Monte Carlo Techniques

Flat Monte Carlo: for each action, simulate to terminal position using random play

choose action w/ highest average

successful in Scrabble, Bridge

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

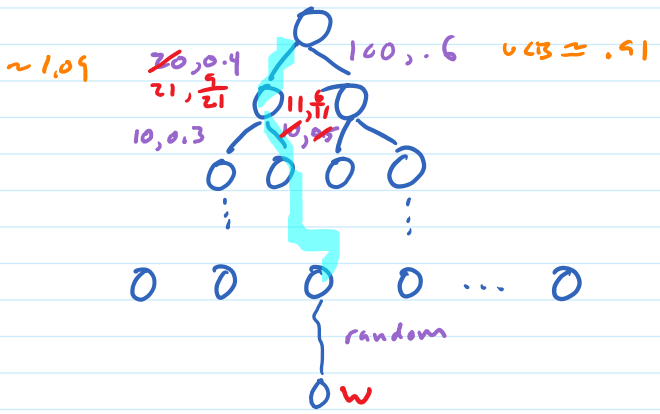
$\bar{r}_j$: obs reward for move j

n_j: # times done playout from move j

Combine with tree search:
(Flat UCB)

build tree to depth d
traverse to leaf according to UCB (note: for min player, minimize $\bar{r}_j - \sqrt{\frac{2 \ln T}{n_j}}$)

T: obs reward for max player



Grow Tree asymmetrically: Monte Carlo Tree Search

Monte Carlo Tree Search

while not out of time / resources

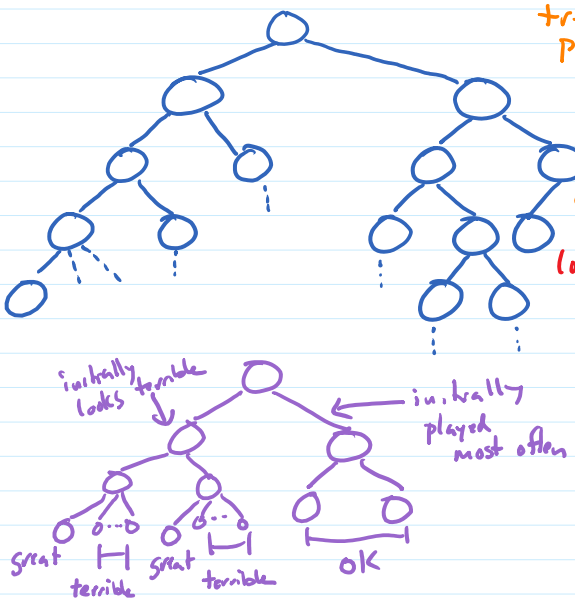
tree policy: traverse tree root → expandable node (has child not in tree) or terminal node

add child

default policy (often random): playout from child

backpropagate result

return best child of root
- highest observed reward
- or move played most often
- (or keep going until same)



UCT = MCTS with UCB as default policy

UCB for trees

advantages: convergent - converges to minimax

anytime - can give suggested move after any iteration
more fine-grained than iterative deepening

no domain knowledge - no heuristic

easily parallelized

leaf parallel - many random playouts in parallel

root parallel - build separate trees in parallel, combine results for branches from root @ end

tree parallel - many traversals/playouts on one tree in parallel
lock tree for updates
(useful when time for playout >> time for update)

disadvantages: no domain knowledge

some games not amenable
trap states - positions with a move that leads to quick loss
common in Chess (MCTS bad)
not in Go (MCTS good)