5

Adversarial Search

Claudia Chirita

School of Informatics, University of Edinburgh

THE UNIVERSITY *of* EDINBURGH
informatics

Based on slides by: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann, Vaishak Belle

5.a

Games. Minimax

GAMES VS. SEARCH PROBLEMS

"Unpredictable" opponent $\rightarrow$ solution is a **strategy**
specifying a move for every possible opponent reply

Time limits $\rightarrow$ unlikely to find goal, must approximate

TYPES OF GAMES

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon, monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble, nuclear war

GAMES

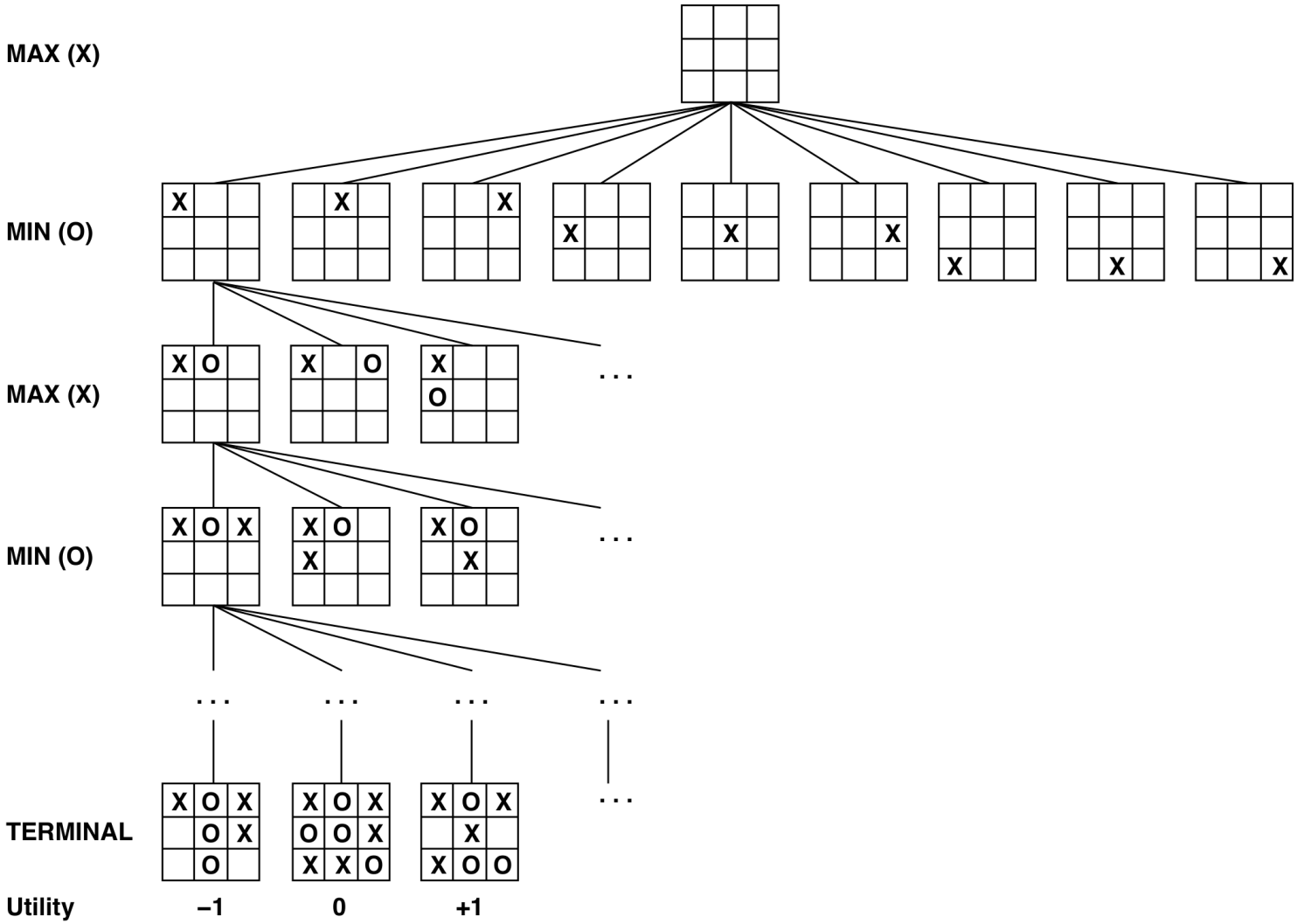
We are interested in zero-sum games of **perfect information**:

deterministic & fully observable

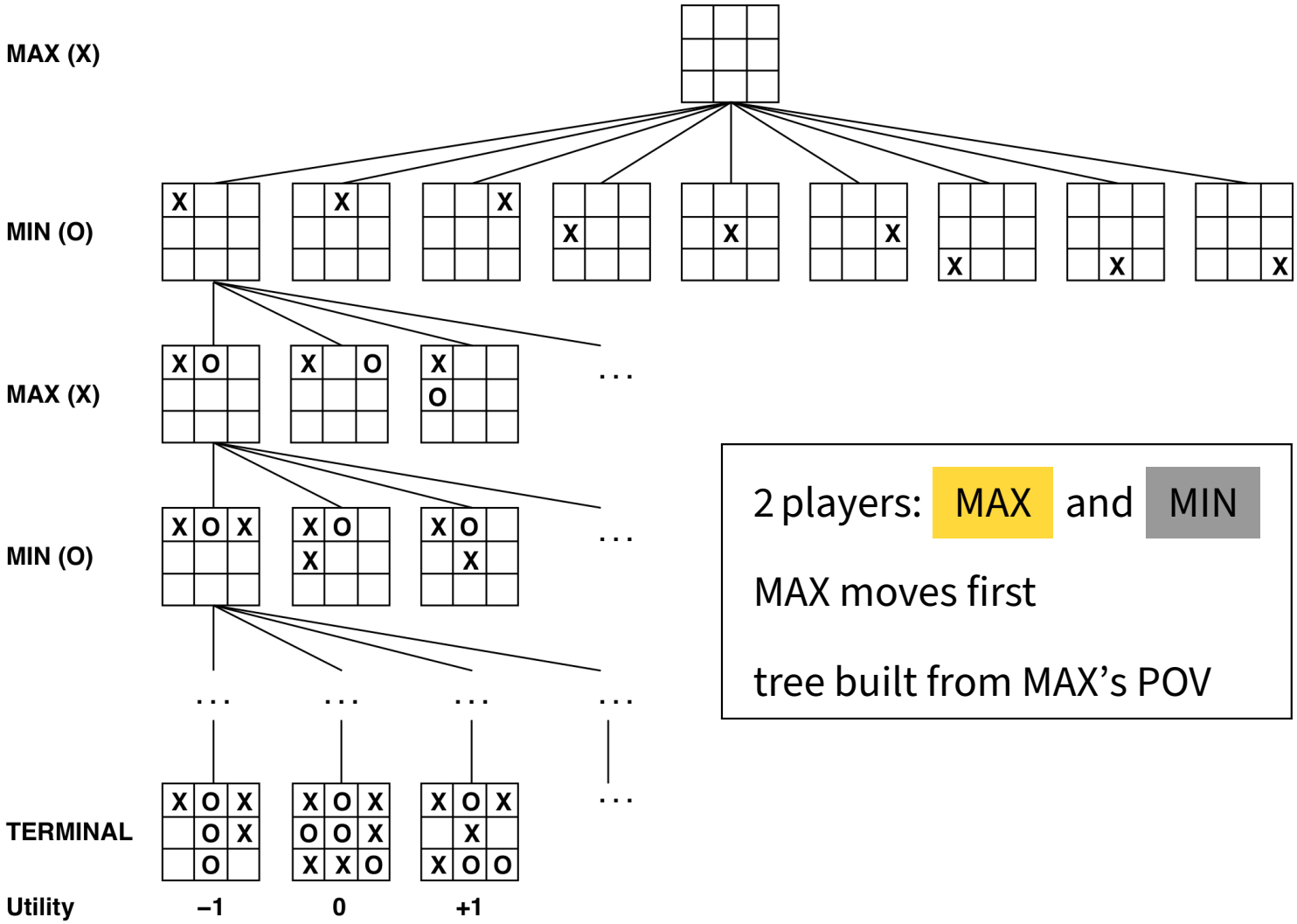
agents act alternately

utilities at end of game are equal and opposite

GAME TREE (2-PLAYER. DETERMINISTIC. TURNS)



GAME TREE (2-PLAYER. DETERMINISTIC. TURNS)



2 players: MAX and MIN

MAX moves first

tree built from MAX's POV

OPTIMAL DECISIONS

Normal search · optimal decision is a sequence of actions leading to a goal state (i.e. a winning terminal state)

ADVERSARIAL SEARCH

MIN has a say in game

MAX needs to find a contingent strategy which specifies:

- MAX's move in initial state then ...
- MAX's moves in states resulting from every response by MIN to the move then ...
- MAX's moves in states resulting from every response by MIN to all those moves, etc. ...

OPTIMAL DECISIONS

Normal search · optimal decision is a sequence of actions leading to a goal state (i.e. a winning terminal state)

ADVERSARIAL SEARCH

minimax value of a node = utility for MAX of being in corresponding state

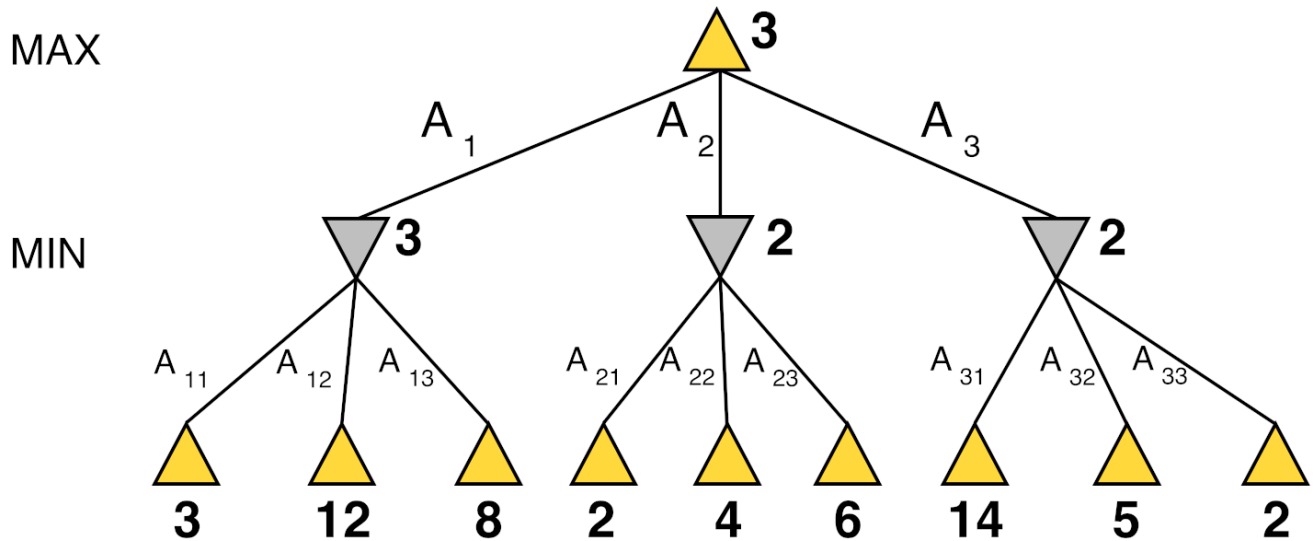
$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

MINIMAX

Perfect play for deterministic, perfect-information games

IDEA choose move to position with highest **minimax value**
= best achievable payoff against best play

EXAMPLE

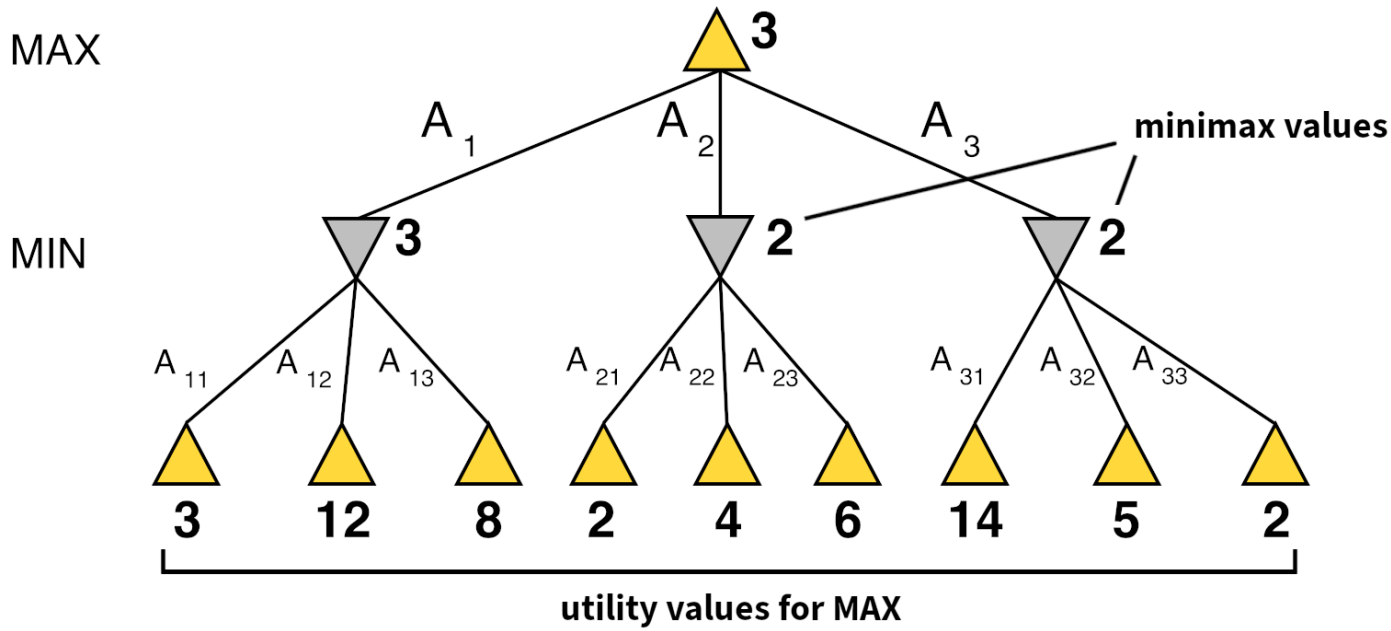


MINIMAX

Perfect play for deterministic, perfect-information games

IDEA choose move to position with highest **minimax value**
= best achievable payoff against best play

EXAMPLE



MINIMAX · ALGORITHM

function MINIMAX-DECISION($state$) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state, a$))

---

function MAX-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

---

function MIN-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow \infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

MINIMAX · ALGORITHM

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

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

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

IDEA proceed all the way down to the leaves of the tree
then minimax values are backed up through tree

MINIMAX · PROPERTIES

- complete
- optimal
- time
- space

MINIMAX · PROPERTIES

complete	Yes	(if tree is finite)
optimal		
time		
space		

MINIMAX · PROPERTIES

complete	Yes	(if tree is finite)
optimal	Yes	(against an optimal opponent)
? time		
space		

MINIMAX · PROPERTIES

complete	Yes	(if tree is finite)
optimal	Yes	(against an optimal opponent)
time	$O(b^m)$	
❓ space		

MINIMAX · PROPERTIES

complete	Yes	(if tree is finite)
optimal	Yes	(against an optimal opponent)
time	$O(b^m)$	
space	$O(bm)$	(depth-first exploration)

CHESS

MINIMAX · PROPERTIES

complete	Yes	(if tree is finite)
optimal	Yes	(against an optimal opponent)
time	$O(b^m)$	
space	$O(bm)$	(depth-first exploration)

CHESS

$b \approx 35, m \approx 100$ for *reasonable* games

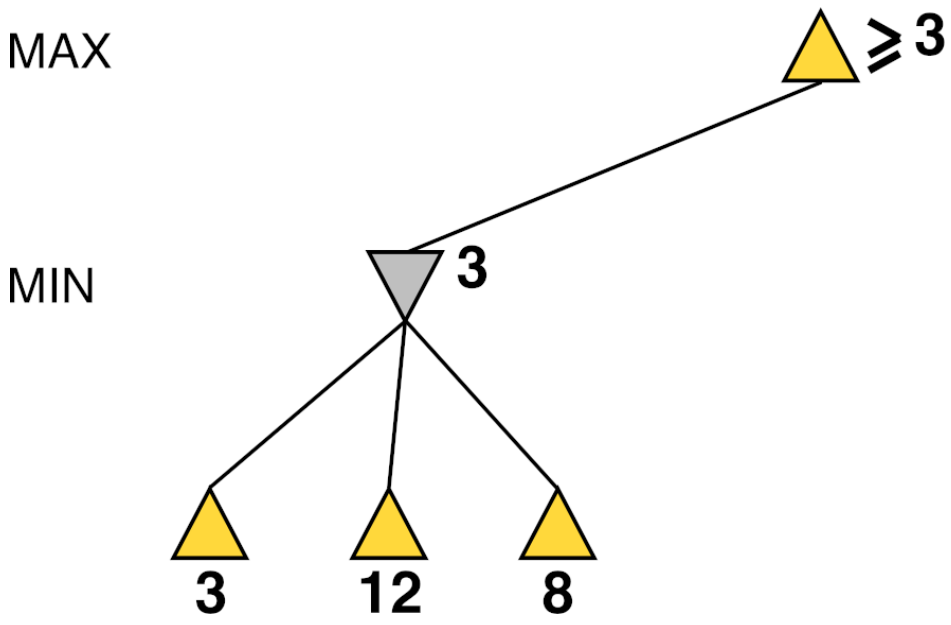
$\rightarrow$ exact solution completely infeasible

We would like to eliminate (large) parts of the game tree.

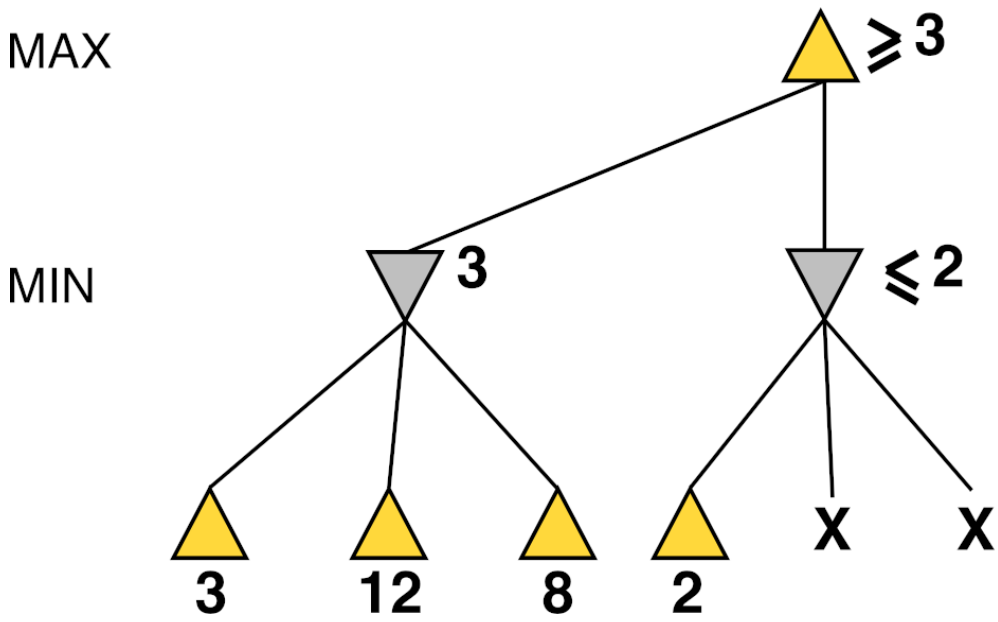
5.b

α-β pruning

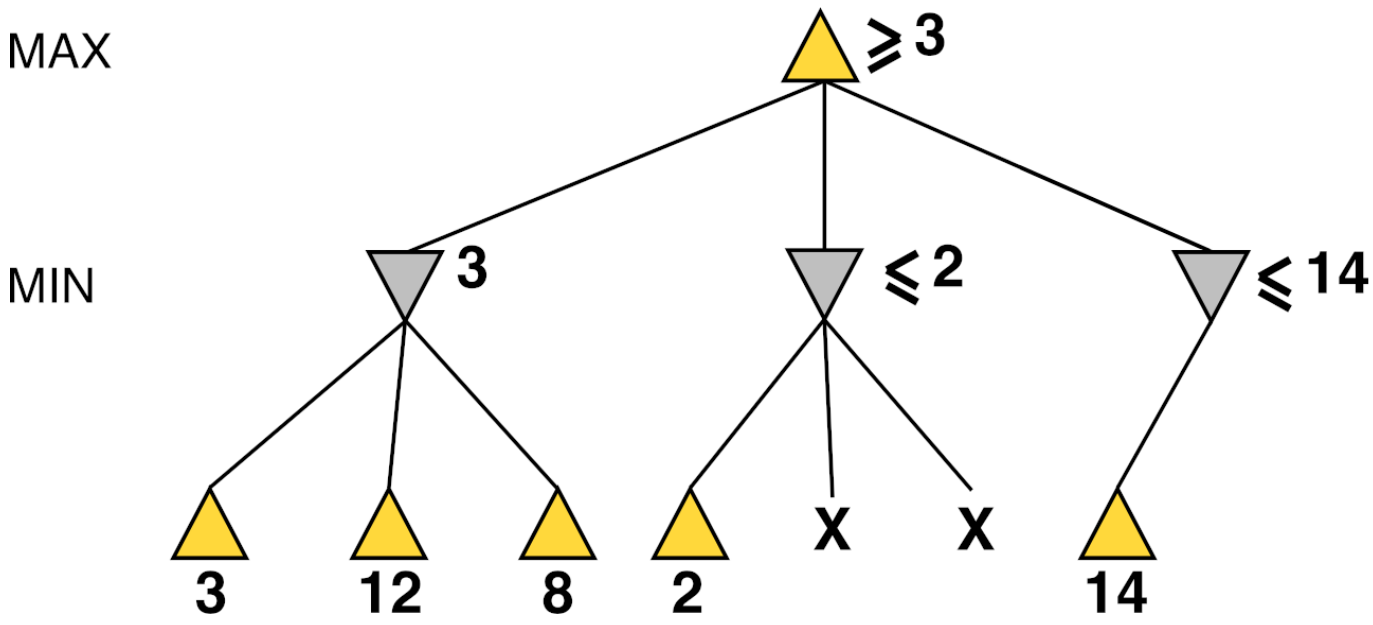
α-β PRUNING · EXAMPLE



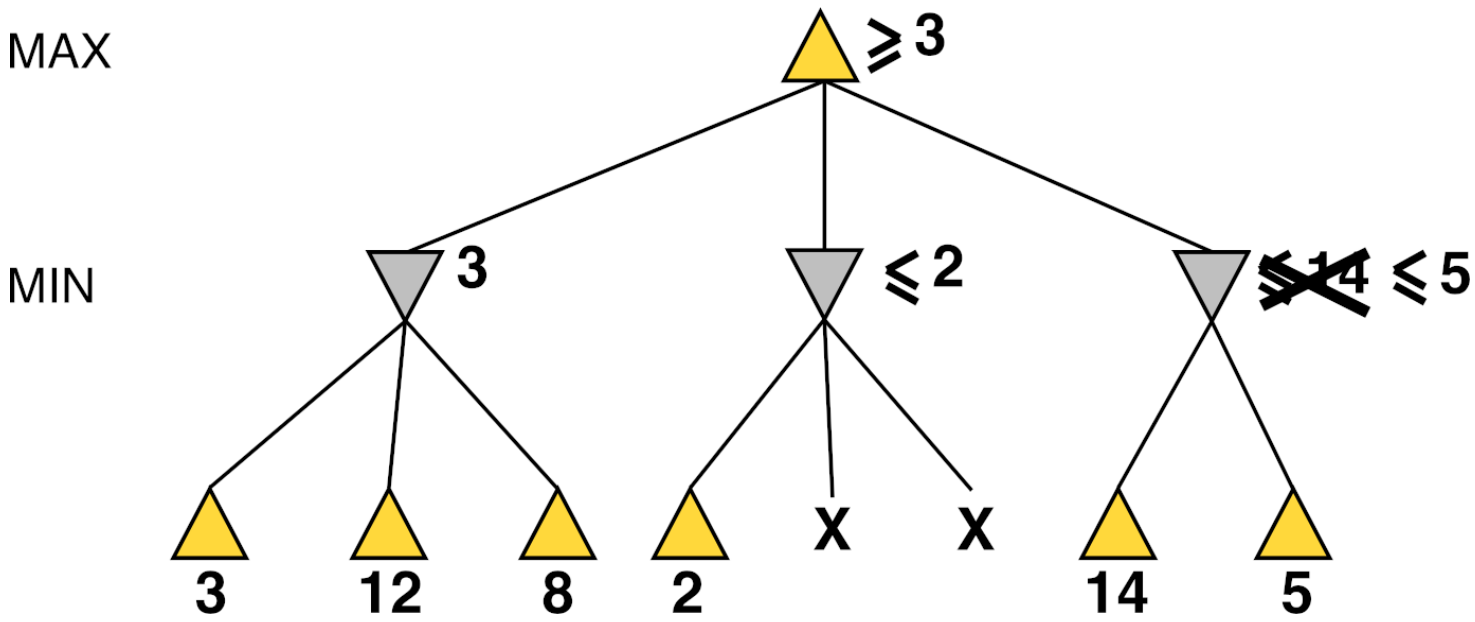
α-β PRUNING · EXAMPLE



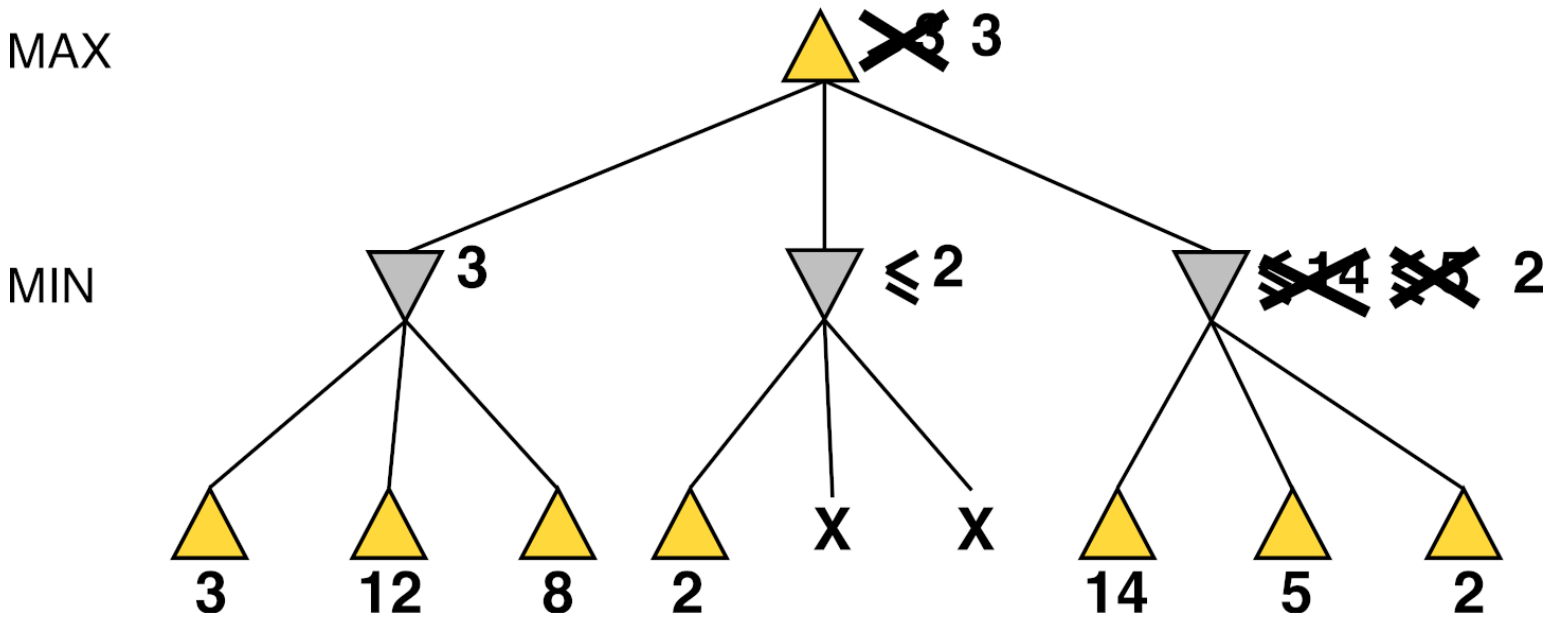
α-β PRUNING · EXAMPLE



α-β PRUNING · EXAMPLE



α-β PRUNING · EXAMPLE



QUESTION TIME!

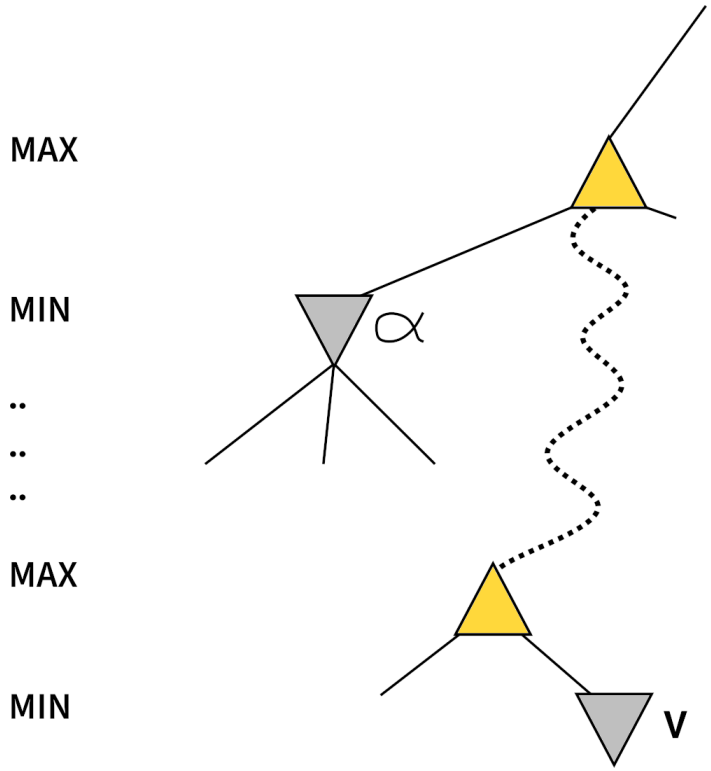
Are the minimax value of root and, hence, the minimax decision **independent** of pruned leaves?

Let pruned leaves have values u and v.

$$\begin{aligned}
\text{MINIMAX}(\textit{root}) &= \max(\min(3, 12, 8), \min(2, u, v), \min(14, 5, 2)) \\
&= \max(3, \min(2, u, v), 2) \\
&= \max(3, z, 2) \quad \text{where } z \leqslant 2 \\
&= 3
\end{aligned}$$

YES!

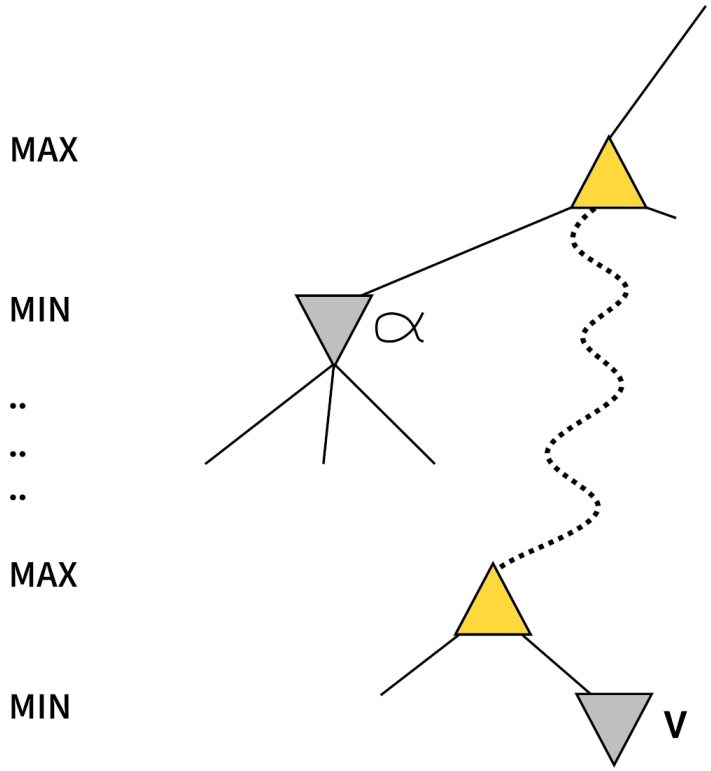
WHY IS IT CALLED α-β ?



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 $\rightarrow$ prune that branch.

WHY IS IT CALLED α-β ?



β is the value of the best (i.e. lowest-value) choice found so far at any choice point along the path for MIN.

If v is worse than β, MIN will avoid it $\rightarrow$ prune that branch.

α-β · ALGORITHM

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

---

function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

α-β · PROPERTIES

Pruning does not affect final result (as we saw for our example).

Good move ordering improves effectiveness of pruning.
(How could the previous tree be better?)

With "perfect ordering", time complexity = $O(b^{m/2})$

$\rightarrow$ branching factor goes from b to $\sqrt{b}$

$\rightarrow$ **doubles** solvable depth compared to minimax

A simple example of the value of reasoning about which computations are relevant (a form of *meta-reasoning*).

5.c

Resource Limits

RESOURCE LIMITS

Standard approach

- Use a cutoff test instead of a terminal test

  e.g. depth limit (perhaps add quiescence search, which tries to search interesting positions to a greater depth than quiet ones)
- Use an evaluation function instead of utility

  i.e. evaluation function that estimates desirability of position

MinimaxCutoff is identical to MinimaxValue except:

TERMINAL-TEST is replaced by CUTOFF

UTILITY is replaced by EVAL

EVALUATION FUNCTIONS

typically linear weighted sum of **features**

$$\text{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

where each w_i is a weight and each f_i is a feature of state s

CHESS

queen = 1, king = 2, etc.

f_i number of pieces of type i on board

w_i value of the piece of type i

CUTTING OFF SEARCH

Does it work in practice?

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

$\rightarrow$ 10^6 nodes per move

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

4-ply lookahead is a hopeless chess player!

4-ply $\approx$ human novice

8-ply $\approx$ typical PC, human master

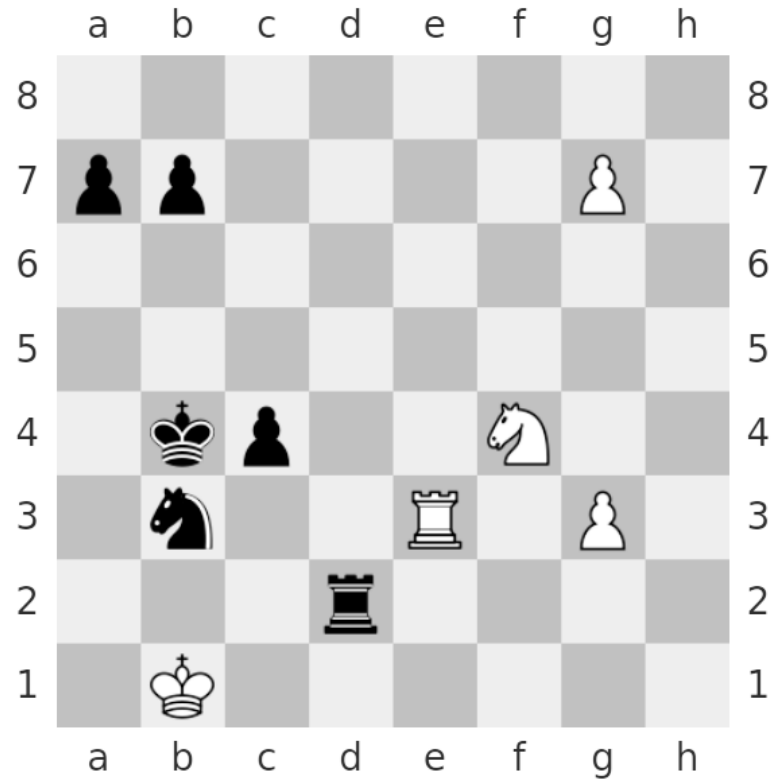
12-ply $\approx$ Deep Blue, Kasparov

DETERMINISTIC GAMES IN PRACTICE

Checkers Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.

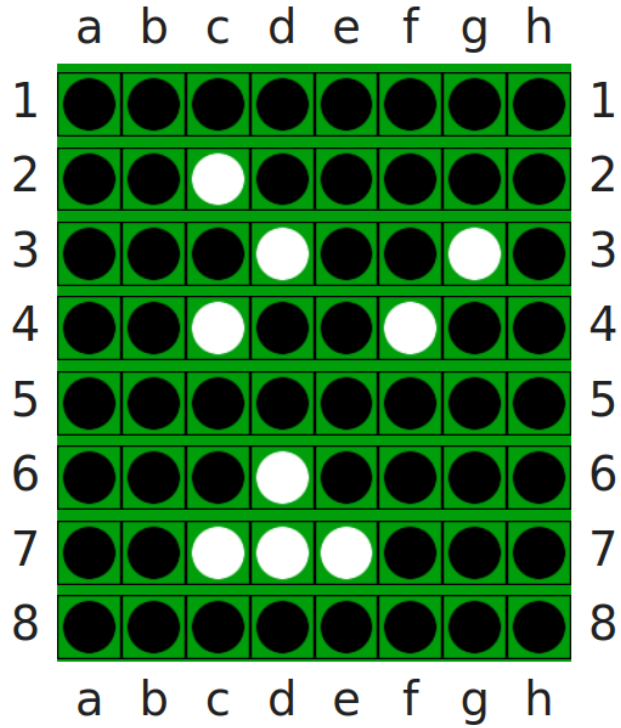
DETERMINISTIC GAMES IN PRACTICE

Chess Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.



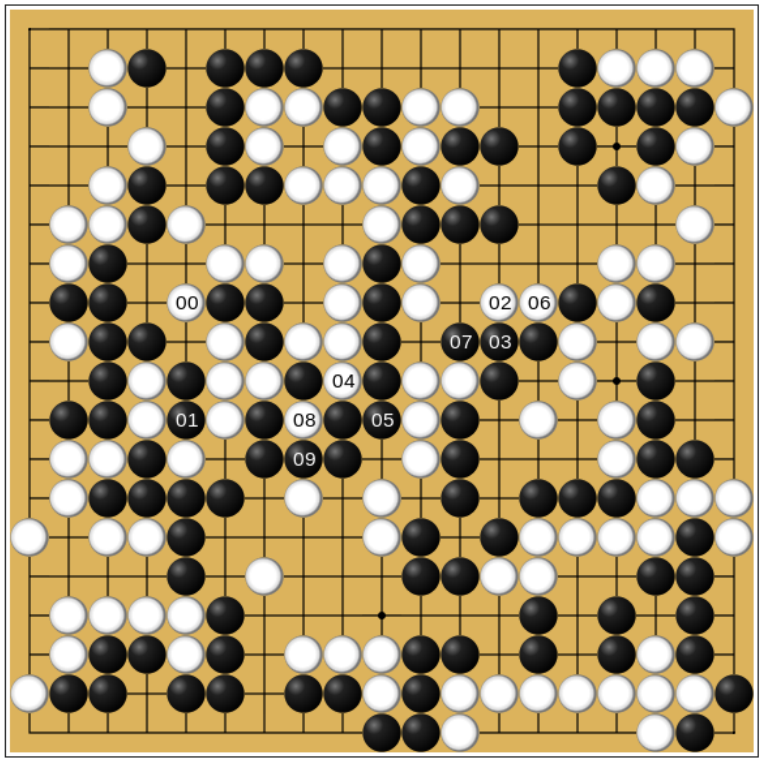
DETERMINISTIC GAMES IN PRACTICE

Othello human champions refuse to compete against computers. Logistello, written by Michael Buro, defeated the human world champion Takeshi Murakami six games to none in 1997. The best Othello programs are now much stronger than any human player.



DETERMINISTIC GAMES IN PRACTICE

Go human champions used to refuse to compete against computers, because they were too bad. In Go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves. In 2017, AlphaGo changed everything.



GAMES?

Games are fun!

They illustrate several important points about AI.

Perfection is unattainable $\rightarrow$ must approximate.

Good idea to think about what to think about.

Public demonstrations of the AI potential for superhuman performance.