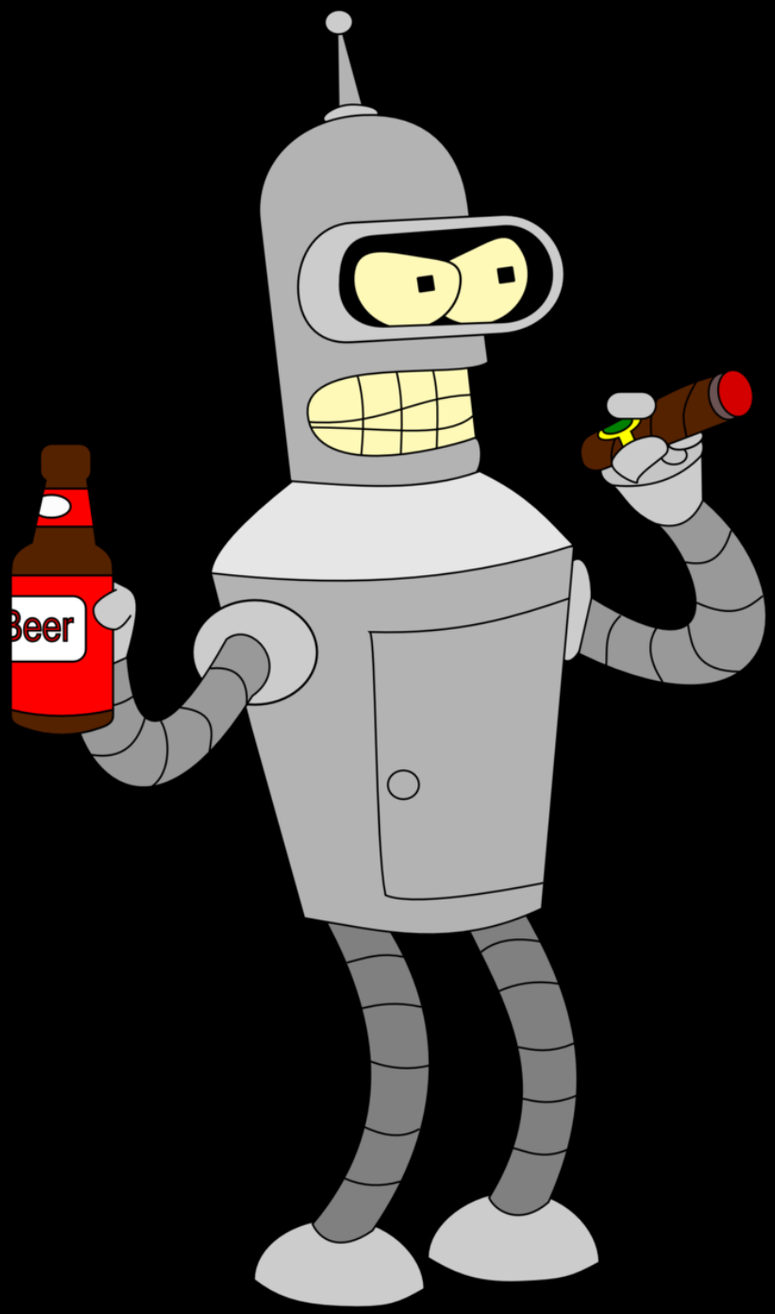
Intelligent agents that outperform human domain experts

Ian Tsybulkin - AI Ukraine 2016

Intelligent Agent



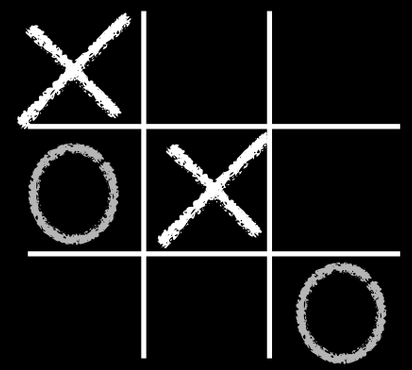
percepts

acts

Environment

State —> Action

Tic-Tac-Toe

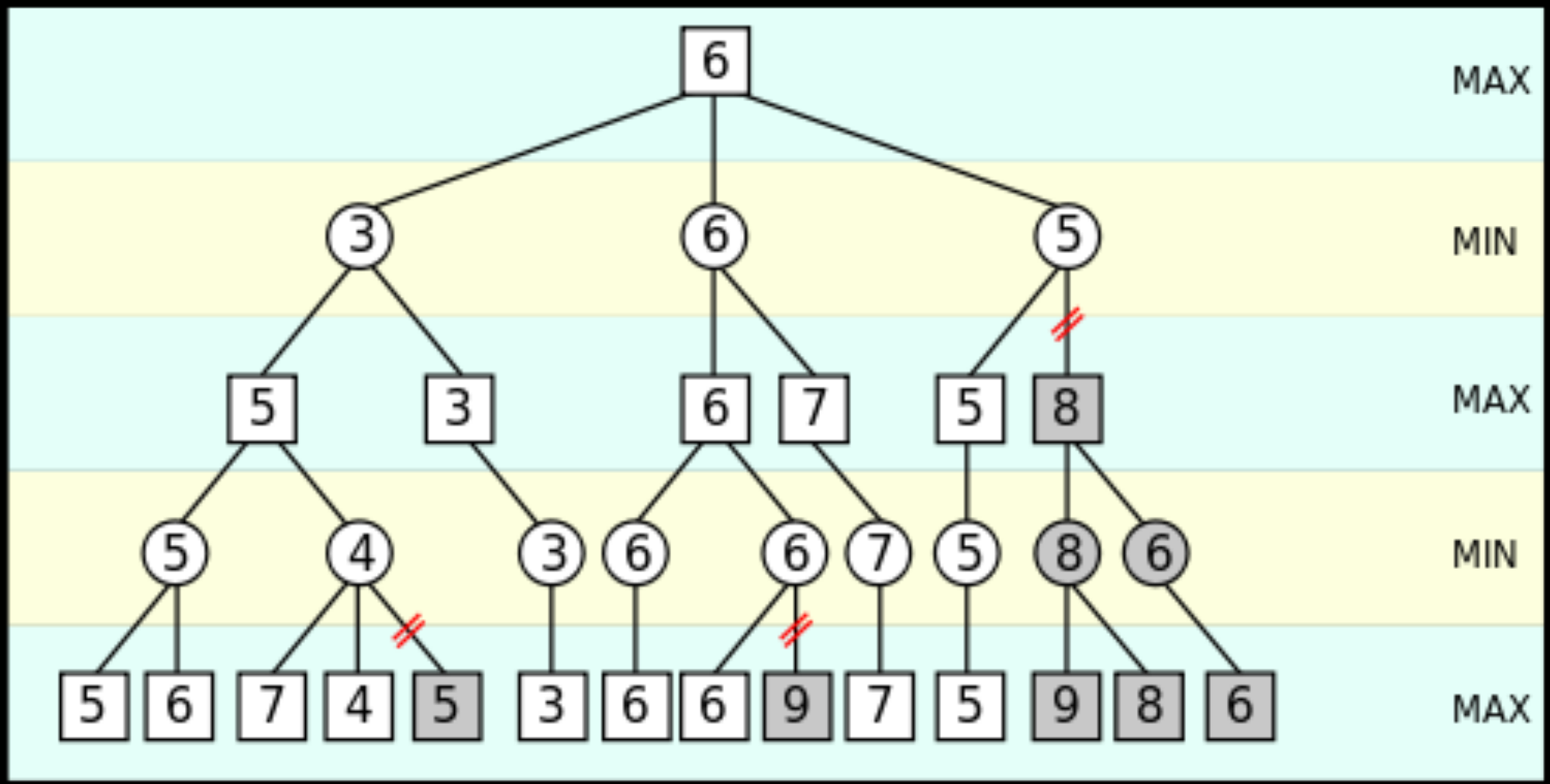
State: 

States:	Moves:
State 1	Move 1
State 2	Move 2
…	…
State N	Move N

Which move to choose

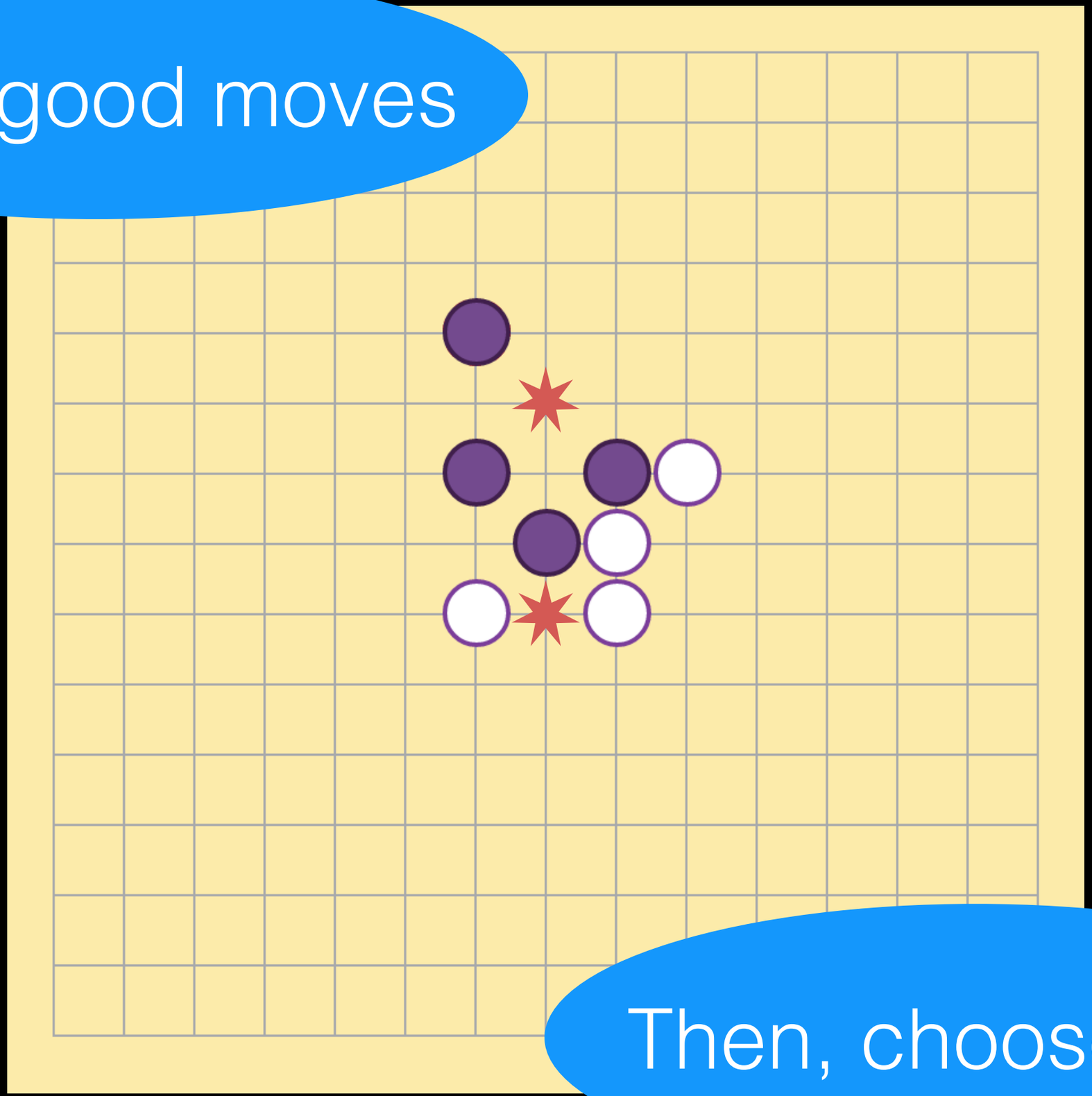
- Agent: State -> Action
- Traversing through the tree
- crafting sophisticated state evaluation function - Deep Blue
- Neural network as a non-liner function

Simple: search



erlangonxen.org/gomoku

Select good moves



Then, choose one!

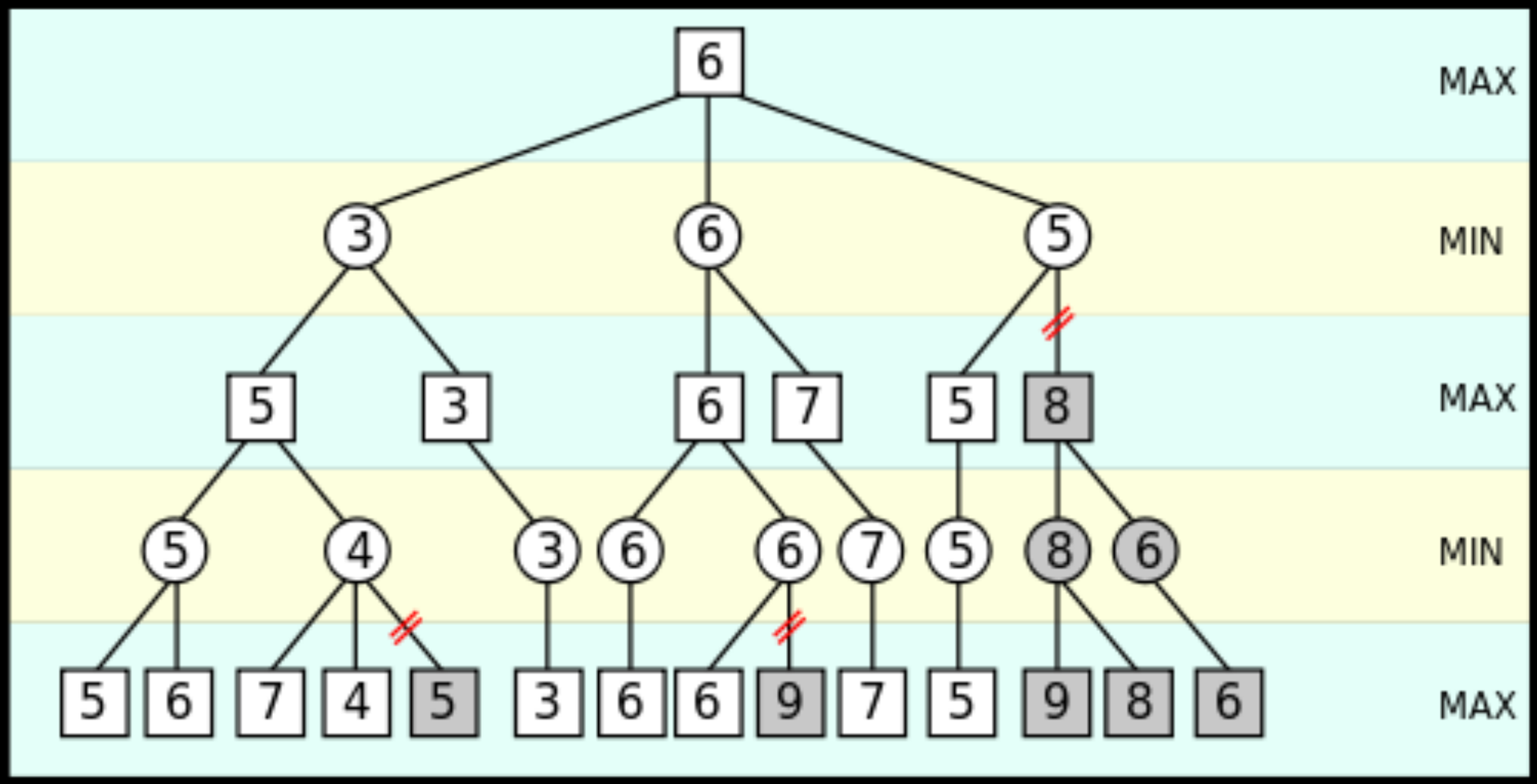
Simple and powerful

- Zero steps ahead
- State -> [Best actions] -> choose randomly
- Extremely time-efficient (like most people do)
- Agent can play against 1000s of people simultaneously using a single computer

Playing against experts

- to think a few steps ahead like a grandmaster
- to be able to evaluate a position accurately

Problem:
how to evaluate the position?

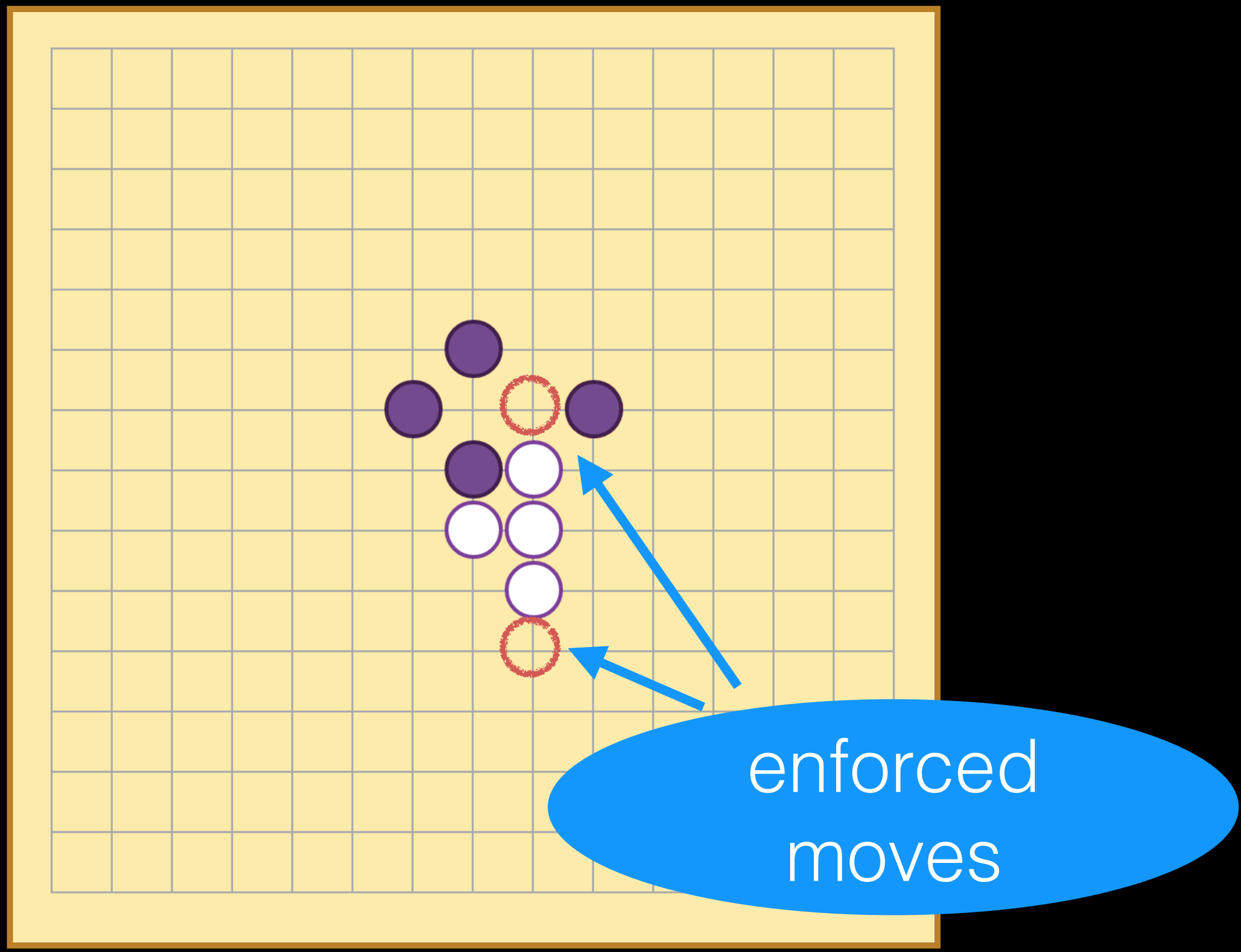


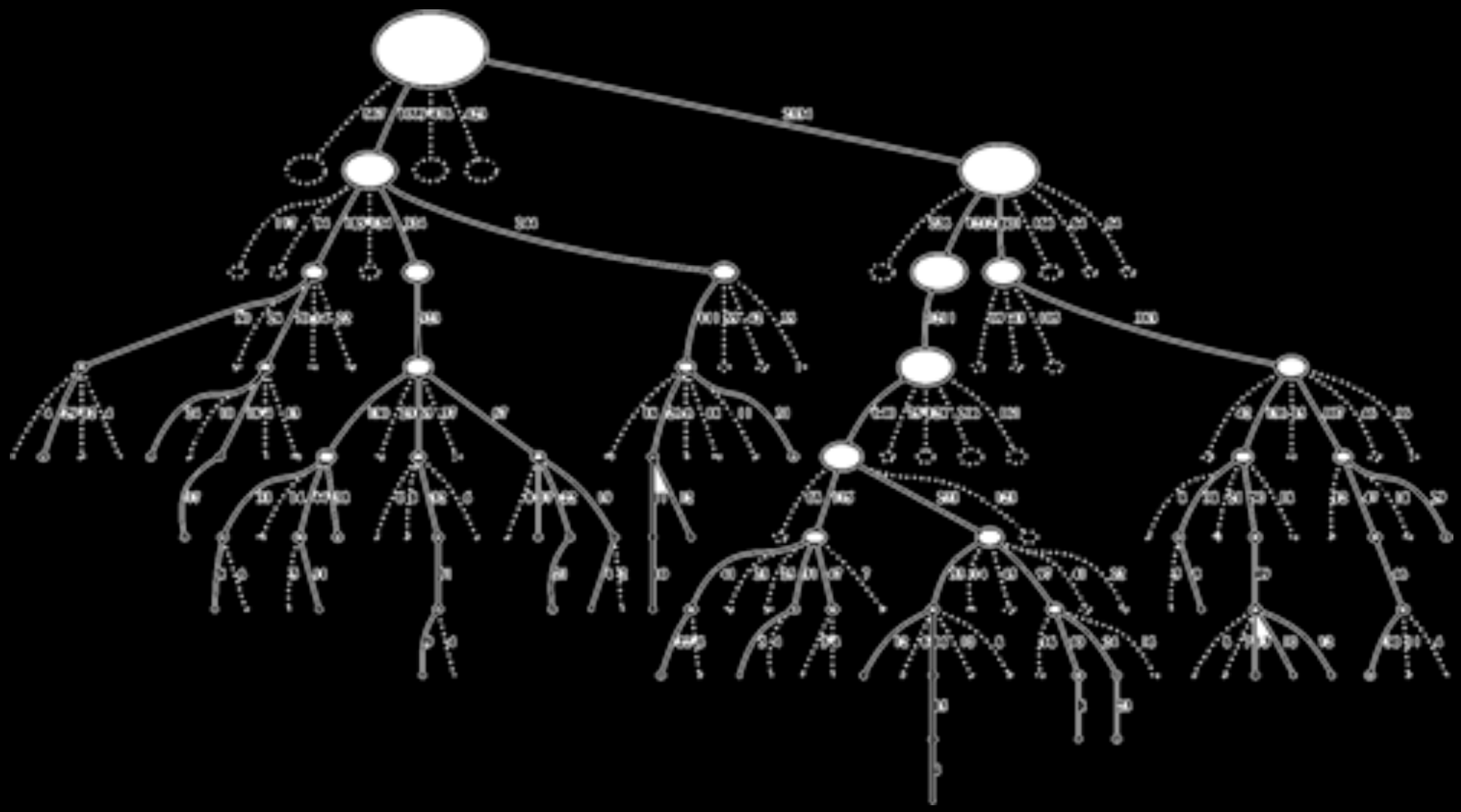
Advanced methods

- State -> [Actions]
- State -> Who has an advantage?
- 2 different functions (neural networks)

Advanced: traversing tree in a different depth

- for enforcing moves you need to look deeper than usually
- assign probabilities to all moves and traverse until a predefined threshold


enforced
moves



$$p(path_k) \sim p_{threshold}$$

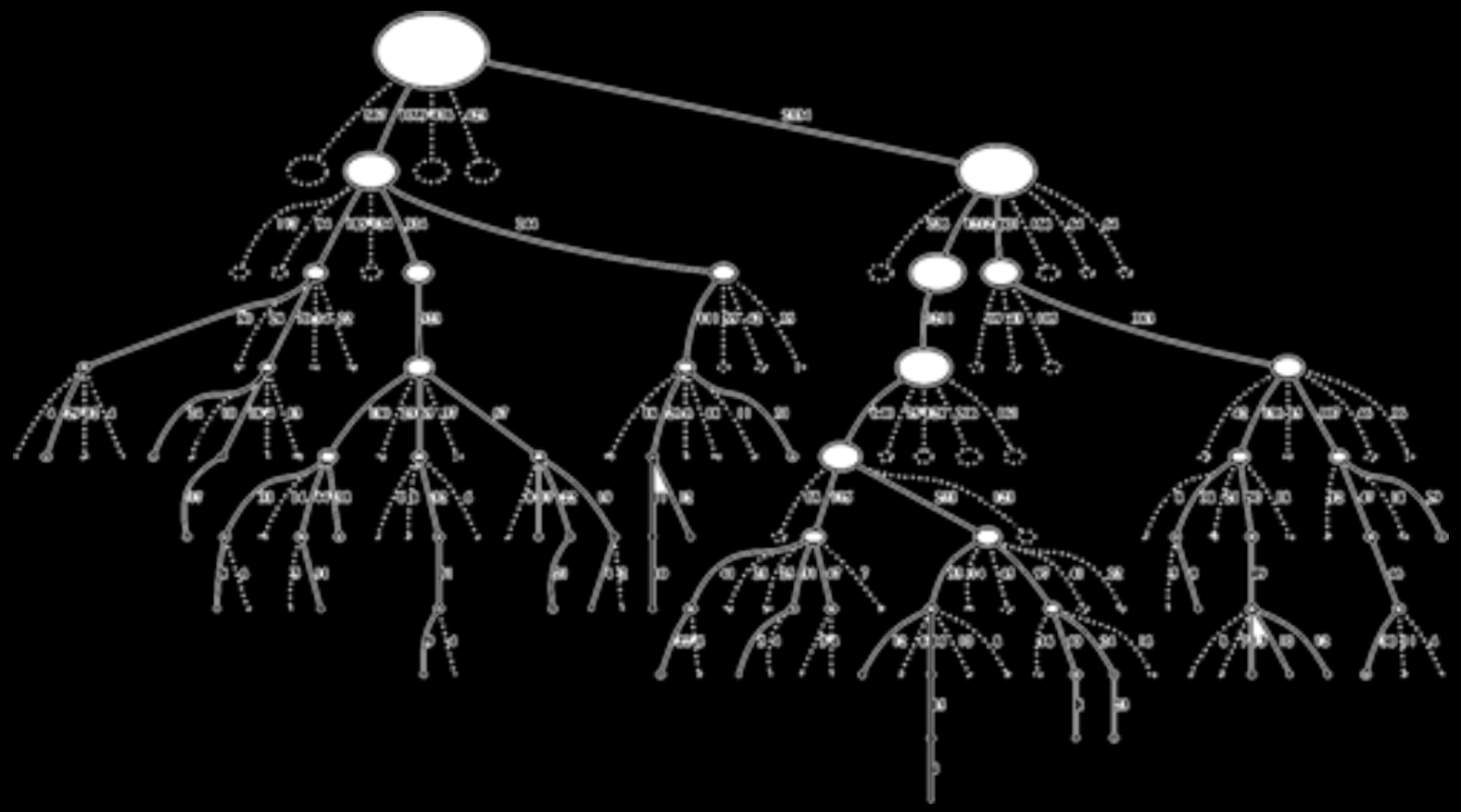
Stochastic approach

Minimax:

- playing against *ideal* opponent
- time consuming
- memory consuming

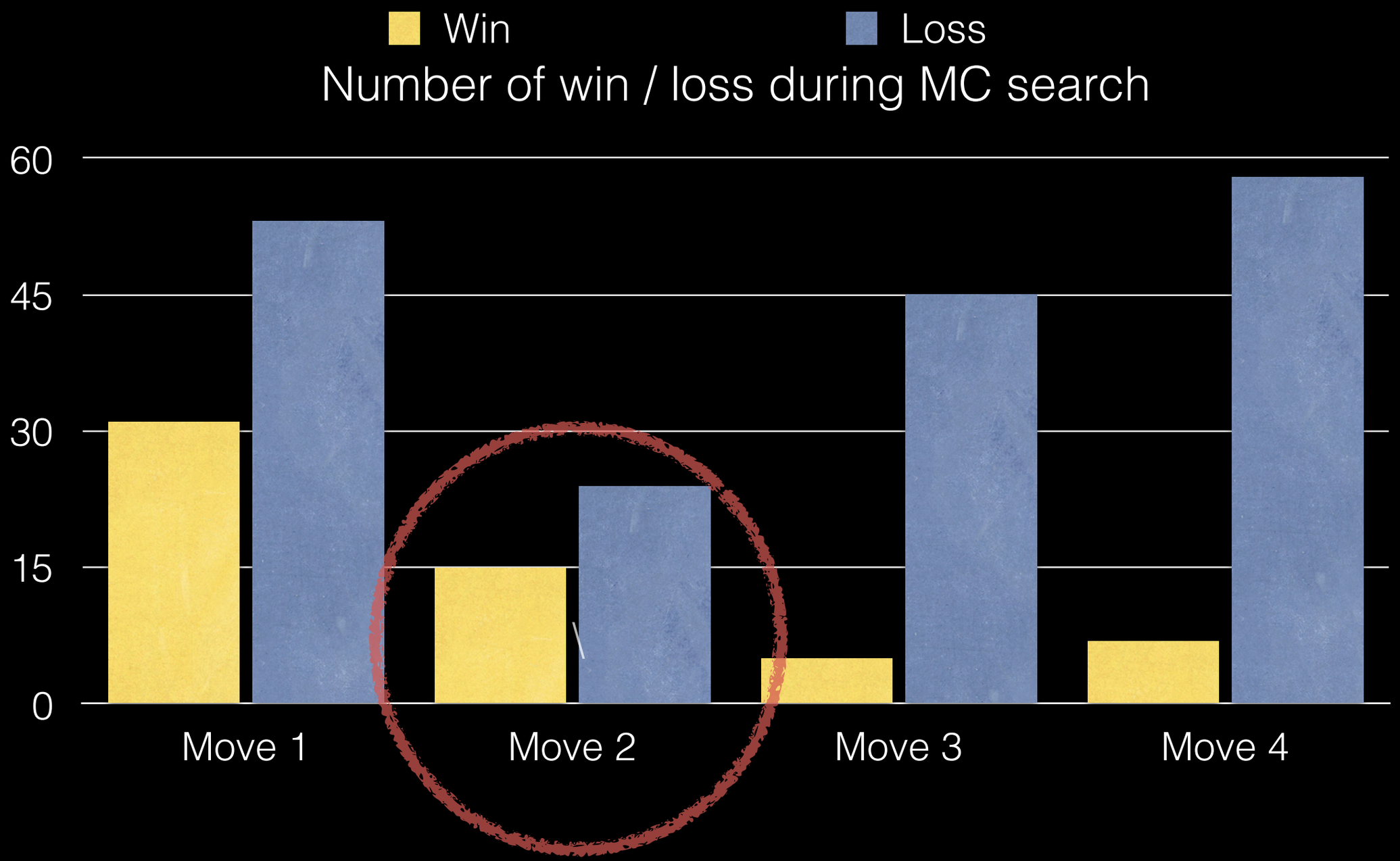
Key idea:
choose actions randomly

Monte Carlo search



- when state is huge
- there is a time limit for a move

Aggressive-defensive style



Learning

- Optimal policy: State -> Action*
- State -> [Actions]
- Reinforcement learning ??
- When state-action space is huge it does not work - curse of dimensionality

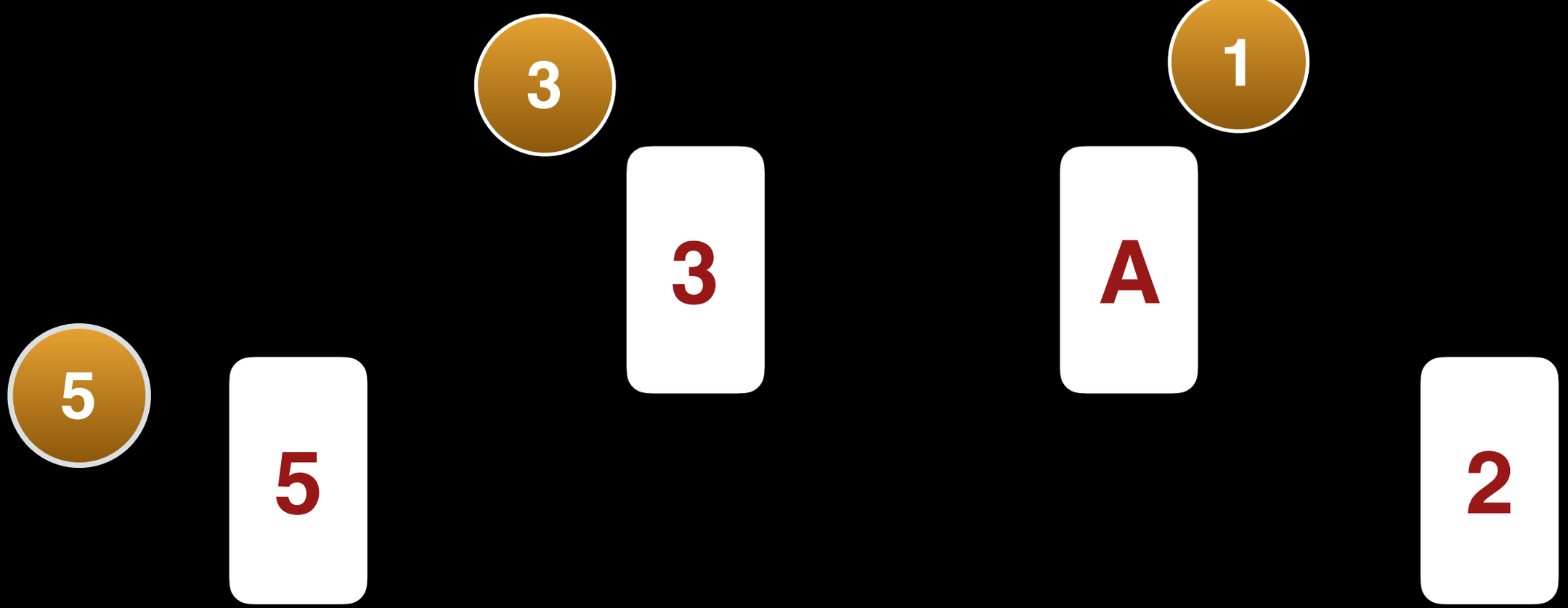
Evolutionary learning. Demo

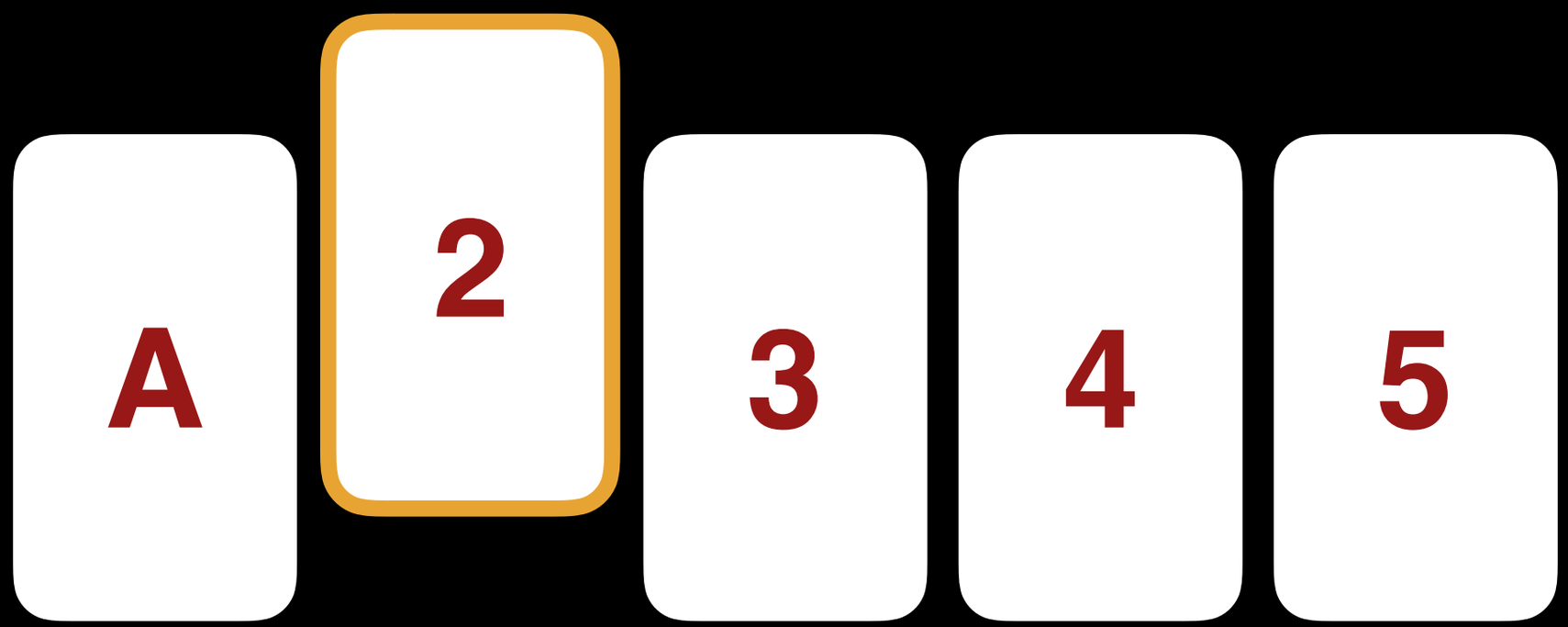
2:0	$w_w \leftarrow w_w$;	$w_i \leftarrow w_w + \delta w$
1.5:0.5	$w_w \leftarrow w_w$;	$w_i \leftarrow w_i + \delta w$
1:1	$w_w \leftarrow w_w$;	$w_i \leftarrow w_i$

Probabilistic ML

- Stochastic applications
- There are a lot of hidden (unknown factors)
- Examples: Poker, Heart Stone

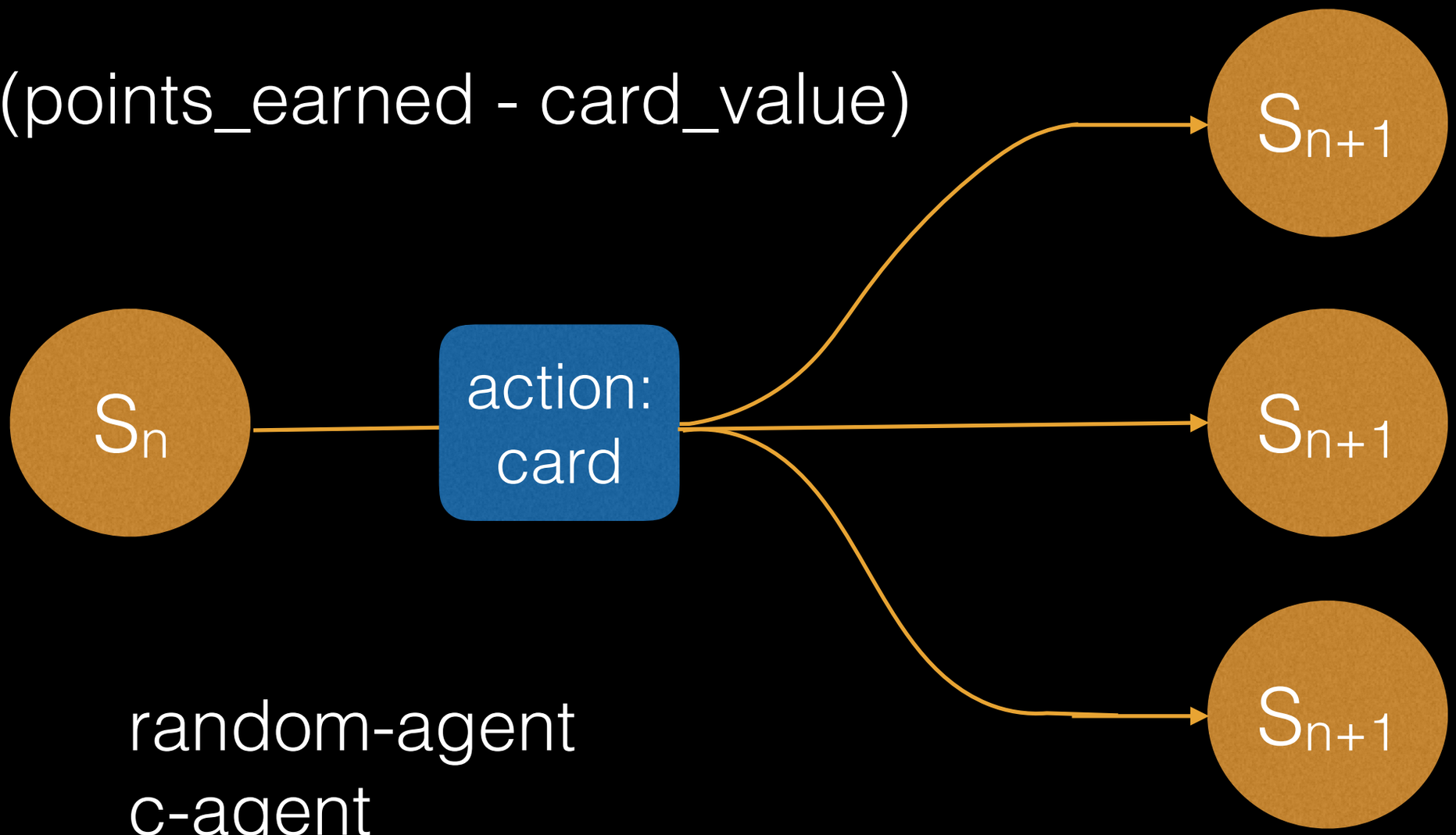
RANDOM HOUSE


3
1
5
3
A
5
2


A
2
3
4
5

State: {score:[2, 0, 0 ,5, 1],
rem_cards:[[1,3,5], [3,5,2], [1,4,5], [2,4,5], [2,3,4]]}

Utility: (points_earned - card_value)

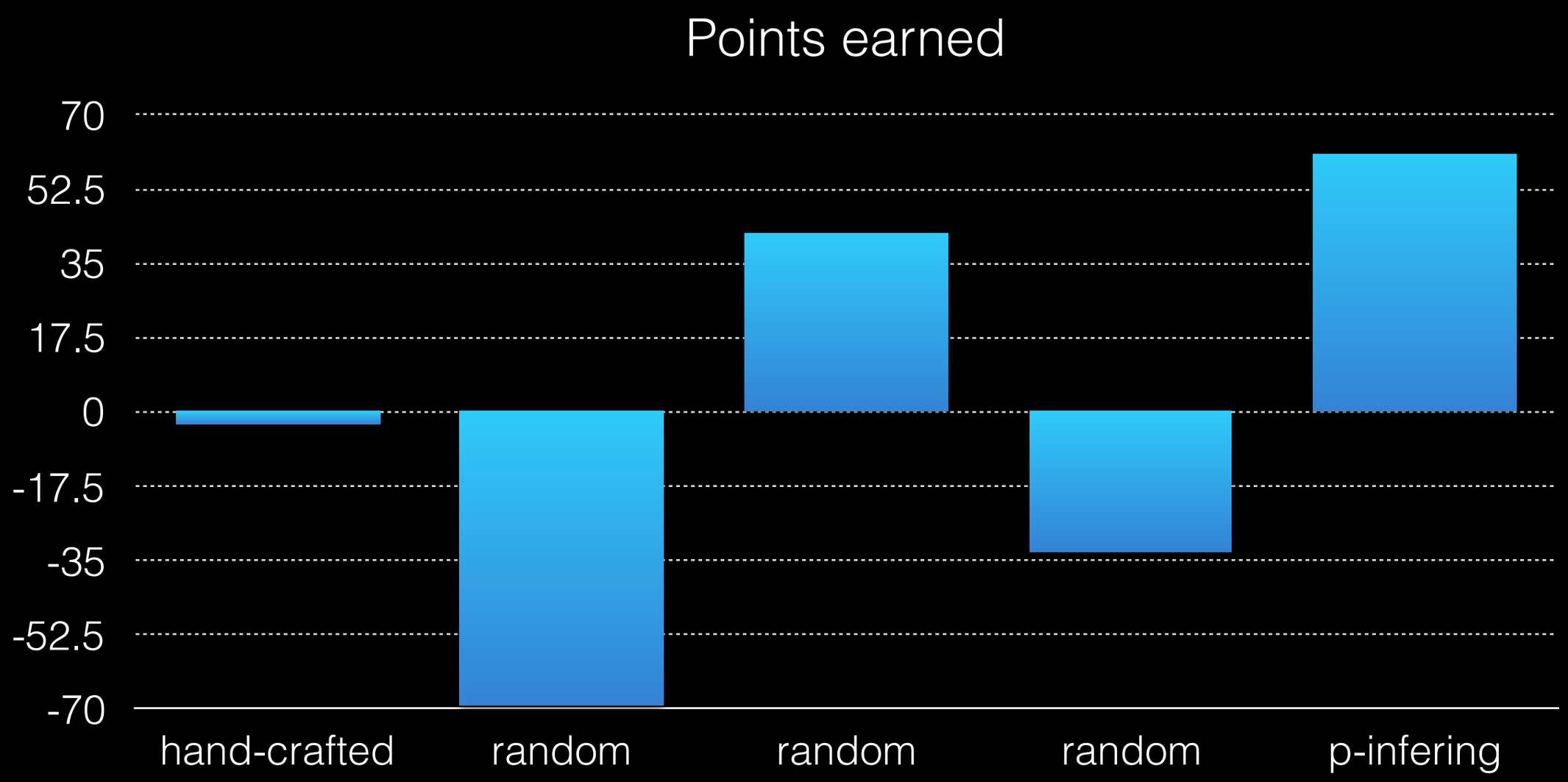


Agents:

random-agent
c-agent
one-step-agent
pinf-agent

http://webppl.org/

Demo. Results



https://github.com/tsybulkin/guess-card

Thank you

ian.tsybulkin@gmail.com
www.facebook.com/ian.tsybulkin
https://github.com/tsybulkin
@tsybulkin