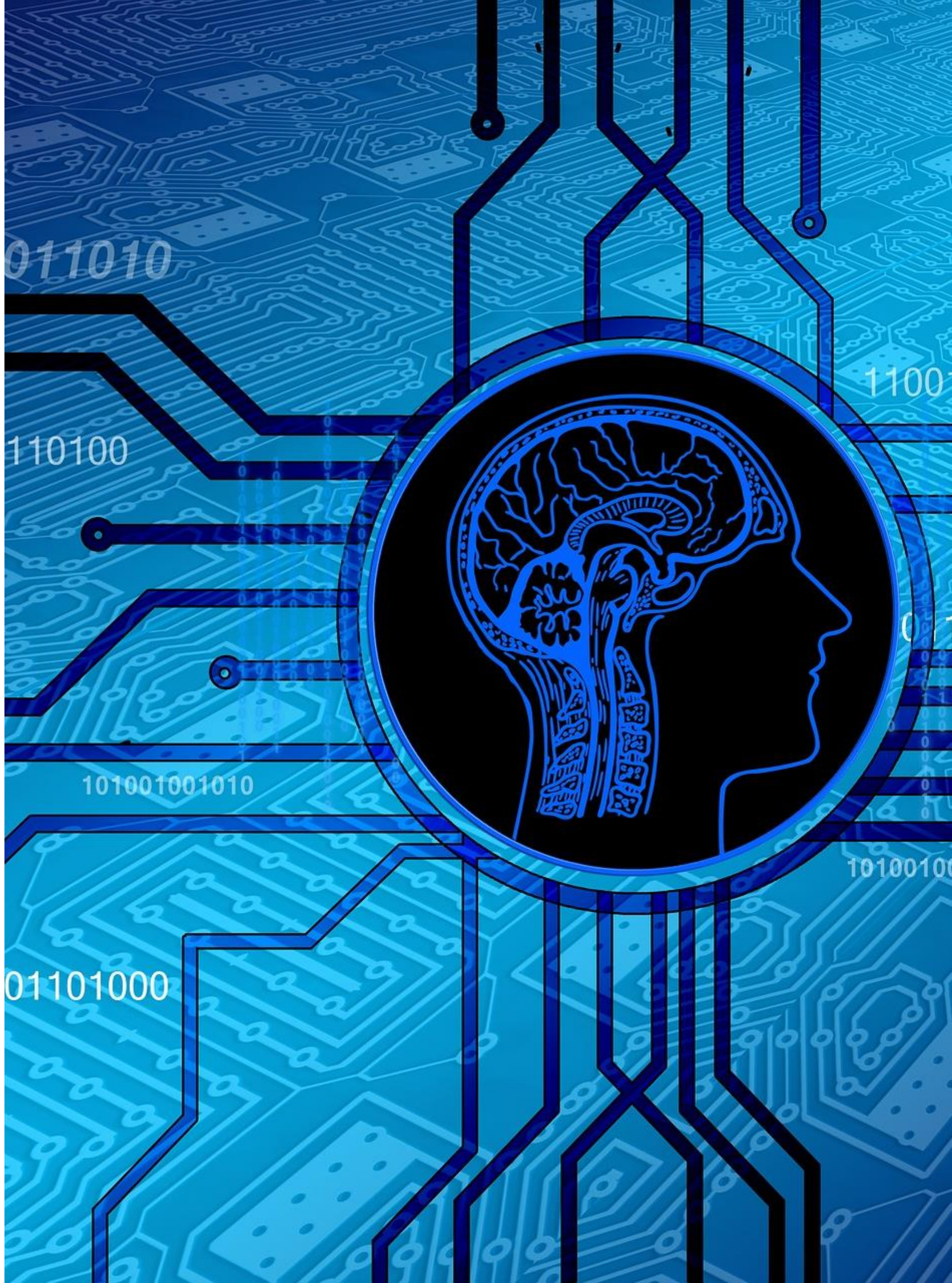
Adversarial Search

Petros Papapanagiotou

Informatics 2D: Reasoning and Agents
Lecture 8



Games vs. search problems

"Unpredictable" opponent → solution is a **strategy** / **policy**

- Specify a move for *every possible* opponent reply

Time limits → unlikely to find goal, must approximate

Discrete!

TYPES OF GAMES	**deterministic**	**chance**
perfect information	chess, checkers	backgammon, monopoly
imperfect information	battleships, stratego	bridge, poker, scrabble

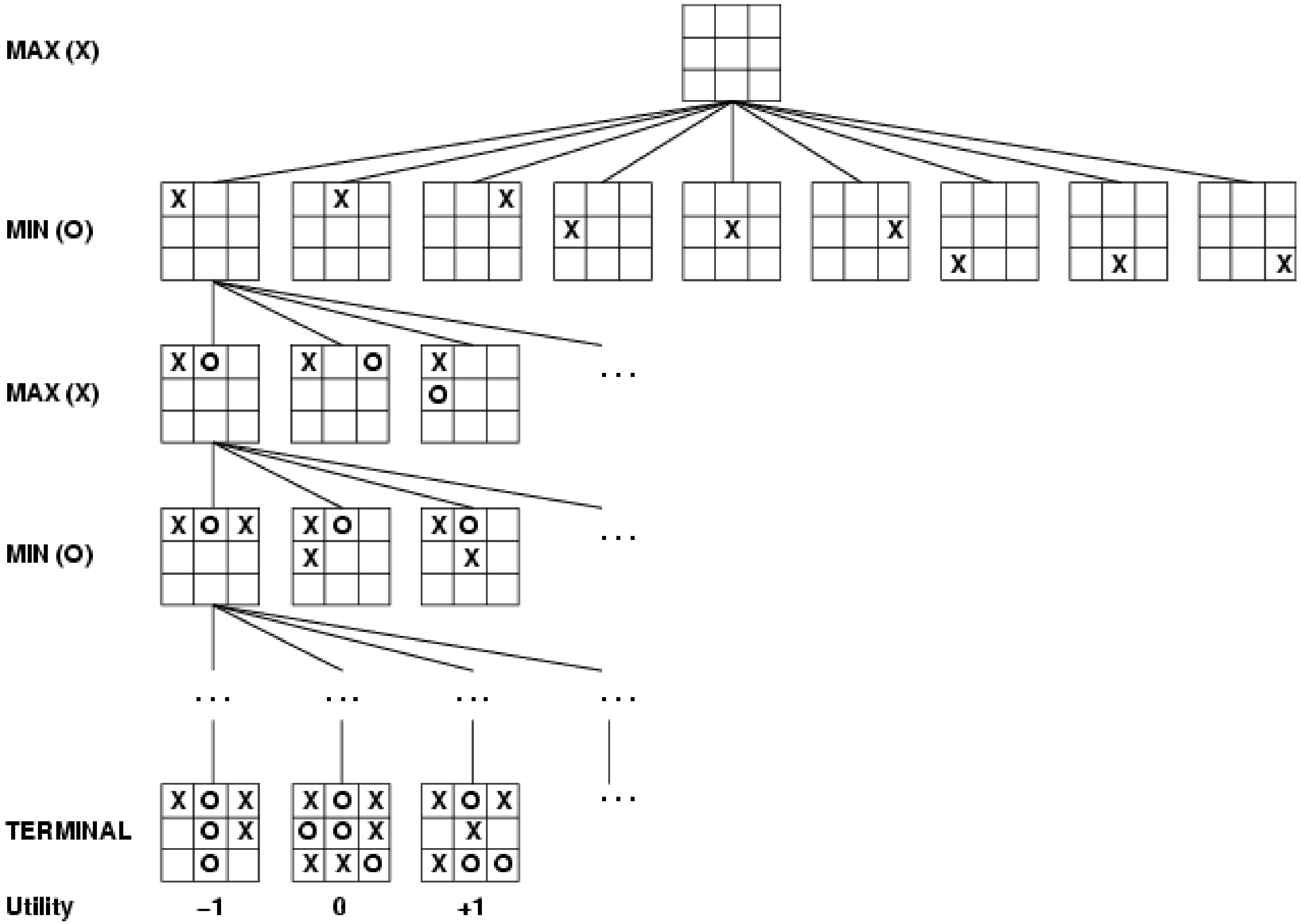
Games vs. search problems

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 (adding up to 0)

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…

Minimax value

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

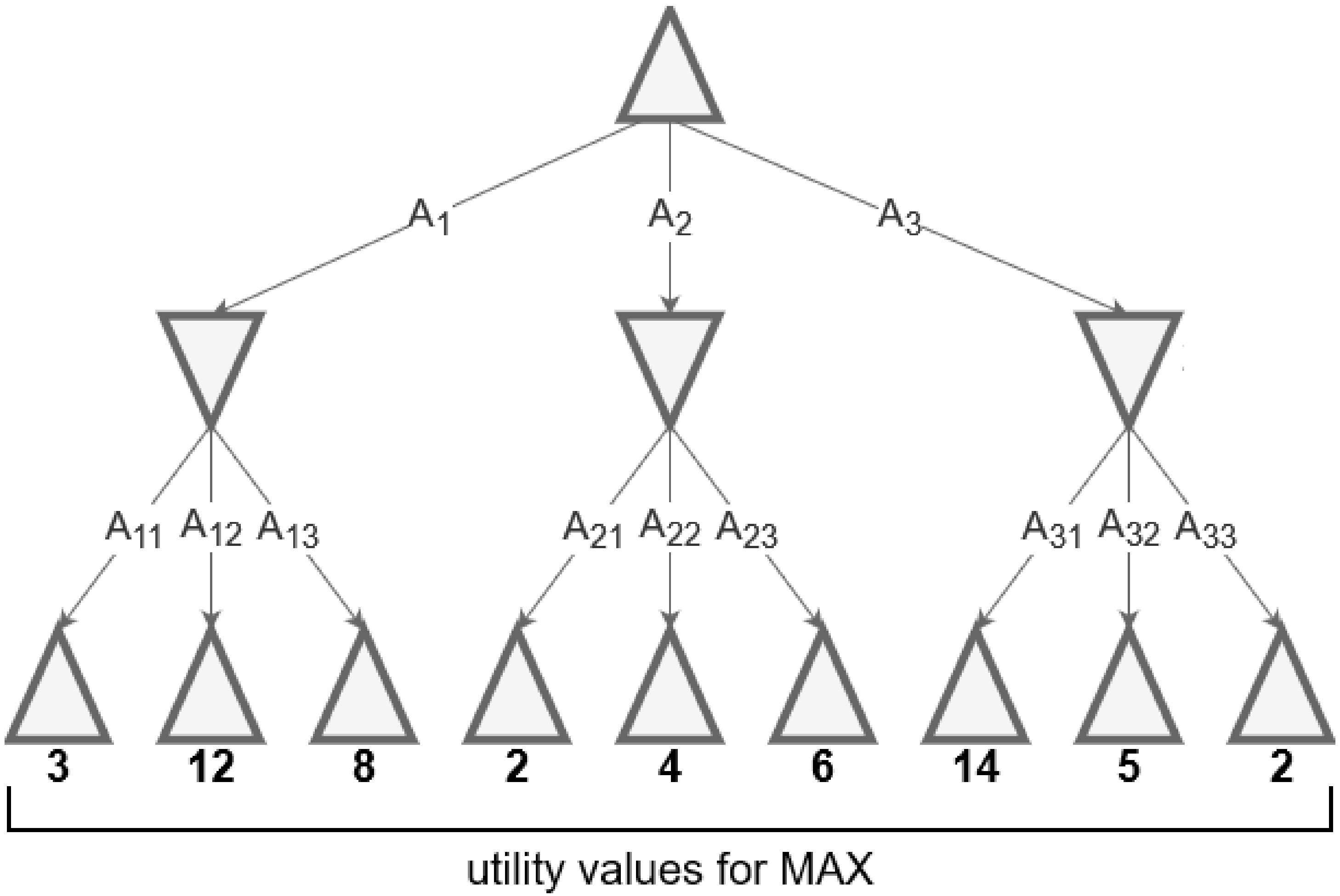
$$
\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}
$$

Minimax

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest minimax value

= best achievable payoff against best play

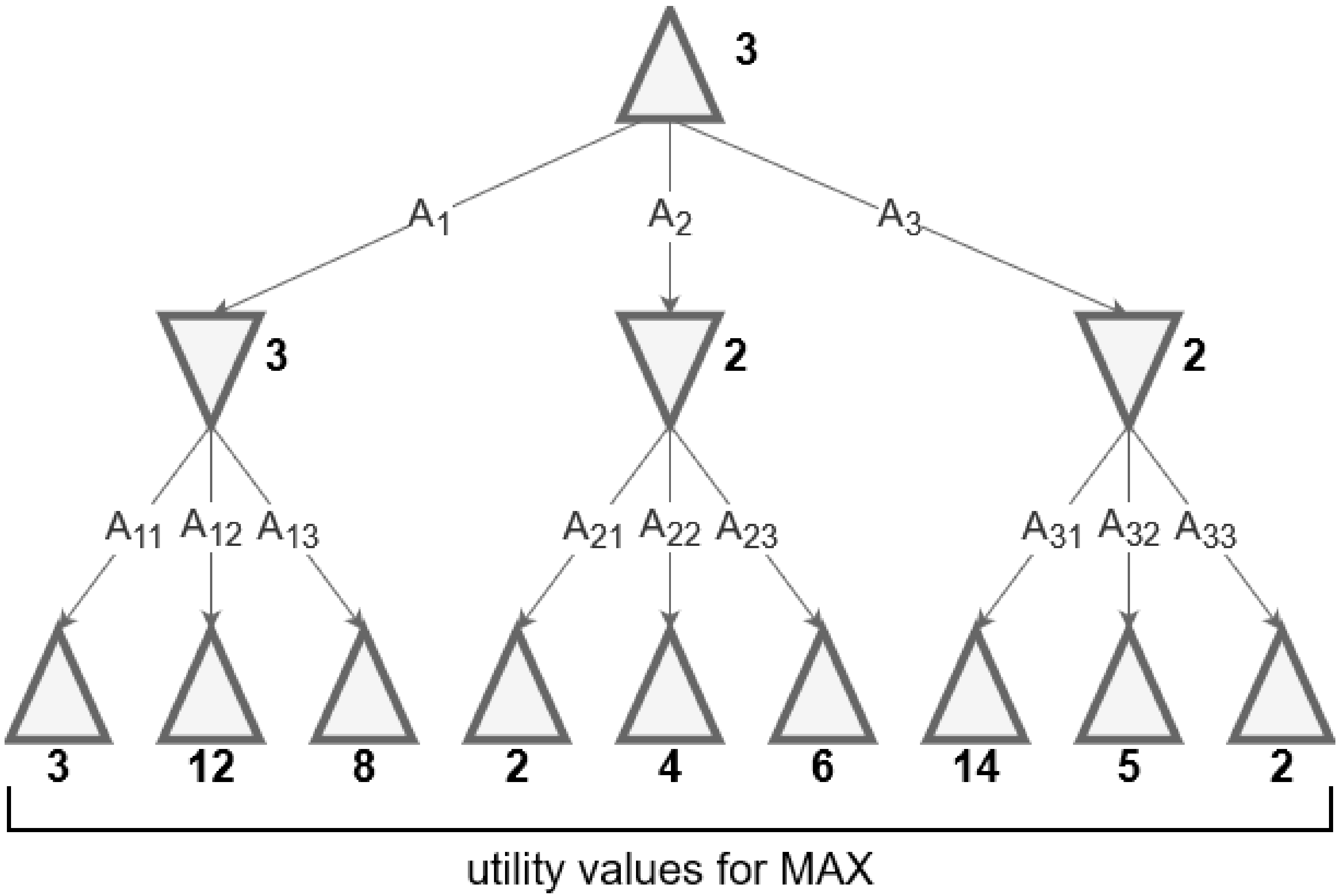


Minimax

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest minimax value

= best achievable payoff against best play



Minimax algorithm

Idea:

- ➢ Proceed all the way down to the leaves of the tree
- ➢ then minimax values are backed up through tree

```
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
```

Properties of Minimax

Complete?

Time complexity?

Space complexity?

Optimal?

Properties of Minimax

Complete?
Yes (if tree is finite)

Time complexity?

Space complexity?

Optimal?

Properties of Minimax

Complete?
Yes (if tree is finite)

Time complexity?
$O(b^m)$

Space complexity?

Optimal?

Properties of Minimax

Complete?
Yes (if tree is finite)

Time complexity?
$O(b^m)$

Space complexity?
$O(bm)$

Optimal?

Properties of Minimax

Complete?
Yes (if tree is finite)

Time complexity?
$O(b^m)$

Space complexity?
$O(bm)$

Optimal?
Yes (against an optimal opponent)

Complexity

For chess, $b \approx 35$, $m \approx 100$ (average ≈ 40) for "reasonable" games

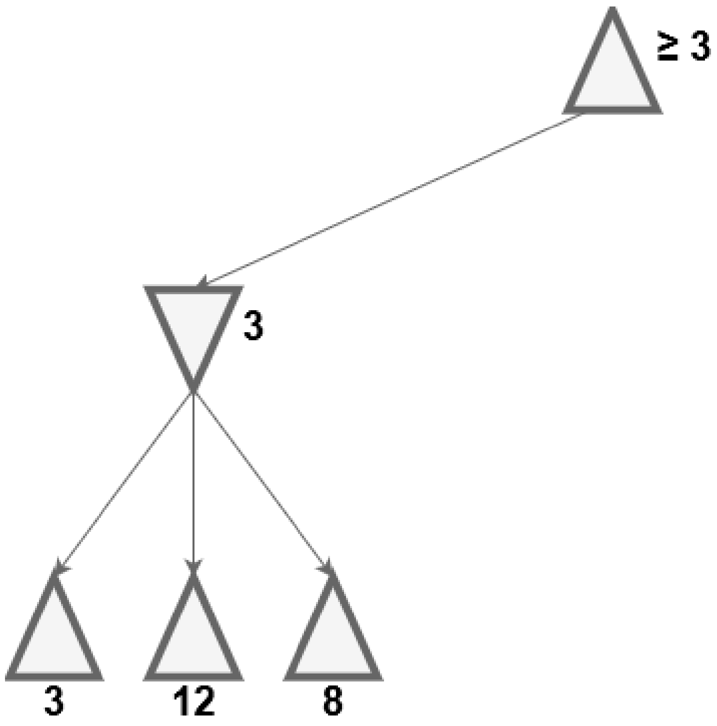
- ➢ exact solution completely infeasible!
- ➢ would like to eliminate (large) parts of game tree

$$35^{40} = 5.791 \cdot 10^{61}$$

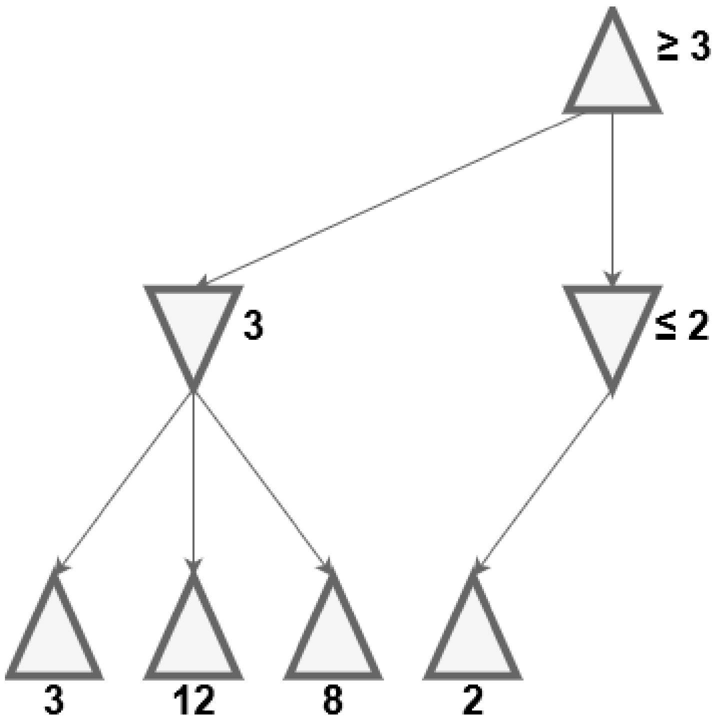
$$35^{100} = 2.552 \cdot 10^{154}$$

α-β pruning

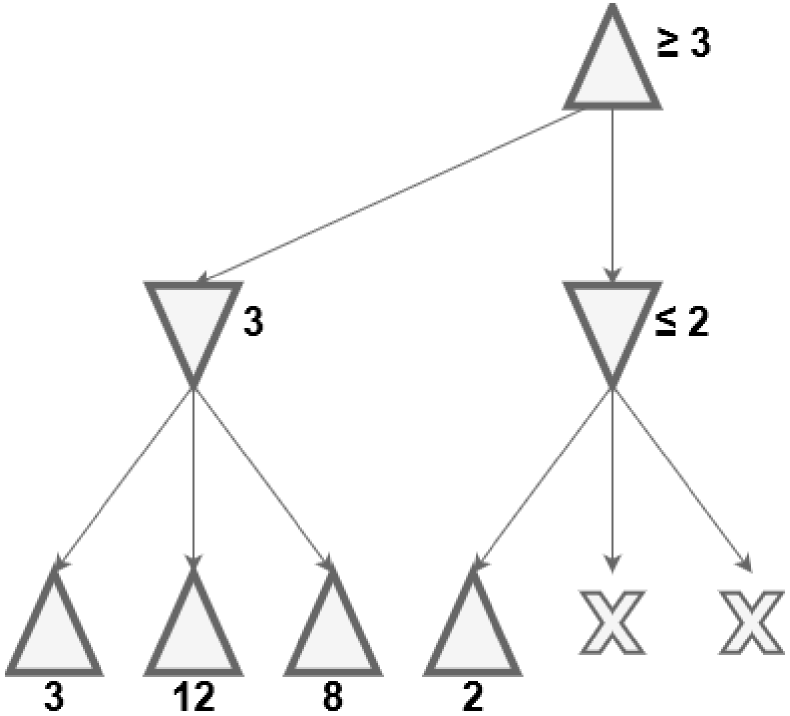
α-β pruning example



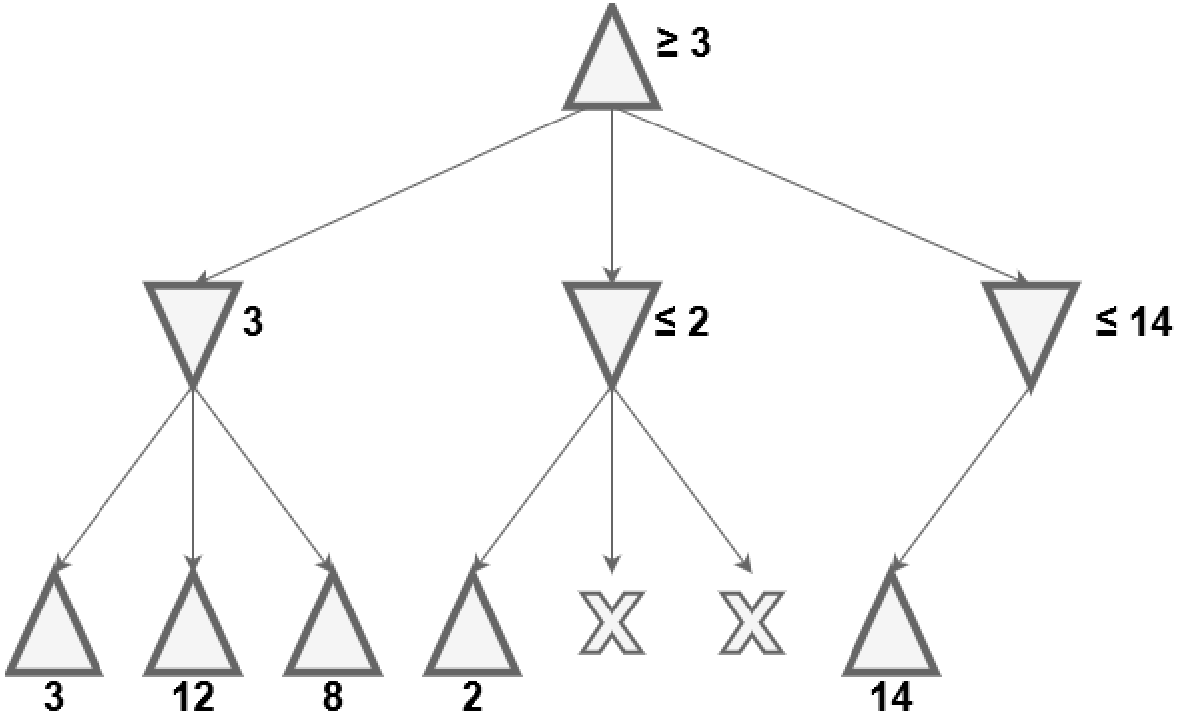
α-β pruning example



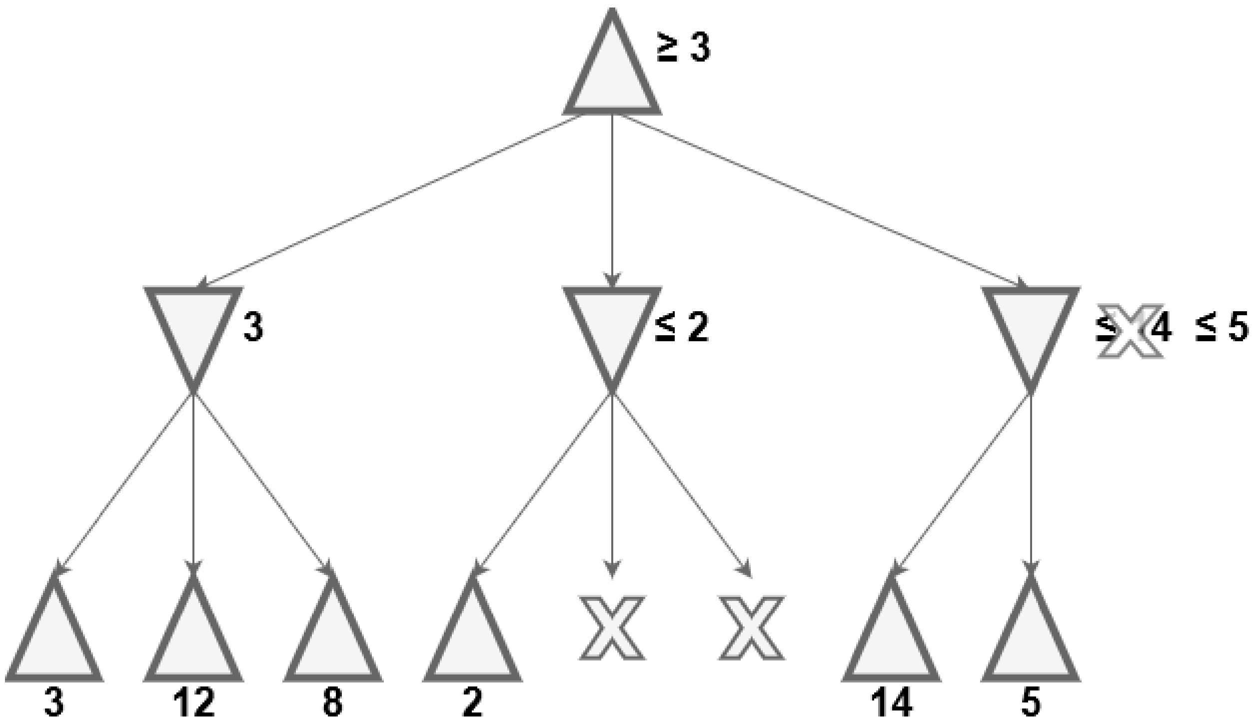
α-β pruning example



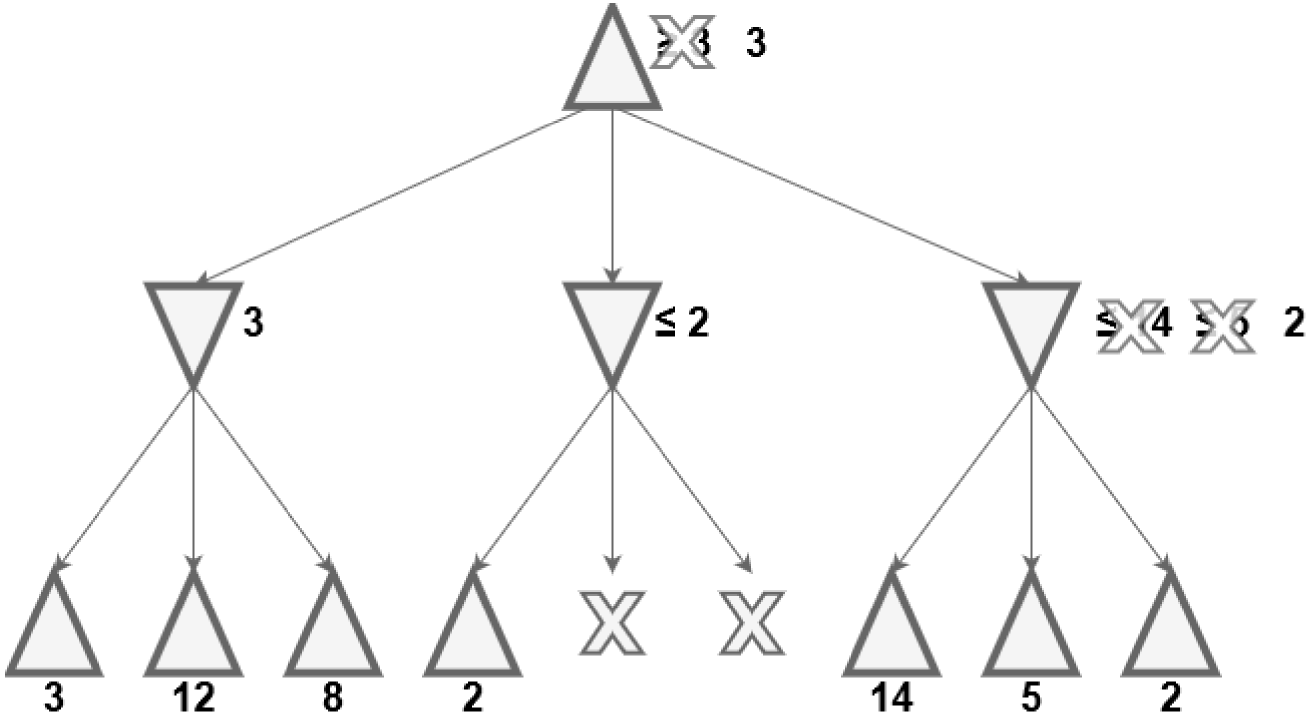
α-β pruning example



α-β pruning example



α-β pruning example



α-β pruning example

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

Let pruned leaves have values u and v,

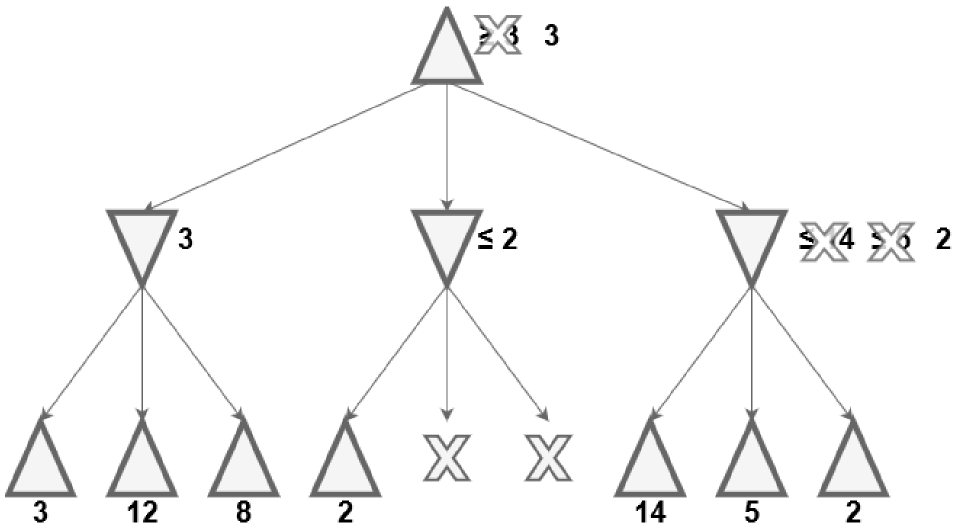
MINIMAX(root) = max(min(3,12,8), min(2,u,v), min(14,5,2))

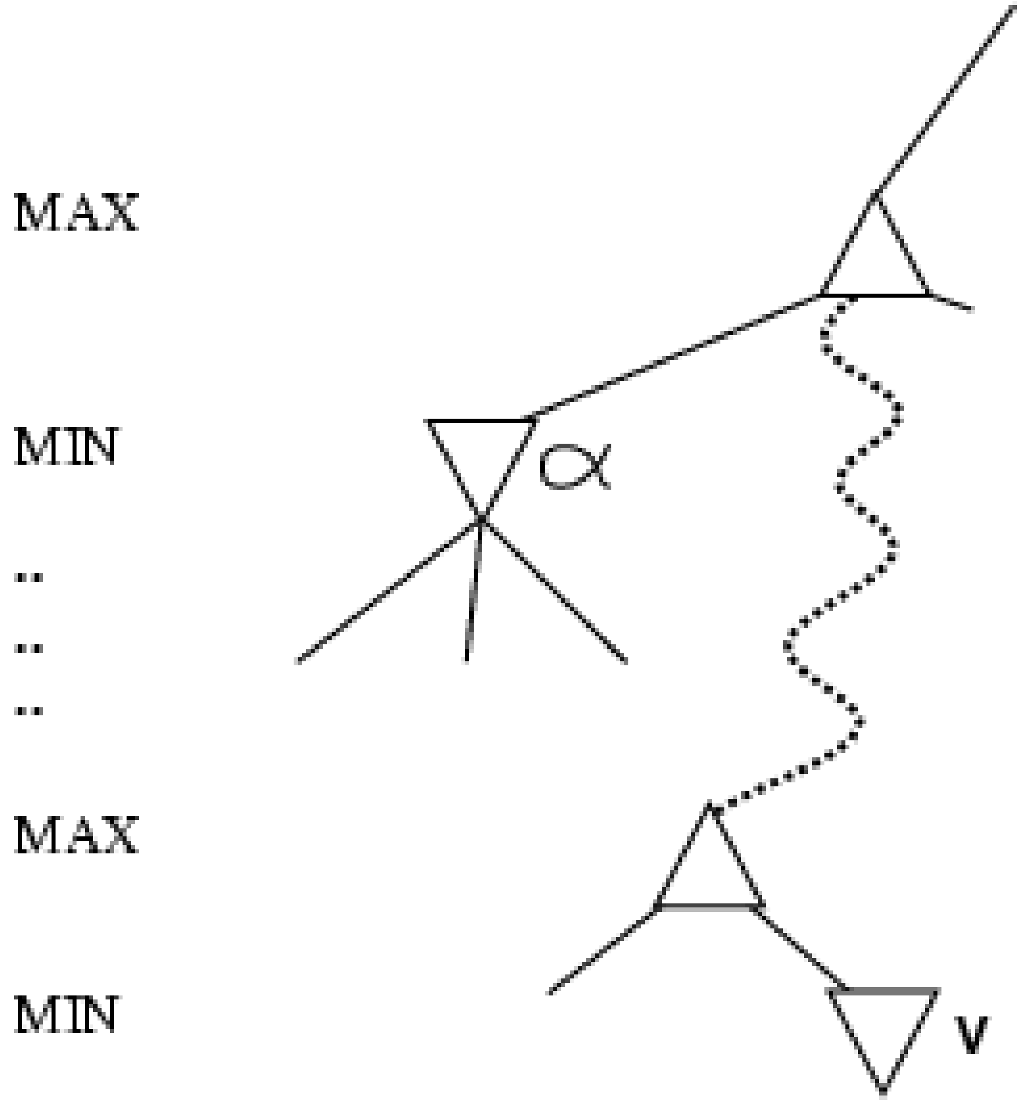
= max(3, min(2,u,v), 2)

= max(3, z, 2) where z ≤ 2

= 3

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
→ prune that branch

β is defined symmetrically for MIN

The *α*-*β* algorithm

α is value of the best i.e. **highest**-value choice found so far at any choice point along the path for MAX

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

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

Complexity of α-β

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

Good move ordering improves effectiveness of pruning

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

- branching factor goes from b to $\sqrt{b}$
- **doubles solvable depth** of search compared to minimax

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

Resource limits

Resource limits

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

- ➢ 10^6 nodes per move
- ➢ $b^m = 10^6$
- ➢ For $b = 35 \rightarrow 35^4 = 1.5 \cdot 10^6 \rightarrow$ so $m \approx 4$

4-ply lookahead is a hopeless chess player!

- ◦ 4-ply ≈ human novice
- ◦ 8-ply ≈ typical PC, human master
- ◦ 12-ply ≈ Deep Blue, Kasparov

Standard approach

Cutoff test

e.g., depth limit (perhaps add quiescence search, which tries to search interesting positions to a greater depth than quiet ones)

Evaluation function

= estimated desirability of position

Standard approach

MinimaxCutoff is identical to *MinimaxValue* except:

1. *TERMINAL-TEST* is replaced by *CUTOFF*
2. *UTILITY* is replaced by *EVAL*

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

Evaluation functions

Often a linear weighted sum of features

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

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

Chess example

- queen = 1, king = 2, etc.
- f_i = number of pieces of type *i* on board
- w_i = value of the piece of type *i*

Deterministic games in practice

https://www.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts/

Checkers

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

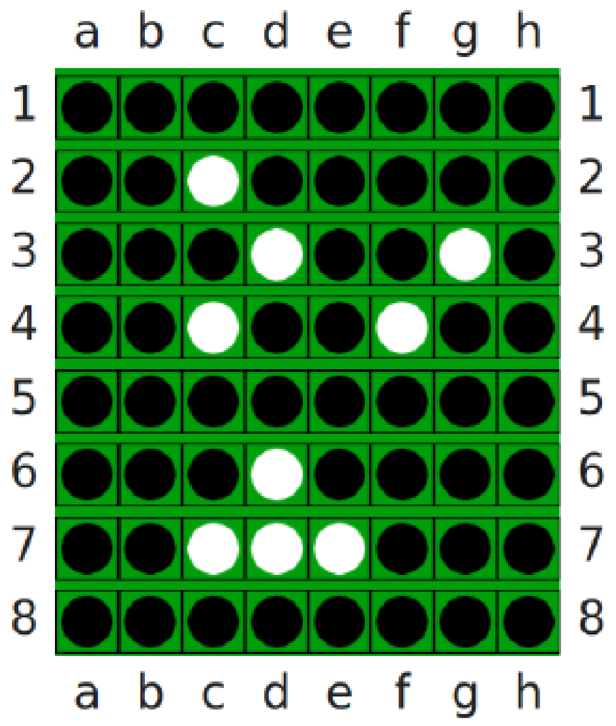
http://jonathanschaeffer.blogspot.com/2012/08/chinook-twenty-years-later.html

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.



https://skatgame.net/mburo/log.html

Chess

Deep Blue defeated human world champion Garry 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.

https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)

Modern Chess

Stockfish

- Uses and advanced version of α-β pruning among other algorithms.
- Recently added a simple neural network in its evaluation.
  - Improved by 100+ ELO points since.
- Analyses 10^8 positions per second (half when using the neural network).

AlphaZero (successor of AlphaGo Zero)

- Based on Monte Carlo tree search, deep neural networks and self-play.
- Analyses 80,000 positions per second.
- Defeated Stockfish with 28W-72D-0L in 2016.

Leela Zero

- Released 2017 with ideas from AlphaGo Zero's paper.
- Believed to have surpassed AlphaZero.
- Neck to neck with modern Stockfish, losing narrowly to it in the last 3 TCEC superfinals.

Go

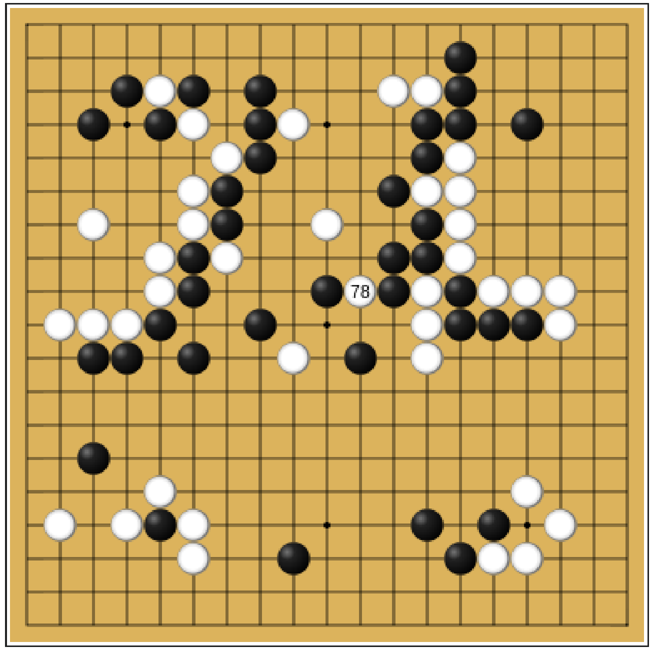
Human champions used to refuse to compete against computers, because they were are too bad.

In Go, *b > 300*, so most programs use pattern knowledge bases to suggest plausible moves.

In 2015 AlphaGo became the first computer program to beat a human professional Go player (Fan Hui) without handicap.

In 2016 AlphaGo beat world's #2 Lee Sedol 4-1.

Evolved into AlphaGo Zero (without human datasets), then AlphaZero, and more recently MuZero (model-free) .



Game 4, Lee Sedol (white) v. AlphaGo (black).
First 78 moves

https://en.wikipedia.org/wiki/Lee_Sedol

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.

Modern AI demonstrating superhuman performance.