CS-E4800 Artificial Intelligence

Jussi Rintanen

Department of Computer Science
Aalto University

February 28, 2019

A Map to Decision-Making and Planning

one initial state	multiple initial states
deterministic actions	non-deterministic actions
full observability	partial observability
finite horizon	infinite horizon
one agent	multiple agents

decision-making by basic state-space search
decision-making under uncertainty (Markov decision processes)
game-theoretic / adversarial decision-making

Centralized versus Distributed

Centralized AI with Distributed Assets

Multi-actor systems (fleet of robots, vehicles, machines) under central control and full communication work like single-agent systems (lecture until now!)

Distributed AI and Multi-Agent Systems

- Collaboration of independent agents
- Competition of independent agents

(Independence = Agents have different "owners")

Autonomous Taxi Cab Service

Exactly like human taxi cab service!

Example

1. Central: Pick up passenger at 321 Garden St.?
2. Car A: Pick-up possible in 4 minutes!
3. Car B: Pick-up possible in 2 minutes!
4. Central: B is closer, car B will pick up.

Problem: Car better off lying about pick-up time $\rightarrow$ Passengers suffer!

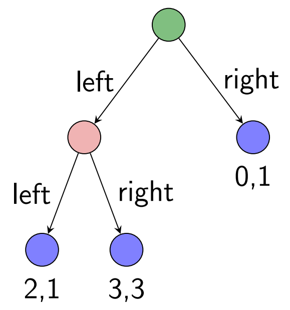
Collaborative situation, but selfishness of agents (or their owners) interferes with system's optimality

This Lecture

1. Basics of Game Theory: How to act in multi-agent scenarios?
   - Maximize own utility/profit
   - Beat the opponent in competition
   - Must reason about opponents' behavior
2. Methods for solving large-scale 2-player zero-sum games:
   - Game tree search: Minimax with $\alpha\beta$ pruning
   - Game tree search: Monte Carlo Search and Monte Carlo Tree Search
   - End-game databases and Transposition tables

   (When search depth is bounded, no guarantees about optimality!)

Strategies in Games



Strategy = choice of action at every stage in a game

		red player	
		left	right
green player	left	2,1	3,3
	right	0,1	0,1

Strategic Games (Normal Form Games)

Players choose strategies, and get pay-offs accordingly:

		player 2	
		strategy C	strategy D
player 1	strategy A	p_{1AC}, p_{2AC}	$p_{1AD}, p_{2AD},$
	strategy B	$p_{1BC}, p_{2BC},$	p_{1BD}, p_{2BD}

(Examples are 2-player, but everything generalizes to n player with $n \geq 2$.)

Dominated Strategies

Should Player 1 choose A or B?

	C	D
A	0,2	1,3
B	2,4	2,1

Strategy B is better for 1, no matter how 2 plays.

Assuming Player 1 is rational and will play B, Player 2 should play C (because $4 > 1$.)

Dominated Strategies and Iterated Strict Dominance

Definition

A strategy is strictly dominated if some other strategy is strictly better for every opponent strategy.

Theorem

1. *If all players are rational,*
2. *all players know that all players are rational and that everybody know this (common knowledge), and*
3. *elimination of strictly dominated strategies leaves one strategy for every agent,*

then these strategies will be played by the agents.

Prisoners' Dilemma

Two suspects are in detention, and asked to give information to police. Options are:

1. Cooperate with police, helping convict the other suspect
2. Not cooperate, getting a small sentence

	shut up	cooperate
shut up	-1,-1	-3,0
cooperate	0,-3	-2,-2

Best if both suspects shut up, if commitment to this could be enforced. But, assuming the other has already committed to a strategy, it is always best to cooperate!

Arms Race

Confrontation between two super-powers:

1. Arm: Acquire lots of weapons, exert threat
2. Don't arm

	don't arm	arm
don't arm	0,0	-3,1
arm	1,-3	-2,-2

Whatever the other does, it is always better to arm
However, (0,0) better than (-2,-2) ($\rightarrow$ Arms treaties)

Mixed Strategies

- When players have opposing goals, randomly selecting between strategies – which forces the opponent to randomize (and compromise his utility) – may produce the highest payoffs.
- example: Paper, Rock, Scissors
- example: bluffing and sand-bagging in Poker
  - With bad cards, bidding higher makes the opponents believe the cards are good
  - With good cards, not bidding higher makes the opponents believe the cards are bad

  Both must be used randomly, to confuse the opponents
  Both are part of optimal strategies for Poker

Examples of Mixed Strategies

Meeting

	A	B	C
A	1,1	0,0	0,0
B	0,0	1,1	0,0
C	0,0	0,0	1,1

versus...

Stalking

	A	B	C
A	-1,1	0,0	0,0
B	0,0	-1,1	0,0
C	0,0	0,0	-1,1

Paper, Rock, Scissors

	P	R	S
P	0,0	1,-1	-1,1
R	-1,1	0,0	1,-1
S	1,-1	-1,1	0,0

Formalization of Normal Form Games

Definition

1. There are players $1, \ldots, n$
2. Player i has a set of pure strategies A_i
3. A strategy profile is $(s_1, \ldots, s_n)$ such that $s_i \in A_i$ for every $i \in \{1, \ldots, n\}$
4. The utility $u_i(s_1, \ldots, s_n)$ of player i is expressed by a function $u_i : A_1 \times A_2 \times \cdots \times A_n \to \mathbb{R}$

Formalization of Normal Form Games

Definition

A mixed strategy σ is a probability distribution on the player's pure strategies A such that $\sum_{s \in A} \sigma(s) = 1$

Definition (Utility of Mixed Strategies)

The utility $u_i(\sigma_1, \ldots, \sigma_n)$ of player i w.r.t. a mixed strategy profile is

$$\sum_{(s_1,\ldots,s_n) \in A_1 \times \cdots \times A_n} u_i(s_1, \ldots, s_n)\sigma_1(s_1) \cdots \sigma_n(s_n)$$

Nash Equilibrium

Definition

A (mixed) strategy profile $(\sigma_1, \ldots, \sigma_n)$ is a Nash equilibrium if for all $i \in \{1, \ldots, n\}$,

$$u_i(\sigma_1, \ldots, \sigma_n) \geq u_i(\sigma_1, \ldots, \sigma_i', \ldots, \sigma_n) \text{ for all } \sigma_i' \neq \sigma_i$$

Nash Equilibrium: Intuitive meaning

NE is a (mixed) strategy profile in which no player can improve its pay-off by changing to some other strategy.

Nash Equilibrium

Disclaimer

It is not always clear how an NE could be reached and which NE would be played.
Nash equilibrium is realistic when

1. the agents fully know the game (the strategies, the pay-offs),
2. the agents cannot communicate and coordinate their actions,
3. the agents cannot bargain (re-distribute pay-offs after the game),
4. game is played only once,
5. there is only one NE.

Nash Equilibrium: Properties

1. Every game has at least one NE
2. Not all games have pure-strategy NE
3. Every pure strategy in an NE is an equally good response to the opponents' strategies (randomization "only" needed to force opponents to randomize)

For 2-player zero-sum games, NE can be found with Linear Programming. More general games require more complex methods.

Example: Mixed Strategies

Bach or Stravinsky (aka Battle of the Sexes, with Ballet and Soccer)

	B	S
B	2,1	0,0
S	0,0	1,2

Nash equilibria in pure strategies: $\langle B, B\rangle, \langle S, S\rangle$, with respective payoffs $\langle 2, 1\rangle, \langle 1, 2\rangle$.
Nash equilibrium $\langle \sigma_1, \sigma_2\rangle$ in mixed strategies with

$$\sigma_1(B) = \tfrac{2}{3} \quad \sigma_1(S) = \tfrac{1}{3}$$

$$\sigma_2(B) = \tfrac{1}{3} \quad \sigma_2(S) = \tfrac{2}{3}$$

and payoffs $\langle \frac{6}{9}, \frac{6}{9}\rangle$.

Tragedy of the Commons

Using jointly-owned resource; cost evenly shared

	0	2	4	6
0	0,0	-1,1	-2,2	-3,3
2	1,-1	0,0	-1,1	-2,2
4	2,-2	1,-1	0,0	-1,1
6	3,-3	2,-2	1,-1	0,0

Using your own resource

	0	2	4	6
0	0,0	0,0	0,0	0,0
2	0,0	0,0	0,0	0,0
4	0,0	0,0	0,0	0,0
6	0,0	0,0	0,0	0,0

Best action: Spend joint resource as much as you can
(Or (under diminishing marginal utility) at least: Spend it more than you would if you had to pay for it in full.)

(Flatmates agree to fill-up fridge every day, divide the cost evenly, and let everybody eat as much as they like. Good idea?)

Game Tree Search

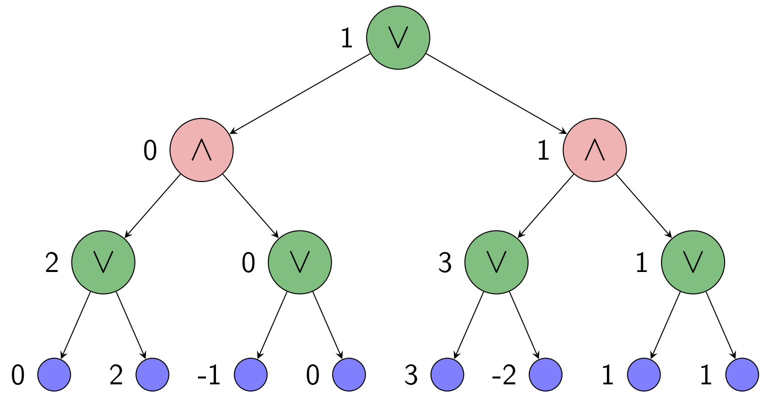
- Two-person multi-stage zero-sum games
- player wins, opponent loses, or vice versa, or it's a draw
- Board games: Checkers, Chess, Backgammon, Go
- Other applications? (Military operations?)
- Issue 1: very large search trees
- Issue 2: focusing search difficult
- Issue 3: search cannot be exhaustive due to I1 (suboptimality)

Basic Game Tree Search by Minimax

- Depth-first search of bounded depth AND-OR tree
- Leaf nodes evaluated with a heuristic value function
  Chess: pieces' value, relative position (mobility, safety of king, ...)
- Values of non-leaf nodes by min or max of children
- AND-nodes (opponent) by minimization
- OR-nodes (player) by maximization

(Special case: whole game-tree covered, winning leafs 1, losing leafs -1, and draws 0)

Minimax Tree Search



The Minimax Algorithm

```
procedure minimax(player,state,depthLeft)
  if depthLeft = 0 or no action is applicable then return value of state;
  player2 := the player other than player;
  if player is minimizing then best := $\infty$ else best := $-\infty$;
  for each action that is applicable for player in state do
    state2 := successor of state w.r.t. action;
    v := minimax(player2,state2,depthLeft-1);
    if player is minimizing
    then best := min(best,v)
    else best := max(best,v);
  return best;
```

Top-level call: minimax(startingplayer,startingstate,depthbound)

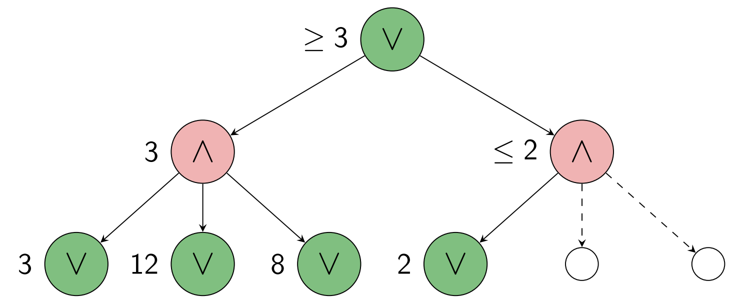
Alpha-Beta Pruning

Idea behind Alpha-Beta Pruning

$\min(x, \max(y, z)) = x$ if $x \leq y$ (α cuts)
$\max(x, \min(y, z)) = x$ if $x \geq y$ (β cuts)
In both cases, z is irrelevant.

Utility of $\alpha\beta$: search tree may be pruned to $\frac{1}{10}, \frac{1}{100}, \frac{1}{1000}, \ldots$

Alpha-Beta pruning example



The Alpha-Beta Algorithm

```
procedure alphabeta(player,state,depthLeft,alpha,beta)
  if depthLeft = 0 or no action is applicable then return value of state;
  player2 := the player other than player;
  if player is minimizing then best := $\infty$ else best := $-\infty$;
  for each action that is applicable for player in state do
    state2 := successor of state w.r.t. action;
    v := alphabeta(player2,state2,depthLeft-1,alpha,beta);
    if player is minimizing
    then best := min(best,v); beta := min(beta,v)
    else best := max(best,v); alpha := max(alpha,v)
    if alpha $\geq$ beta then break;
  return best;
```

Top-level call:
alphabeta(startingplayer,startingstate,depthbound,$-\infty$,∞)

Transposition Tables

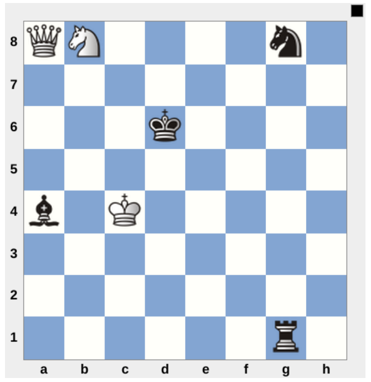
- Depth-first used in games like chess because astronomic state spaces: algorithms that require storing all visited states not feasible.
- Need to utilize memory for pruning, without exhausting it
- DFS can reach a state in multiple ways
  $\Longrightarrow$ Multiple copies of the same subtree
- Transposition tables: Cache states encountered during DFS; retrieve value of already-encountered states, rather than repeating search
- When table full, delete low-importance states

Endgame Databases

- In games with a limited number of simple (late) states/configurations, compute their value by exhaustive game-tree search and store for later use.
- Another form of caching, constructed once, before game-playing

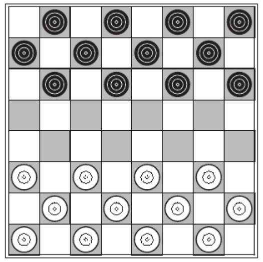
Endgame databases

- All ≤ 7 piece states solved in 2012
- 7-piece DB is 140 TB; 6-piece DB is 1.2 TB
- Black to check-mate in 545 moves:

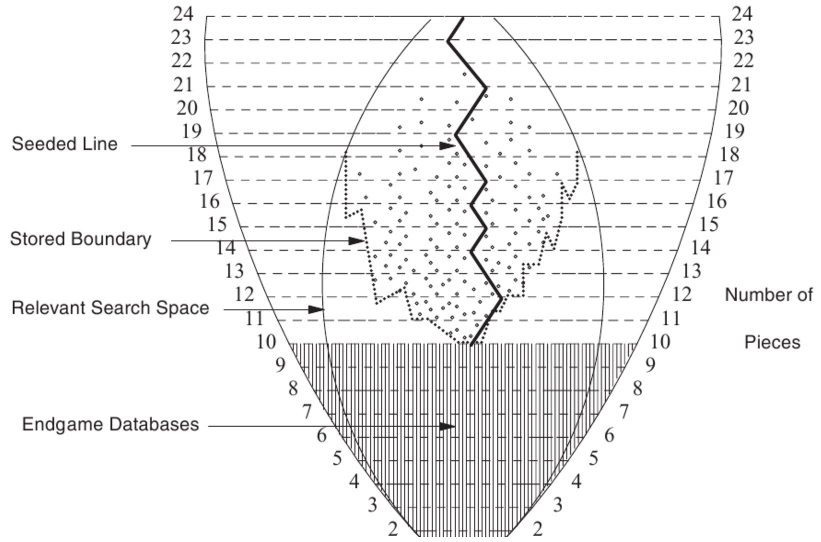


Checkers is solved

- Optimal play of Checkers ($5 \cdot 10^{20}$ states) ends in a draw (Schaeffer et al., 2007)
- The solution consists of:
  - AND-OR tree from initial state ($\sim 10^7$ nodes)
  - Leaf nodes evaluated from endgame database with all ≤ 10 piece positions: consists of $3.9 \cdot 10^{13}$ states; computations took 2001-2005



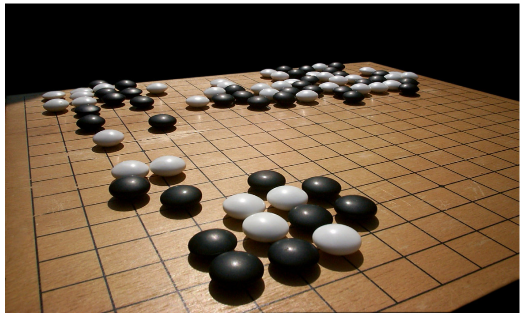
Checkers is solved



Monte Carlo Methods

- DFS not working well for some types of games
  - Too many states
  - Heuristics don't guide search well
  - Information gained during search not utilized
- Monte Carlo methods
  - Sample randomly full game-plays
  - Focus search according to promising game-plays
  - Works even without heuristics, e.g. for Go
- Similar methods used also for large MDPs (e.g. in robotics)
- Disclaimer: For lots of games $\alpha - \beta$ beats MC methods!!!

Go



- Two-player fully-observable deterministic zero-sum board game
- Has been a big challenge for computers

Rules of Go

Go is played on 19×19 square grid of points, by players called Black and White.
Each point on the grid may be colored black, white or empty.

A point P, not colored C, is said to reach C, if there is a path of (vertically or horizontally) adjacent points of P's color from P to a point of color C.

Clearing a color means emptying all points of that color that don't reach empty.

Starting with an empty grid, the players alternate turns, starting with Black.

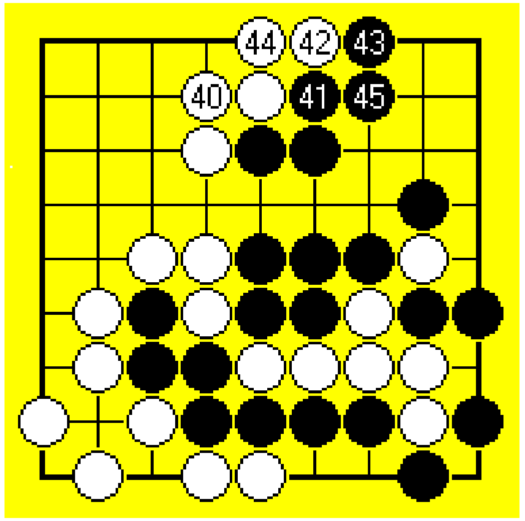
A turn is either a pass; or a move that doesn't repeat an earlier grid coloring.

A move consists of coloring an empty point one's own color; then clearing the opponent color, and then clearing one's own color.

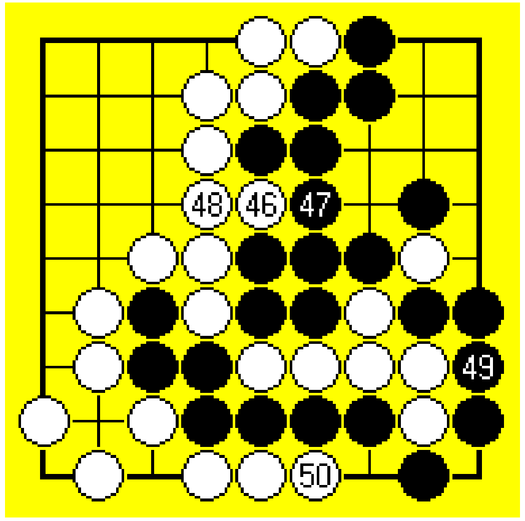
The game ends after two consecutive passes.

A player's score is the number of points of her color, plus the number of empty points that reach only her color. White gets 6.5 points extra.
The player with the higher score at the end of the game is the winner.

Example game of 9×9 Go



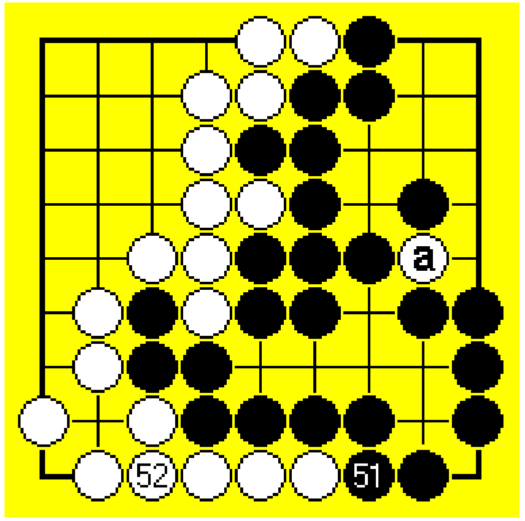
Example game of 9×9 Go



Why is Go difficult for computers?

- Go is visual and thus easy for people
  Could not show 10 Chess moves in one image
- Branching factor far larger than in Chess (avg. 250 vs. 35)
- Evaluation of board configurations difficult

Example game of 9×9 Go

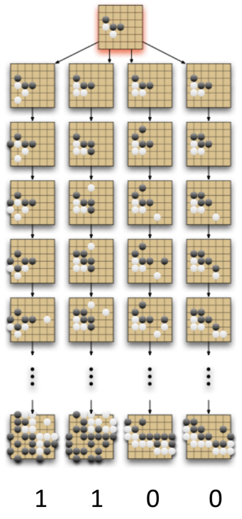


Paradigm shift in 2006

- Computer Go was progressing slowly
  (weak amateur level)
- In 2006, Monte-Carlo methods surpassed traditional tree search
- In 2015
  - All competitive programs use Monte Carlo
  - 19×19 is strong amateur level
  - 9×9 is professional level
  - 5×6 is solved, solving 6×6 feasible
- In 2016, board-evaluation by neural networks → AlphaGo beats human champions

Monte Carlo Search (MCS)

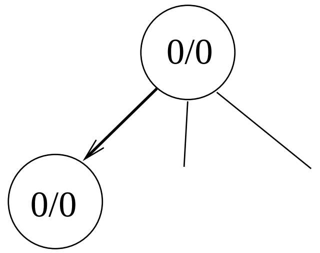
- Try out every possible action
- *Several* randomized plays:
    - Choose actions randomly
    - Stop only after game ends
- Score each gameplay according to who wins
- Best action is one with most wins
- Notice: No search *tree* here, only evaluation of current action alternatives



Monte Carlo Tree Search (MCTS)

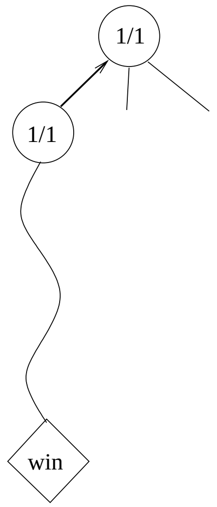
- Extension of simulation/sampling-only Monte Carlo search
- Generate a search tree, with leafs evaluated by randomized simulation
- Allows to focus on most promising gameplays

Example (Single Agent)

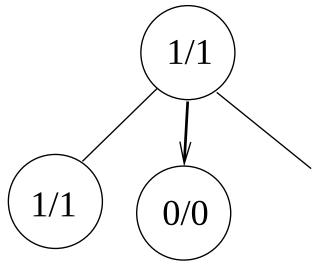


Show number of wins/trials for each node

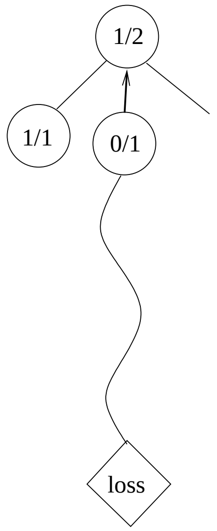
Monte Carlo Tree Search



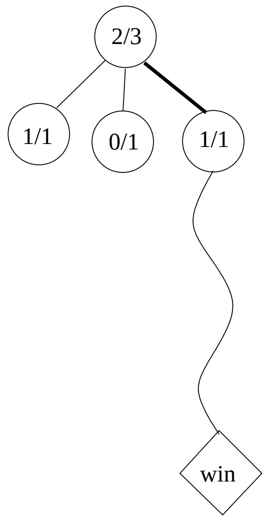
Monte Carlo Tree Search



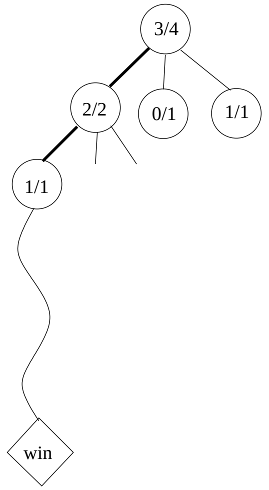
Monte Carlo Tree Search



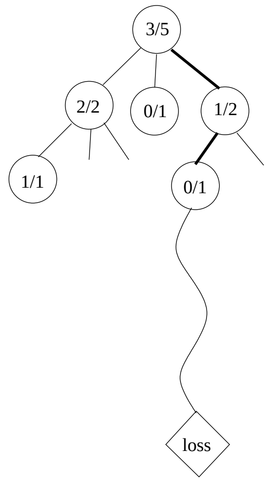
Monte Carlo Tree Search



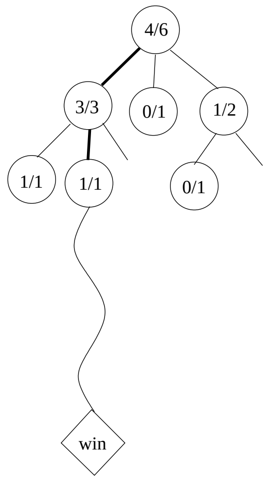
Monte Carlo Tree Search



Monte Carlo Tree Search



Monte Carlo Tree Search



Monte Carlo Tree Search

- Which tree node to choose for next expansion or trial?
- Incomplete information: results of previous trials
- Choose one with few trials with high rewards
  (low confidence)
- Choose one with many trials with lower rewards (high confidence)