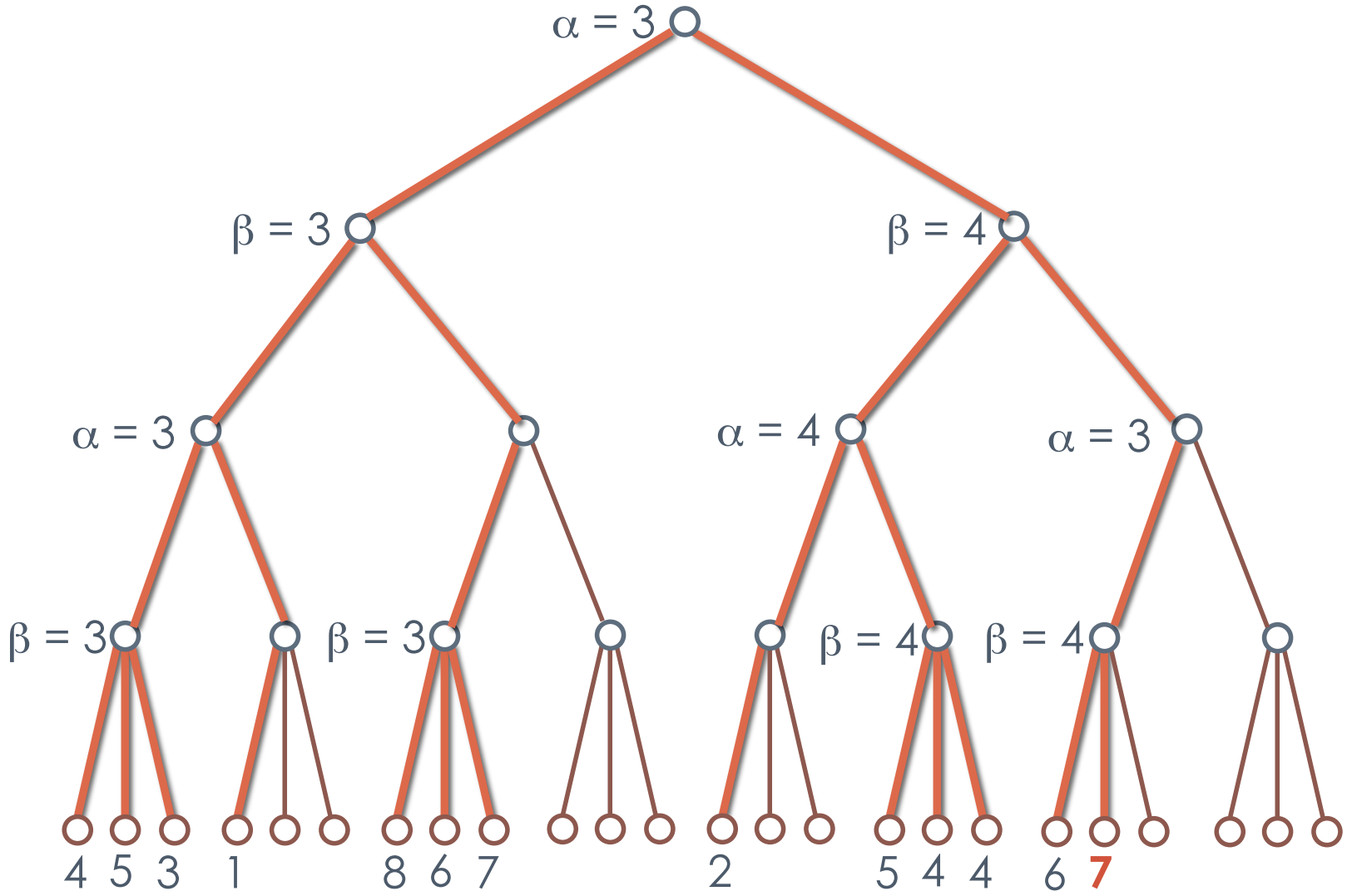
Algoritmo de corte α-β



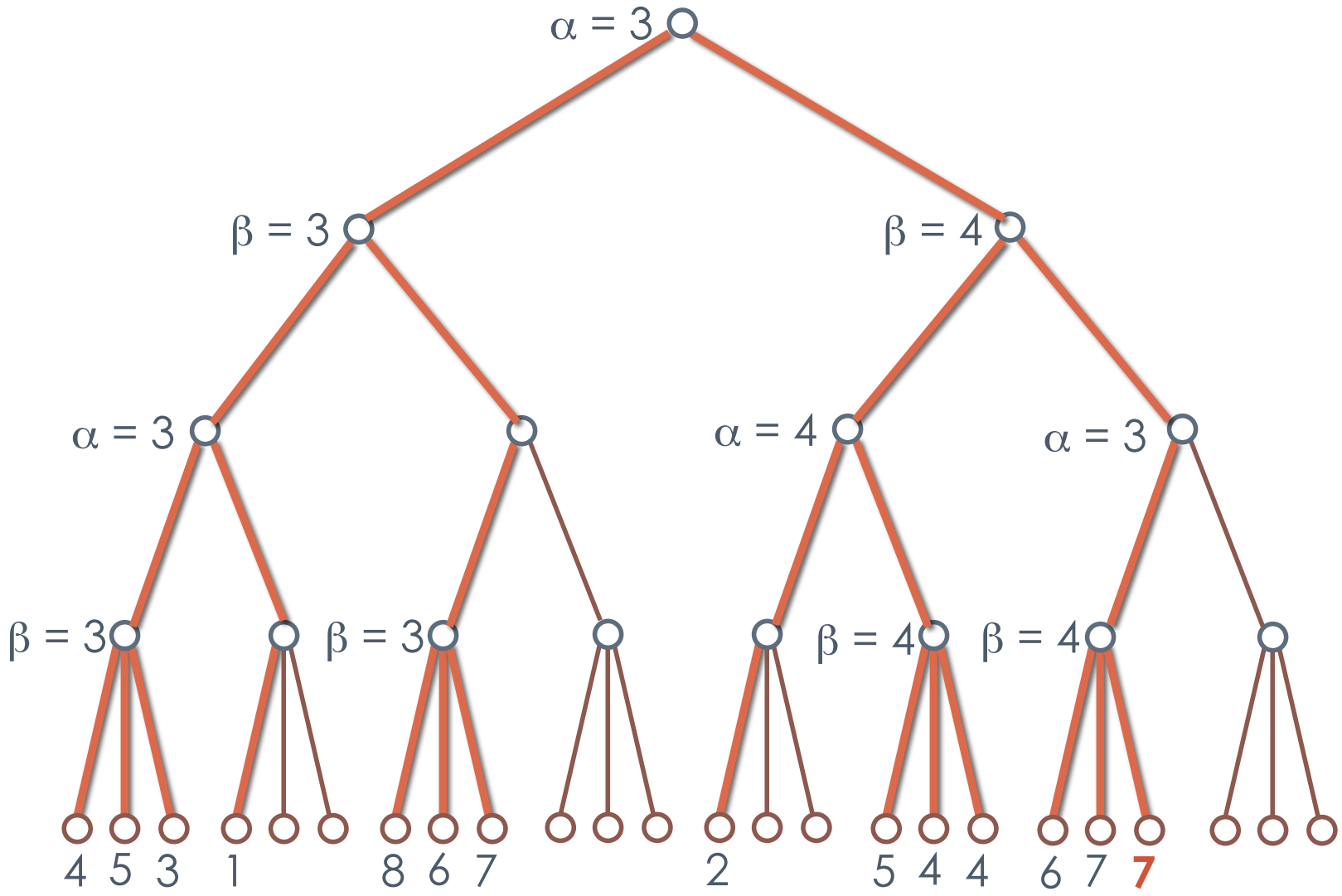
Algoritmo de corte α-β



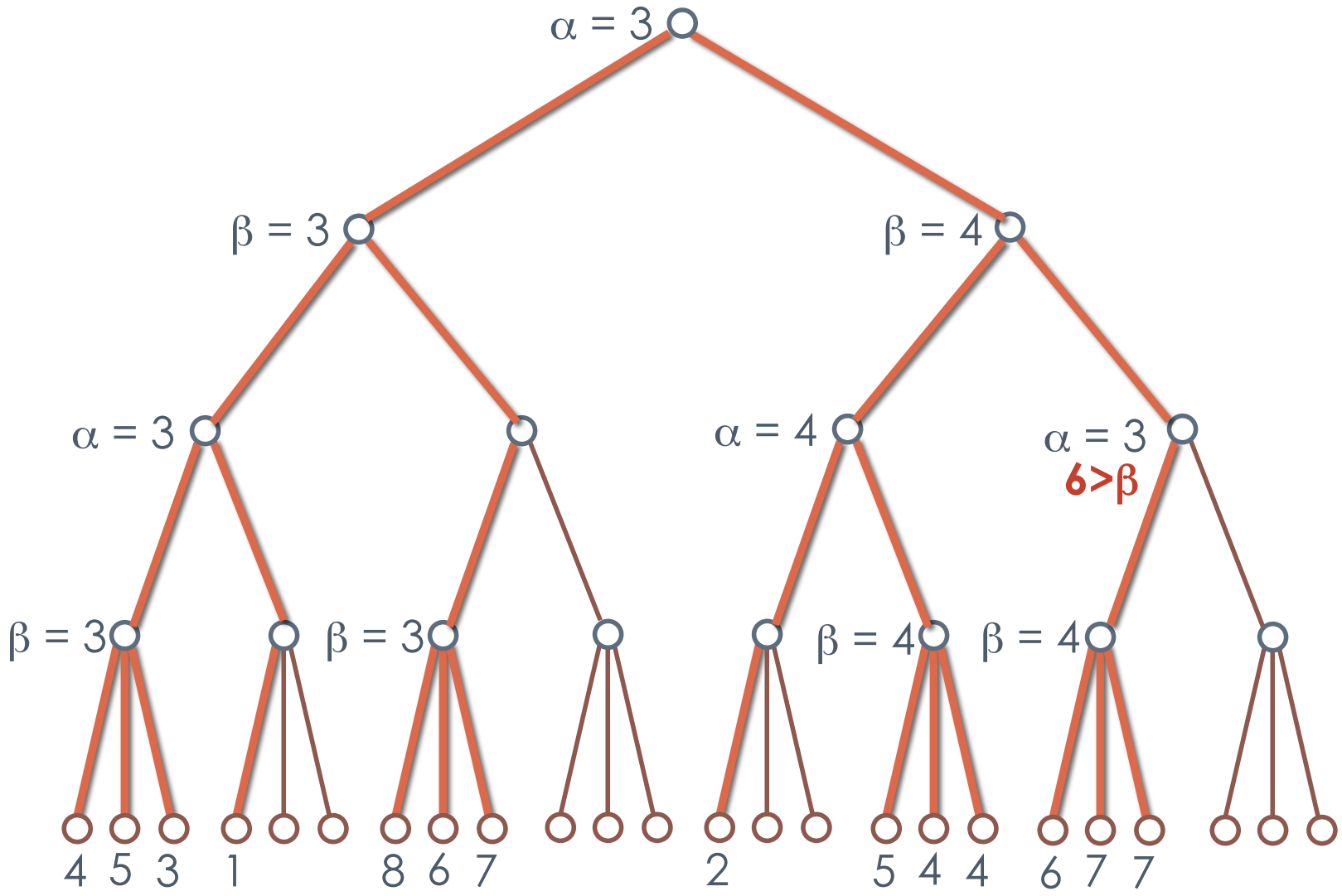
Algoritmo de corte α-β



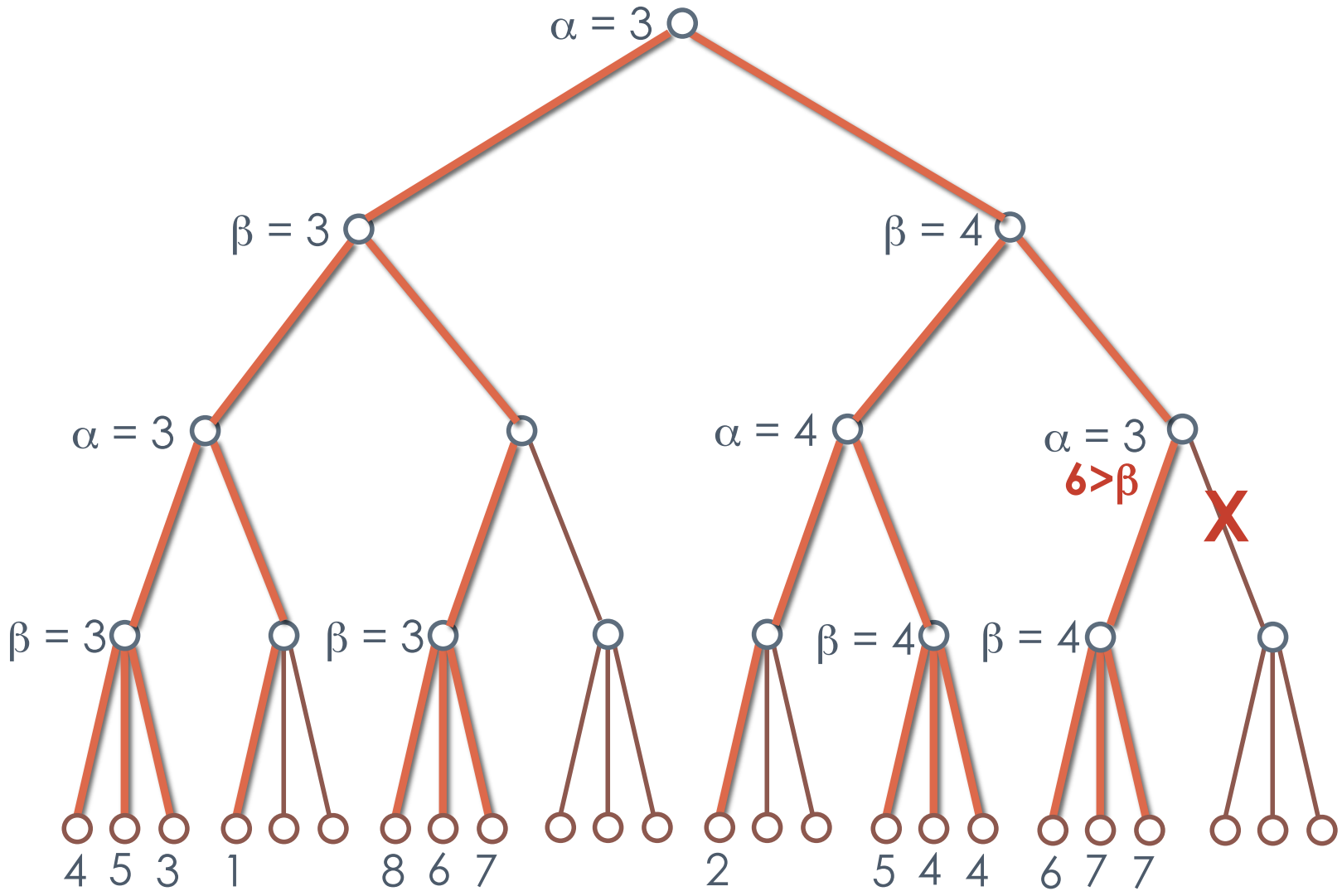
Algoritmo de corte α-β



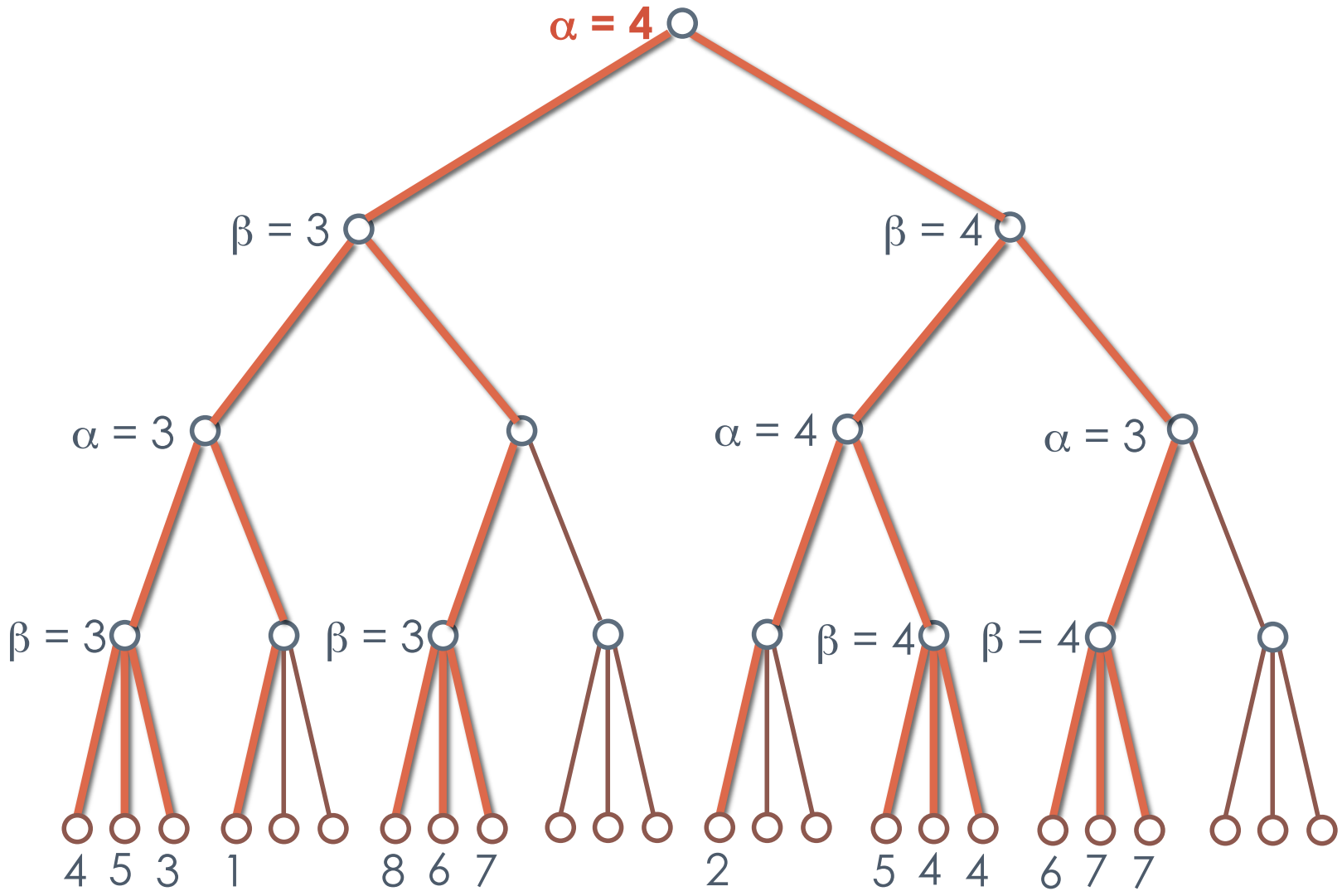
Algoritmo de corte α-β



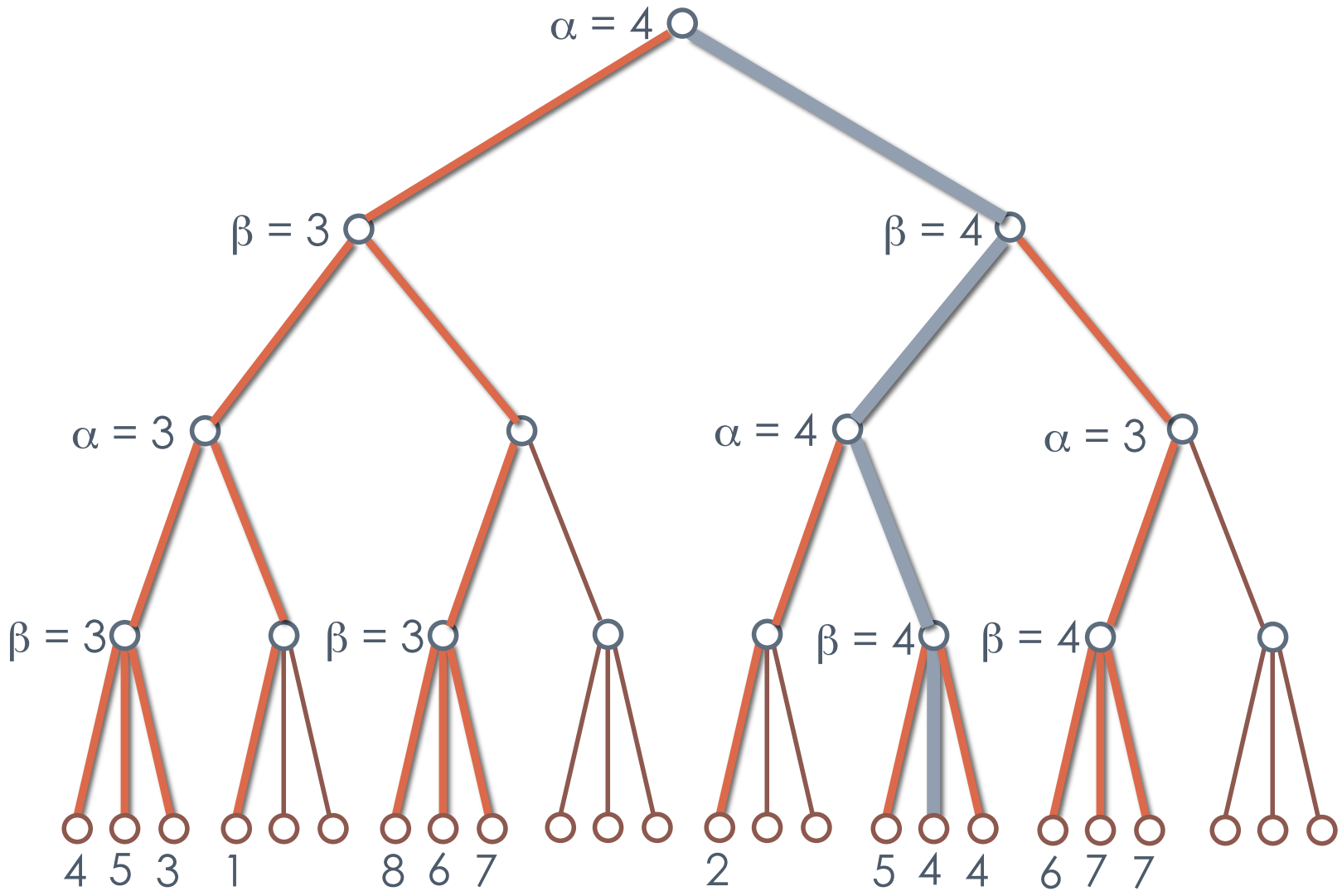
Algoritmo de corte α-β



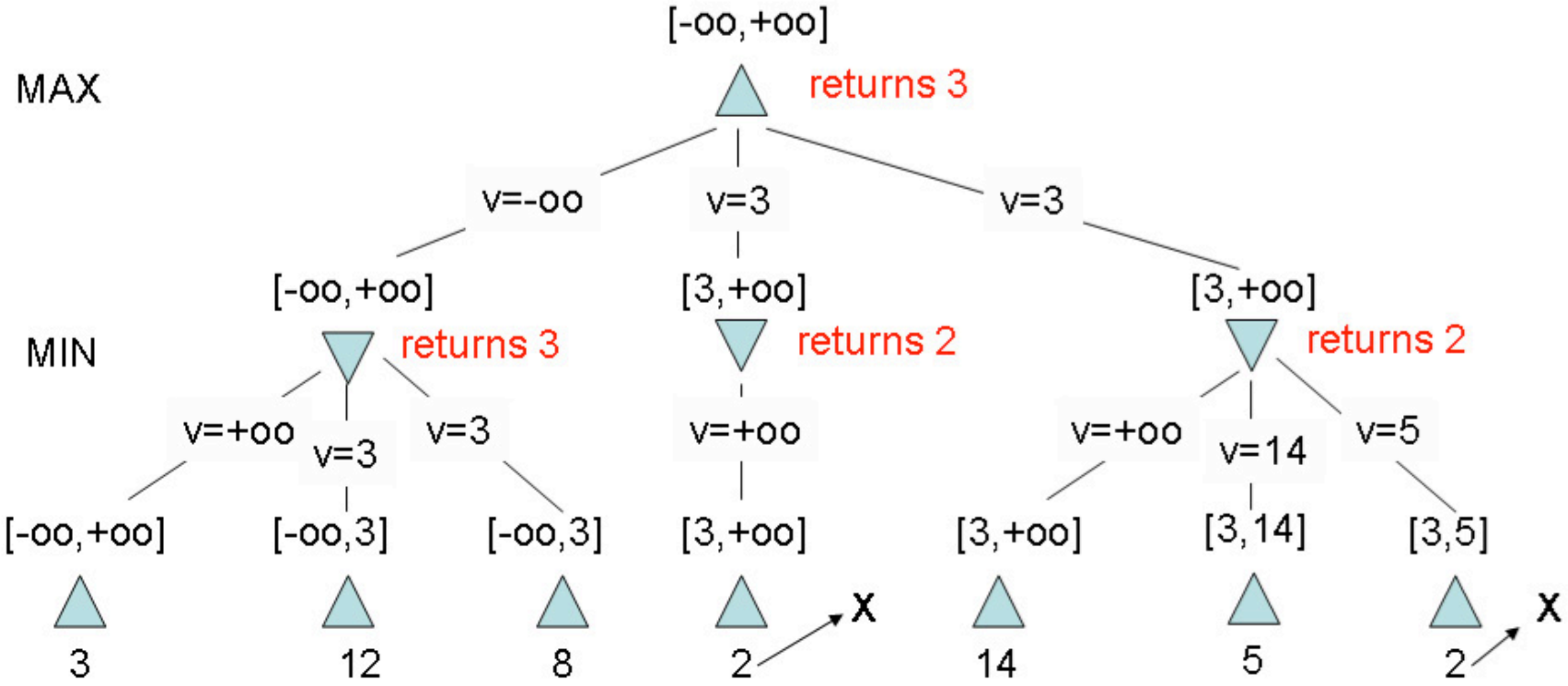
Algoritmo de corte α-β



Algoritmo de corte α-β



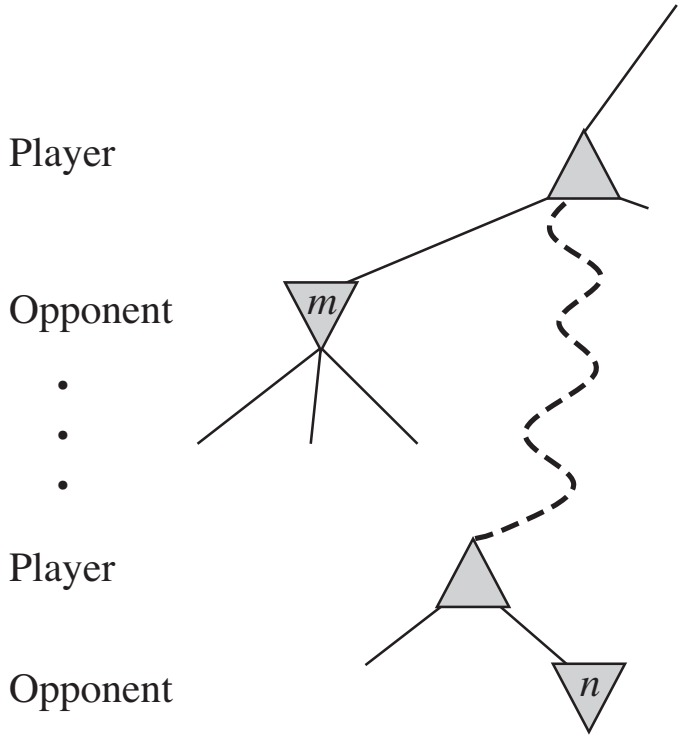
Execução do Algoritmo α-β



- Legenda:
- [α,β] Valor dos parâmetros na invocação
- v = u Valor da variável v antes da chamada recursiva
- **X** Corte após retorno da chamada recursiva
- returns u valor MINIMAX de saída da chamada

Corte α-β: o caso geral

- Considere-se um nó n algures na árvore
- Se um jogador tiver uma melhor escolha:
  - Num pai de n
  - Ou noutro ponto de escolha mais acima
- O nó n nunca será escolhido.
- Quando se sabe o suficiente sobre n, o seu ramo pode ser cortado.
- É válido tanto para MAX (m>n) como para MIN (m<n).



Propriedades do α-β

- Corte não afecta o resultado final
- Uma boa ordenação das jogadas melhora o efeito do corte
- Com “ordenação perfeita”, complexidade temporal = $O(b^{m/2})$
  - duplica profundidade da procura
- Um exemplo simples do valor do raciocínio sobre quais as computações relevantes (uma forma de metaraciocínio)

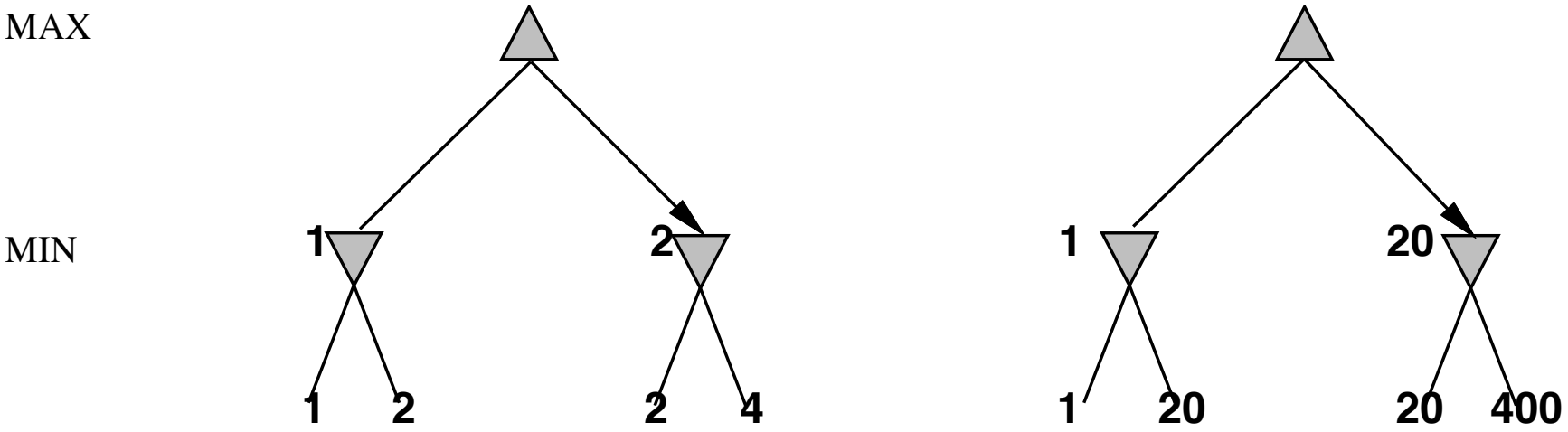
Recursos Limitados

- O Minimax com corte α-β ainda requer a avaliação de muitos nós terminais, que podem estar em profundidades elevadas.
- Pode ser impossível dentro de limites de tempo razoáveis.
- Solução:
  - Limitar a profundidade
    - Substituir **TERMINAL-TEST** por **CUTOFF-TEST**
  - Aplicar função heurística
    - Substituir **UTILITY** por **EVAL**

Funções de avaliação (EVAL)

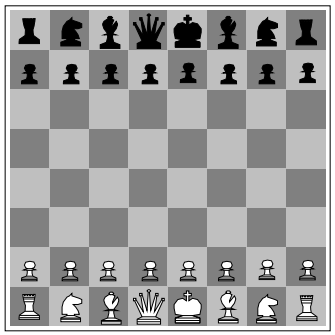
- Produzem uma estimativa da utilidade esperada de um jogo a partir de uma dada posição
- Performance do algoritmo depende da qualidade de EVAL
- Requesitos de EVAL:
  - Deve preservar a ordem dos nós terminais estabelecida por UTILITY.
  - Deve ser calculada de forma eficiente
  - Para nós não terminais, deve ser fortemente correlacionada com a real chance de ganho.

Valores exactos não são importantes

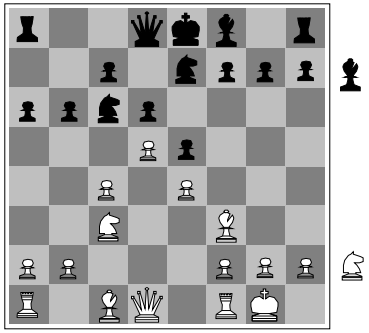


- O comportamento é preservado com qualquer transformação monótona da utilidade.
- Apenas a ordem interessa:
  - recompensa em jogos deterministas comporta-se como uma função de utilidade ordinal.

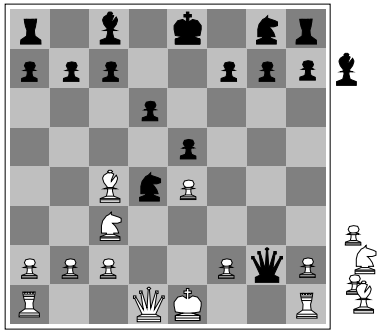
Funções de avaliação



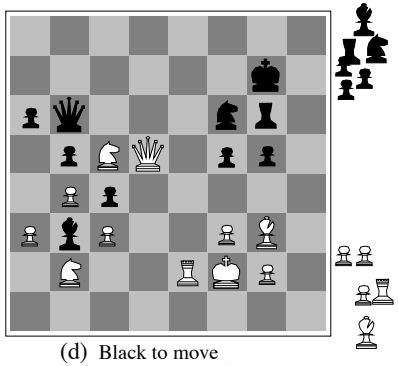
(a) White to move
Fairly even



(b) Black to move
White slightly better



(c) White to move
Black winning



(d) Black to move
White about to lose

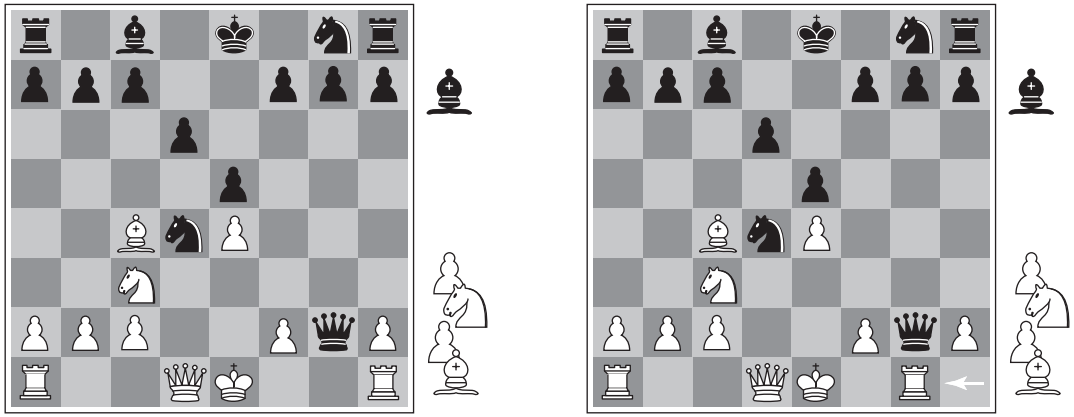
- *No Xadrez, habitualmente soma pesada linear de características*

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

  - fx(s) = diferença do número de raínhas
  - fy(s) = controlo do centro do tabuleiro
  - Etc…
- Por vezes é necessário combinações não lineares. E.g.:
  - Um par de bispos vale mais do que o dobro do valor de um bispo.
  - Um bispo vale mais no final do jogo.

Corte da Procura

- Como cortar um nó?
  - Limitar a profundidade a um número fixo (naïve).
  - Aprofundamento progressivo.
  - Problema dos estados não quiscentes (estáveis)

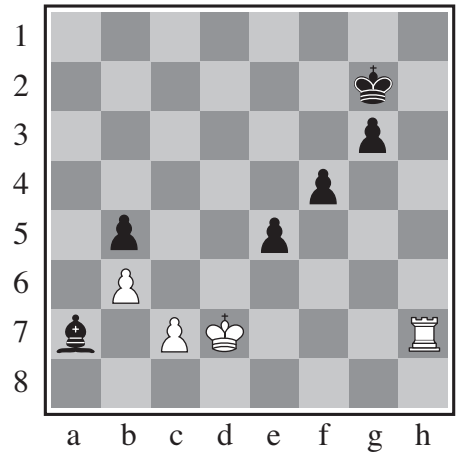


(a) White to move (b) White to move

  - Em (a), as negras têm vantagem. Em (b), apesar de uma função de avaliação baseada no valor das peças retornar o mesmo valor, as brancas têm uma grande vantagem (captura da rainha negra)
  - Solução: apenas realizar cortes em nós em que o valor da função de avaliação estabilizou (sem alterações substanciais nas próximas jogadas)

Corte da Procura

- Mais difícil de resolver é o problema do horizonte (morte adiada), que acontece quando uma jogada terrível pode ter os seus efeitos adiados de maneira a ficarem invisíveis.



- O bispo negro está condenado (torre: h8-a8-a7).
- No entanto, as negras podem adiar a captura fazendo xeques sucessivos com os peões.
- Assim, o jogador das negras adiará a captura do bispo para além do horizonte, julgando que os sucessivos xeques pelos peões conseguem salvar o bispo, o que não é verdade.

Jogos deterministas na prática

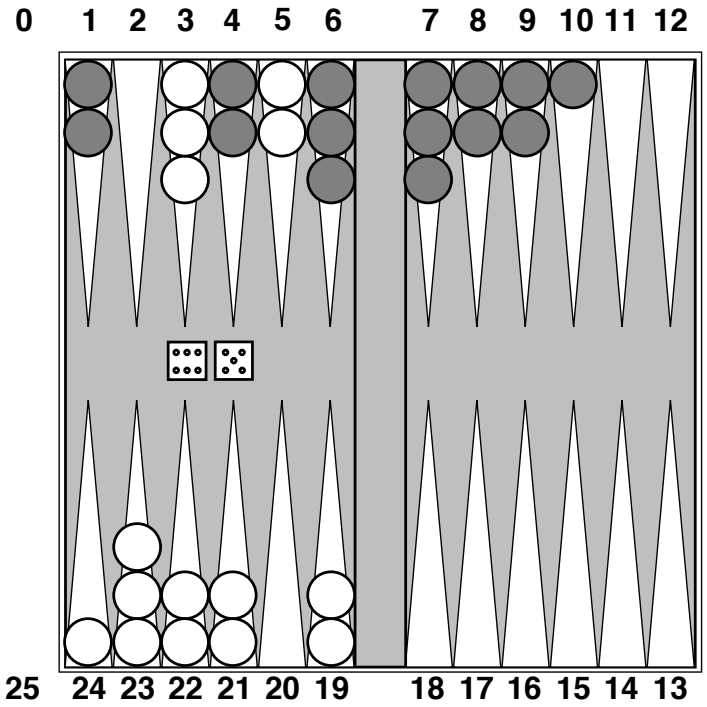
- **Damas**: Chinook terminou com o reinado de 40 anos do campeão mundial Marion Tinsley em 1994. Utilizou uma base de dados de final de jogo definindo a estratégia perfeita para todas as posições com 8 ou menos peças no tabuleiro, num total de 443,748,401,247 posições.

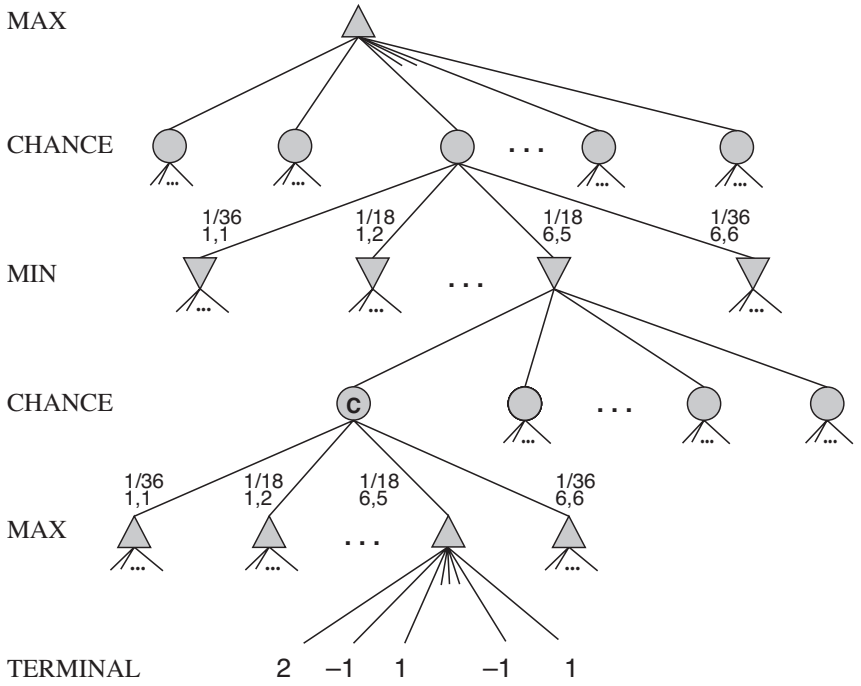
- **Xadrez**: Deep Blue derrotou o campeão mundial humano Gary Kasparov num encontro a 6 partidas em 1997. Deep Blue procura 200 milhões de posições por segundo, utiliza avaliação muito sofisticada, e recorres a métodos para estender algumas linhas de procura até 40 jogadas.

- **Othello**: campeões humanos recusam-se a competir contra computadores, que são demasiado bons.

- **Go**: até há pouco tempo, os campeões humanos recusavam-se a competir contra computadores, por serem péssimos jogadores. Recentemente, já em 2016, o Alpha Go derrotou Lee Sedol – um profissional de 9-dan – num encontro a 5 partidas. O AlphaGo usa um misto de pesquisa Monte Carlo e redes neuronais para avaliar posições e estratégias.

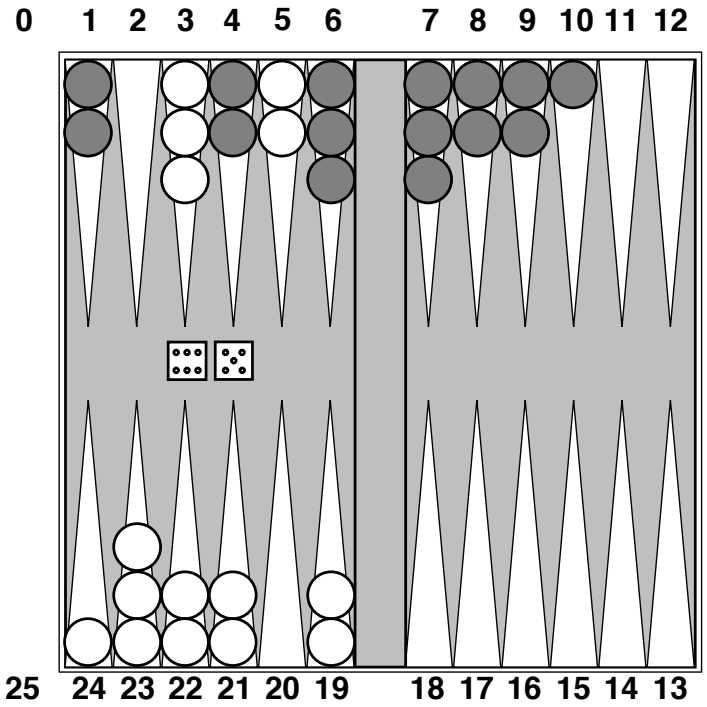
Jogos estocásticos

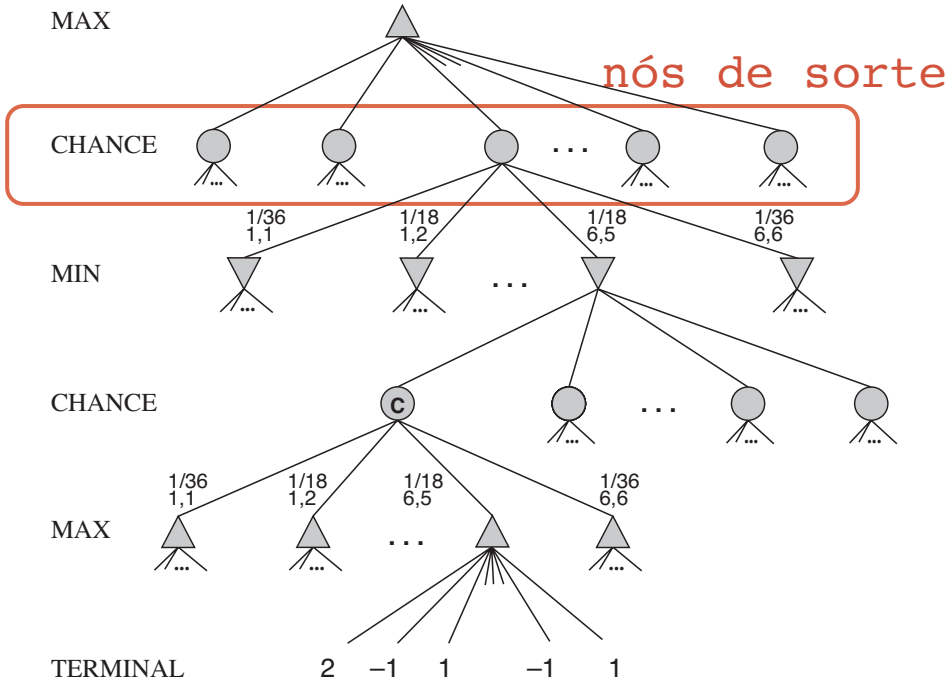




- Jogadas possíveis:
  - (5-10,5-11), (5-11,19-24),(5-10,10-16) e (5-11,11-16)

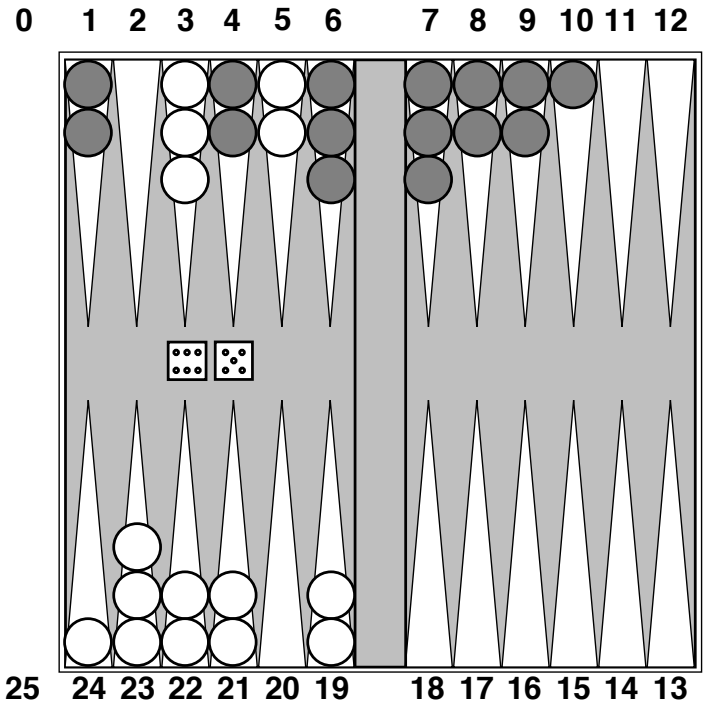
Jogos estocásticos

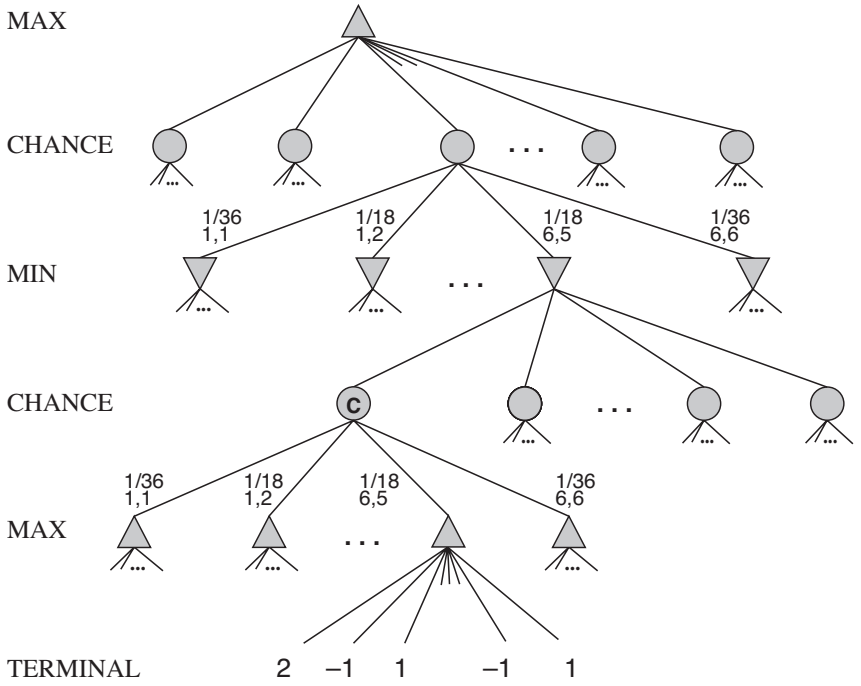




- Jogadas possíveis:
  - (5-10,5-11), (5-11,19-24),(5-10,10-16) e (5-11,11-16)
- [1,1],…, [6,6] com 1/36, todos os restantes com 1/18.

Jogos estocásticos





- [1,1],…, [6,6] com 1/36, todos os restantes com 1/18.
- Não é possível calcular um valor preciso de minimax, apenas um valor esperado

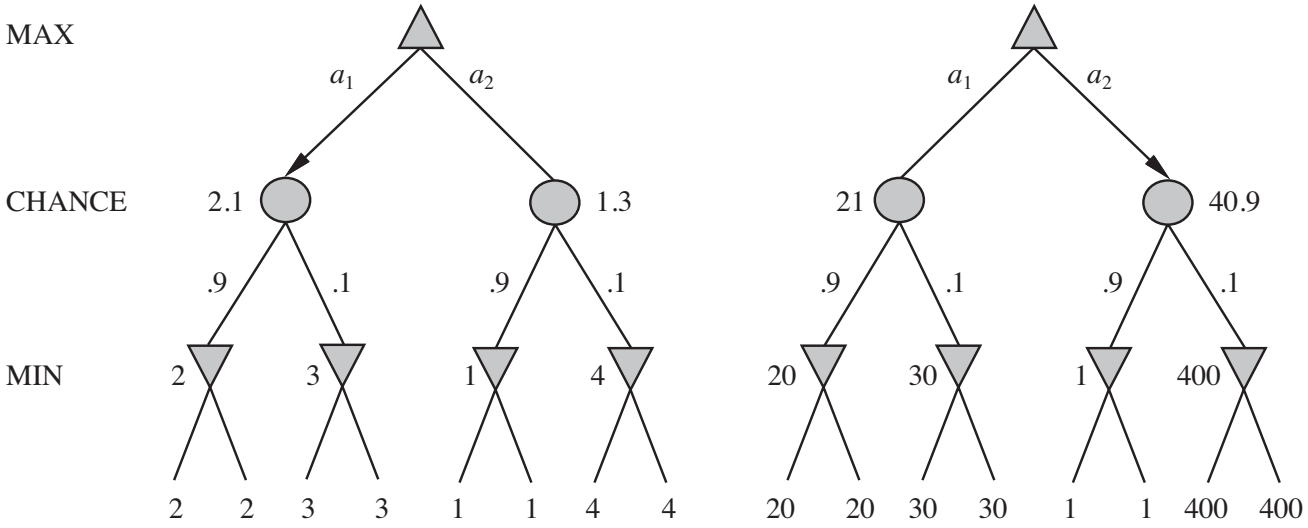
Valor minimax esperado

$$\text{EXPECTEDMINIMAX}(n) = \begin{cases} \text{UTILITY}(n) & \text{Se } n \text{ é nó terminal} \\ \max_{s \in successors(n)} \text{EXPECTEDMINIMAX}(s) & \text{Se } n \text{ é nó de MAX} \\ \min_{s \in successors(n)} \text{EXPECTEDMINIMAX}(s) & \text{Se } n \text{ é nó de MIN} \\ \sum_{s \in successors(n)} P(s).\text{EXPECTEDMINIMAX}(s) & \text{Se } n \text{ é nó CHANCE} \end{cases}$$

- Como MINIMAX, excepto que se tem em conta os nós de sorte.
- EXPECTEDMINIMAX fornece estratégia óptima

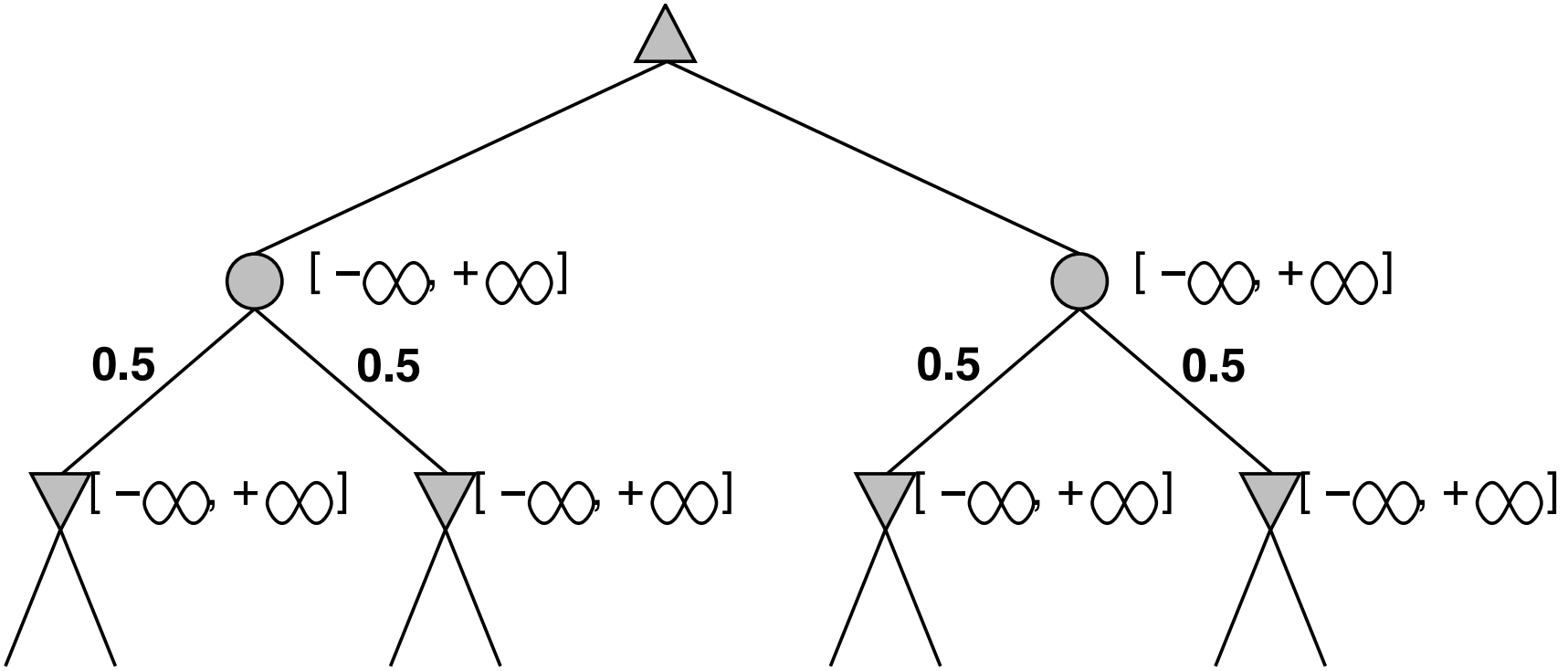
Função de avaliação de posição em jogos estocásticos

- Ao contrário dos jogos deterministas, os valores concretos são importantes

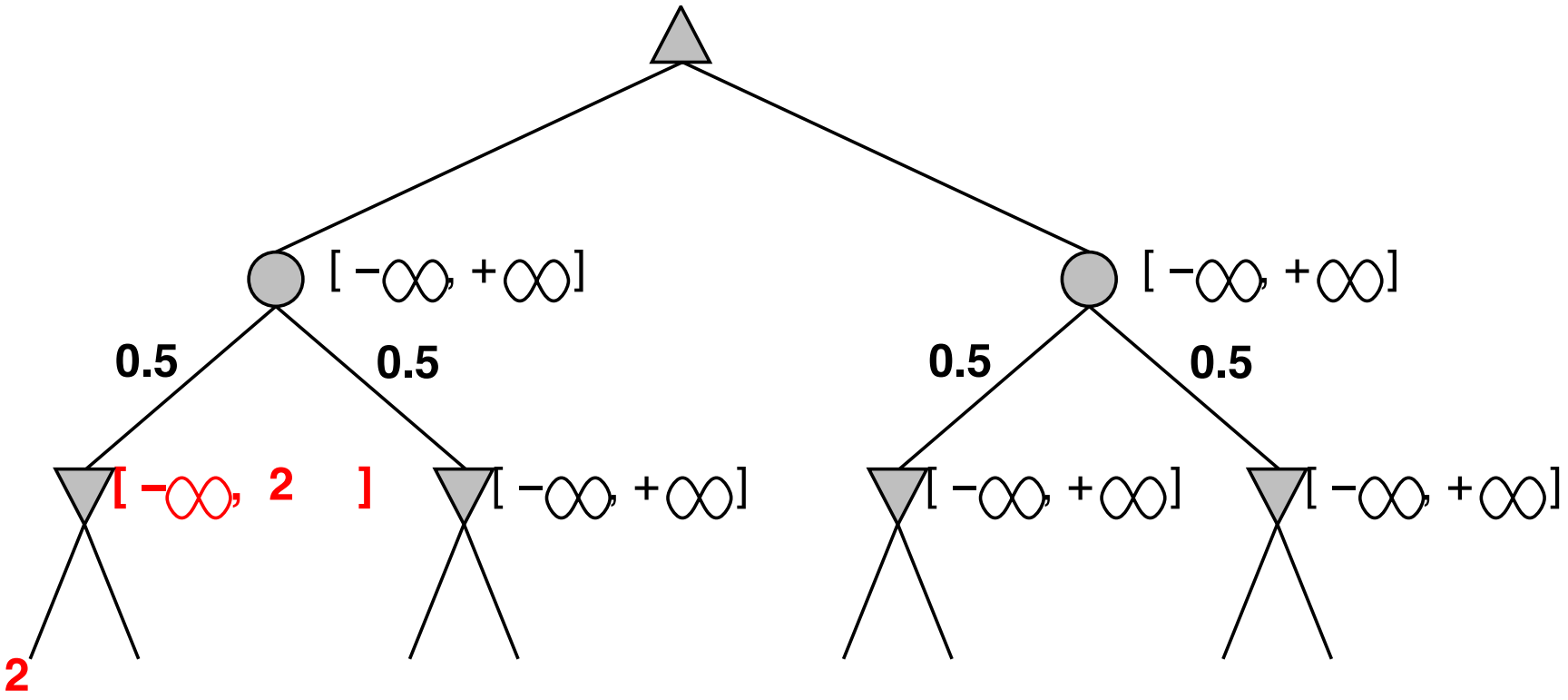


- Esquerda: a1 ganha
- Direita: a2 ganha
- O comportamento é preservado apenas com transformação linear positiva de EVAL.
- Logo, EVAL deve ser proporcional à utilidade esperada.

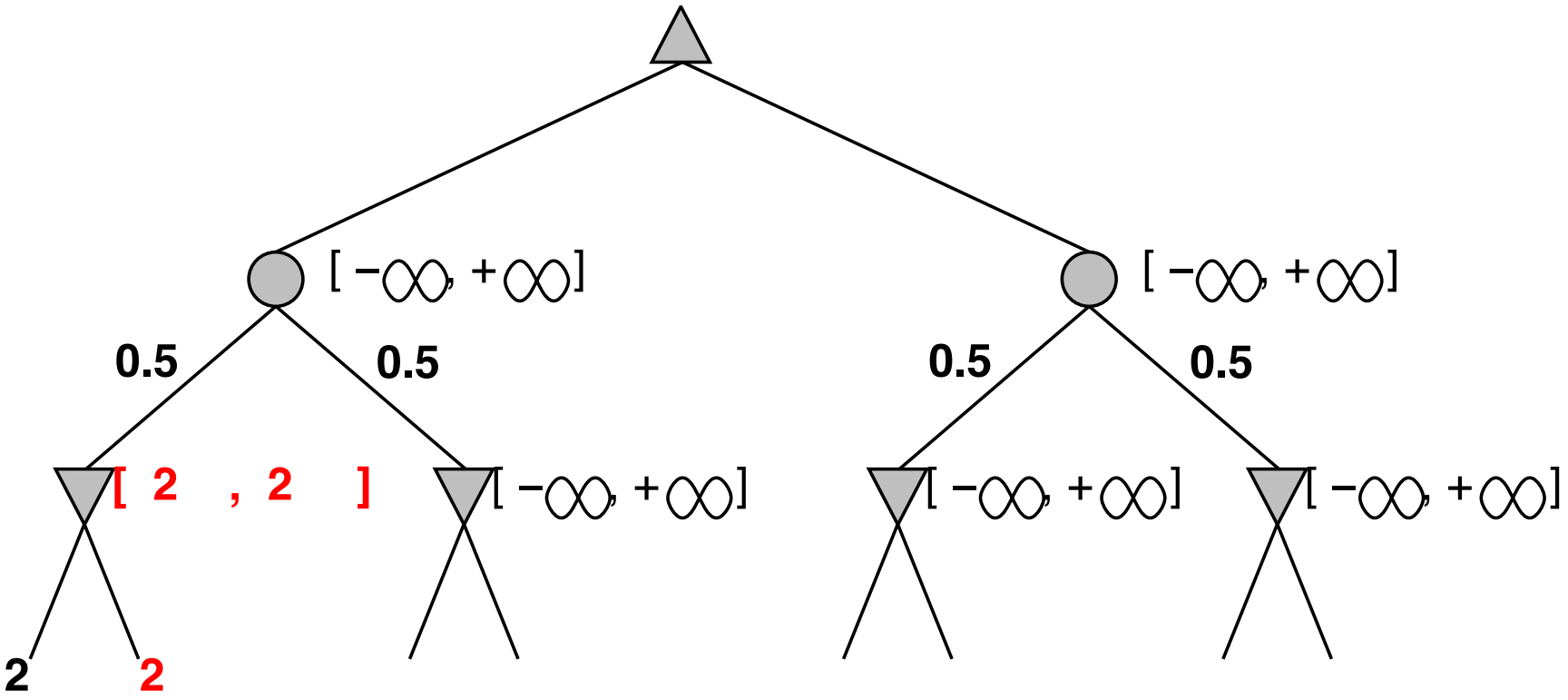
Exemplo de corte α-β em jogos estocásticos



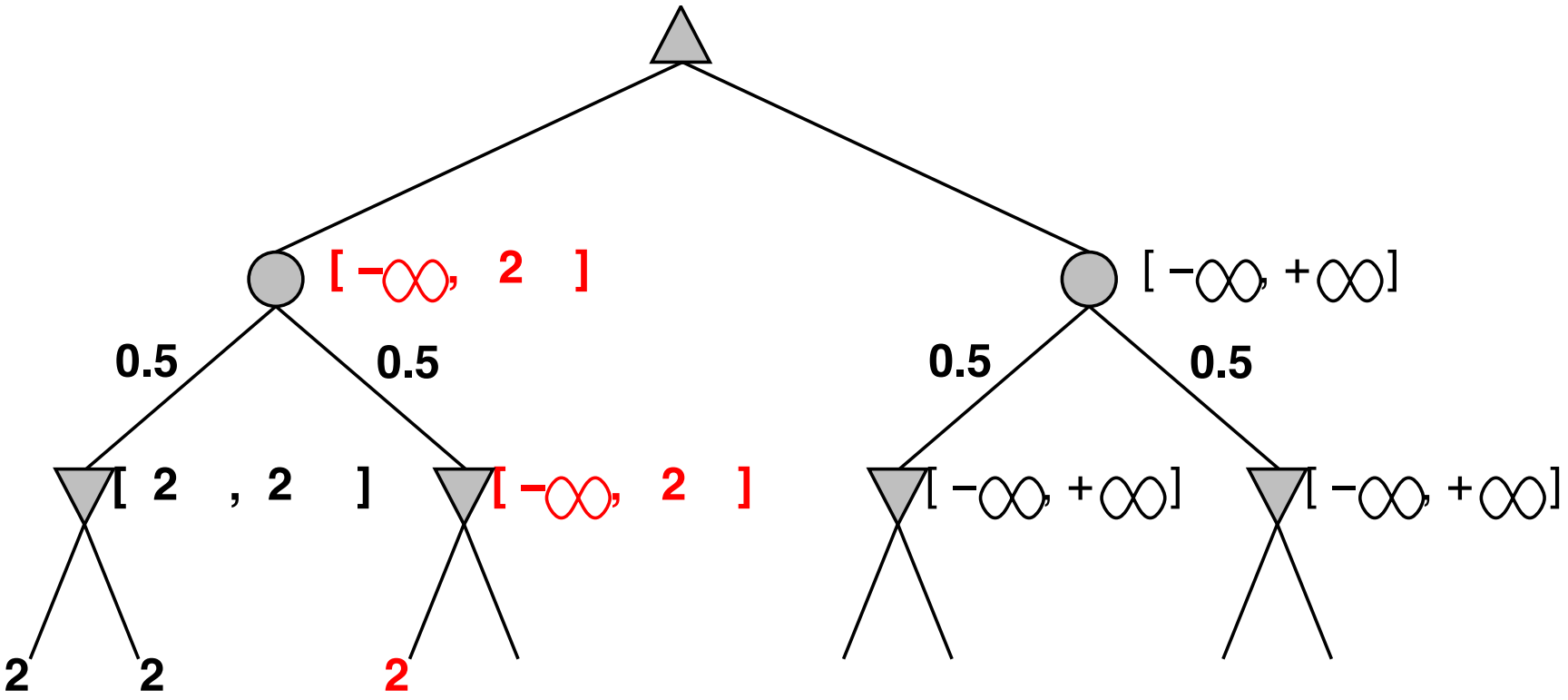
Exemplo de corte α-β em jogos estocásticos



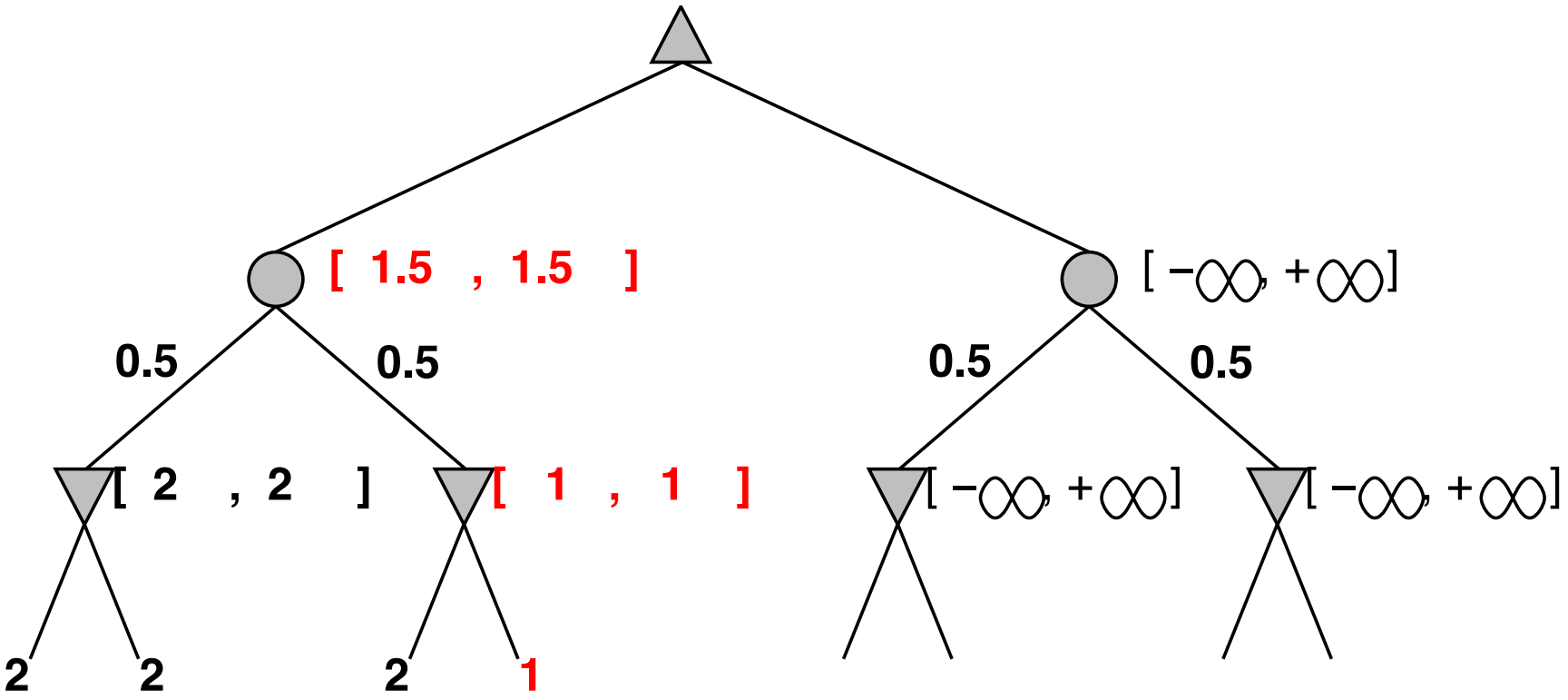
Exemplo de corte α-β em jogos estocásticos



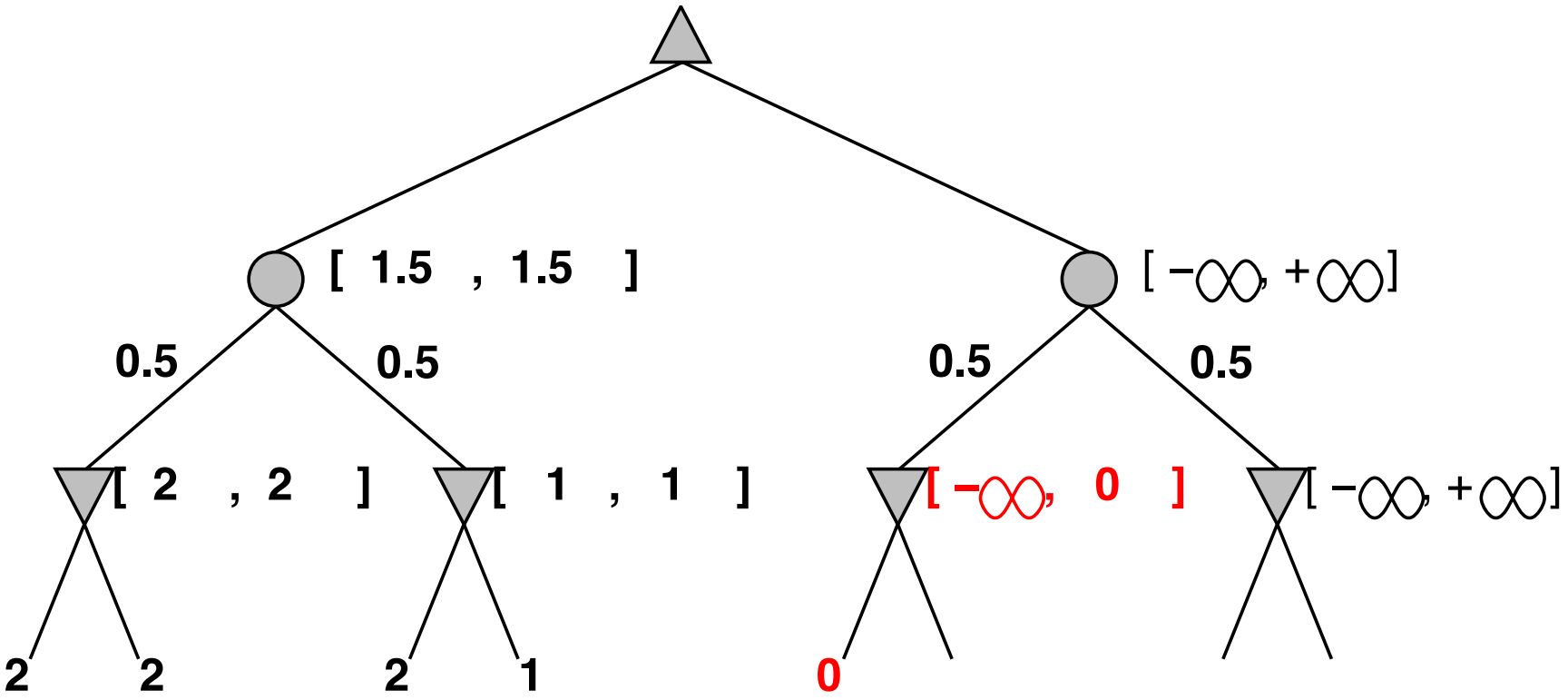
Exemplo de corte α-β em jogos estocásticos



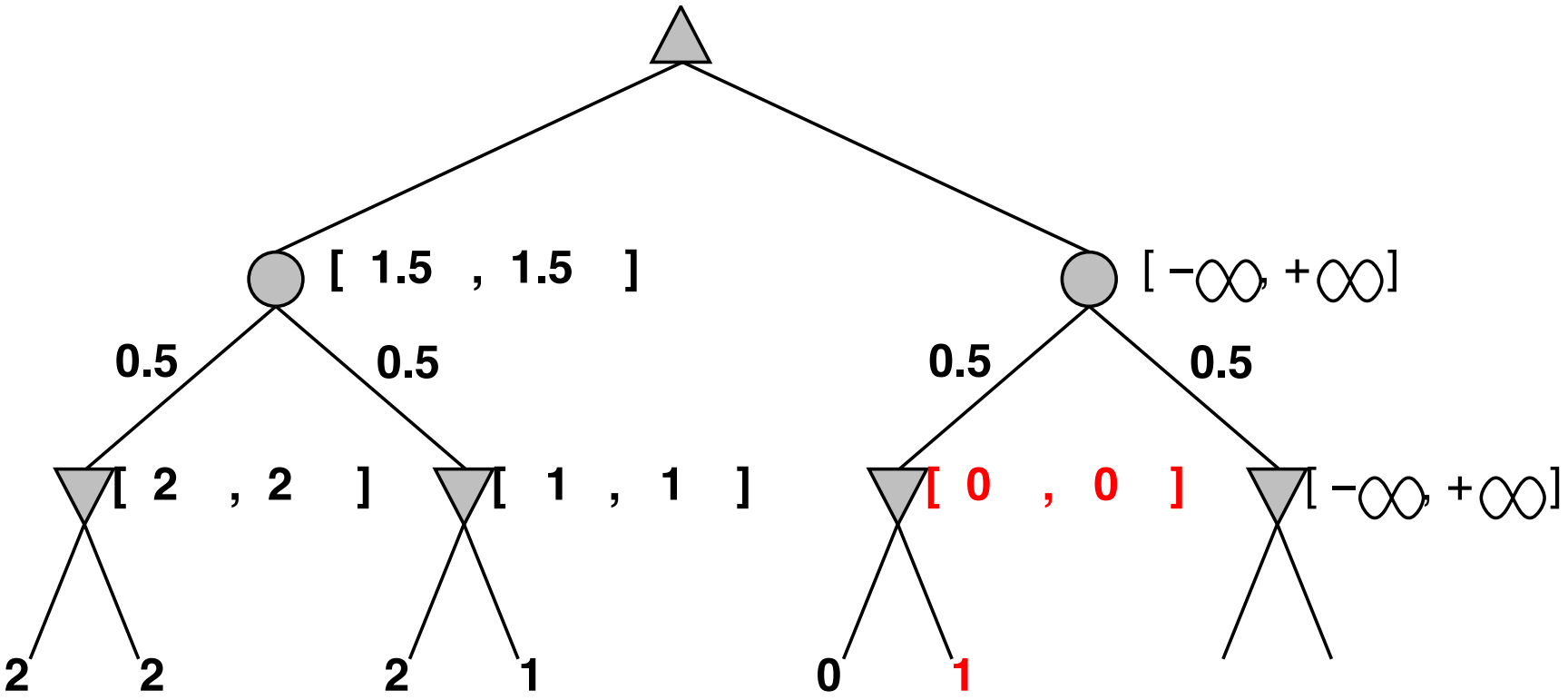
Exemplo de corte α-β em jogos estocásticos



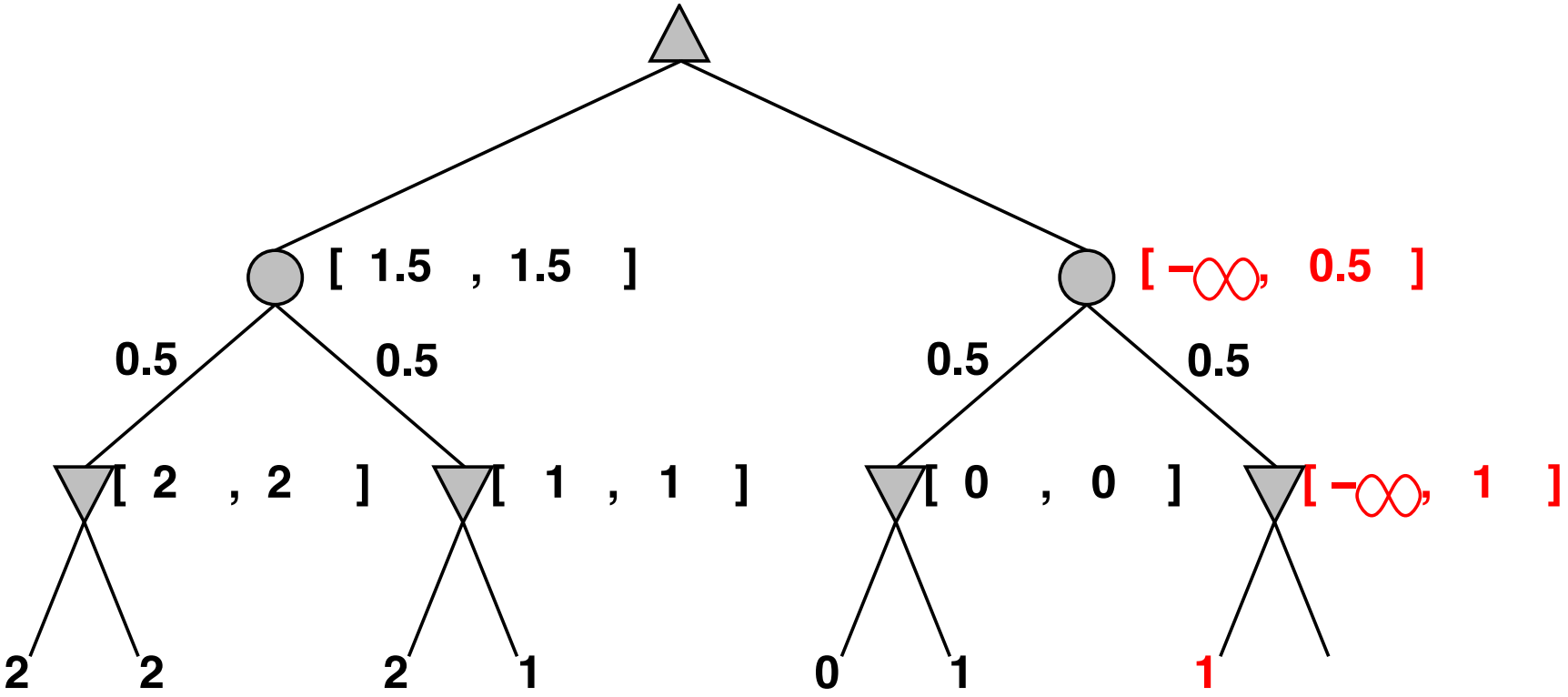
Exemplo de corte α-β em jogos estocásticos



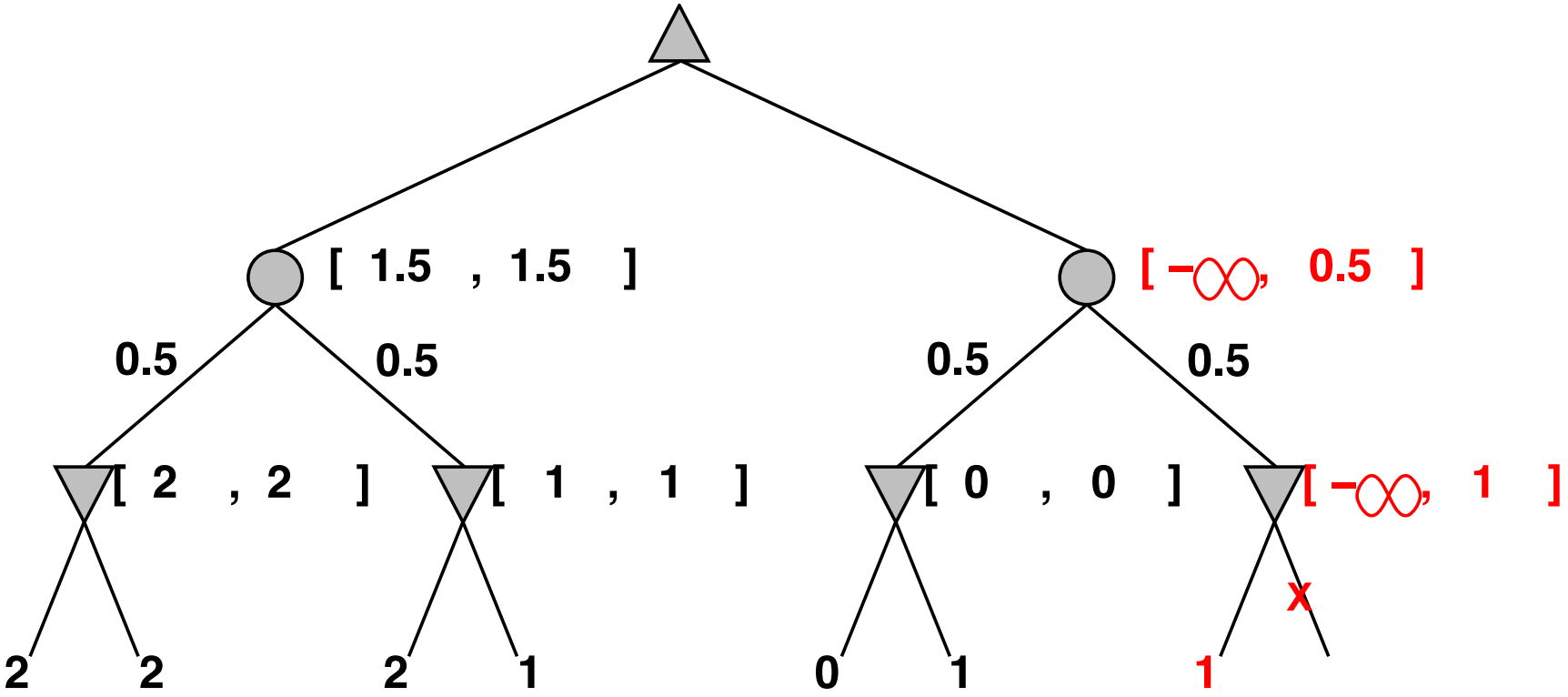
Exemplo de corte α-β em jogos estocásticos



Exemplo de corte α-β em jogos estocásticos

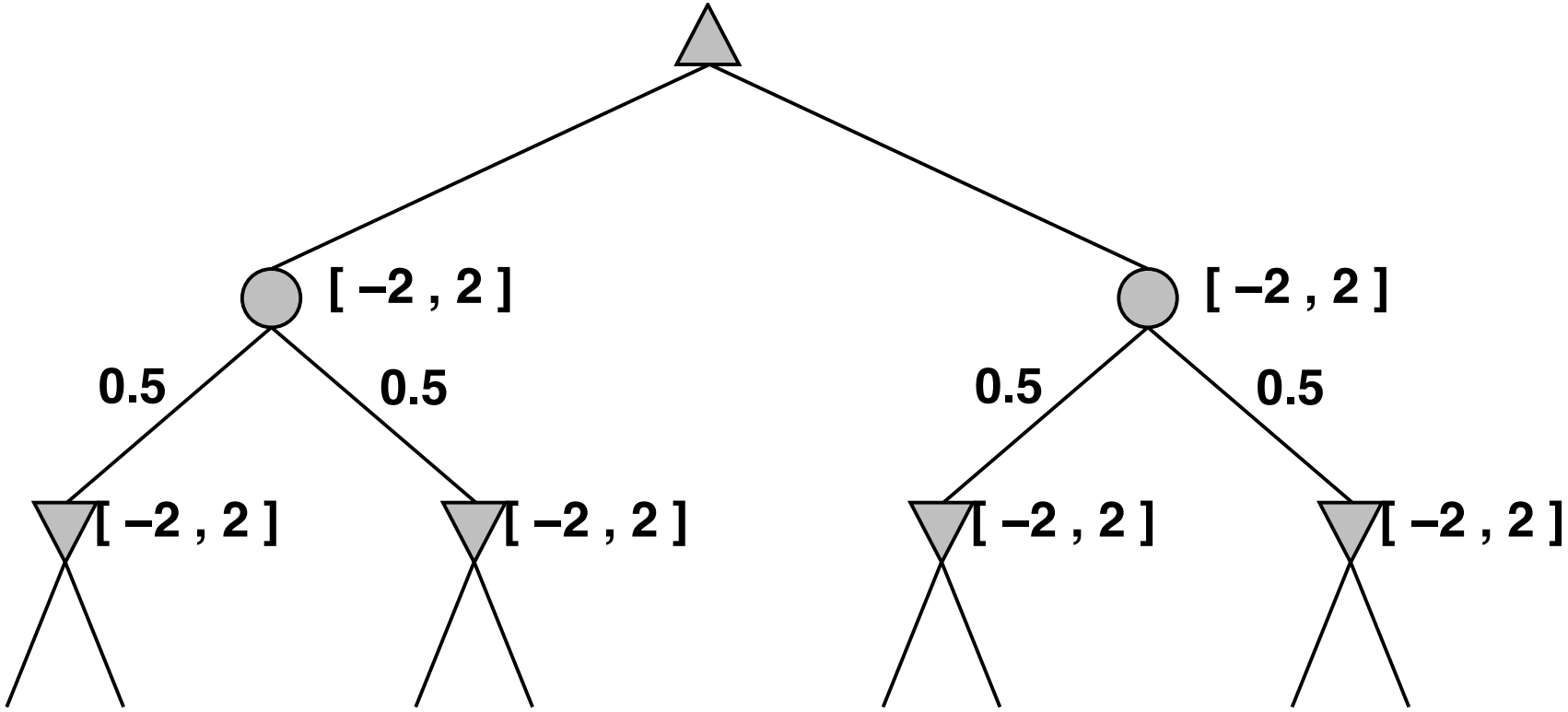


Exemplo de corte α-β em jogos estocásticos



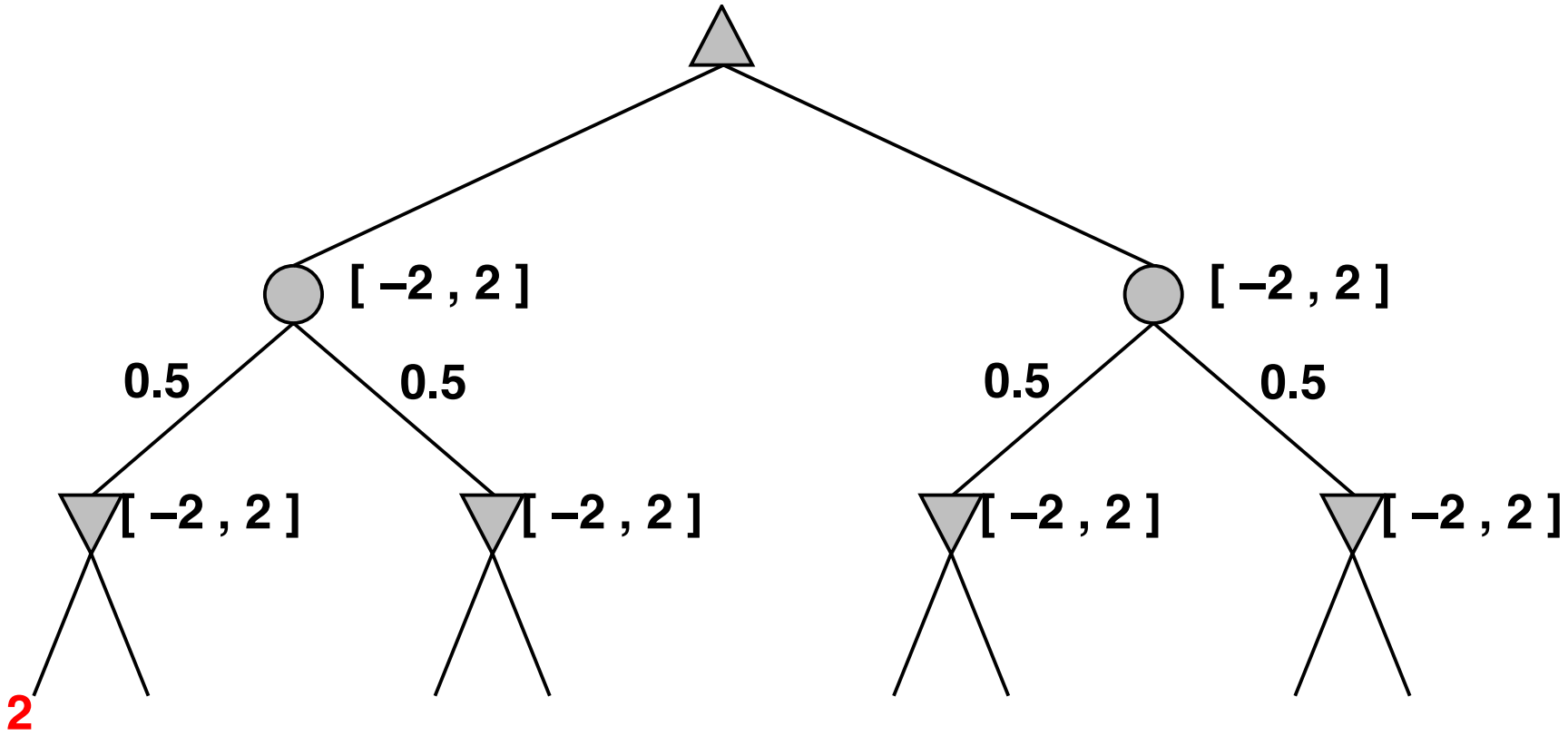
Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados



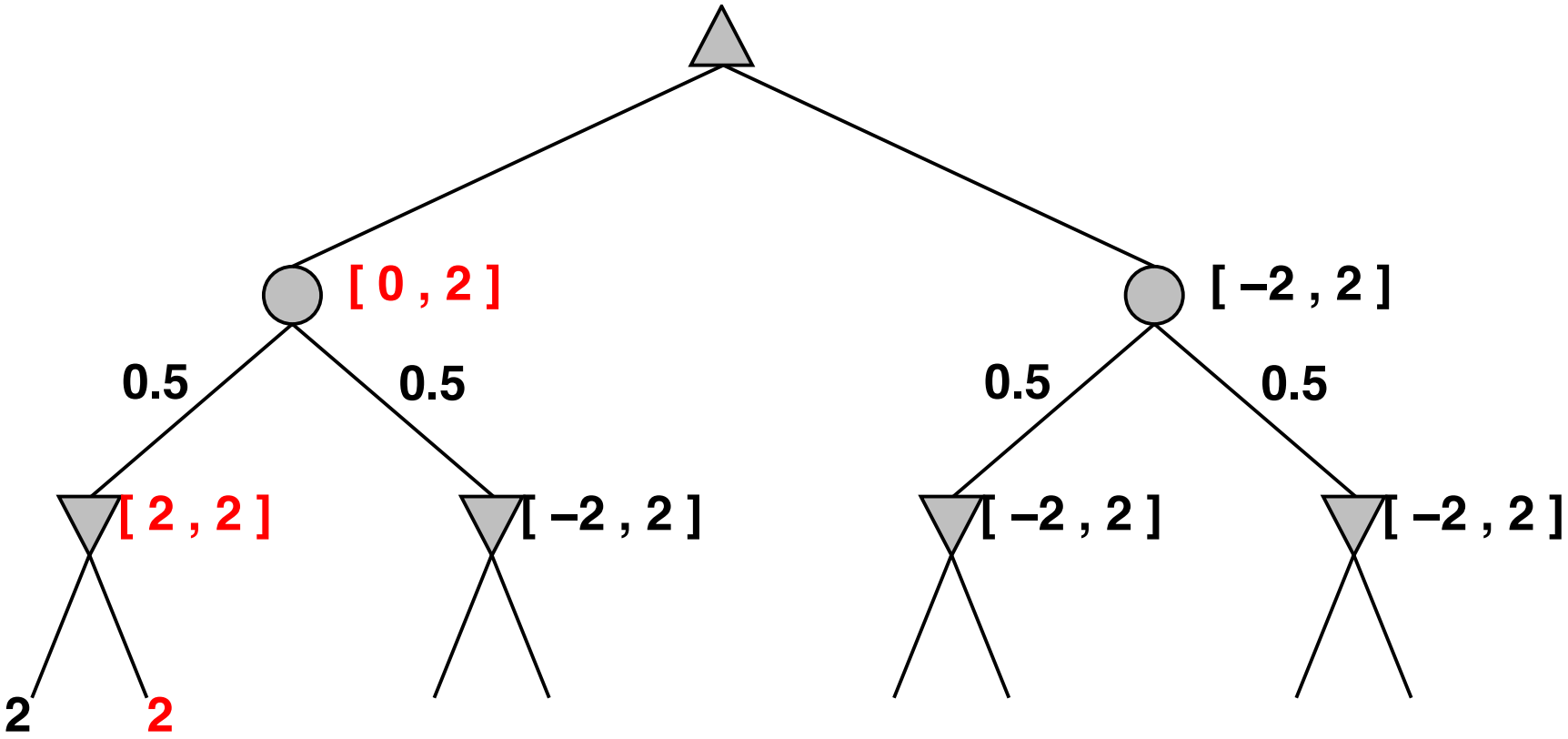
Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados



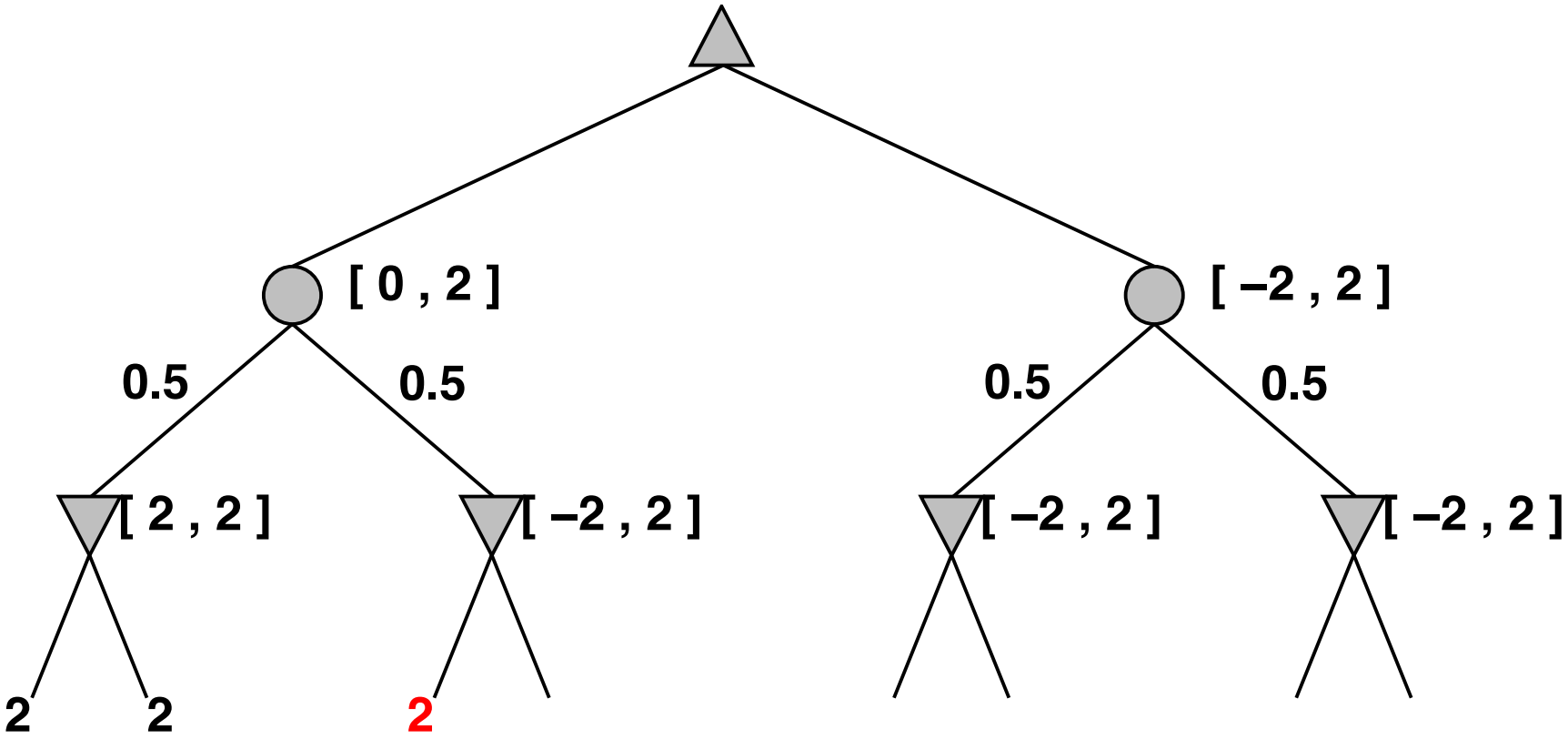
Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados



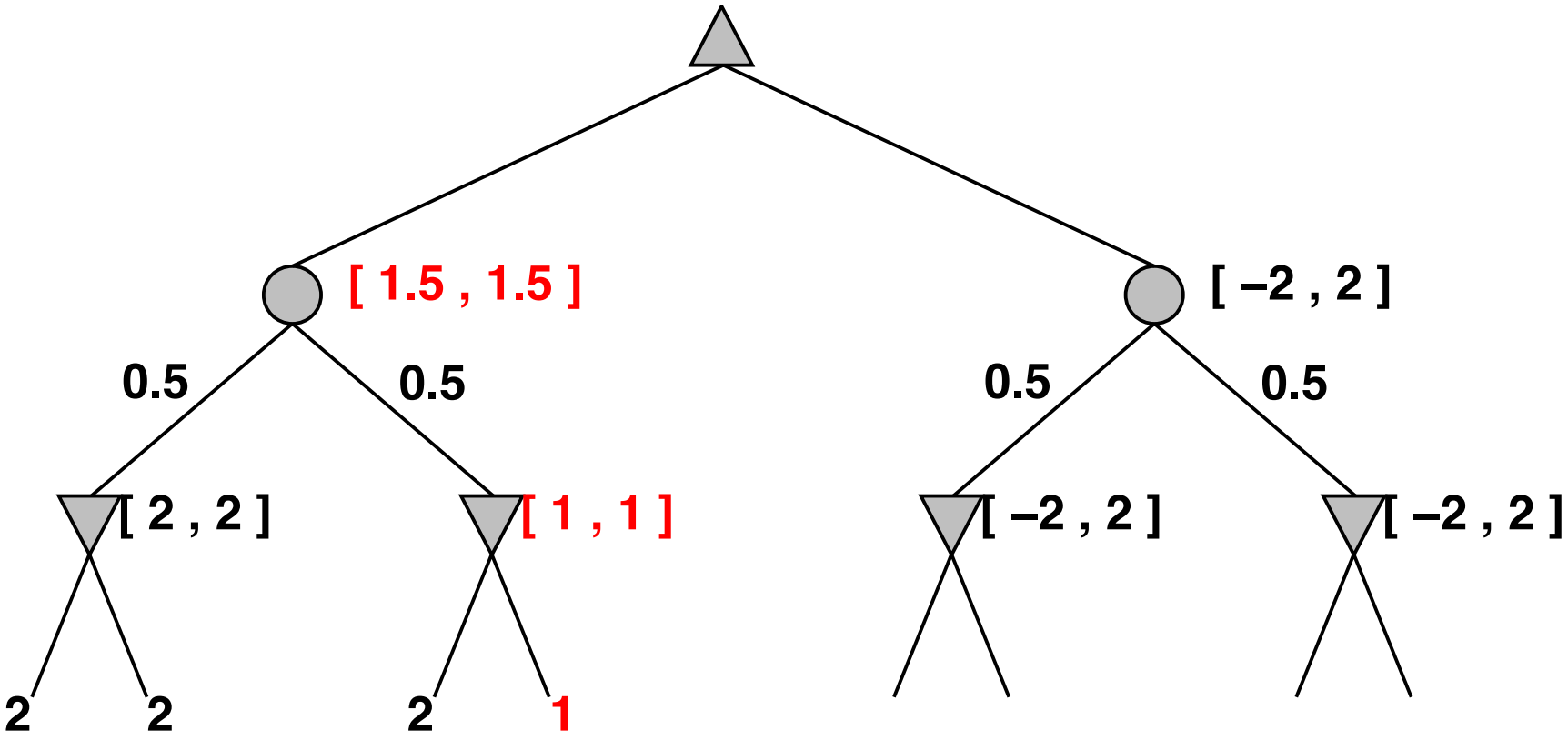
Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados



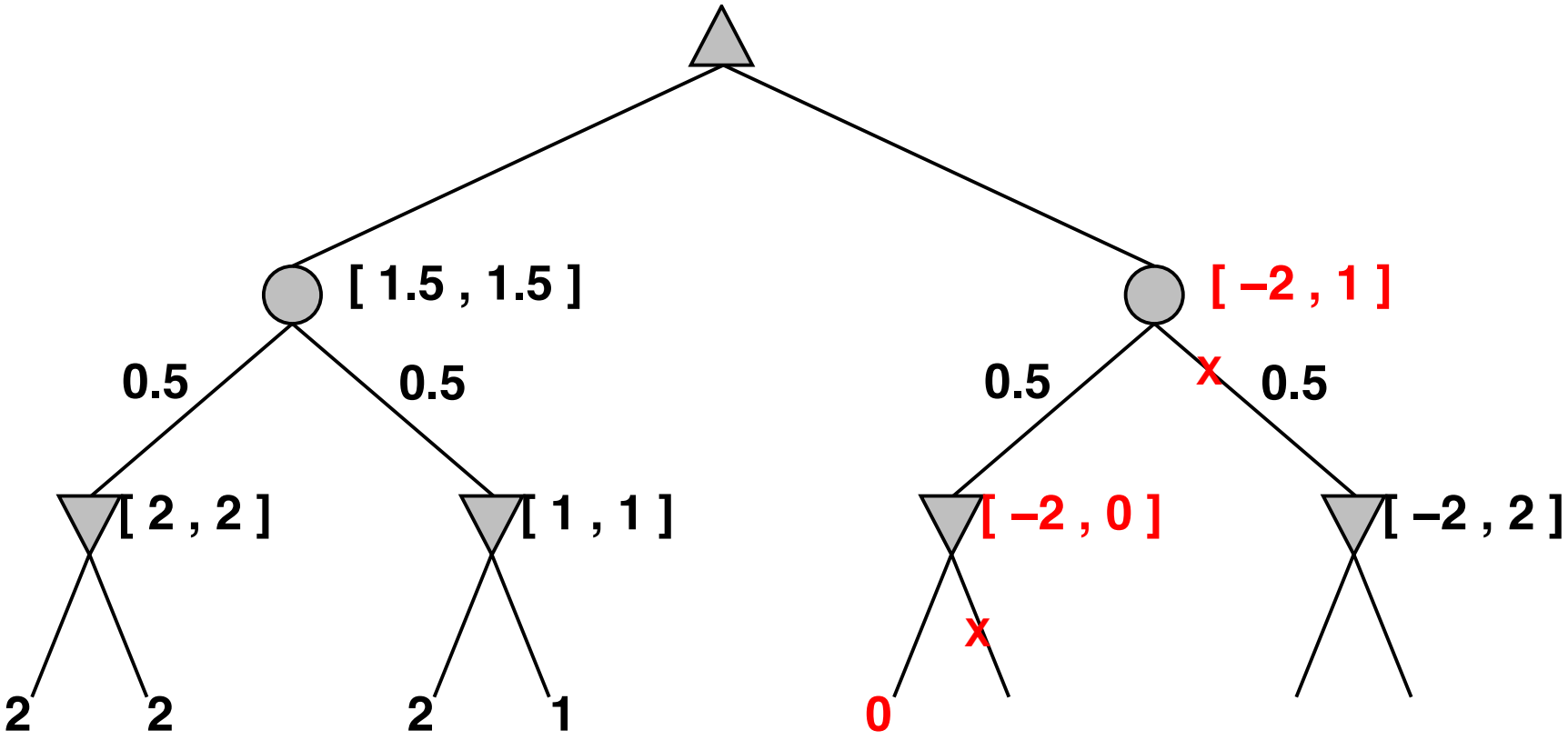
Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados



Exemplo de corte α-β em jogos estocásticos

- Pode-se cortar mais se os valores das folhas forem limitados

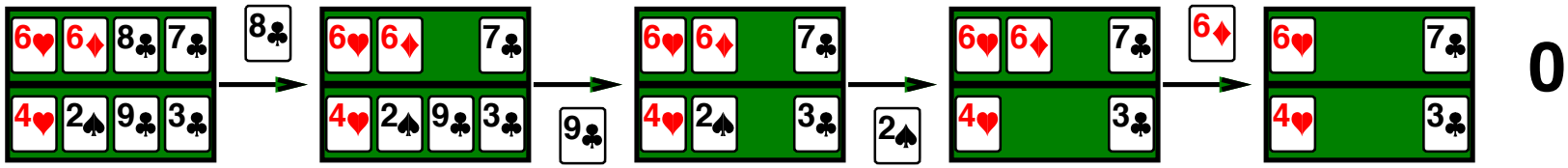


Jogos com informação imperfeita

- Exemplo: jogos de cartas, em que as cartas iniciais do adversário são desconhecidas
  - Tipicamente pode-se calcular a probabilidade de cada distribuição de cartas pelos jogadores
  - Aparentemente semelhante a um lançamento de dados no início do jogo
- Ideia: calcular o valor minimax de cada acção em cada mão, e escolher a acção com maior valor esperado de entre todas as mãos.*
- Caso especial: Se uma acção é óptima para todas as mãos então é óptima.*
- GIB, melhor programa de bridge actualmente, aproxima esta ideia
  - gerando 100 mãos consistentes com a informação de apostas actuais
  - escolhe a acção que ganha mais vazas em média

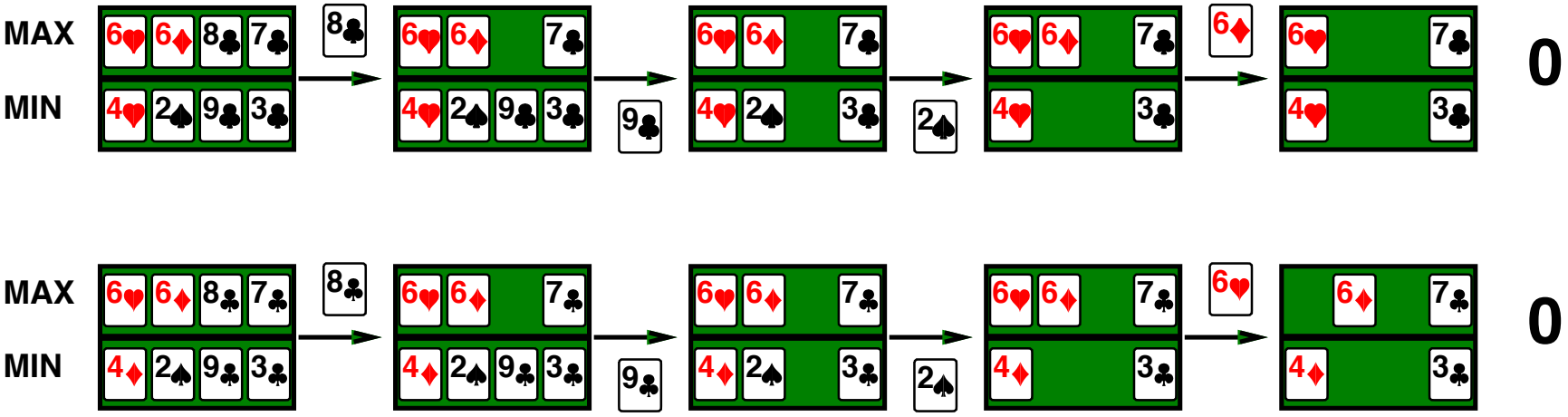
No entanto…

- Mão de quatro cartas, MAX joga primeiro



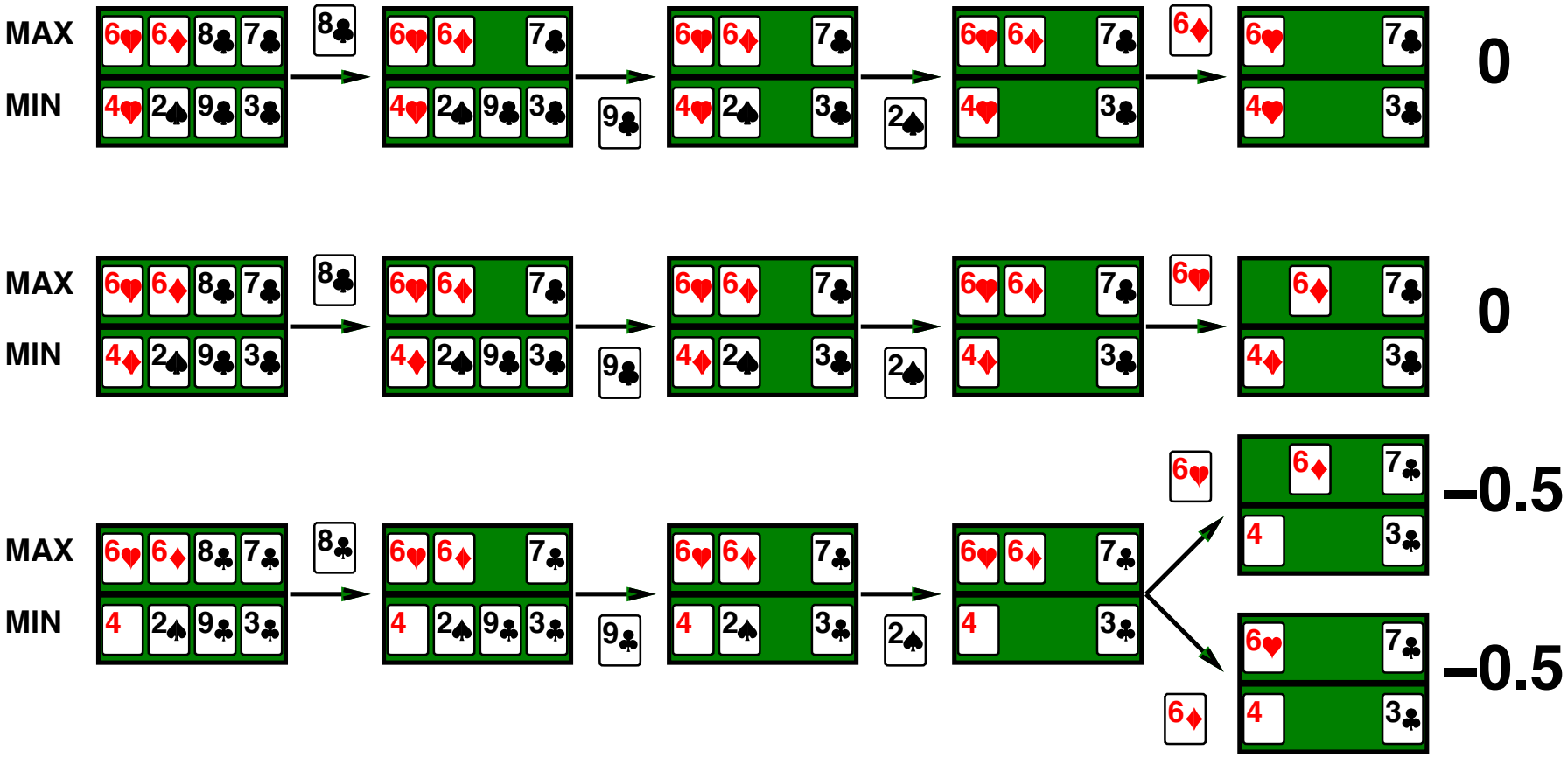
No entanto…

- Mão de quatro cartas, MAX joga primeiro



No entanto…

- Mão de quatro cartas, MAX joga primeiro



Exemplo de senso comum

- Estrada A leva a um pequeno pote de moedas de ouro
- Estrada B leva a um cruzamento:
- se for pela esquerda encontra um monte de jóias;
- se for pela direita é atropelado por um autocarro.

- Estrada A leva a um pequeno pote de moedas de ouro
- Estrada B leva a um cruzamento:
- se for pela esquerda é atropelado por um autocarro.
- se for pela direita encontra um monte de jóias;

- Estrada A leva a um pequeno pote de moedas de ouro
- Estrada B leva a um cruzamento:
- adivinhe correctamente e encontra um monte de jóias;
- adivinhe incorrectamente e é atropelado por um autocarro.

Análise Correcta

- * Intuição segundo a qual o valor de uma acção é a média de todos os seus valores em todos os estados possíveis é **ERRADA**

- Com obervação parcial, valor de uma acção depende do estado de informação ou estado de crença em que o agente se encontra

- Pode gerar e procurar a árvore de estados de informação

- Origina os seguintes comportamentos racionais
  - Agir para obter informação
  - Fazer sinais ao parceiro
  - Agir aleatoriamente para minimizar a descoberta de informação

Sumário

- Jogos ilustram vários aspectos importantes da IA
- perfeição é inatingível ⇒ tem de se aproximar
- boa ideia pensar sobre o que se vai pensar
- a incerteza limita a atribuição de valores a estados
- Os jogos estão para IA como a Fórmula I está para a indústria automóvel