

Adversarial Search

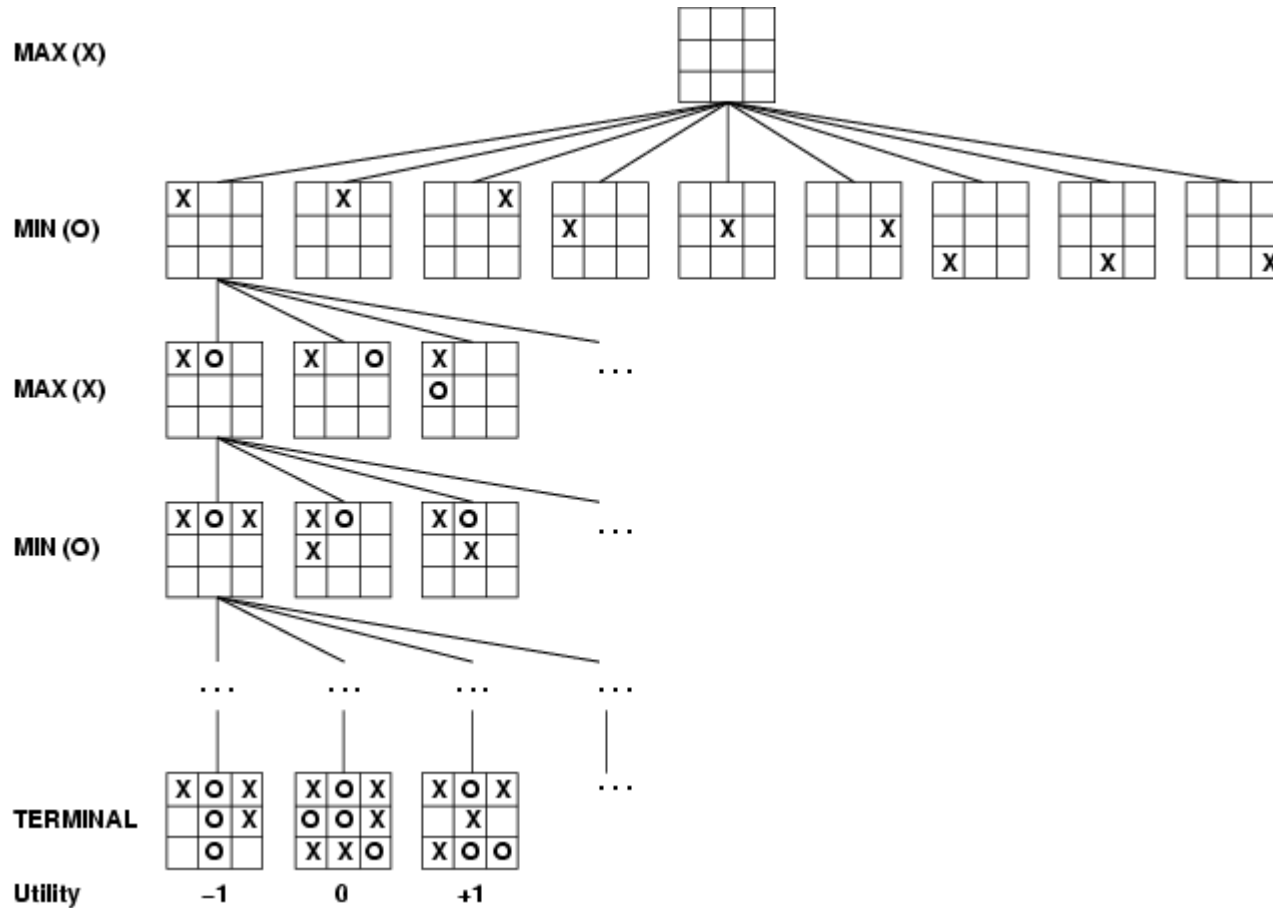
Chapter 6

Section 1 – 4

Games vs. search problems

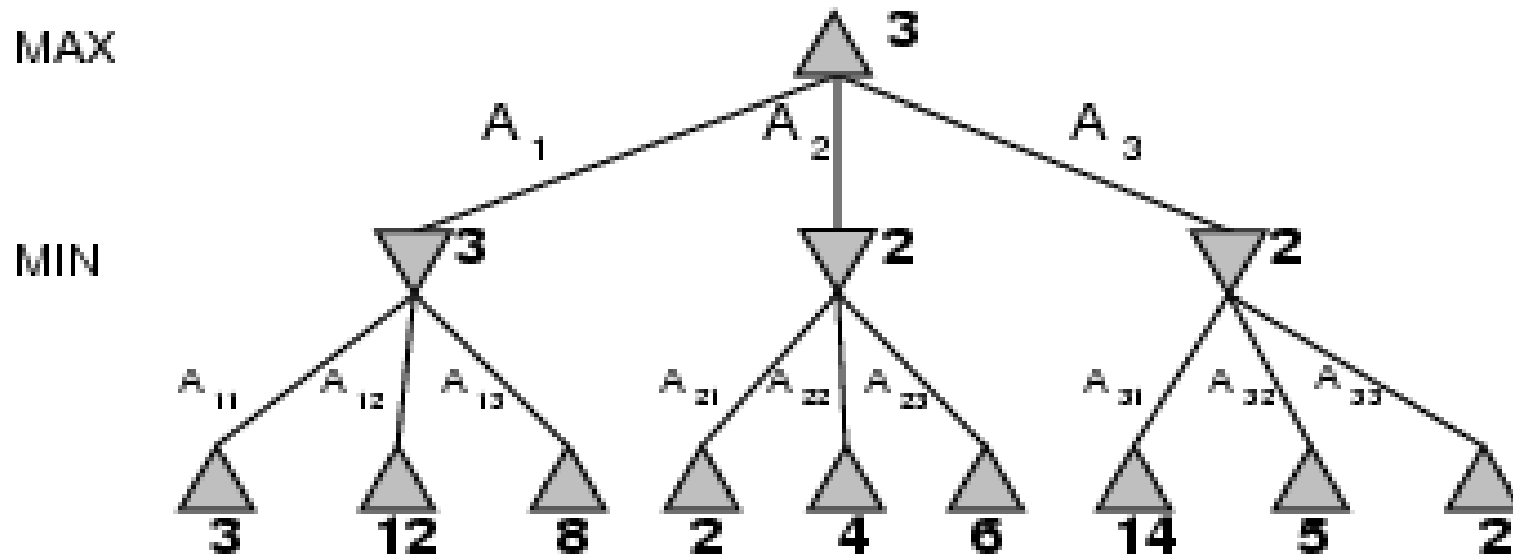
- "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate

Game tree (2-player, deterministic, turns)



Minimax

- E.g., 2-ply game:
- Perfect play for deterministic games
- Idea: choose move to position with highest **minimax value**
= best achievable payoff against best play



Minimax algorithm

function MINIMAX-DECISION(*state*) *returns an action*

$v \leftarrow \text{MAX-VALUE}(\textit{state})$

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**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$

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**

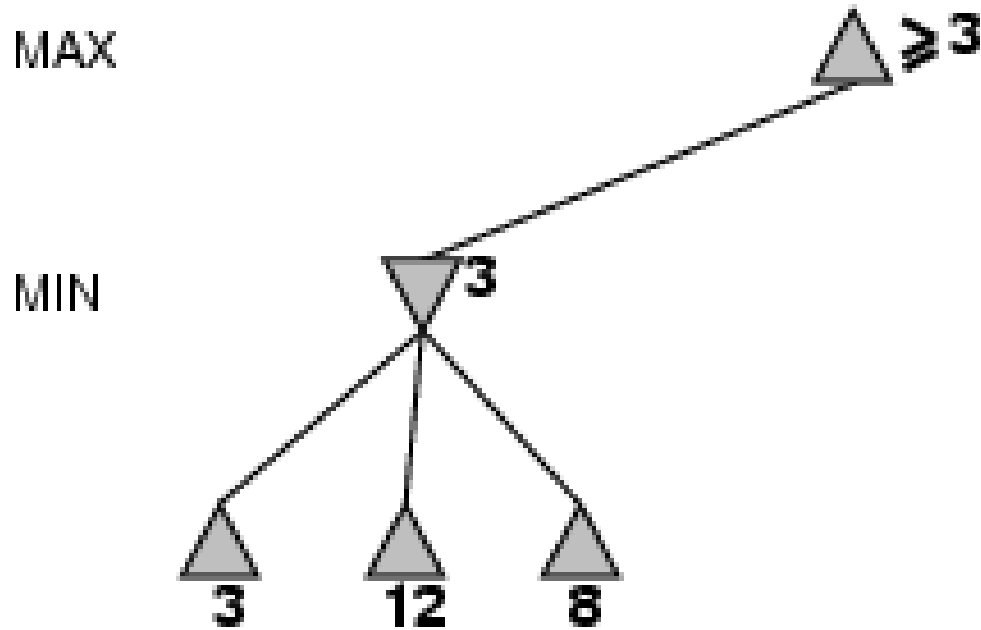
$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$

return *v*

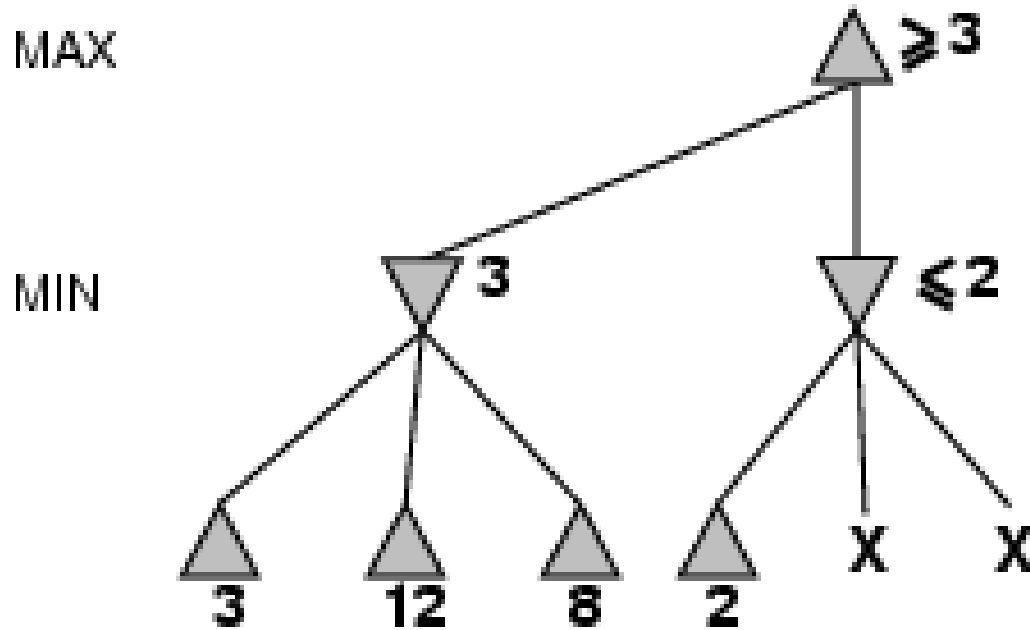
Properties of minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$ (depth-first exploration)
- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
→ exact solution completely infeasible

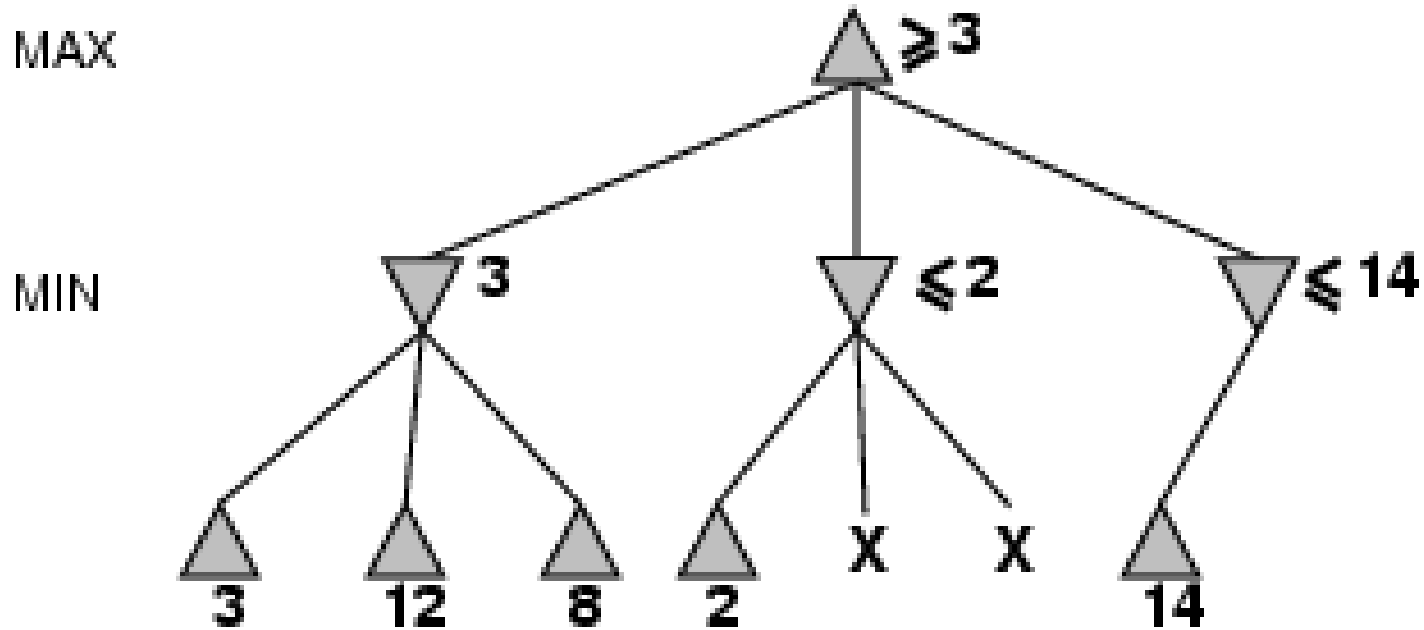
α - β pruning example



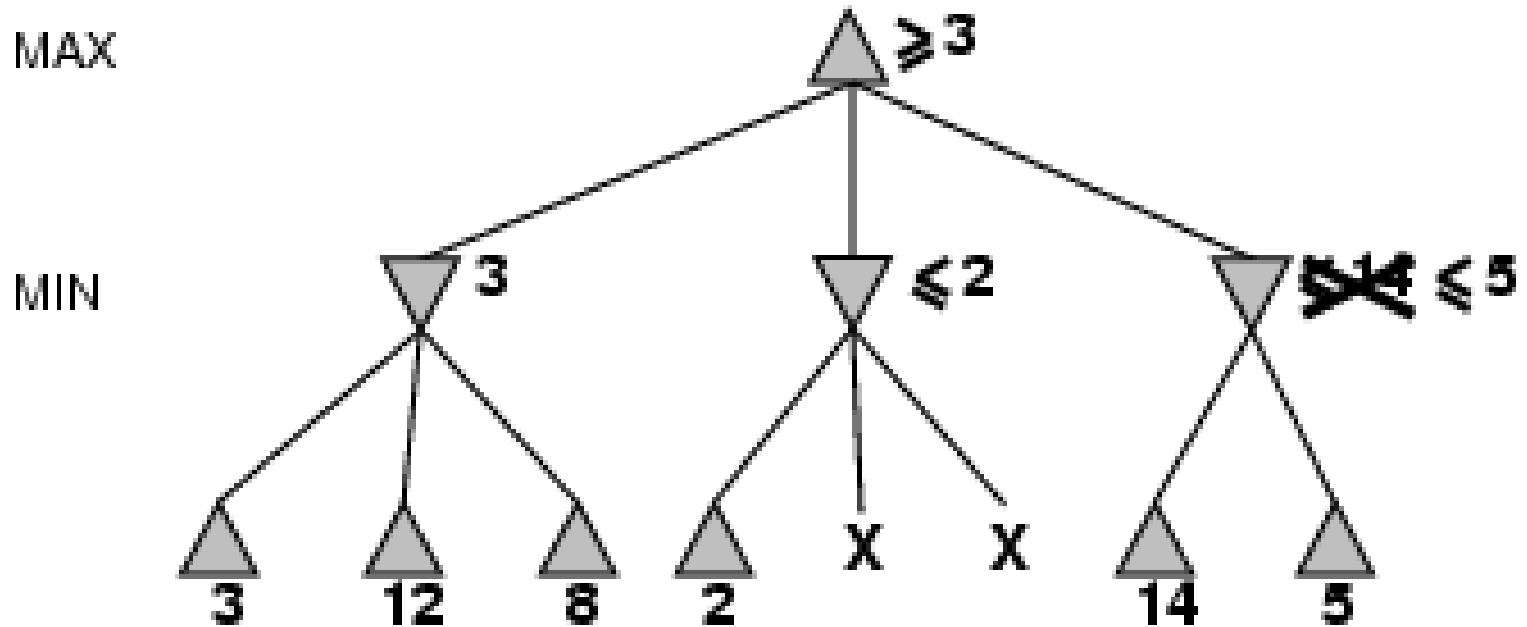
α - β pruning example



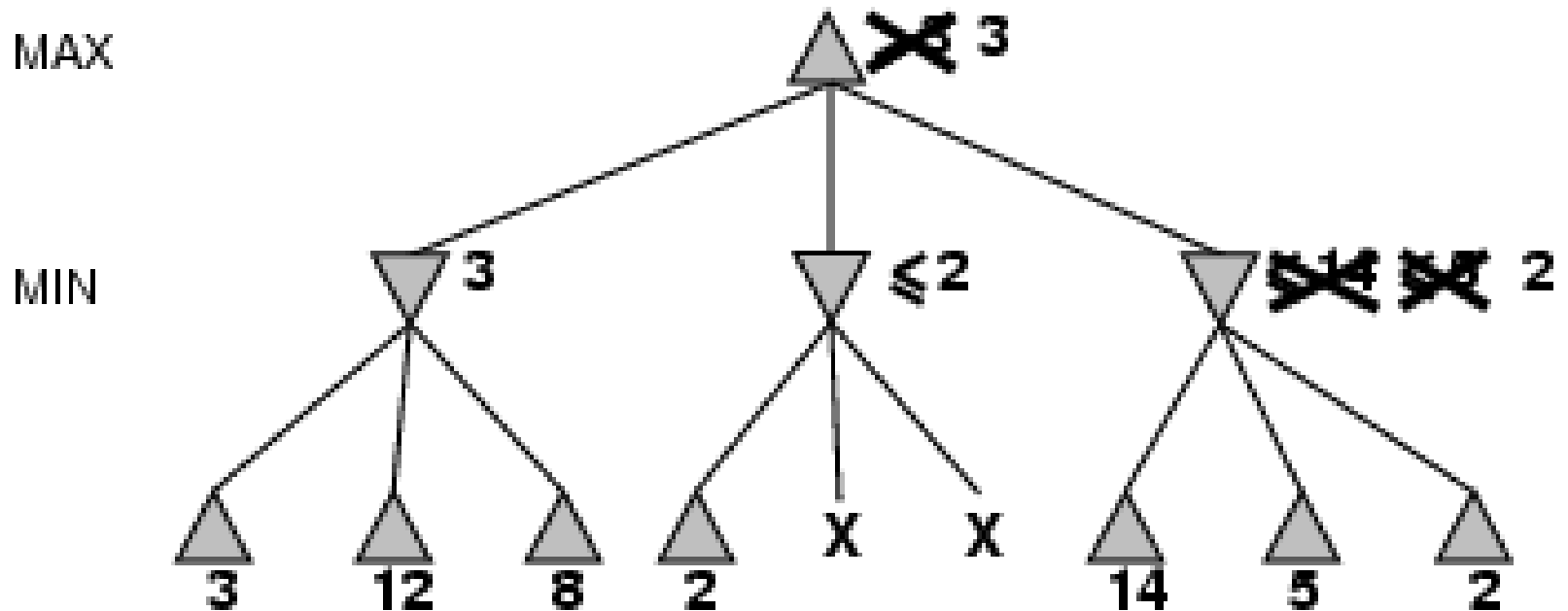
α - β pruning example



α - β pruning example



α - β pruning example



Properties of α - β

- Pruning **does not** affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
 - **doubles** depth of search

Why is it called α - β ?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
- If v is worse than α , *max* will avoid it
→ prune that branch
- Define β similarly for *min*

MAX

MIN

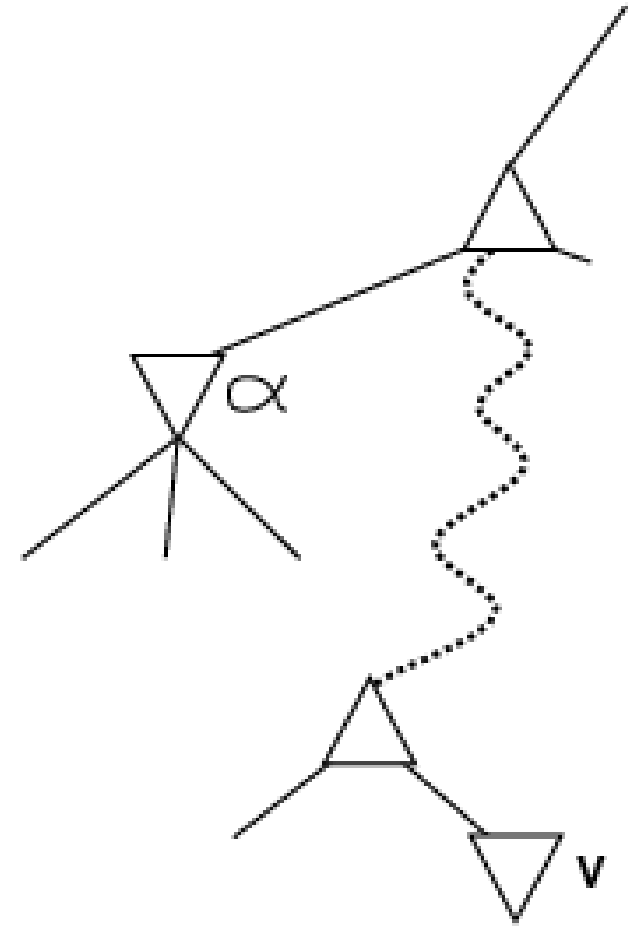
..

..

..

MAX

MIN



The α - β algorithm

function ALPHA-BETA-SEARCH(*state*) *returns an action*

inputs: *state*, current state in game

$v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$

return the *action* in SUCCESSORS(*state*) with value v

function MAX-VALUE(*state*, α , β) *returns a utility value*

inputs: *state*, current state in game

α , the value of the best alternative for MAX along the path to *state*

β , the value of the best alternative for MIN along the path to *state*

if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow -\infty$

for a, s in SUCCESSORS(*state*) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$

if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v

The α - β algorithm

```
function MIN-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  inputs: state, current state in game
            $\alpha$ , the value of the best alternative for MAX along the path to state
            $\beta$ , the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow +\infty$ 
  for  $a, s$  in SUCCESSORS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s, \alpha, \beta))$ 
    if  $v \leq \alpha$  then return  $v$ 
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return  $v$ 
```

Resource limits

Suppose we have 100 secs, explore 10^4 nodes/sec

→ 10^6 nodes per move

Standard approach:

- **cutoff test:**
e.g., depth limit
- **evaluation function**
= estimated desirability of position

Evaluation functions

- For chess, typically **linear** weighted sum of **features**

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

- e.g., $w_1 = 9$ with

$f_1(s) = (\text{number of white queens}) - (\text{number of black queens}), \text{ etc.}$

Cutting off search

MinimaxCutoff is identical to *MinimaxValue* except

1. *Terminal?* is replaced by *Cutoff?*
2. *Utility* is replaced by *Eval*

Does it work in practice?

$$b^m = 10^6, b=35 \rightarrow m=4$$

4-ply lookahead is a hopeless chess player!

- 4-ply \approx human novice
- 8-ply \approx typical PC, human master
- 12-ply \approx Deep Blue, Kasparov

Summary

- Games are fun to work on!
- They illustrate several important points about AI
- perfection is unattainable → must approximate
- good idea to think about what to think about