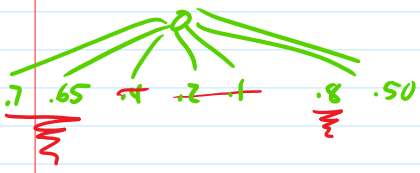
Monte Carlo Techniques

Flat Monte Carlo: for each action simulate to terminal using random play
choose action with highest observed average

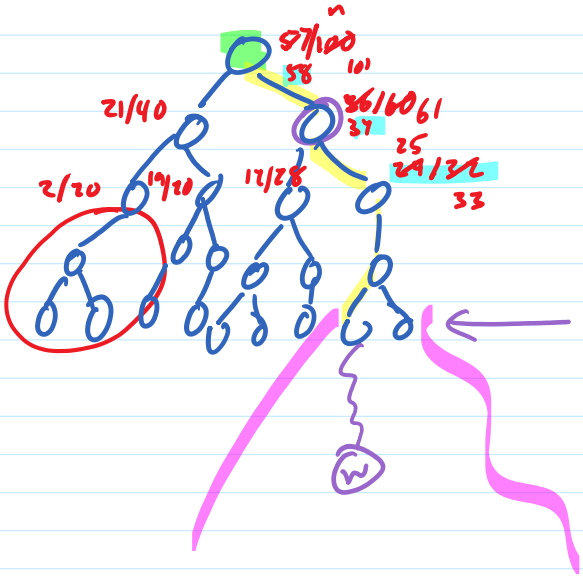


worked well in Scrabble, Bridge

Combine with UCB: choose action max $\bar{r}_j + \sqrt{\frac{2 \ln T}{n_j}}$

Combine with tree search:
(Flat UCB)

build tree to depth d
traverse to leaf using UCB at each level
at leaf do random playout (move randomly until end)
propagate stats back



Grow Tree asymmetrically: Monte Carlo Tree Search

Multi-Armed Bandit

Given unknown probability distributions $R_1, \dots, R_K$ with means $\mu_1, \dots, \mu_K$ $\quad \mu^* = \max_i \mu_i$

Choose indices $i_1, i_2, \dots$ to optimize payout

Regret = difference between reward obtained and best possible expectation

$$\rho_T = \underbrace{T}_{\text{total plays so far}} \cdot \mu^* - \sum_{t=1}^{T} \hat{r}_t$$

$\hat{r}_t$ = reward at time t (what we got from arm i_t at time t)

"optimal" means "zero regret"

$$P\left(\lim_{T\to\infty} \frac{\rho_T}{T} = 0\right) = 1$$

Ex:

Arm 1		Arm 2		Arm 3	
prob	payout	prob	payout	prob	payout
$\frac{1}{3}$	2	$\frac{1}{4}$	$\frac{3}{2}$	$\frac{1}{100}$	200
$\frac{2}{3}$	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{99}{100}$	0
		$\frac{1}{2}$	0		

$\mu_1 = \frac{1}{3}\cdot 2 + \frac{2}{3}\cdot 0 = \frac{2}{3}$

$\mu_2 = \frac{1}{4}\cdot\frac{3}{2} + \frac{1}{4}\cdot\frac{1}{2} + \frac{1}{2}\cdot 0 = \frac{1}{8}$

$\mu_3 = \frac{1}{100}\cdot 200 + 0 = 2 = \mu^*$

uniform rotation: cycle through each arm $\quad 1\ 2\ 3\ 1\ 2\ 3 \dots$

$$\lim_{T\to\infty} \frac{\rho_T}{T} = \lim_{T\to\infty} \frac{T\cdot\mu^* - (\hat{r}_1 + \hat{r}_2 + \hat{r}_3 + \dots)}{T}$$

$$\frac{T\cdot\mu^*}{T} - \lim_{T\to\infty} \frac{(\hat{r}_1 + \hat{r}_2 + \hat{r}_3 + \dots)}{T} \qquad 0$$

$$\mu^* - \lim_{T\to\infty} \left(\frac{\hat{r}_1 + \hat{r}_4 + \hat{r}_7 + \dots}{T} + \frac{\hat{r}_2 + \hat{r}_5 + \dots}{T} + \frac{\hat{r}_3 + \hat{r}_6 + \dots}{T} \right)$$

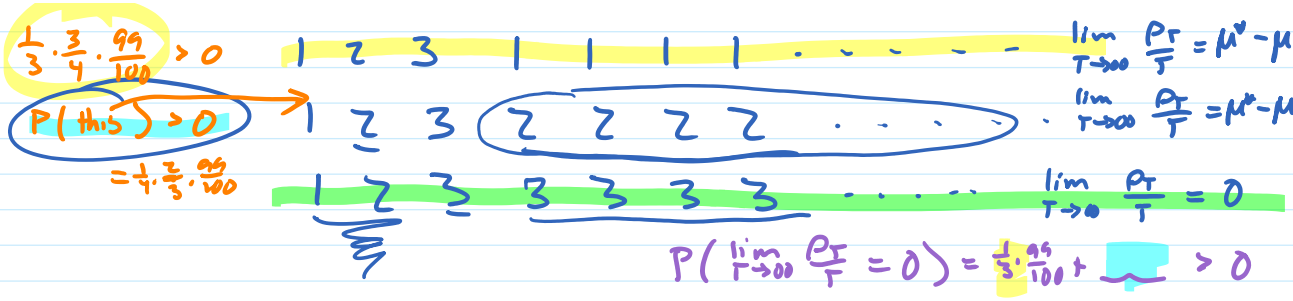
$$\mu^* - \left(\tfrac{1}{3}\mu_1 + \tfrac{1}{3}\mu_2 + \tfrac{1}{3}\mu_3\right)$$

$$\tfrac{1}{3}(\mu^* - \mu_1) + \tfrac{1}{3}(\mu^* - \mu_2) > 0$$

($\mu^* - \mu_1 > 0$, $\mu^* - \mu_2 > 0$)

greedy: play each once, play arm with highest payout forever

$\frac{1}{3}\cdot\frac{3}{4}\cdot\frac{99}{100} > 0$ $\quad 1\ 2\ 3\ 1\ 1\ 1\ 1 \dots$ $\quad \lim_{T\to\infty} \frac{\rho_T}{T} = \mu^* - \mu_1 > 0$



ε-greedy : play one round, play best observed with prob $1-\varepsilon$
play randomly with prob ε

$$\lim_{T\to\infty} \frac{\rho_T}{T} > \underbrace{\frac{\varepsilon}{K}}_{>0} \cdot \underbrace{(\mu^* - \mu_i)}_{>0} > 0$$

prob of playing non-optimal arm i is at least this

zero regret
Aver

Choose arm j that maximizes $\bar{r}_j + \sqrt{\frac{2\ln T}{n_j}}$

UCB
(upper confidence bound)

exploitation: $\bar{r}_j$ — observed mean reward for arm j

exploration term: $\sqrt{\frac{2\ln T}{n_j}}$ — tunable parameter; total plays; total plays for arm j

Monte Carlo Tree Search

Until out of time

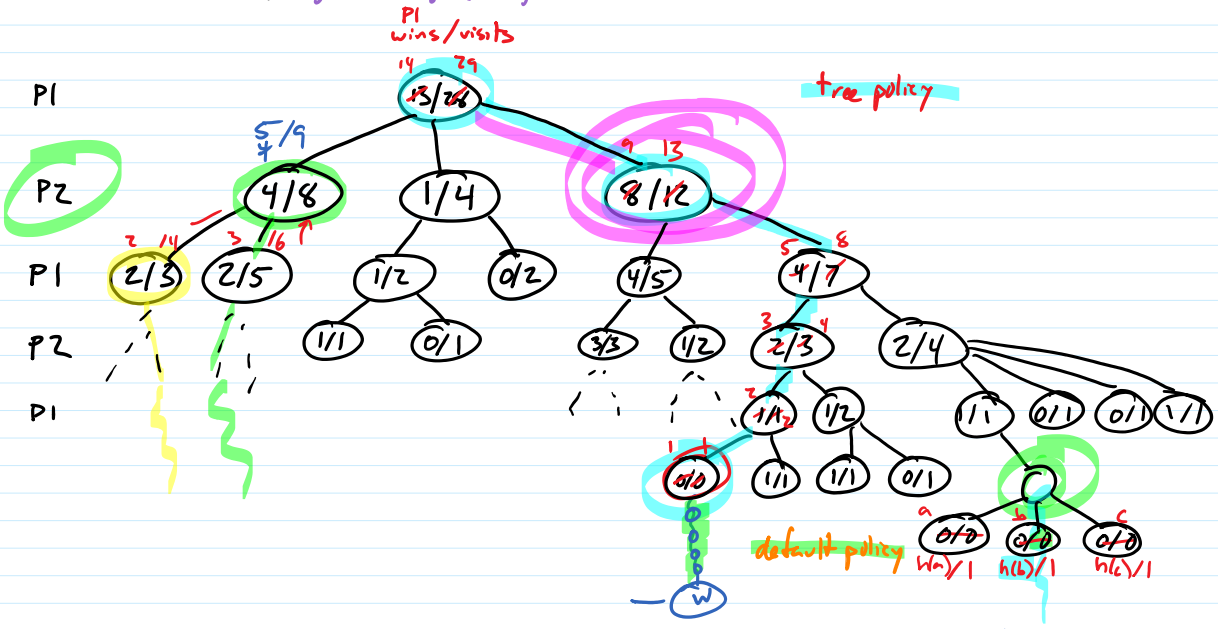
traverse tree — root → expandable node (not all children in tree yet) or terminal node
tree policy

expand — add missing child (choose arbitrarily)

simulate — play to end from added child
default policy

update — back propagate stats along path root → expanded child

select move — based on current stats
↳ highest average (or highest visit count)



UCT = MCTS w/ UCB as tree policy (usually random as default policy)
UCB for trees

advantages: convergent — converges to minimax

anytime — returns a move after any number of iterations

no domain knowledge — no heuristic

easily parallelized

- leaf — multiple parallel playouts from newly added nodes
- tree — parallel traversals in same tree (requires locking)
- root — separate tree on each CPU, combine results (add statistics, take highest mean or count) or vote

disadvantages: no domain knowledge — doesn't use heuristic (but can add in)

some games not amenable

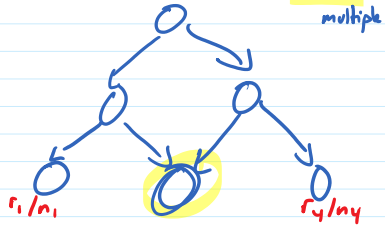
trap state — state where most moves are bad, one good move — MCTS takes a long time

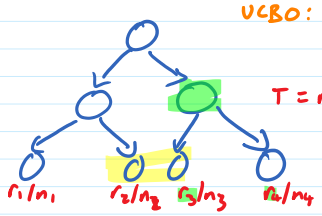
chess has many → MCTS alone doesn't work well

so doesn't → MCTS works well

MCTS adapted for games that aren't trees

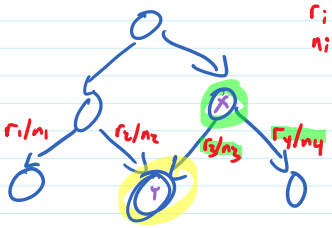
(some states reachable through multiple paths)





UCB0: ignore multiple paths; duplicate states if reachable along different paths

$T = n_3 + n_4$



r_i = total reward after traversing edge

n_i = total times edge traversed

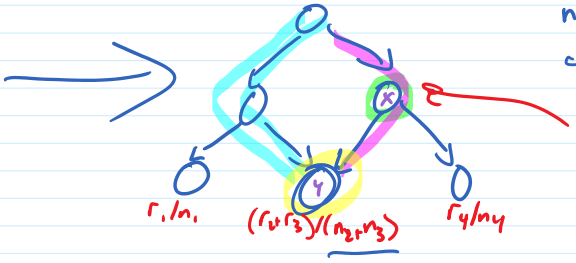
UCB1: ignore other paths (= UCB0 but without duplicates)

ex: when evaluating $Y \in X$ compute

$$\frac{r_3}{n_3} + \sqrt{\frac{c \ln (n_2 + n_y)}{n_3}}$$

UCB2: combine multiple edges in exploit term

$$\frac{r_2 + r_3}{n_2 + n_3} + \sqrt{\frac{c \ln (n_2 + n_y)}{n_3}}$$



no guarantee on convergence (that I have seen)

$$\frac{r_2 + r_3}{n_2 + n_3} + \sqrt{\frac{c \ln (n_2 + n_3 + n_y)}{n_2 + n_3}}$$