RL: Lecture 19

Harvey Mudd College

April 8, 2020

Neil Rhodes

Course Logistics

MCTS HW 9 is out

~~Quiz 9 postponed~~

MCTS Programming Assignment 6

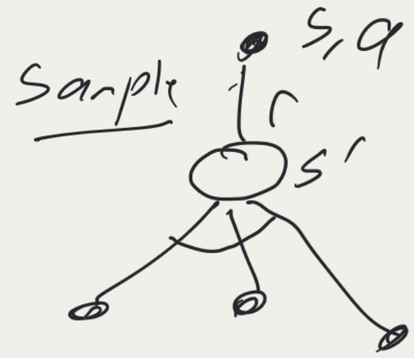
Midterm 2

Monday's lecture

Expected updates > sample updates?

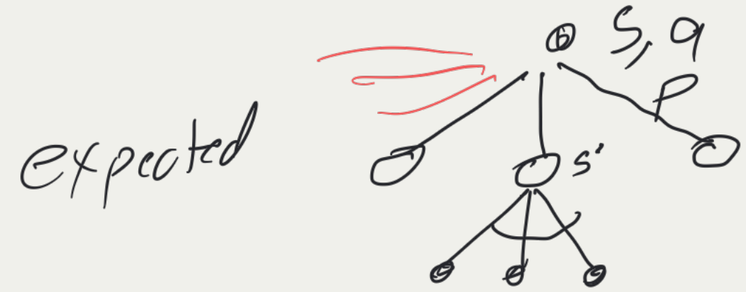
Sample update (less computation):

$$Q(s,a) = Q(s,a) + \alpha[R + \gamma \max_{a'} Q(S', a') - Q(s,a)]$$

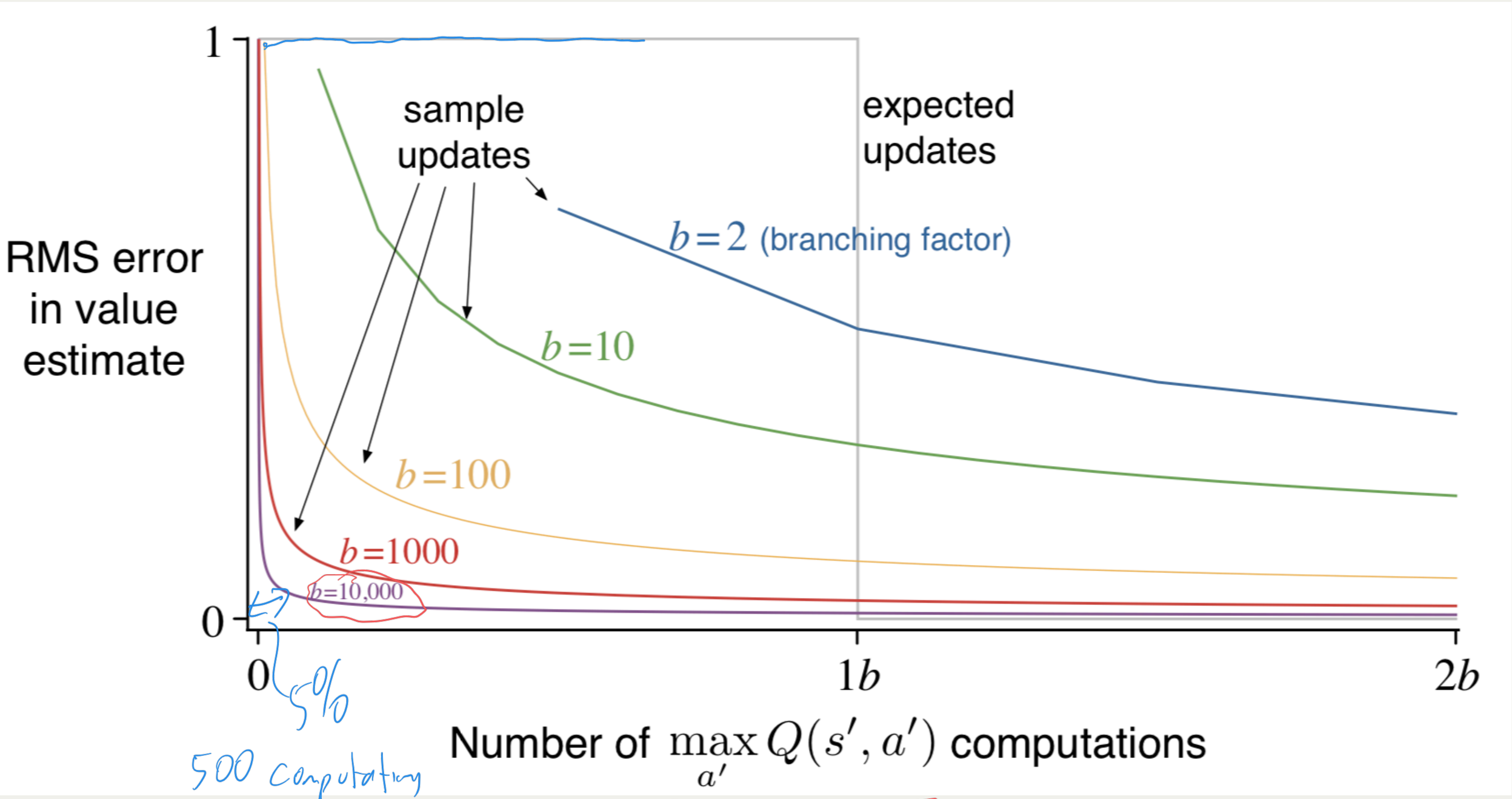


Expected update (more accurate): could be expensive

$$Q(s,a) = \sum_{s',r} p(s', r|s,a)\alpha[r + \gamma \max_{a'} Q(s', a')]$$



Expected updates > sample updates?



500 computations

5%

1) Sample updates can
1) more quickly update $Q(s,a)$ allowing other updates to be more accurate
2) approximate min q values in time it takes to do an exact update

Trajectory Sampling

alternative to uniform or prioritized sweeping

Sampling:

- Uniform

Dyna-Q+

choose a state-action pair @ random
DO one-step update

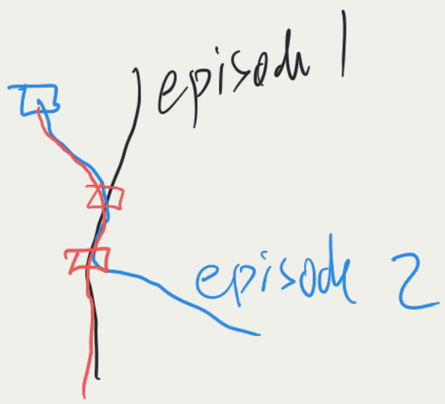
vs.

- According to on-policy distribution

episode seq of state/action from environment

trajectory seq of state actions from model

trajectory Sampling
start w/ a start state
keep sampling until we run out of time or complete an episode



When to plan

- In the background
- At decision time

Always be planning

when a decision is to be made, do some planning

What is a rollout algorithm?

term comes from backgammon

these days we don't physically roll the dice

1. Begin with current state
2. Simulate trajectories starting with that state (following a rollout policy)

start w/ rollout policy

3. Choose action with highest estimated value

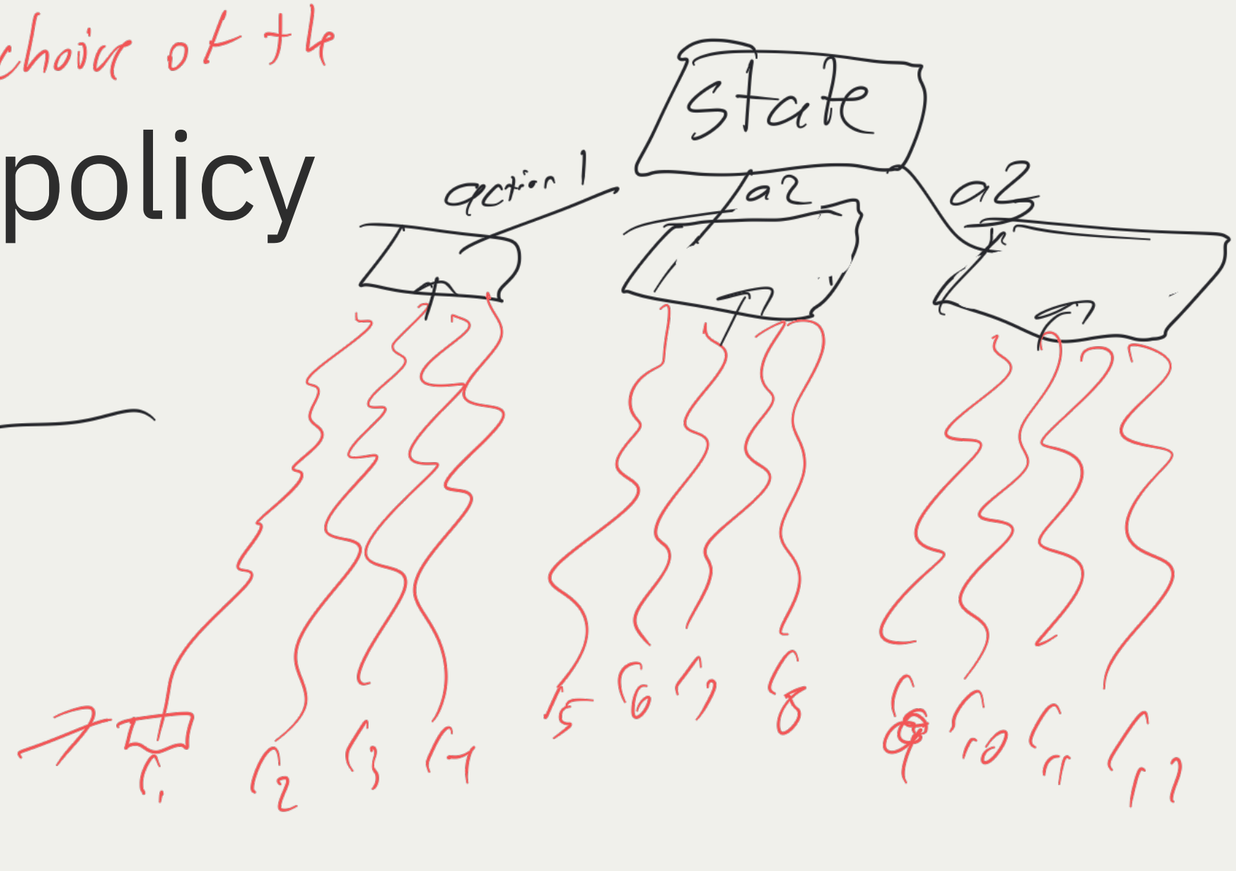
A rollout algorithm improves the rollout policy

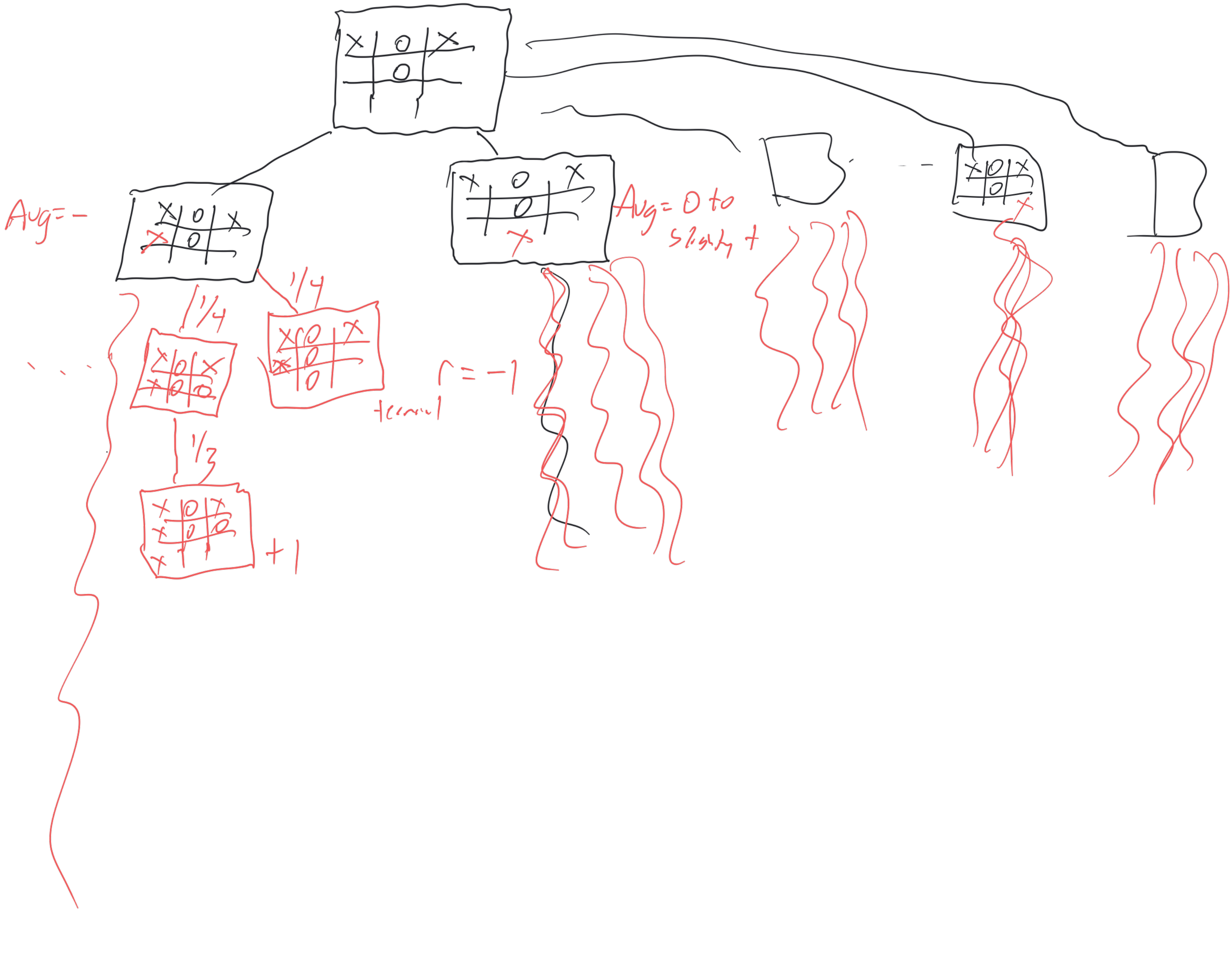
current action choice of the

$\pi(\text{start state}) \rightarrow a_3$

---

often, the rollout policy is uniform random

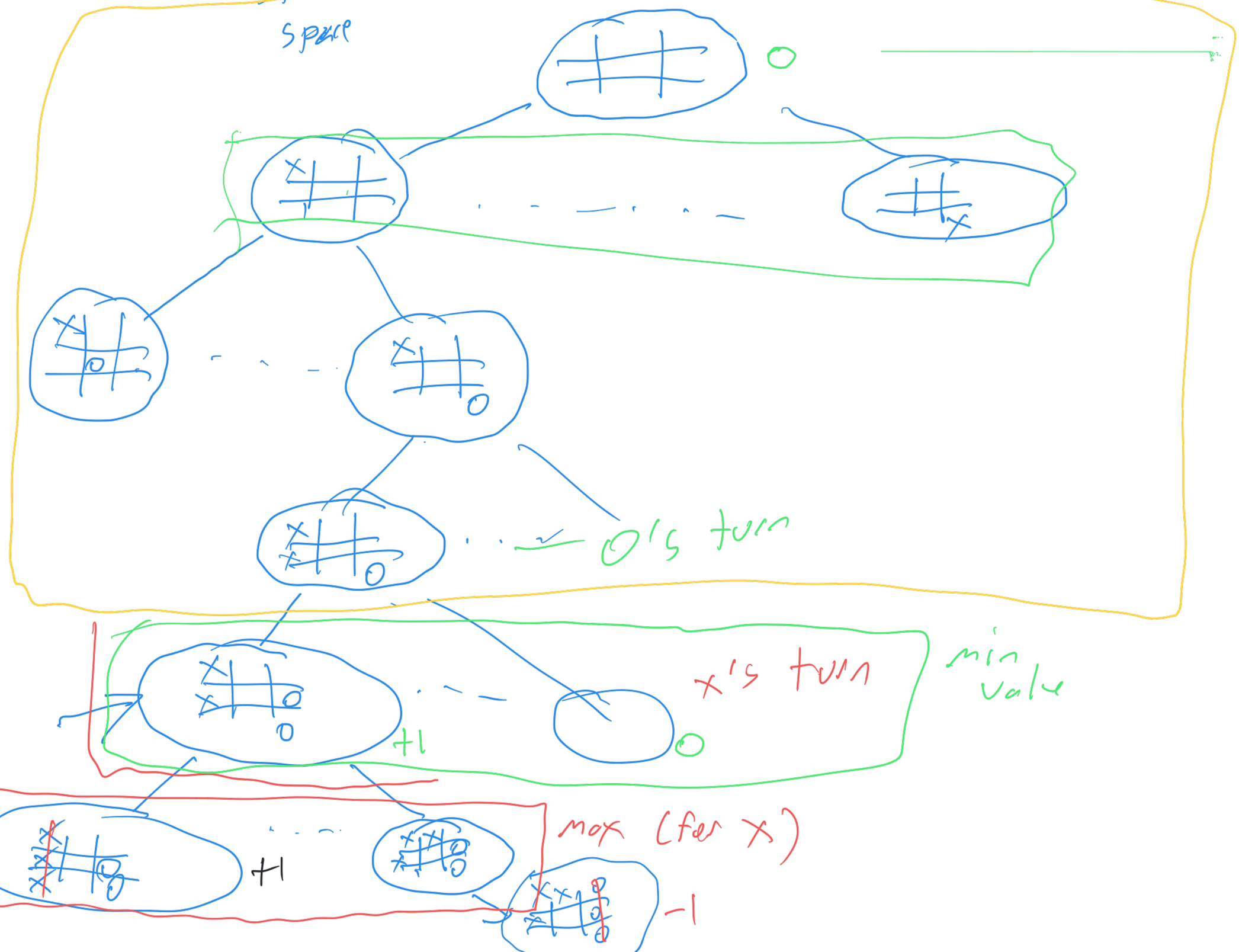



Avg = -
1/4
1/4
1/4
1/3
+1
terminal
r = -1
Avg = 0 to slightly +

Searching a state space

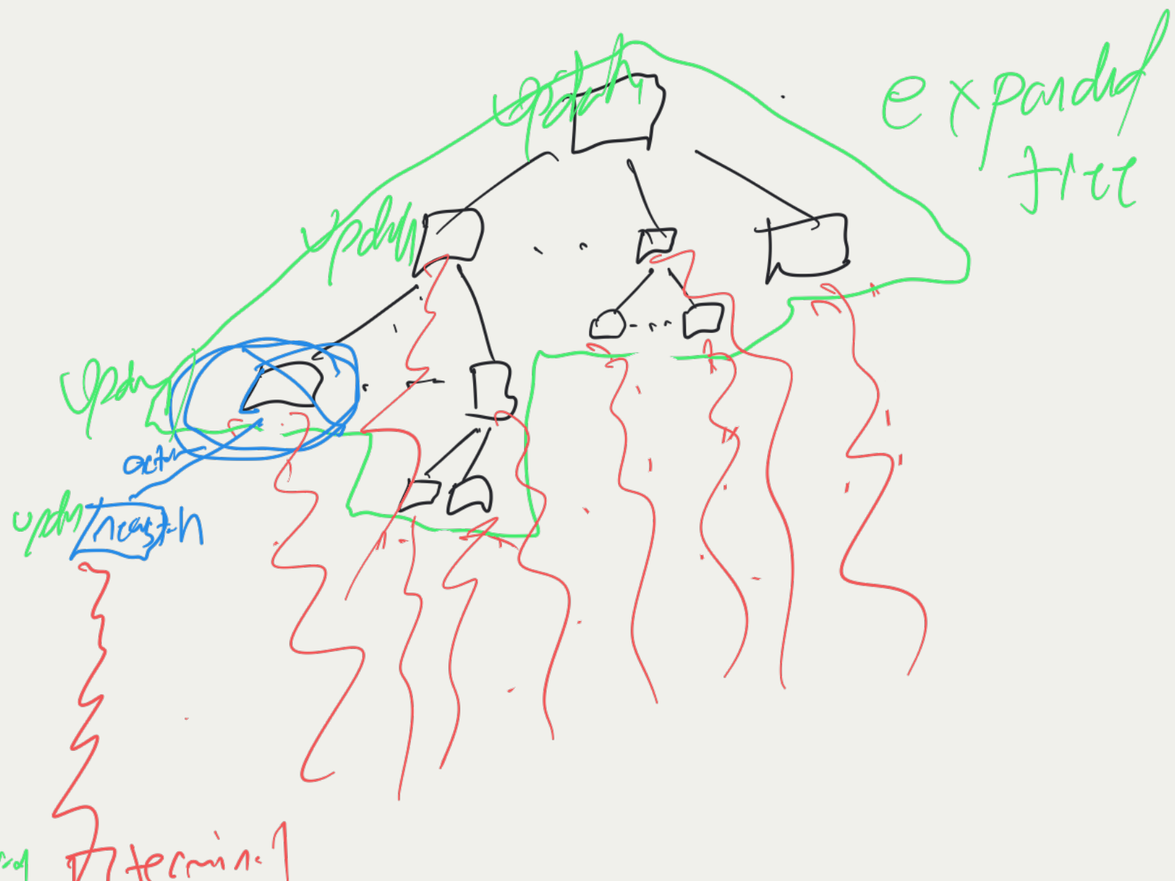
Search state space

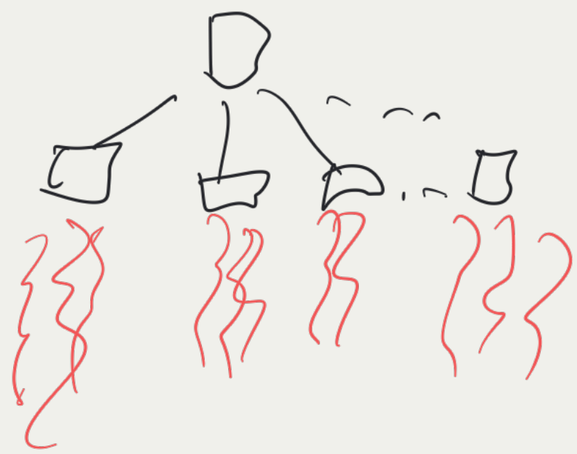
2-player competitive game



Monte Carlo Tree Search (MCTS)

Repeat while time remaining starting with current state:
1. Selection: Select a leaf node in the expanded tree
2. Expansion: Expand a child of the leaf node
3. Simulation: Follow rollout-policy from expanded node to simulate complete episode
4. Backup: Backup action values to nodes in the tree





Selection: choose a leaf node in the MCTS tree

Traverse MCTS tree until reach a node with unexpanded children.

Use Upper Confidence Bound for Trees (UCT) to decide most promising child.

$q(v)$: Total simulation reward for node v

$n(v)$: Total number of visits (simulation backups) for node v

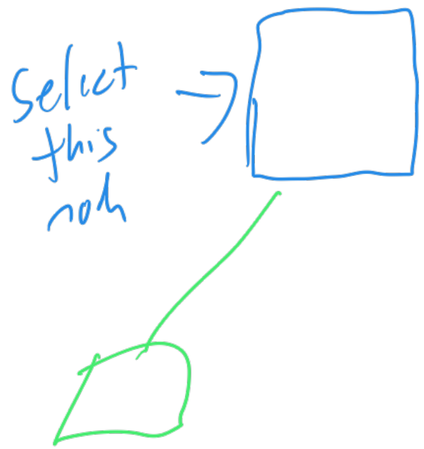
$$UCT(v) = \frac{q(v)}{n(v)} + c\sqrt{\frac{\log n(v.parent)}{n(v)}}$$

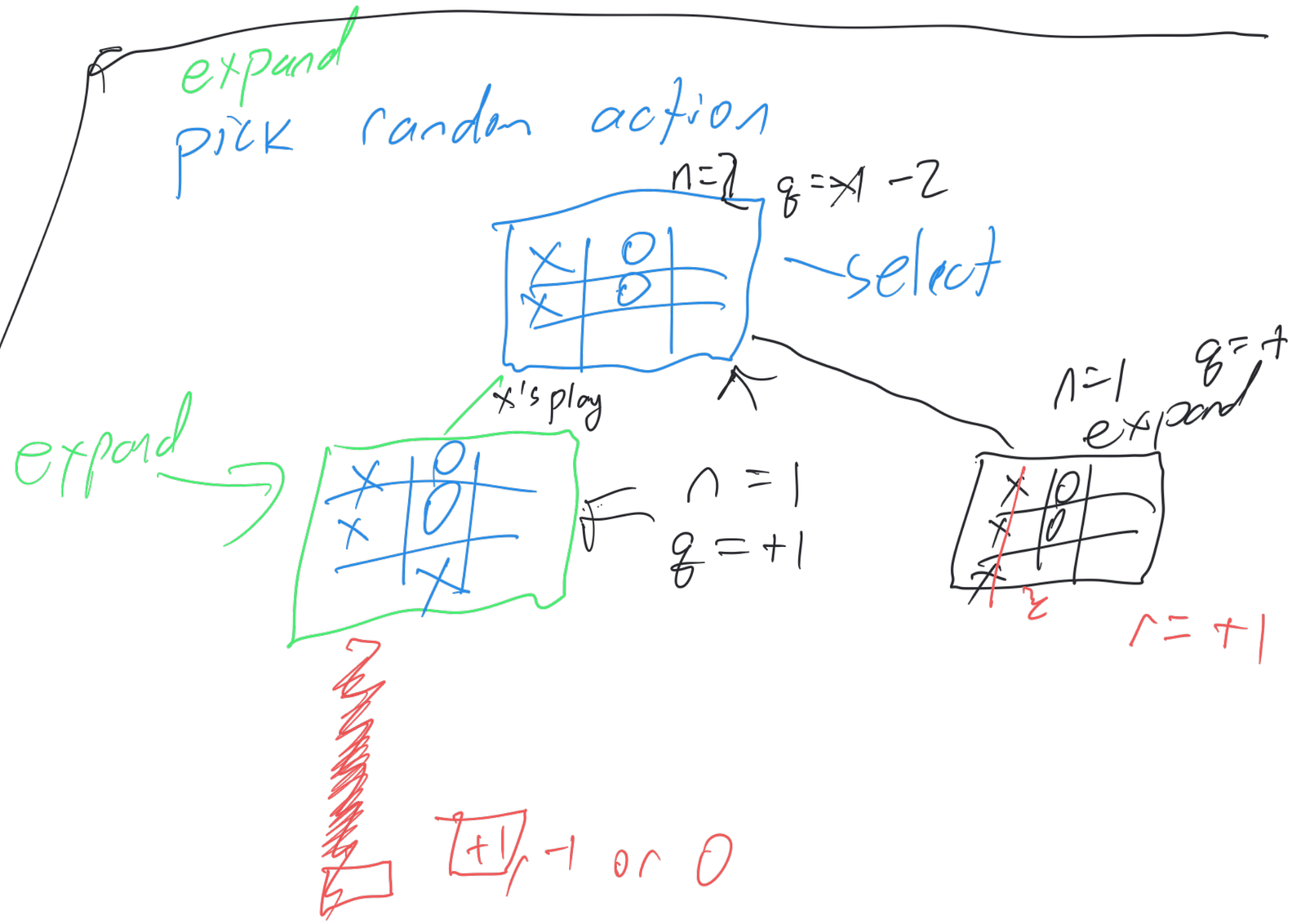
exploitation

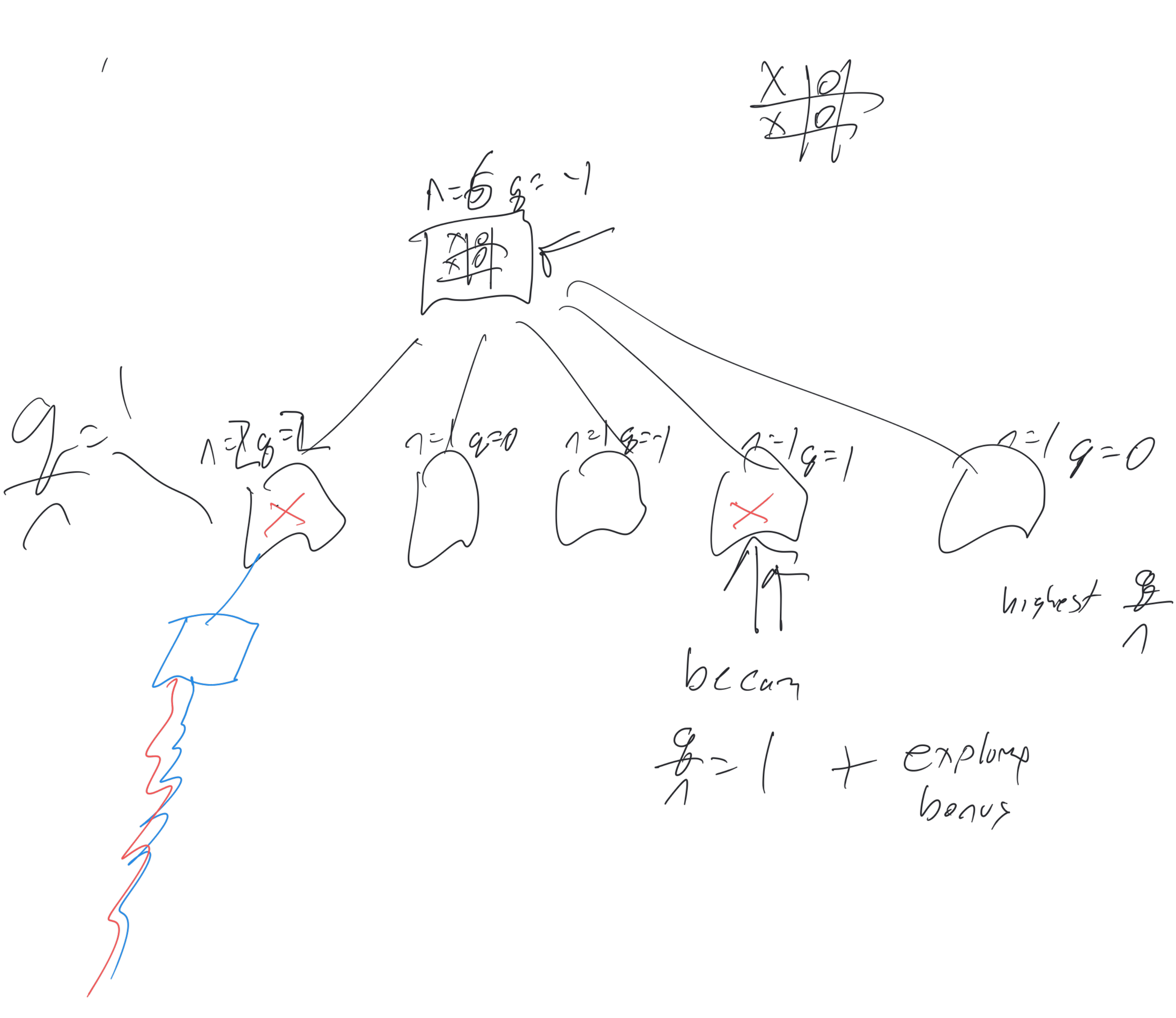
exploration

use uct

first starting MCTJ






highest
becam
= 1 + explomp bonus

Expansion: expand selected node

If selected node is not terminal, choose an untried action and and create a new MCTS node for the state that generates

Simulation: rollout starting at expanded node

Monte Carlo simulation using rollout policy until terminal state is reached.

Record total reward.

Backup: backup action values to nodes in the MCTS tree

Update $n(v)$ and $q(v)$ for each node v in the MCTS tree from simulation node up to root.

Note: If two-player competitive game, adjust reward to reflect who made the move.

For example, if reward is +1 (player 1 won):

- For v reached from player 1 move, increment $q(v)$
- For v reached from player 2 move, decrement $q(v)$

Selecting a final action

Probably don't want exploration term in UCT

- Child with highest $\frac{q(v)}{n(v)}$, or
- Child of root with highest $N(v)$—it's the one that was explored the most so must have been most promising overall.