Artificial Intelligence

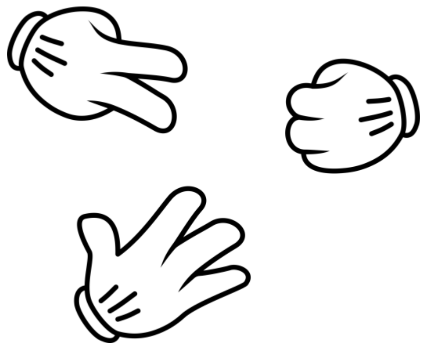
Steve James

Game Theory & Adversarial Search

What is game theory?

- Field involving games, answering such questions as:
  - How should you play games?
  - How do most people play games?
  - How can you create a game that has certain desirable properties?

What is a game?



What is a game?

- Game is defined by:
  - Initial state
  - $Player(s)$: decision-making entities
  - $Actions(s)$: available actions
  - $Result(s, a)$: successor function or transition model
  - $Terminal(s)$: is the game over/state terminal
  - $Utility(s, p)$: the value for the game ending in s for player p

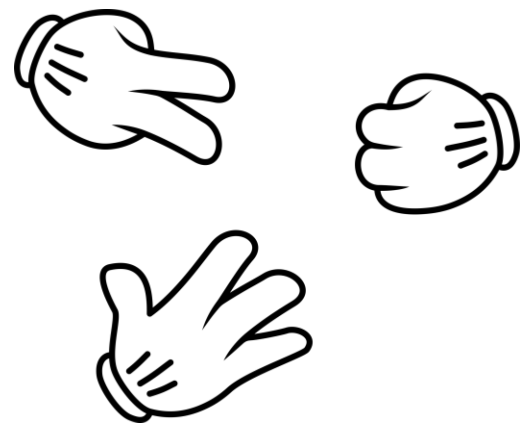
Why study game theory in AI?

- Making good decisions ⊆ AI
- Making good decisions in games ⊆ Game Theory
- AI often created for situations that can be thought of as games

Types of games

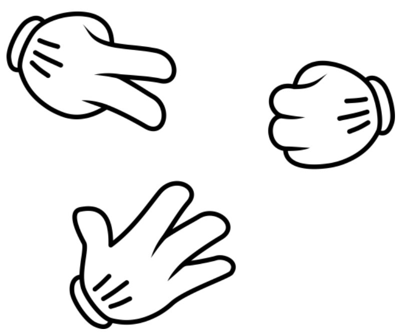
- Sequential

- Simultaneous



Types of games

- Constant-sum



- Variable-sum

Classic game theory

- 2-player, one-turn, simultaneous-move games

	R	P	S
R	½, ½	0, 1	1, 0
P	1, 0	½, ½	0, 1
S	0, 1	1, 0	½, ½

Strategy

- Strategy: A specification of what to do in every single non- terminal state of the game
  - Functions from states to (probability distributions over) legal actions
    - Pure vs. Mixed

- Examples:
  - Trading: I’ll accept an offer of R2m or higher, but not lower
  - Chess: Full lookup table of moves and actions to make

Best response

- What’s the best strategy in rock-paper-scissors?
  - Depends on what the other player is doing!
  - If we knew it, then we could choose the best strategy (optimisation)
  - But we don’t know what they want!
    - How to reason when we don’t know opponent’s strategy

Dominated strategies

- Strategy s is dominated by s^* if s^* always gives higher payoff

	C	D
C	3, 3	0, 5
D	5, 0	1, 1

Iterated dominance

	L	C	R
U	6, 1	1, 0	6, 2
M	1, 4	0, 5	5, 5
D	3, 4	4, 3	2, 0

Iterated Dominance

- Iterated Elimination of Dominated Strategies (IEDS)
  - Won’t always produce a unique solution
  - Common Knowledge of Rationality (CKR)
  - “Faithful Approach”

Conservative approach: Maximin

- Ensures best worst-case scenario

	L	C	R
U	6, 1	1, 0	6, 2
M	1, 4	0, 5	5, 5
D	3, 4	4, 3	2, 0

Nash equilibrium

- Strategy profile: specification of strategies for all players
- Nash equilibrium: strategy profile such that players are mutually best-responding
- In other words: From a NE, no player can do better by switching strategies alone

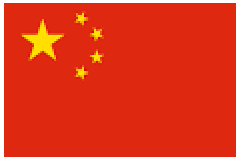
Stag hunt

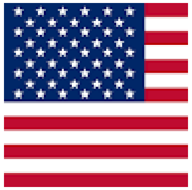
- Strategy s is dominated by s^* if s^* always gives higher payoff

	B	S
B	2, 2	2, 0
S	0, 2	3, 3

Also: play B with prob $\frac{1}{3}$ is an NE!

Diplomacy/Society





	E	R
E	Economy ↑ planet ↓	Economy ↓ planet ↓
R	Economy ↓ planet ↓	Economy → planet ↑

Properties

- There is always at least one NE
  - Might be mixed

- If IEDS produces unique solution, it is a NE

Now

- Let's consider finding pure strategies in
  - Sequential
  - Alternating
  - Constant-sum (zero-sum)
  - Many-turn
  - Perfect information

Games are big

- Tic-tac-toe $\sim 10^3$
- Connect Four $\sim 10^3$
- English draughts $\sim 10^{23}$
- Othello $\sim 10^{28}$
- Chess $\sim 10^{44}$
- Shogi $\sim 10^{71}$
- # atoms in observable universe $\sim 10^{82}$
- Twixt $\sim 10^{140}$
- Go (19x19 board) $\sim 10^{170}$

Size of state space
(reachable states)

Zero-sum games

Can extend to n

- Assume two players, competitive
  - Player 1 wins, player 2 loses and vice versa
- Game is defined by:
  - Initial state
  - $Player(s)$: whose turn it is
  - $Actions(s)$: available actions
  - $Result(s, a)$: successor function or transition model
  - $Terminal(s)$: is the game over/state terminal
  - $Utility(s, p)$: the value for the game ending in s for player p
- Zero sum game: sum of utilities for all players is constant: e.g. Win = +1, Draw = 0, Loss = -1

Two player, zero-sum

- Two players, MAX and MIN
- We are MAX, try to maximise utility
- Opponent is MIN, tries to minimise utility
- Denote $V(s)$ as utility at a given state
  - Utility is known at terminal states
- Start at root node, expand tree
  - Players alternate turns
  - At level 0, us to play. Level 1, them to play, etc
- We want to compute optimal play, assuming our opponent is also optimal

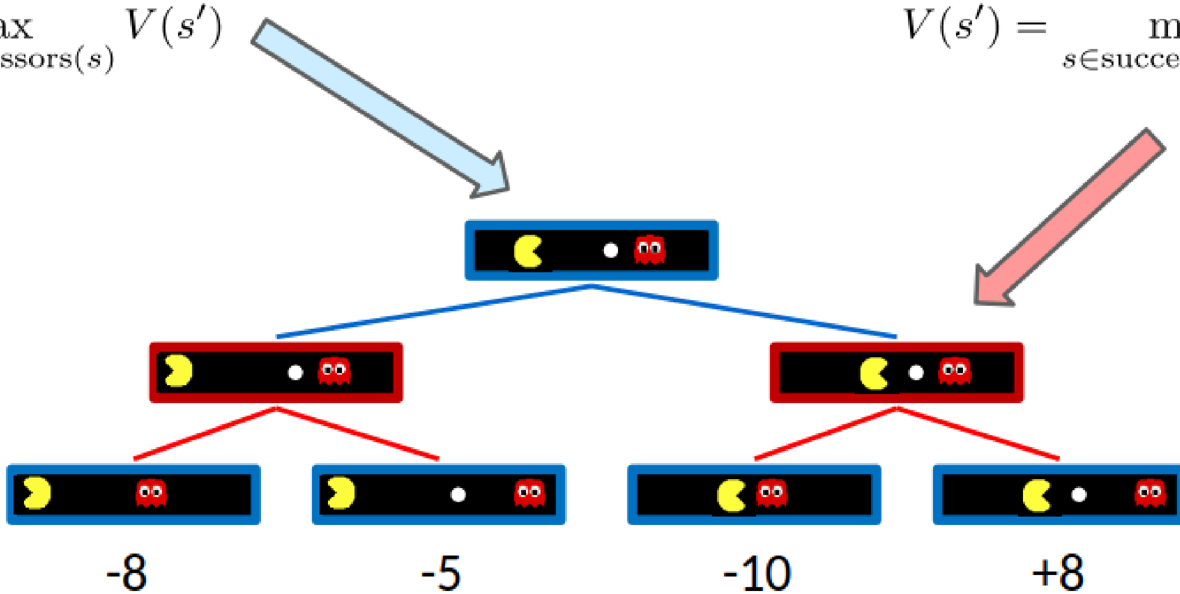
Each level is called a ply

Example

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent's Control:

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$



Terminal States:

$$V(s) = \text{known}$$

Calculating minimax

- Want to calculate $V(s)$ for all s
- If s is terminal, use utility function directly
- else if player to play is MAX:
  - Value is best maximising value at state
- else player to play is MIN:
  - Value is best minimising value at state

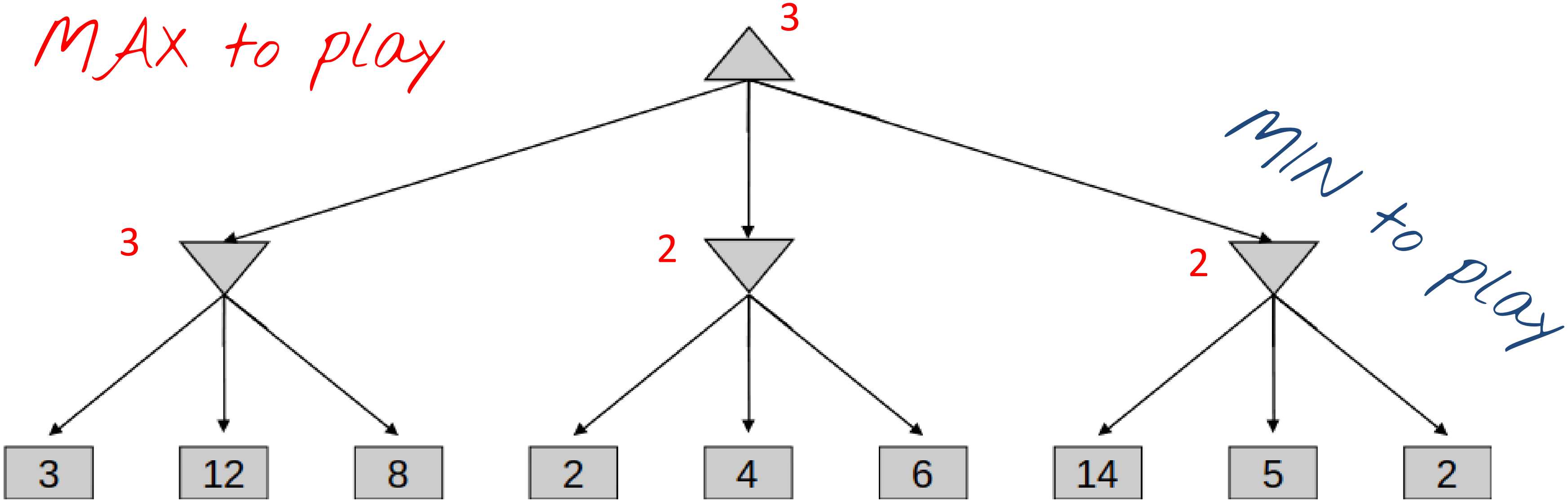
```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

$$
\textsc{Minimax}(s) =
\begin{cases}
\textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\
\max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\
\min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min}
\end{cases}
$$

Example



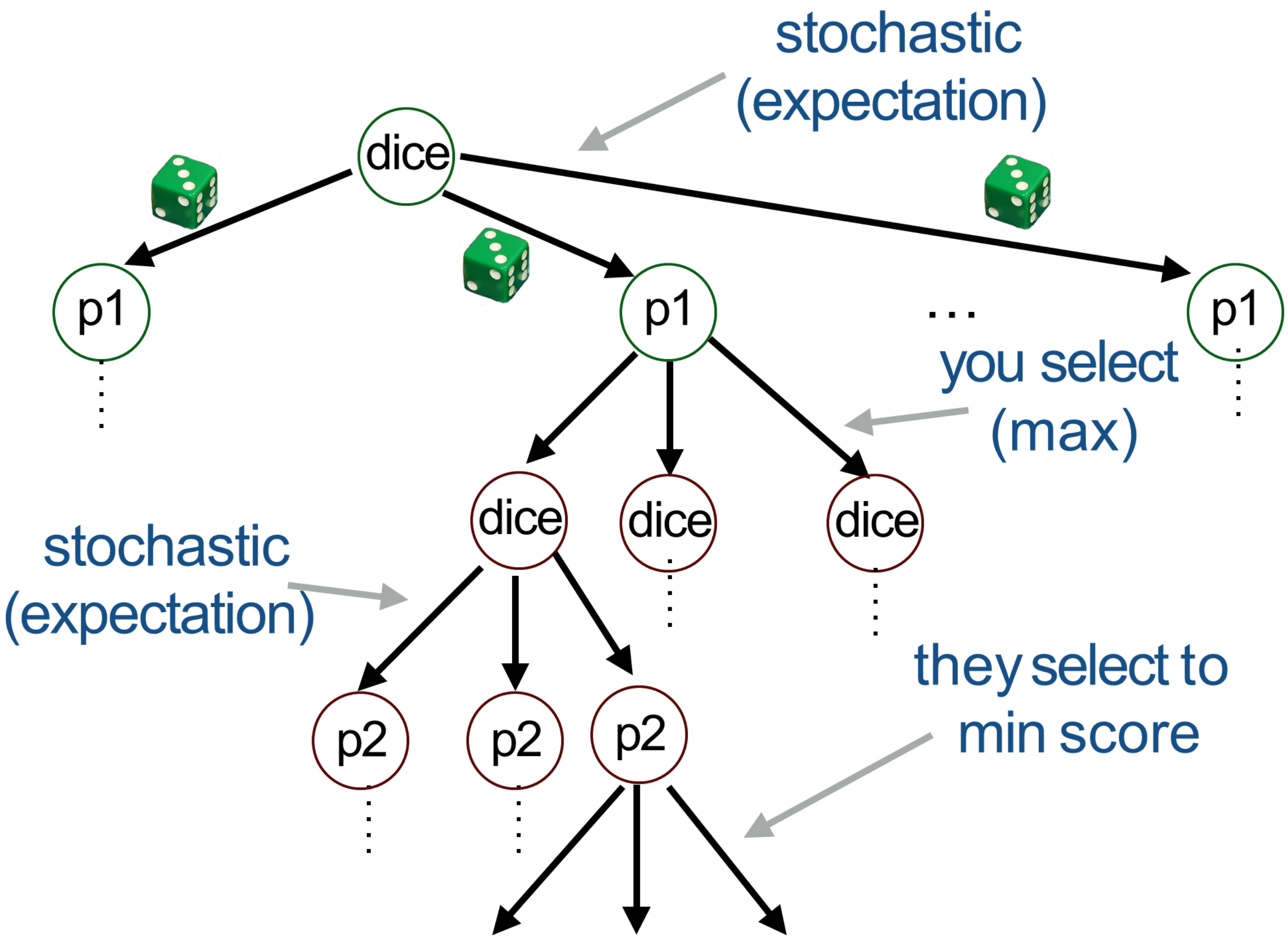
Game of chance

- What if there is stochasticity in the game?

Stochasticity

- Be aware of who is choosing at each level
- Sometimes it is:
  - You
  - Opponent
  - Random generator
- We already have min/max nodes
  - So now add chance node

Expectimax



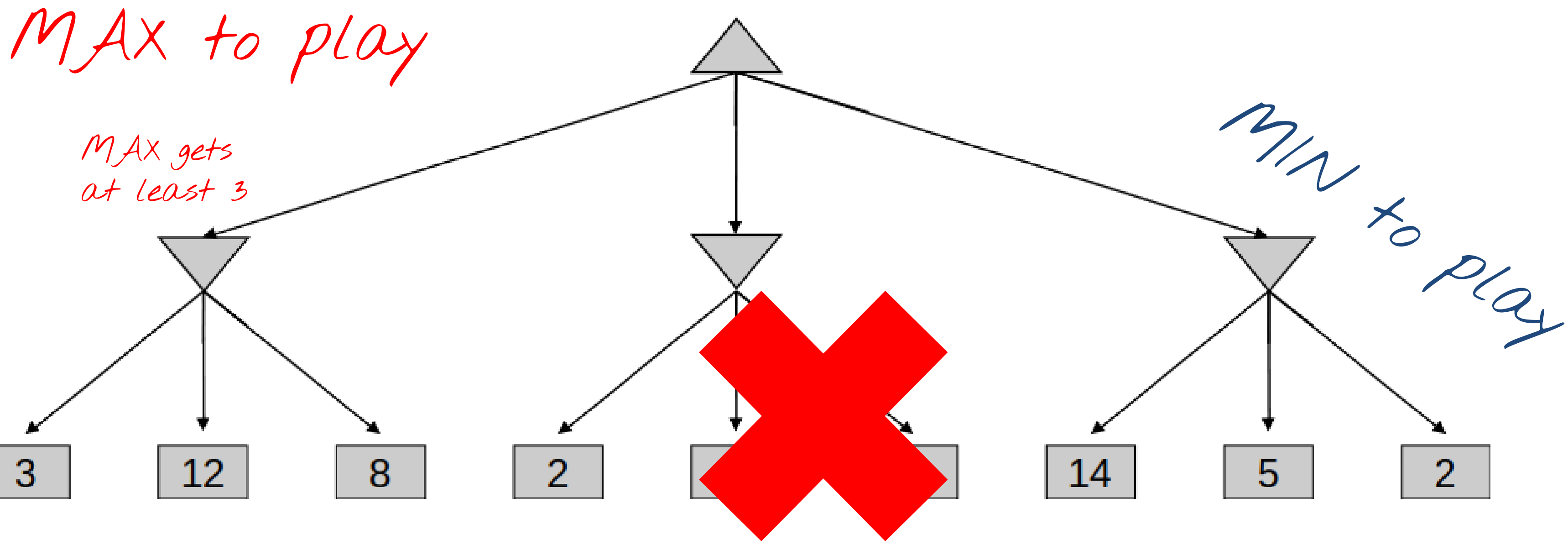
stochastic
(expectation)
dice
p1
p1
...
p1
you select
(max)
dice
dice
dice
stochastic
(expectation)
they select to
min score
p2
p2
p2

Minimax properties

- Like DFS:
  - Time: $O(b^m)$
  - Space: $O(bm)$

- But instead of searching for single goal, we need to exhaustively try everything!
  - And we only get value at leaf nodes
- Chess, for e.g., $b \sim 20, m \sim 70$
  - Exact solution infeasible
  - So what do we do?

Pruning the tree



$\alpha\beta$-pruning

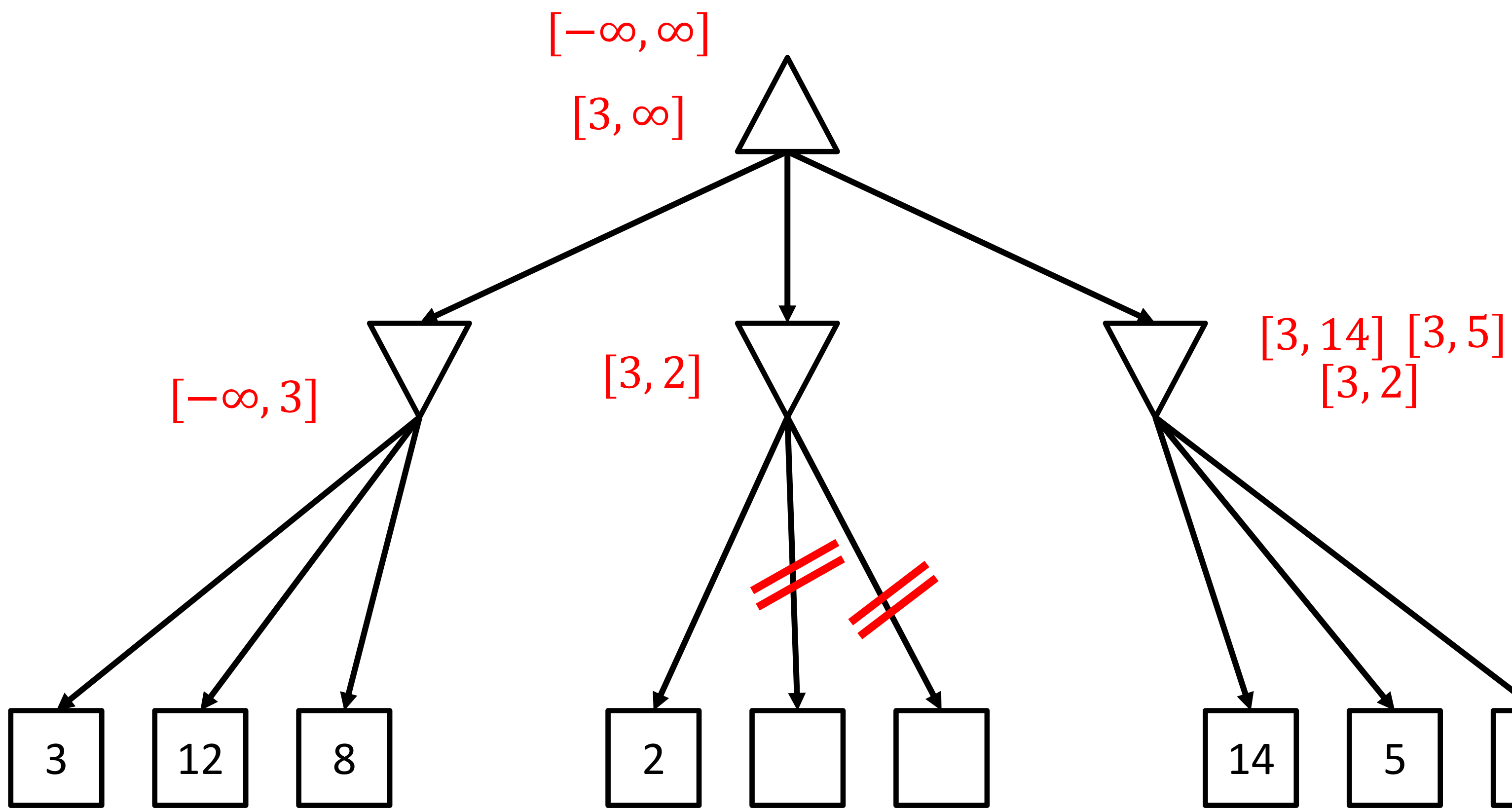
- α – minimum score MAX is guaranteed of
- β – maximum score MIN is guaranteed of

Worst case guarantees

- If at a given min node, $\beta < \alpha$
  - Then MIN can guarantee a score that makes MAX sad ☹
  - So MAX will never go down this road
  - No need to expand rest of node's children!

Could be pruning entire subtrees!

- Symmetric argument for other way around
- If children are expanded in optimal order, complexity is halved: $O(b^{m/2})$



α – minimum score MAX is guaranteed of

β – maximum score MIN is guaranteed of

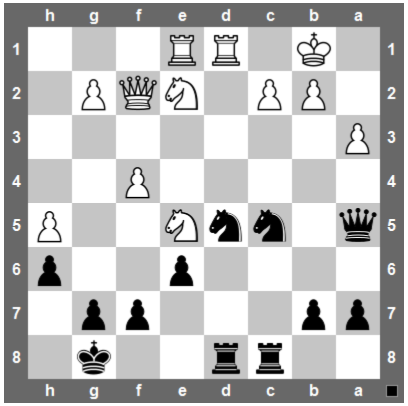
Depth-limited search

- Even with pruning, can’t reach leaf nodes in real games
- So must limit depth of search
  - Must replace utility function with estimate (like heuristic in A*)
  - No longer optimal

- More plies = better performance
- Given time budget, use IDS!

Evaluation functions

- Estimate of utility of non-terminal state
- Ideally: want actual minimax value of state
  - But this is unknown
- In practice: use domain knowledge
  - E.g. $\text{eval}(s) = w_1(|pawns_w - pawns_b|) + w_2(|bishops_w - bishops_b|) + \cdots$



- Tradeoff between complexity vs depth
  - More complex eval function may be more accurate, but longer to compute → less time to search deeper
    - Stockfish: fast eval function, huge depth
    - Komodo: slow, complex eval function, less depth

Other improvements

- Base algorithm of $\alpha\beta$ + IDS + eval function
- Transposition tables: stored previous states and their evals
- Aspiration windows: pretend that the $\alpha\beta$ window is smaller than it is
- Evaluation functions optimised from data (machine learning etc)
- Move ordering: try certain classes of moves first (e.g. captures, then regular moves)

Board games

- “ … board games are more or less done and it's time to move on.”