Backup plans:

If this Zoom meeting ends prematurely, five-minute break, check Piazza
Multi-Armed Bandit

$$a_1 \quad a_2 \quad a_3 \quad a_4 \quad \ldots \quad a_n$$

$$R_1 \quad R_2 \quad R_3 \quad R_4 \quad \ldots \quad R_n$$
Multi-Armed Bandit for Game Tree Search
Key ideas of Monte Carlo Tree Search:

1. View move selection as a multi-armed bandit problem
2. Evaluate moves by simulating games
Multi-Armed Bandit for Game Tree Search

What move should I try?
Multi-Armed Bandit for Game Tree Search

Simulate games with each “arm”
Multi-Armed Bandit for Game Tree Search

What move should I try on each simulated game?
Monte-Carlo Tree Search (MCTS) Terms

• Leaf node:  
A state in the game tree that has successors for which no games have been simulated  
(has one or more “arms” that have never been pulled)

• Terminal node: End of game state

• Playout/rollout: Simulating a game from a leaf node to a terminal node
Three Steps in MCTS

• Selection: Make move choices until a leaf node S is reached
Three Steps in MCTS

(a) Selection
Three Steps in MCTS

(a) Selection
Three Steps in MCTS

(a) Selection
Three Steps in MCTS

(a) Selection
Three Steps in MCTS

• Selection: Make move choices until a leaf node S is reached
• Expansion: Create a new successor state S’ for an untried action
  Simulation: Play a game until you reach a terminal node
Three Steps in MCTS

(a) Selection

(b) Expansion and Simulation

black wins
Three Steps in MCTS

• Selection: Make move choices until a leaf node S is reached
• Expansion: Create a new successor state S’ for an untried action
  Simulation: Play a game until you reach a terminal node
• Backpropagation: Update game statistics for the path from S’ up to the root
Three Steps in MCTS

(a) Selection
black wins

(b) Expansion and Simulation

(c) Backpropagation
MCTS(state):
while TIME-REMAINING() do
    leaf ← SELECT(tree)
    child ← EXPAND(leaf)
    result ← SIMULATE(child)
    BACKPROPAGATE(result, child)
return argmax \#playouts(apply(a, state))
    a∈A
MCTS(state):
while TIME-REMAINING() do
    leaf ← SELECT(tree)
    child ← EXPAND(leaf)
    result ← SIMULATE(child)
    BACKPROPAGATE(result, child)
return argmax \#playouts(apply(a, state))
\text{a} \in A

Which move gives the game state with most playouts
Three Steps in MCTS

How do we pick moves?

(a) Selection
(b) Expansion and Simulation
(c) Backpropagation
Remember This?
(UCB)

Algorithm:

Pull each arm once
For $i \leftarrow 1$ to $n$  \{  $\text{Sum}_i \leftarrow R(\text{arm}_i)$;  $\text{N}_i \leftarrow 1$;  $\text{N} \leftarrow n$  \}  /* Initialization */
Loop Forever

$\text{best} \leftarrow \text{argmax} \left[ \frac{\text{Sum}_i}{\text{N}_i} + c \sqrt{\frac{\ln \text{N}}{\text{N}_i}} \right]$

pull arm $a_{\text{best}}$ and get reward $r$

$\text{Sum}_{\text{best}} \leftarrow \text{Sum}_{\text{best}} + r$;  $\text{N}_{\text{best}} \leftarrow \text{N}_{\text{best}} + 1$;  $\text{N} \leftarrow \text{N}+1$
Picking a Move During Selection and Expansion (UCT – Upper Confidence bound applied to Trees)

\[
\text{Sum}_i = \# \text{ of wins} \\
N_i = \# \text{ of times } i \text{ was tried} \\
N = \# \text{ of simulations thus far (N(parent(i))}
\]

\[
\text{best} \leftarrow \arg\max_{1 \leq i \leq n} \left[ \frac{\text{Sum}_i}{N_i} + c \sqrt{\frac{\ln N}{N_i}} \right]
\]

Lets you control how much exploration
Three Steps in MCTS

How do we pick moves?

(a) Selection

(b) Expansion and Simulation

black wins

(c) Backpropagation
Three Steps in MCTS

How do we pick moves?
UCB heuristic

(a) Selection
(b) Expansion and Simulation
(c) Backpropagation
Three Steps in MCTS

(a) Selection
(b) Expansion and Simulation
(c) Backpropagation

How do we pick moves?
Picking a Move During Simulation

• Light playout: Pick uniformly at random

• Heavy playout: Make a biased selection
  • Simulation statistics
  • Game knowledge

Trade off: Slower run time vs missing a move
Benefits

• Doesn’t use an evaluation function!
• Time is linear in depth
• Handles large number of actions
• Let’s you make a move when a timer goes off (to manage time)
  (“anytime algorithm”)

28