Monte Carlo Tree Search

By the end, you will know…

- Why we use Monte Carlo Search Trees
- The pros and cons of MCTS
- How it is applied to Super Mario Brothers and Alpha Go

Outline

I. Pre-MCTS Algorithms
II. Monte Carlo Tree Search
III. Applications

Motivation

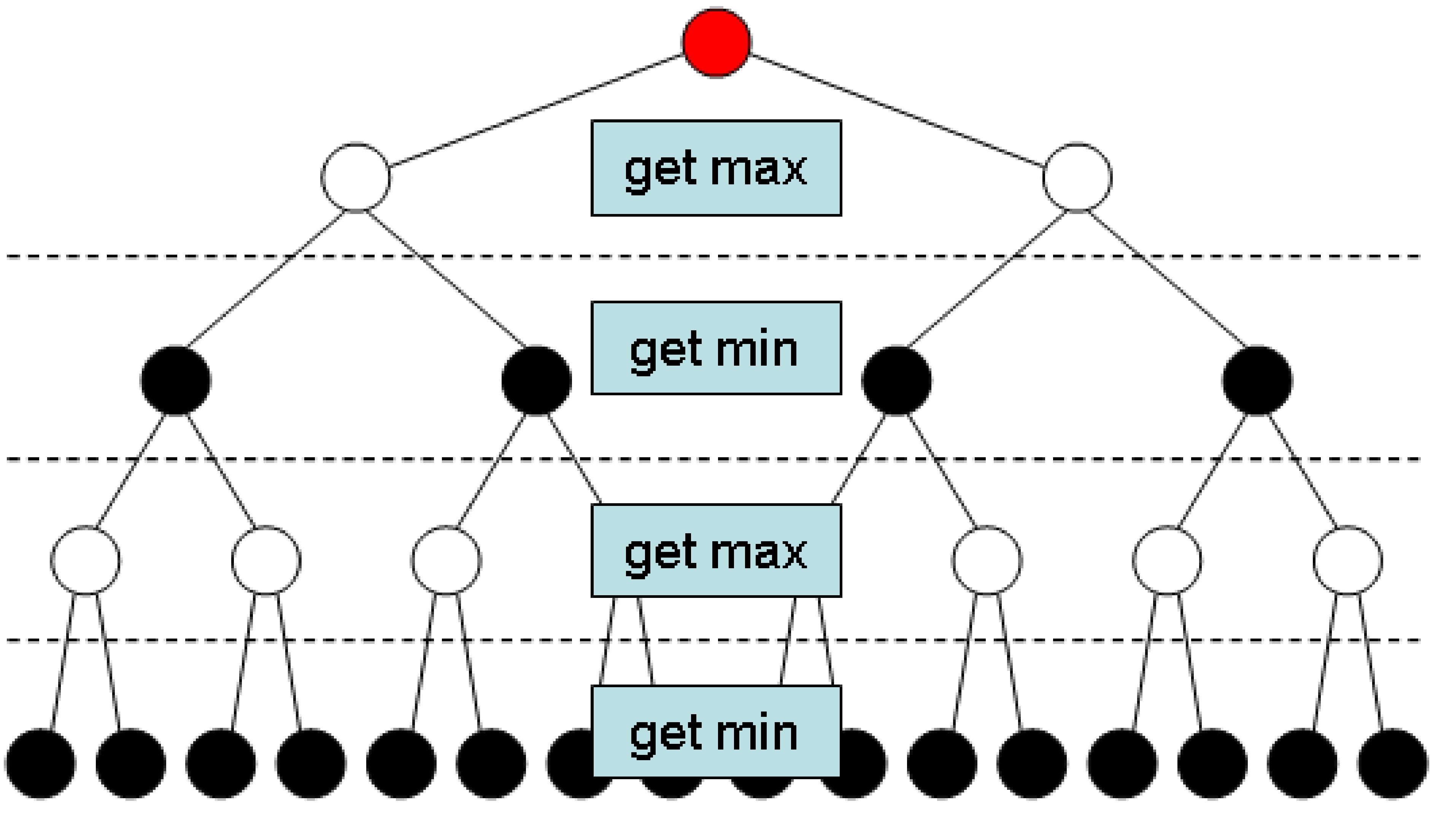
- Want to create programs to play games
- Want to play optimally
- Want to be able to do this in a reasonable amount of time

	Deterministic	Nondeterministic (Chance)
Fully Observable	Chess Checkers Go	Backgammon Monopoly
Partially Observable	Battleship	Card Games

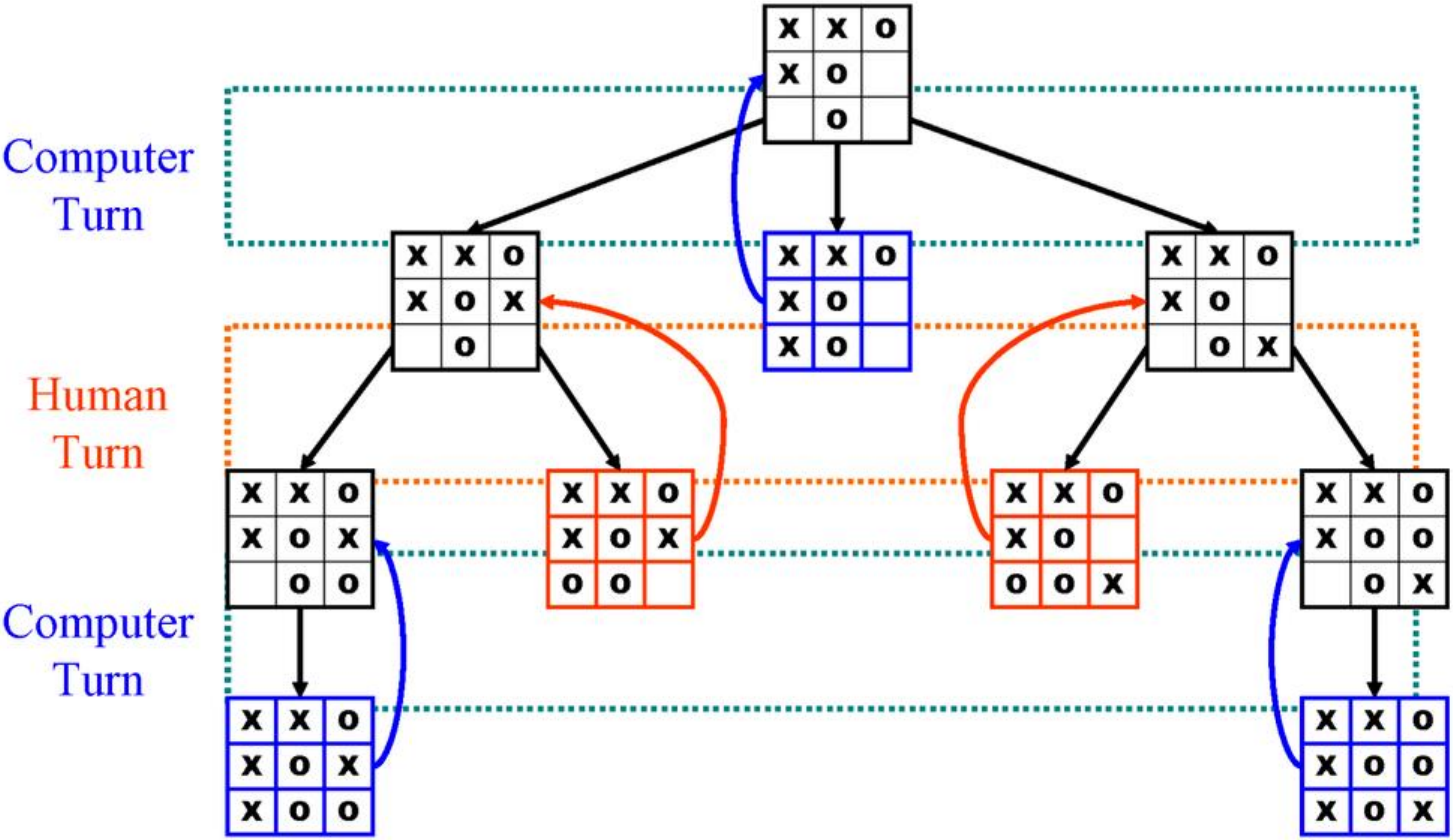
Pre-MCTS Algorithms

- Deterministic, Fully Observable Games
- "Perfect information"
- Can construct a tree that contains all possible outcomes because everything is fully determined

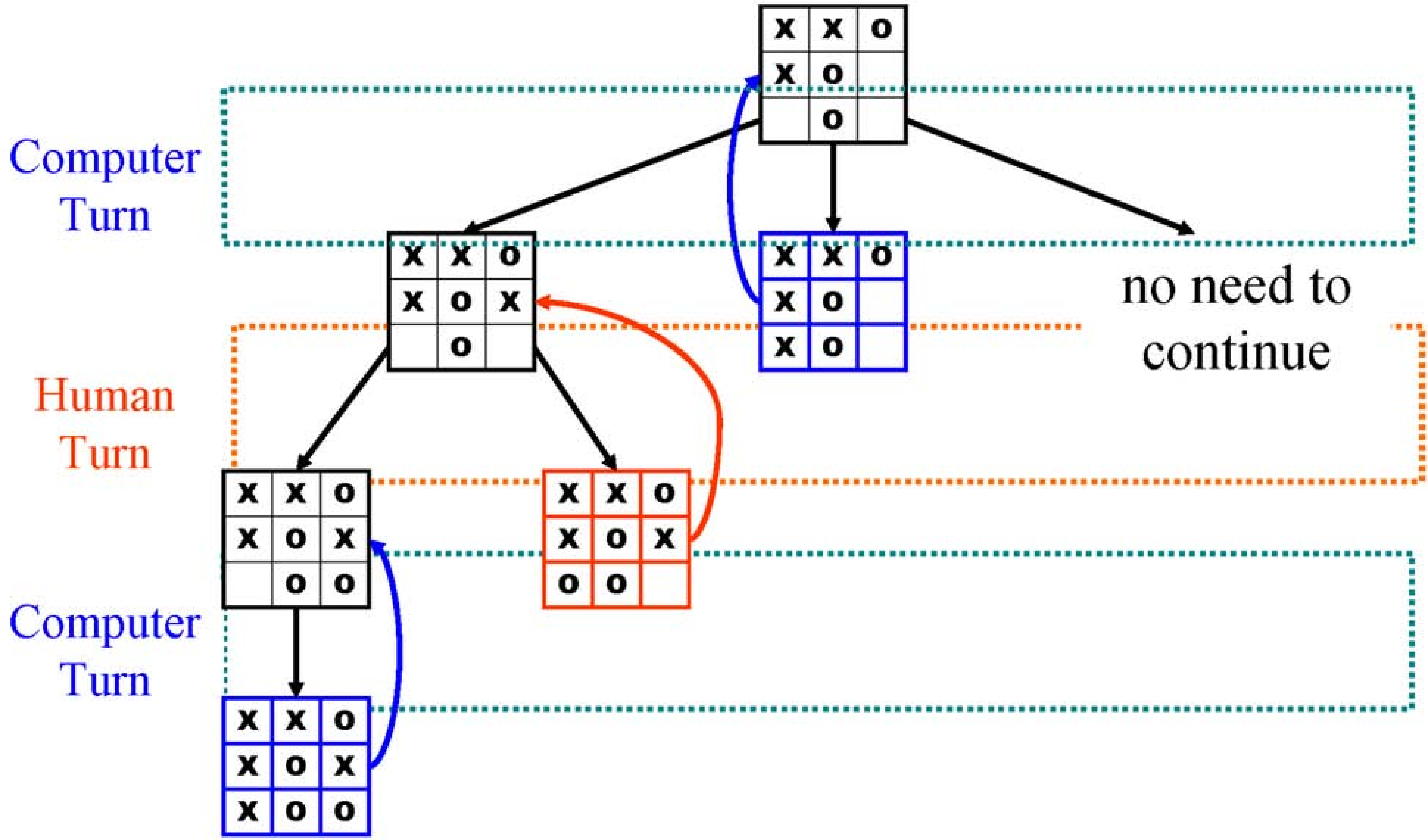
Minimize the maximum possible loss



Minimax



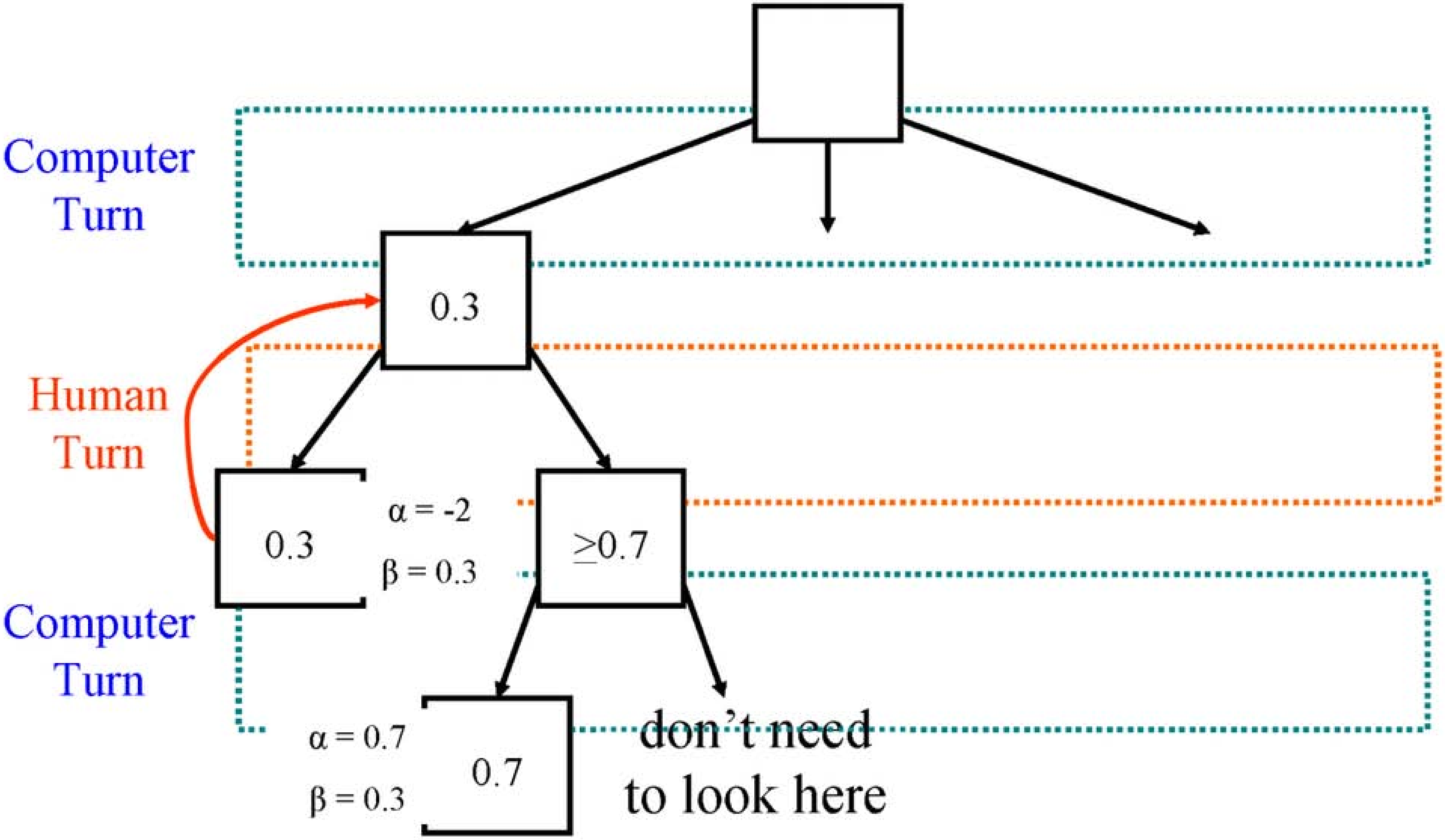
Simple Pruning



Alpha-Beta Pruning

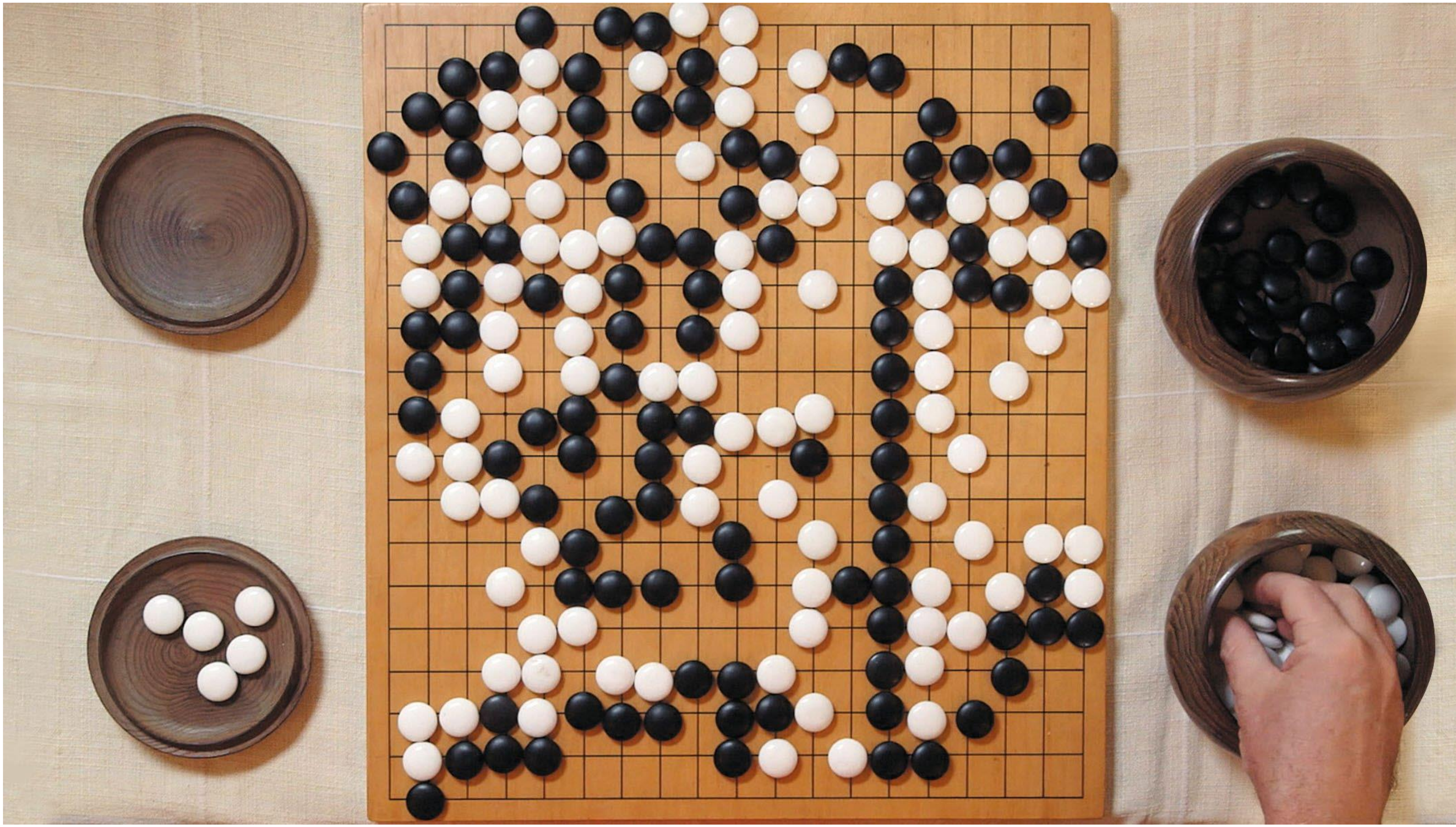
- Prunes away branches that cannot influence the final decision

Alpha - Beta



Computer
Turn
0.3
Human
Turn
0.3
α = -2
β = 0.3
≥0.7
Computer
Turn
α = 0.7
β = 0.3
0.7
don't need
to look here

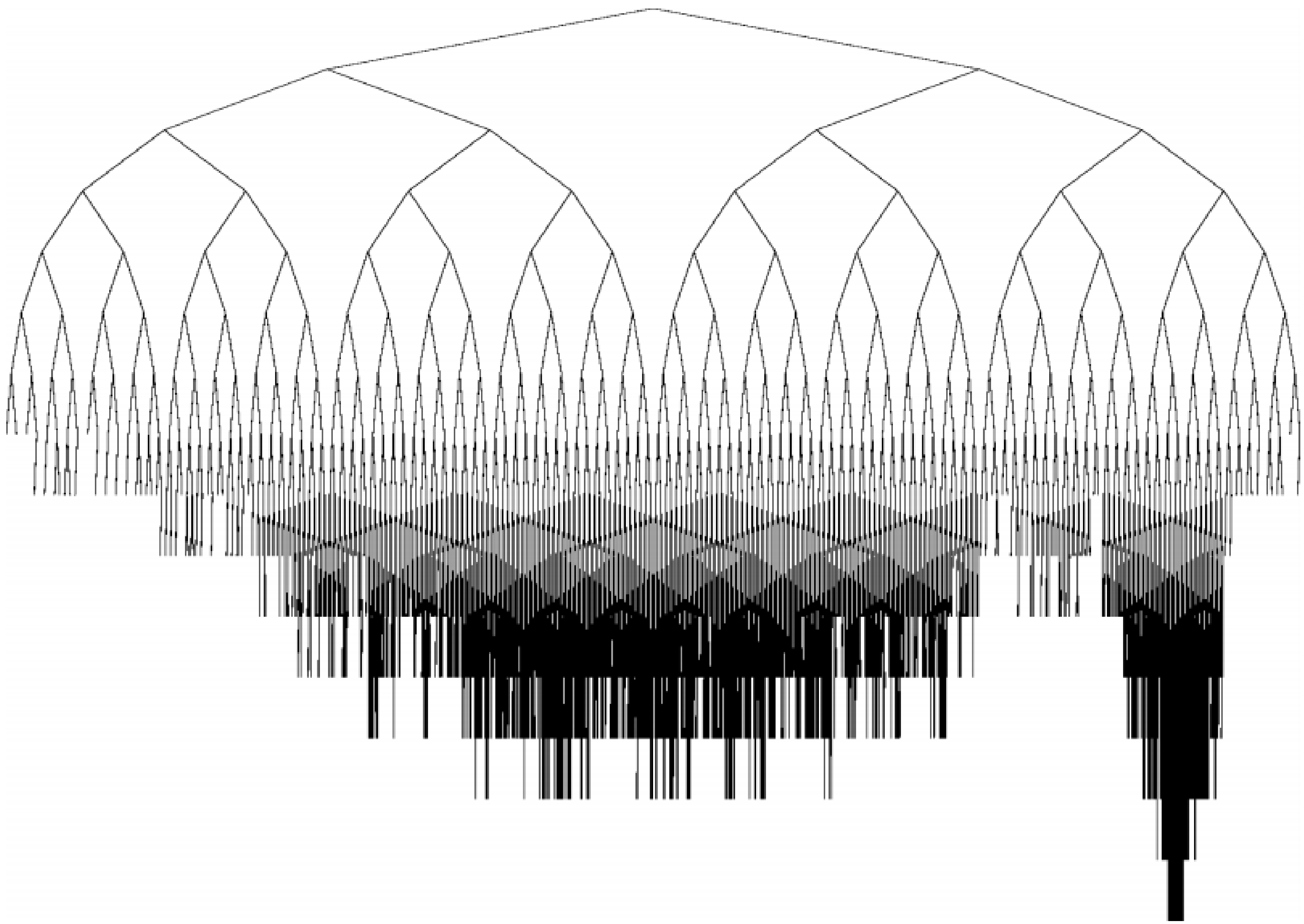
2^4 vs. 2^{250}



Outline

I. Pre-MCTS Algorithms
II. Monte Carlo Tree Search
III. Applications

Asymmetric Tree Exploration



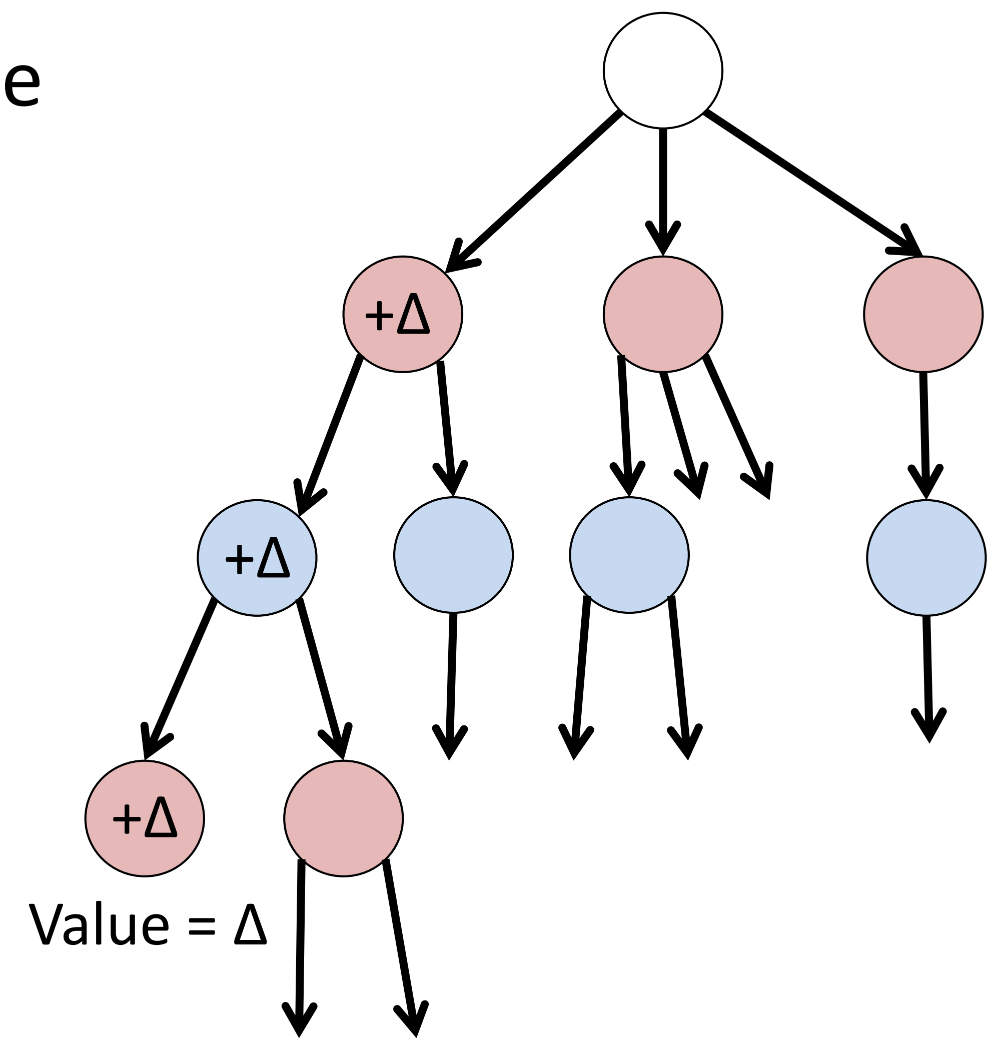
From *Bandit Algorithms for Tree Search*, Coquelin and Munos, 2007

MCTS Outline

1. Descend through the tree
2. Create new node
3. Simulate
4. Update the tree

Repeat!

5. When you're out of time,
Return "best" child.

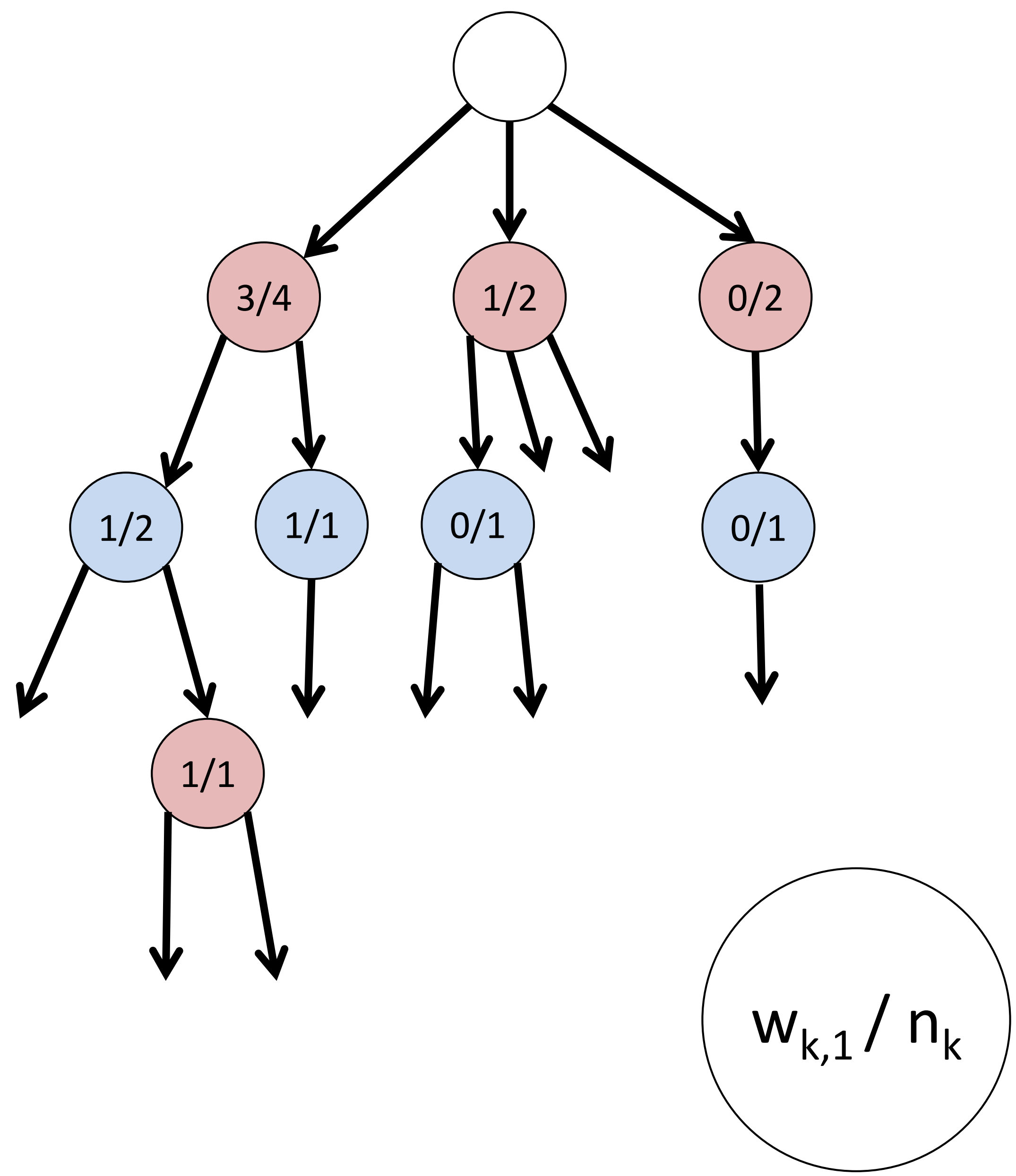


What do we store?

For game state k:

n_k = # games played involving k

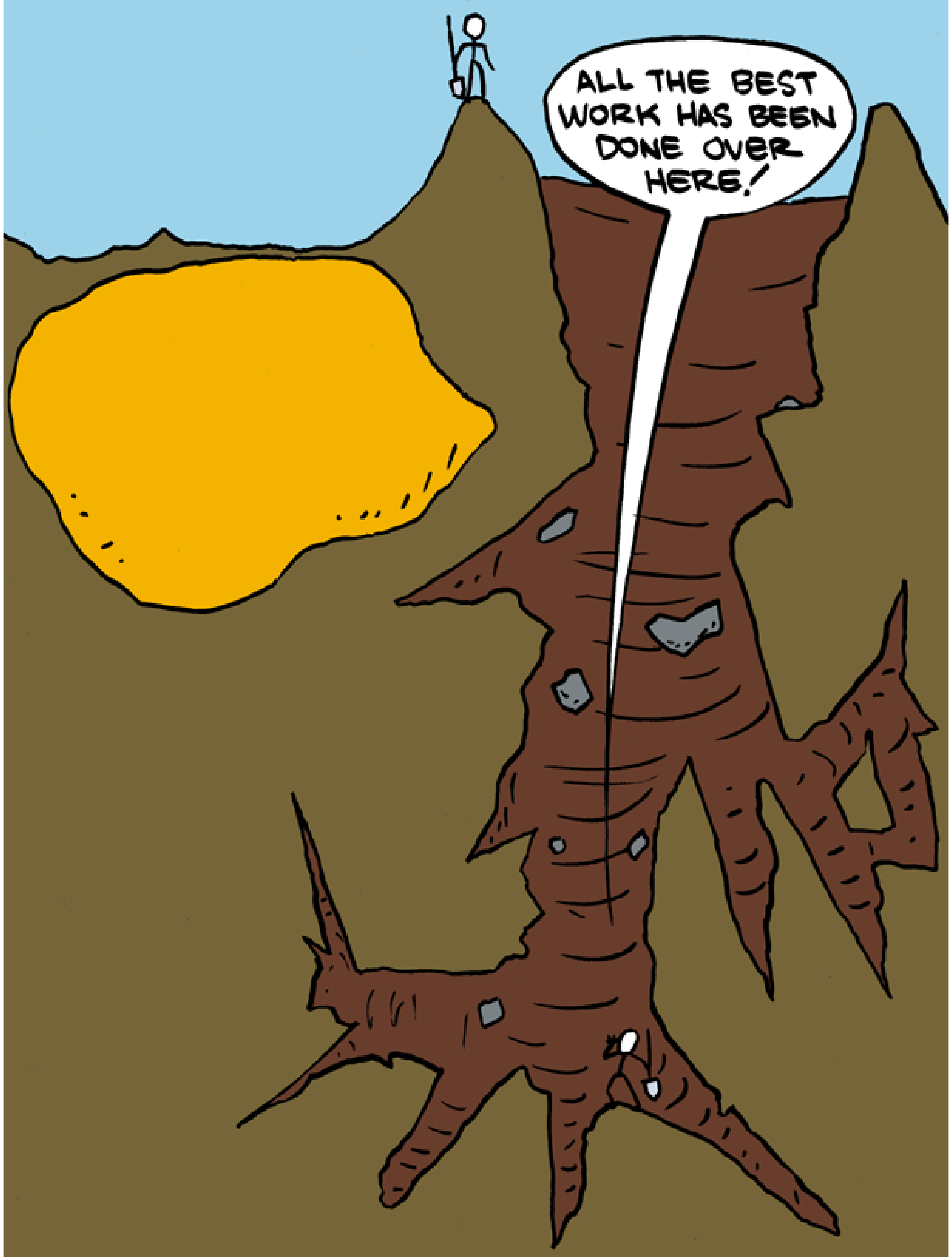
$w_{k,p}$ = # games won (by player p) that involved k



1. Descending

We want to **expand**,

but also to **explore**.



1. Descending

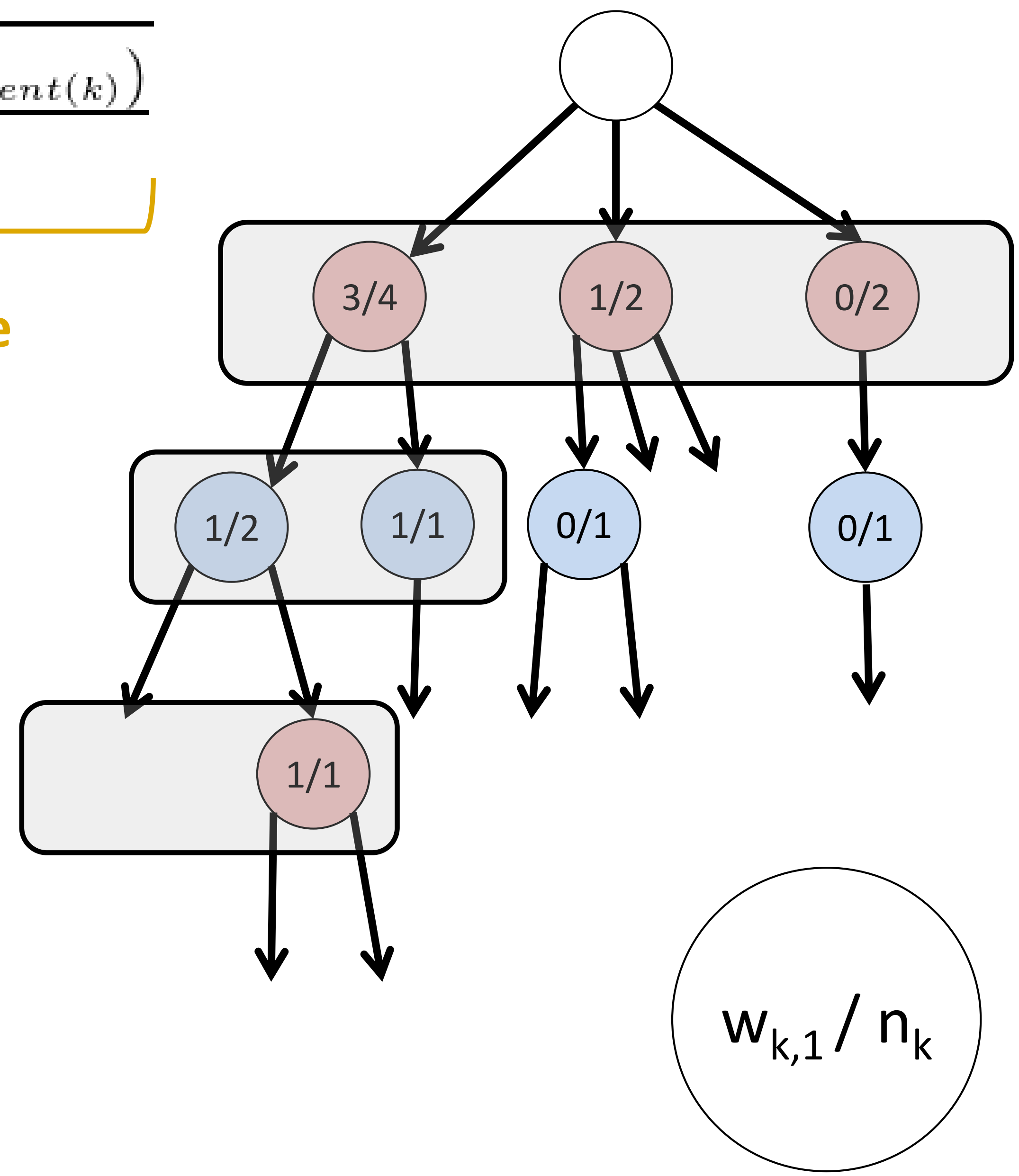
Solution: *Upper Confidence Bound*

$$UCB1(k,p) = \underbrace{E[win|k,p]}_{\text{expand}} + \underbrace{C\sqrt{\frac{2\ln\left(n_{parent(k)}\right)}{n_k}}}_{\text{explore}}$$

$$\approx \frac{w_{k,p}}{n_k} + C\sqrt{\frac{2\ln\left(n_{parent(k)}\right)}{n_k}}$$

At each step,

maximize UCB1(k, p)

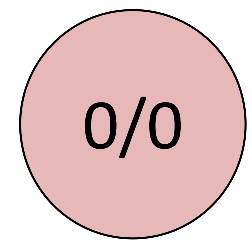


2. Expanding

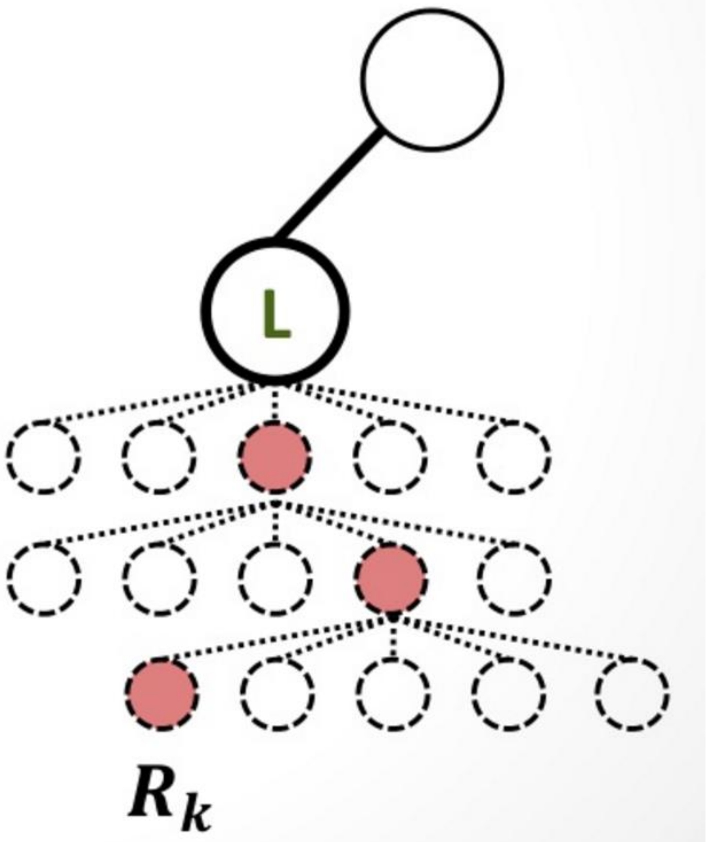
Not very complicated.

Make a new node!

Set $n_k = 0$, $w_k = 0$



3. Simulating



Simulating a real game is hard.

Let's just play the game out randomly!

If we win, $\Delta = +1$. If we lose or tie, $\Delta = 0$.

X		X
O		
X	O	O

X	X	X
O	O	
X	O	O

X wins

X		X
O	X	O
X	O	O

X wins

X		X
O	O	X
X	O	O

O wins

X		

A lot of options...

4. Updating the Tree

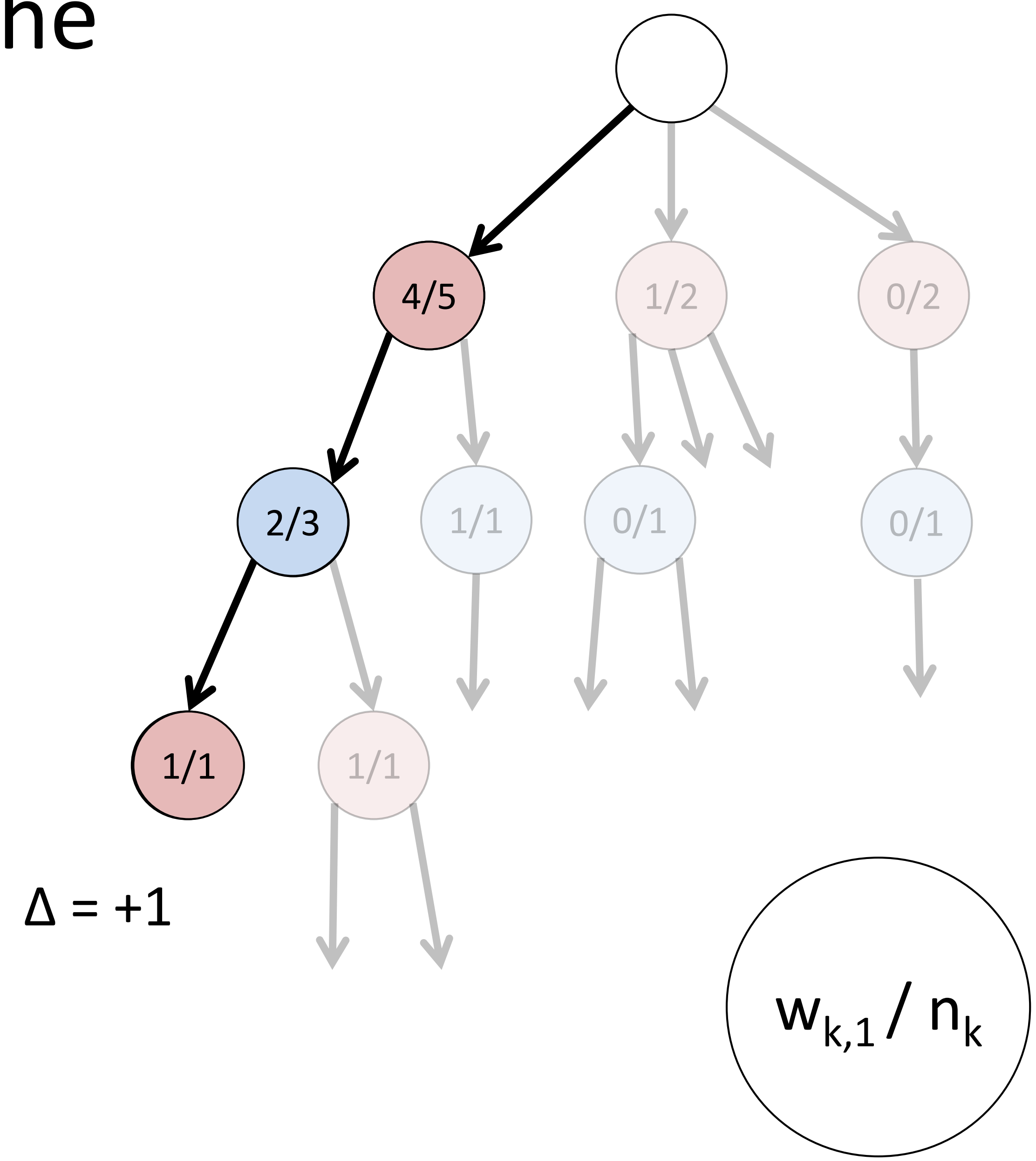
Propagate recursively up the parents.

Given simulation result Δ,

for each k:

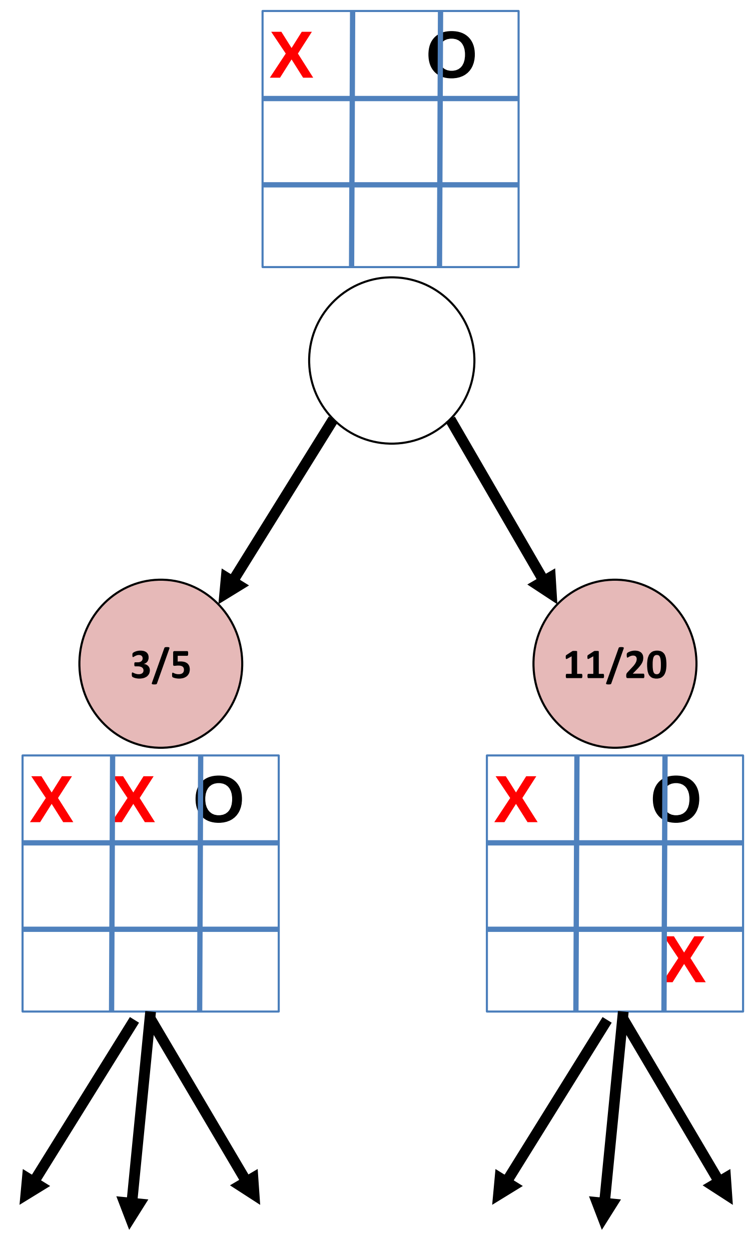
$$n_{k\text{-new}} = n_{k\text{-old}} + 1$$

$$w_{k,1\text{-new}} = w_{k,1\text{-old}} + \Delta$$



5. Terminating

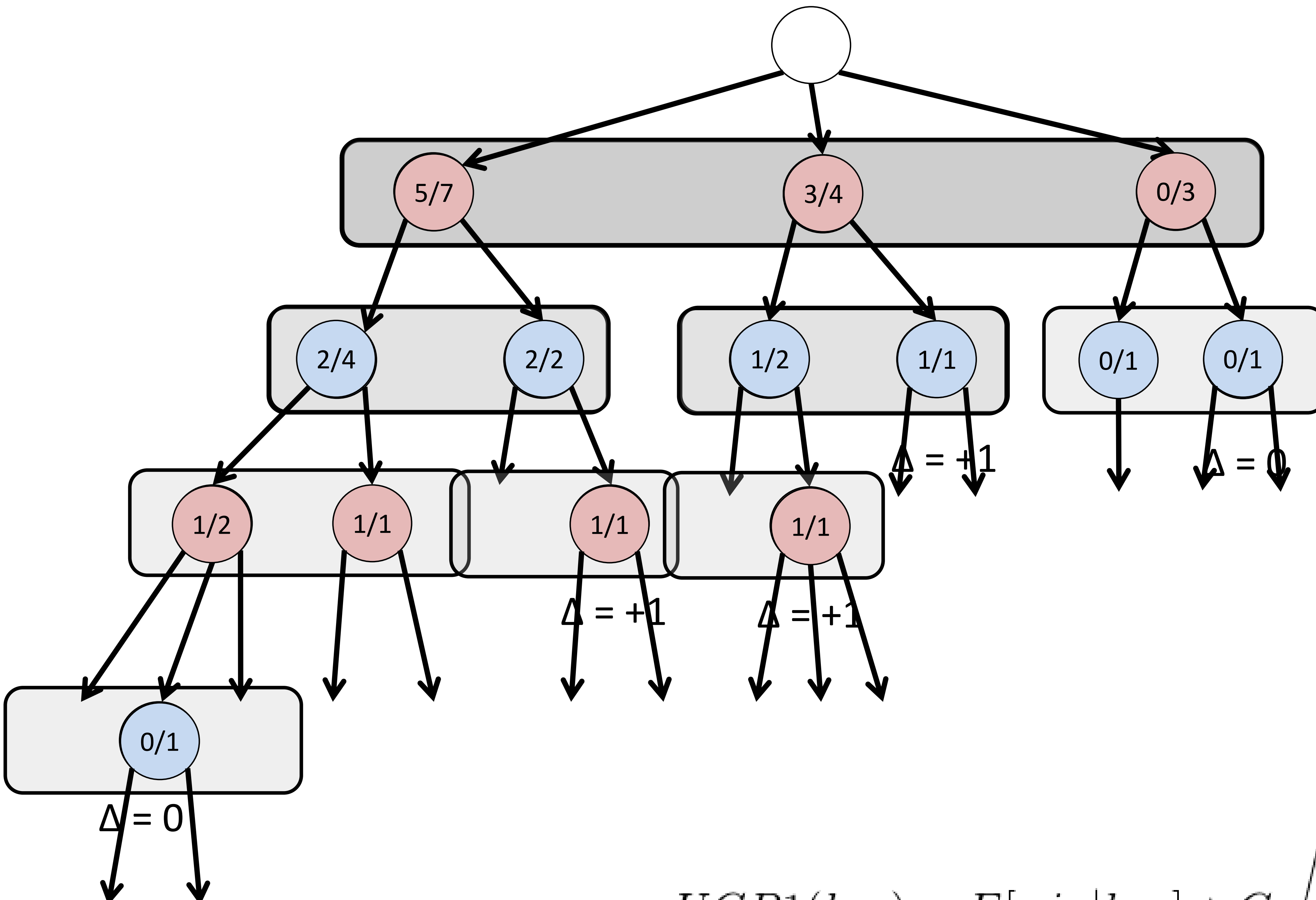
Return the best-ranked first ancestor!



What determines "best"?

- Highest E[win|k]

- Highest E[win|k] AND most visited



$$UCB1(k,p) = \underbrace{E[win|k,p]}_{\text{expand}} + \underbrace{C\sqrt{\frac{2\ln\left(n_{parent(k)}\right)}{n_k}}}_{\text{explore}}$$

Why use MCTS?

Pros:

- Grows tree asymmetrically, balancing expansion and exploration
- Depends only on the rules
- Easy to adapt to new games
- Heuristics not required, but can also be integrated
- Can finish on demand, CPU time is proportional to answer quality
- Complete: guaranteed to find a solution given time
- Trivially parallelizable

Cons:

- Can’t handle extreme tree depth
- Requires ease of simulation, massive computation resources
- Relies on random play being “weakly correlated”
- Many variants, need expertise to tune
- Theoretical properties not yet understood

Screenshots of video games removed due to copyright restrictions.

Outline

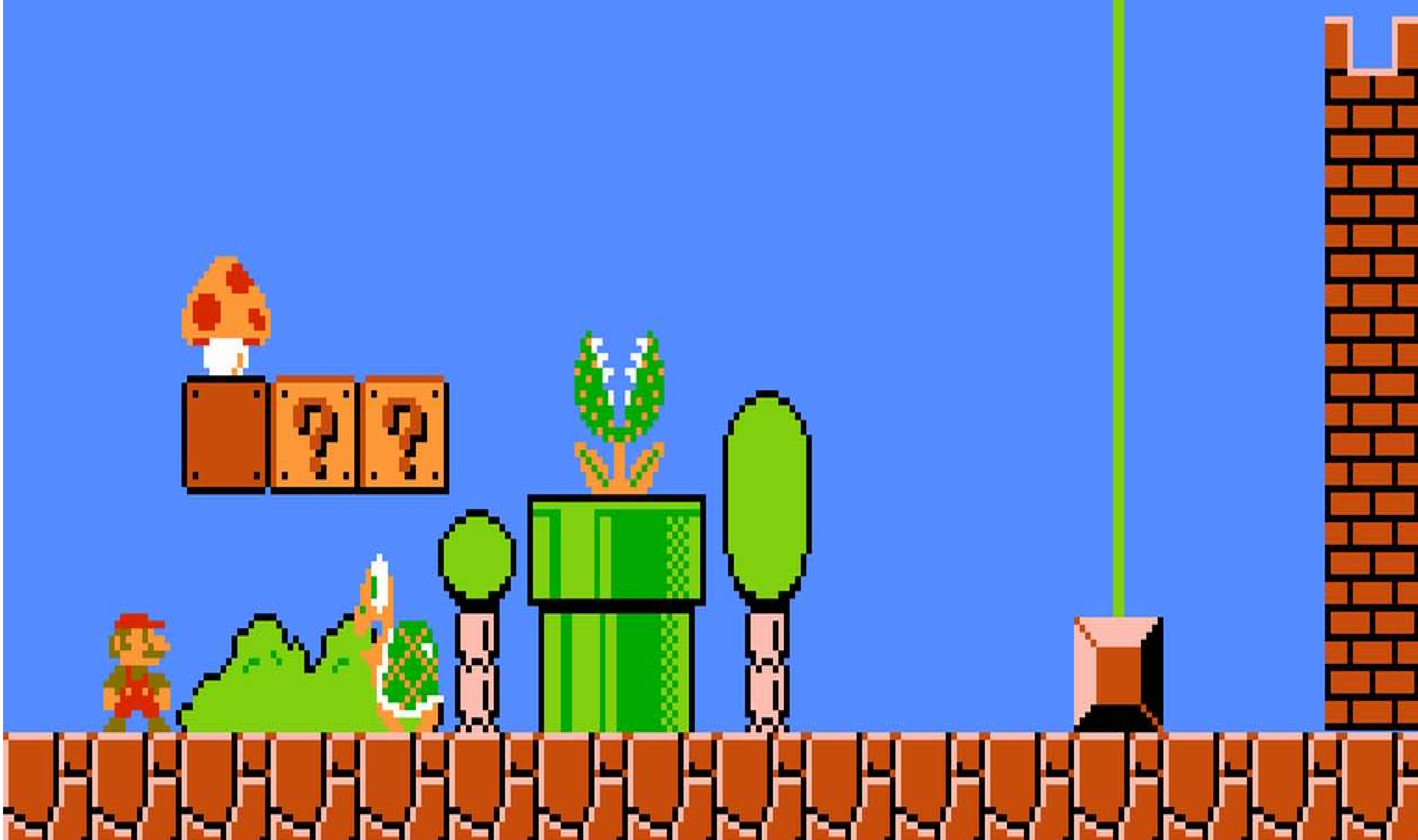
I. Pre-MCTS Algorithms
II. Monte Carlo Tree Search
III. Applications

… Wait for it…

Part III

Applications

MCTS-based Mario Controller!

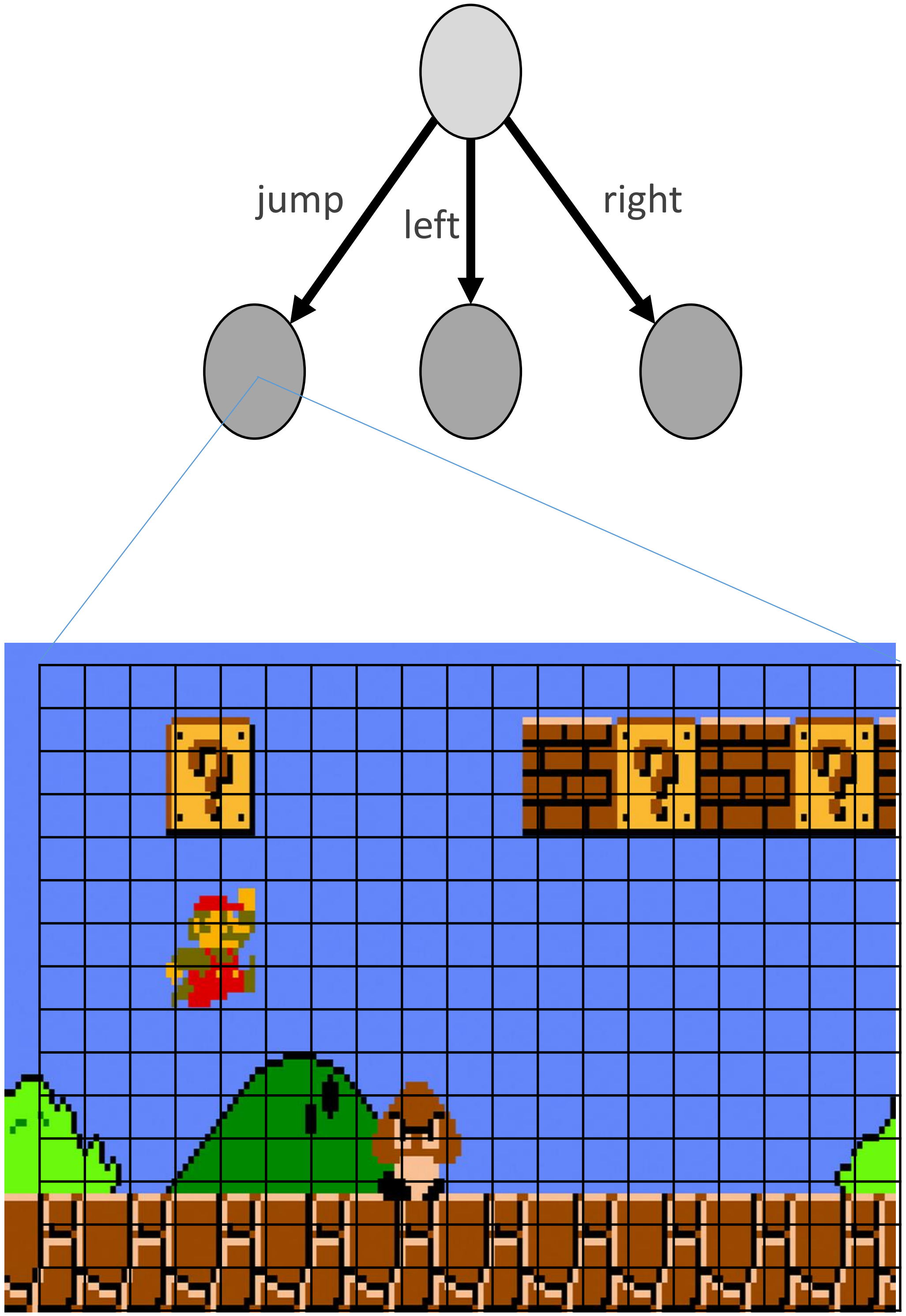


MCTS modifications for Super Mario Bros

- Single player
- Multi-simulation
- Domain knowledge
- 5-40ms computation time

Problem Formulation

- Nodes
  - State
    - Mario position, speed, direction, etc
    - Enemy position, speed, direction, etc
    - Location of blocks
    - etc
  - Value
- Edges
  - Mario's possible action (right, left, jump, etc)



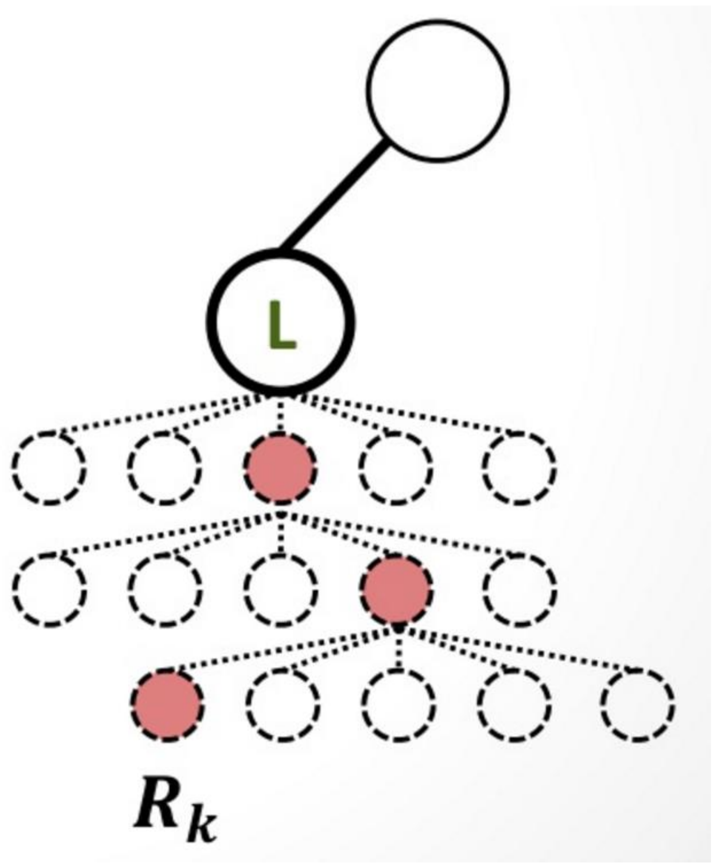
Calculating Simulation Result

Domain Knowledge: multi-objective weighted sum

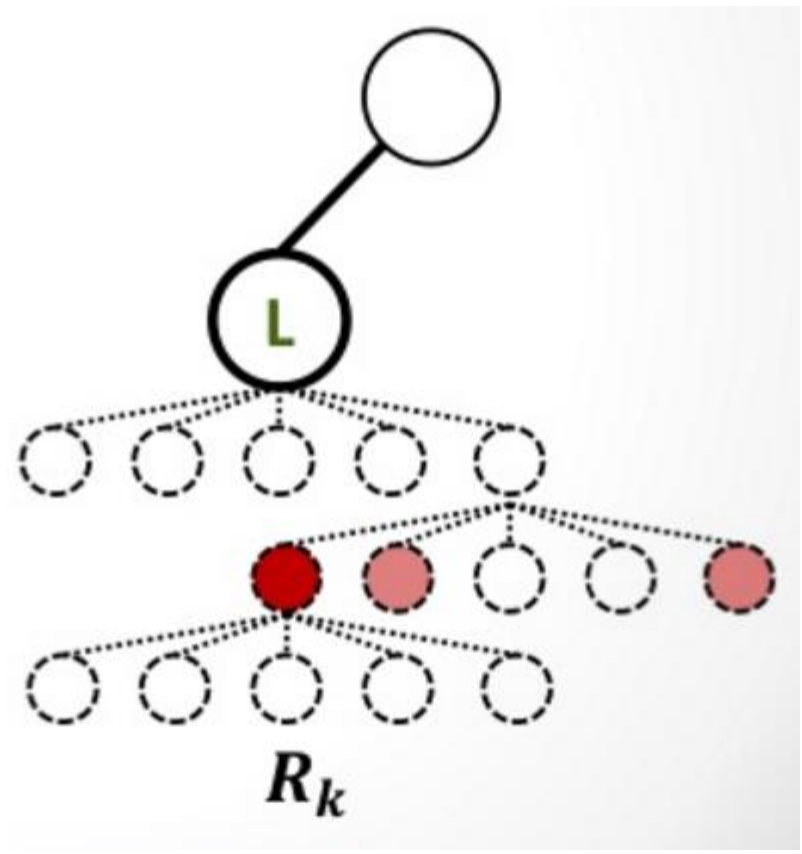
Distance	0.1	hiddenBlocks	24	marioStatus	1024
Flower	64	killsByStomp	12	timeLeft	2
Mushrooms	58	killsByFire	4	marioMode	32
greenMushrooms	1	killsByShell	17	Coins	16
Hurts	-42	killsTotal	42	Stomps	1

Simulation type

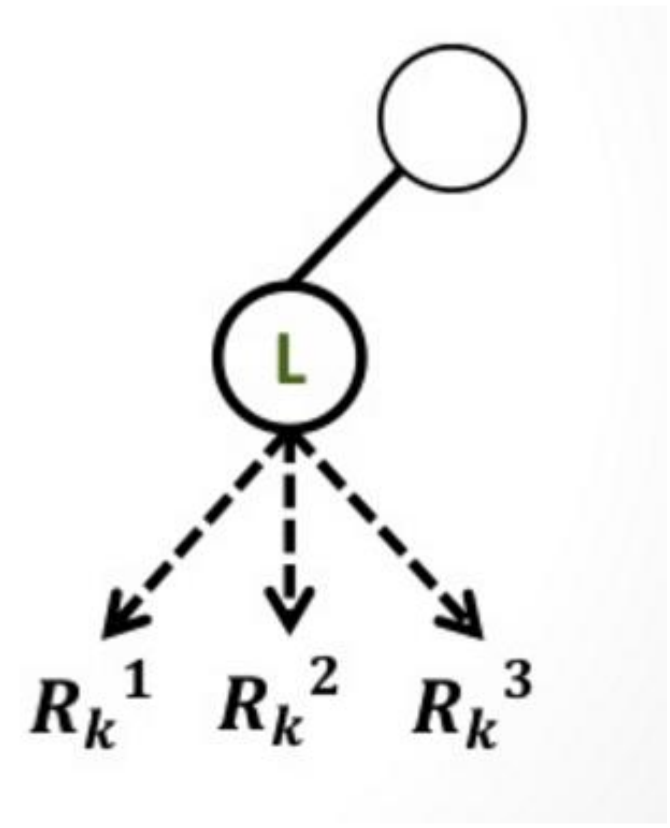
Regular



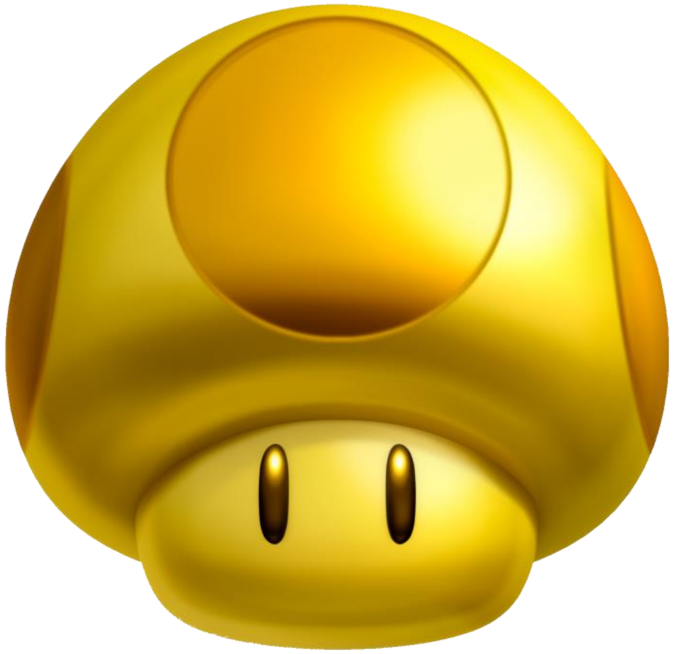
Best of N



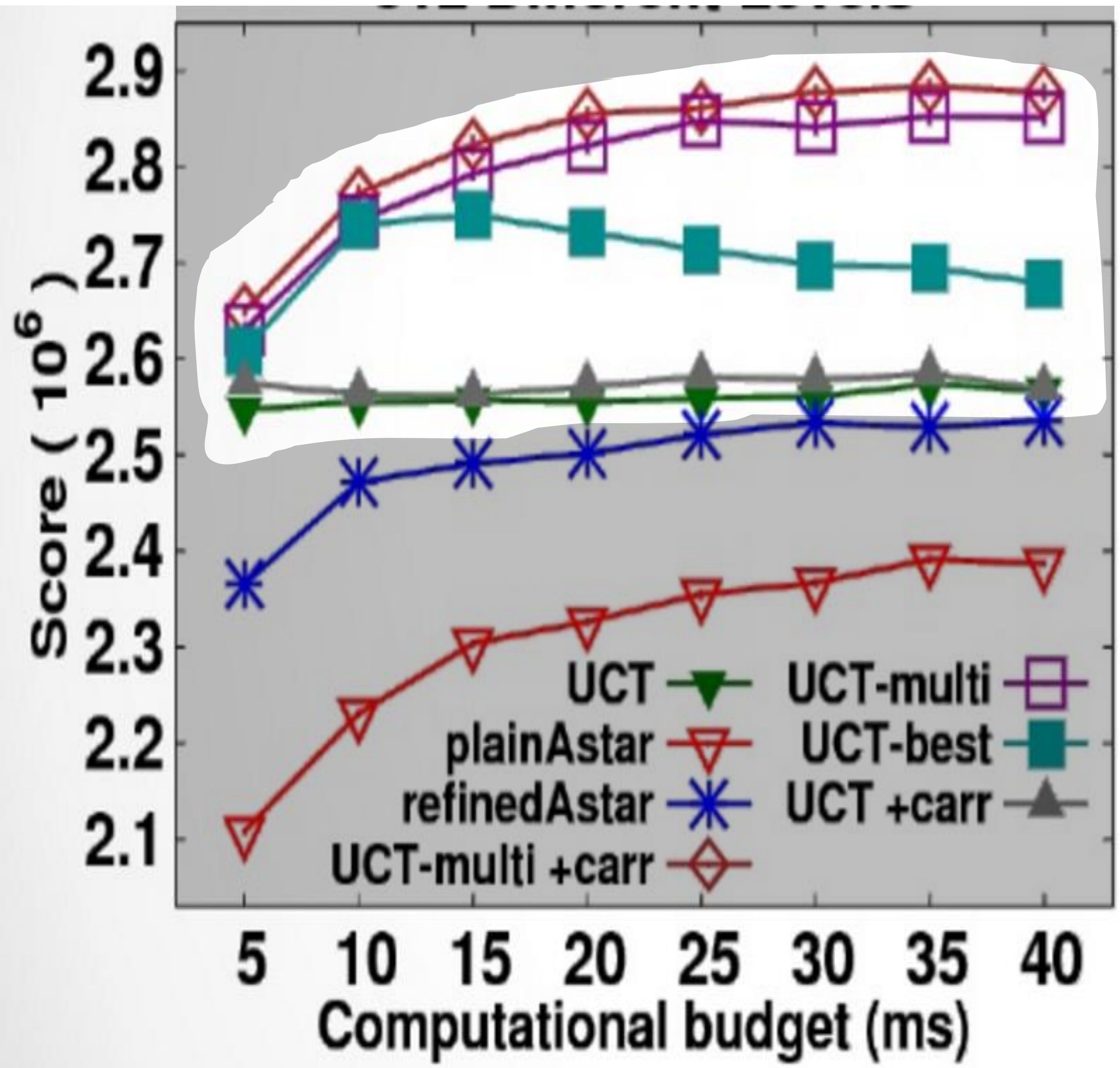
Multi-Simulation

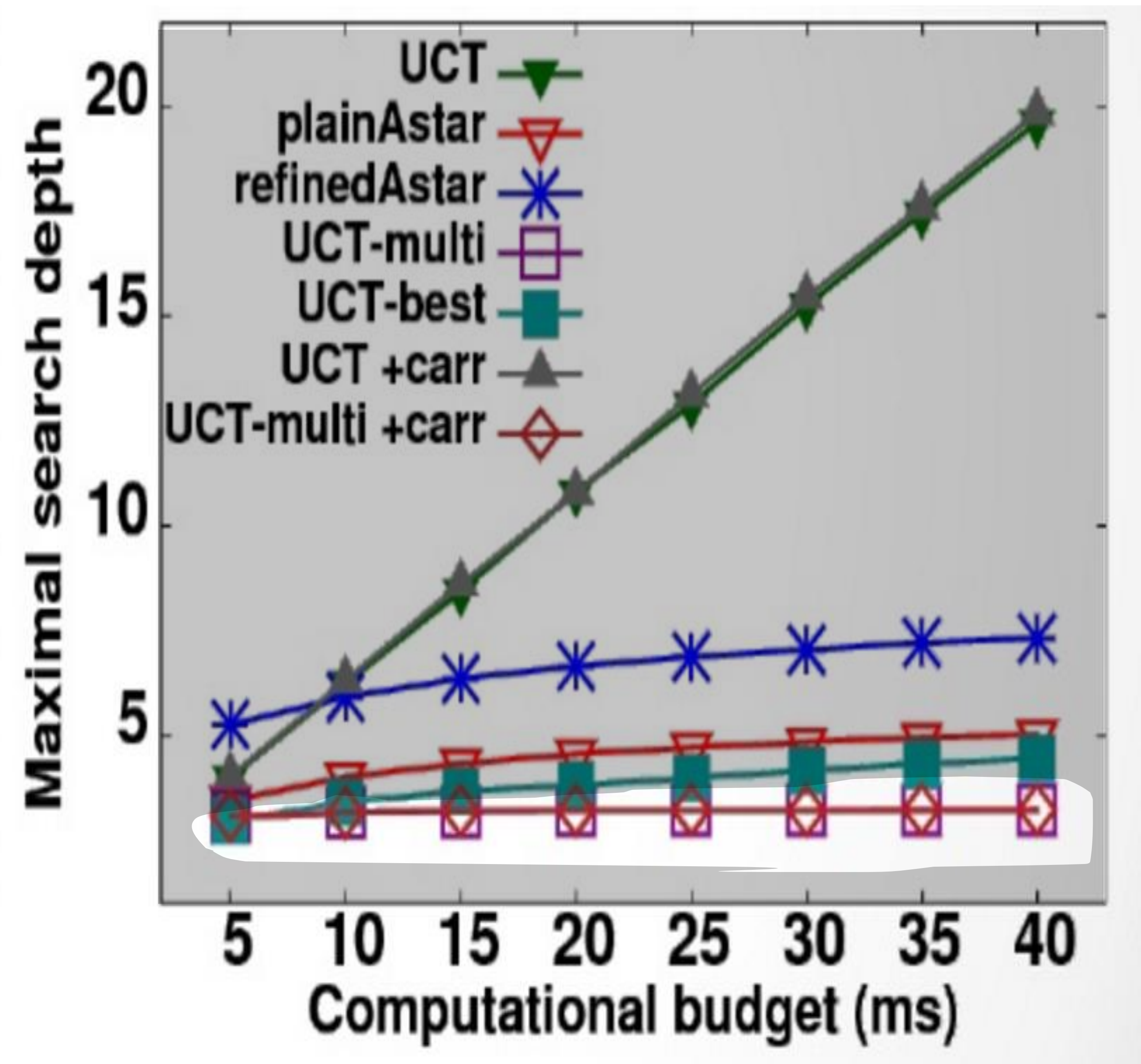


Demo



Results





Outperforms Astar

AlphaGo

The Rules

- Board is 19x19. Starts empty.
- Players alternate placing one stone.
- Capture enemy stone by surrounding
- A player’s territory is all the area surroun
- Score = Territory + Captured pieces

Go vs Chess

GO	CHESS
250 options	35 options
150 turns	80 turns
10^{761} games	10^{120} games

MCTS modifications for Go

- Combines Neural Networks with MCTS
  - 2 Policy Networks (slow and fast)
  - 1 Value Network

2 Policy Networks

- Input is the game state, as an image
- Output is a probability distribution over legal actions
- Supervised learning on 30 million positions from human expert games

Slow Policy Network	Fast Policy Network
57% accuracy	24% accuracy
3,000 microseconds	2 microseconds

Policy Network – Reinforcement Learning

Next step: predict winning moves, rather than expert human moves

Policy Networks play against themselves!

Tested best Policy Network against Pachi

- Pachi relies on 100,000 MCTS simulations at each turn
- AlphaGo's Policy Network won 85% of the games (3ms per turn)
- Intuition tends to win over long reflection in Go?

Value Network

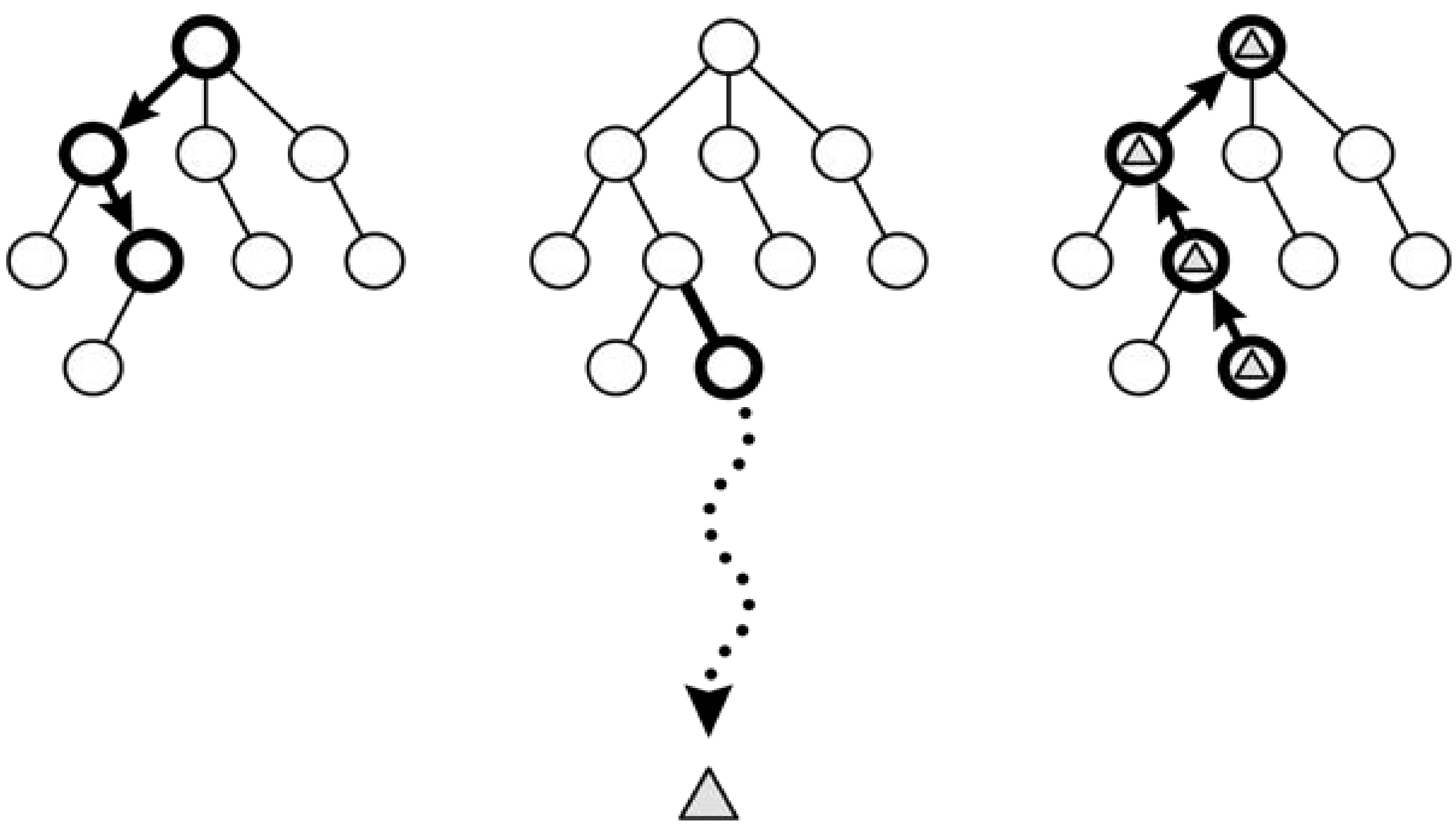
Trained on positions from the Policy Network's reinforcement learning

- Similar to evaluation function (as in DeepBlue), but *learned* rather than designed.
- Predictions get better towards end game

Using Neural Networks with MCTS

Slow Policy Network guides tree search

Value of state = Fast Policy Network simulation + Value Network Output



Why use Policy and Value Networks?

They work hand-in-hand.

The VN learns from the PN, and the PN is improved by the VN.

- Value Network Alone
  - Would have to exhaustively compare the value of all children
    - PN Predicts the best move, narrows the search space by only considering moves that are most likely victorious

- Policy Network Alone
  - Unable to directly compare nodes in different parts of the tree
  - VN gives estimate of winner as if the game were played according to the PN
    - Values direct later searches towards moves that are actually evaluated to be better

Why combine Neural Networks with MCTS?

- How does MCTS improve a Policy Network?
  - Recall: MCTS (Pachi) beat the Policy Network in 15% of games
  - Policy Network is just a *prediction*
  - MCTS and Monte-Carlo rollouts help the policy adjust towards moves that are actually evaluated to be good

- How do Neural Networks improve MCTS?
  - The Slow Policy more intelligently guides tree exploration
  - The Fast Policy Network more intelligently guides simulations
  - Value Network and Simulation Value are complementary

AlphaGo vs Other AI

AI name	Elo rating
Distributed AlphaGo (2015)	3140
AlphaGo (2015)	2890
CrazyStone	1929
Zen	1888
Pachi	1298
Fuego	1148
GnuGo	431

Distributed AlphaGo won 77% of games against single-machine AlphaGo

Distributed AlphaGo won 100% of games against other AI

AlphaGo vs Lee Sedol

AlphaGO	Lee Sedol
4 wins	1 win
3,586 Elo	3,520 Elo

Only one human with a higher Elo....

Ke Jie (Elo 3,621)

Timeline

- **1952** – computer masters Tic-Tac-Toe
- **1994** – computer master Checkers
- **1997** – IBM's Deep Blue defeats Garry Kasparov in chess
- **2011** – IBM's Watson defeats to Jeopardy champions
- **2014** – Google algorithms learn to play Atari games

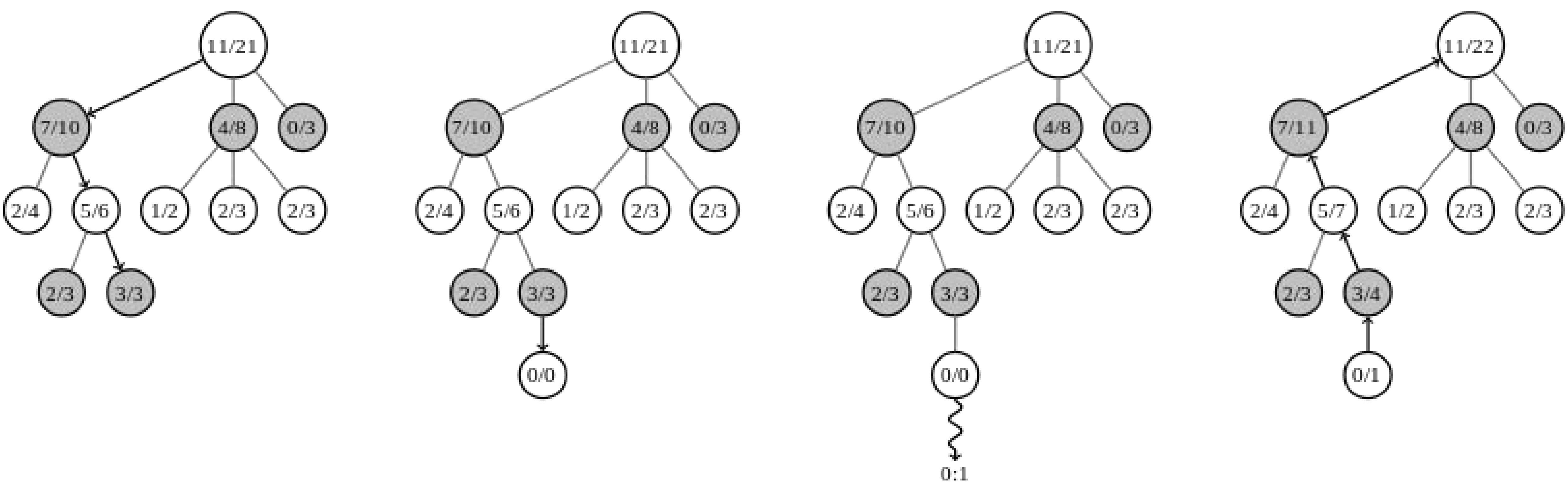
- **2015** – Wikipedia: *"Thus, it is very unlikely that it will be possible to program a reasonably fast algorithm for playing the Go endgame flawlessly, let alone the whole Go game."*

- **2015** – Google's AlphaGo defeats Fan Hui (2-dan player) in Go
- **2016** – Google's AlphaGo defeats Lee Sedol 4-1 (9-dan player) in Go

Conclusion

- MCTS expands the search tree based on random sampling of the search space (game board).

1. Descend   2. Create New Node   3. Simulate   4. Update



References

Mario: http://www.slideshare.net/ssuser7713a0/monte-carlo-tree-search-for-the-super-mario-bros

AlphaGo Full: http://airesearch.com/wp-content/uploads/2016/01/deepmind-mastering-go.pdf

AlphaGo Summary: https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/

Sample Tree