Markov Decision Processes

Online Algorithms

CS4246/CS5446

AI Planning and Decision Making

Online search

So far

- Sequential decision problems & MDP

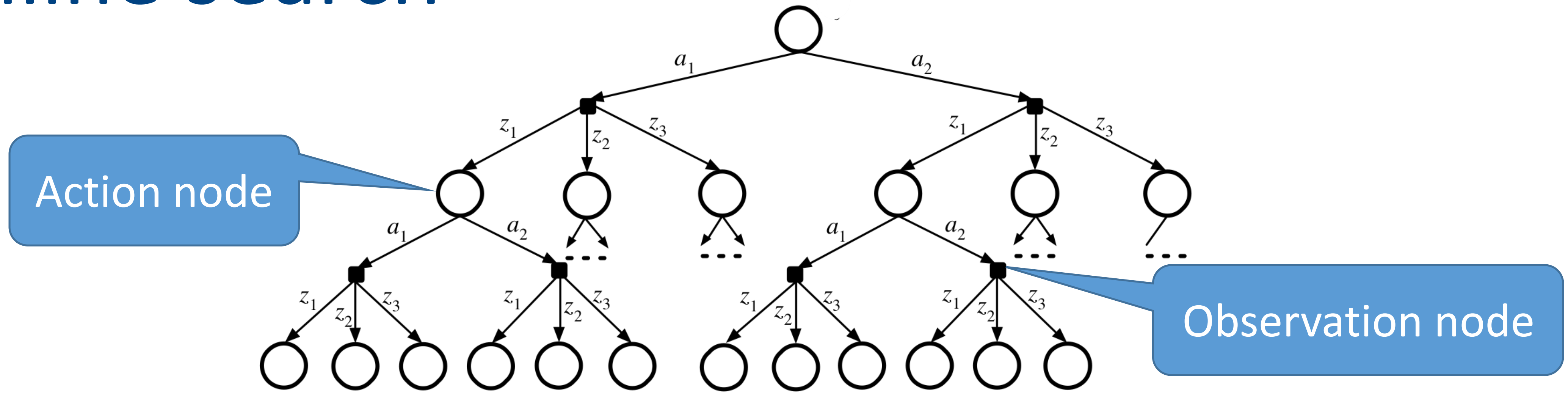
- Solution mechanisms
  - Value iteration
  - Policy iteration

Curse of dimensionality

- State space grows exponential with the number of variables

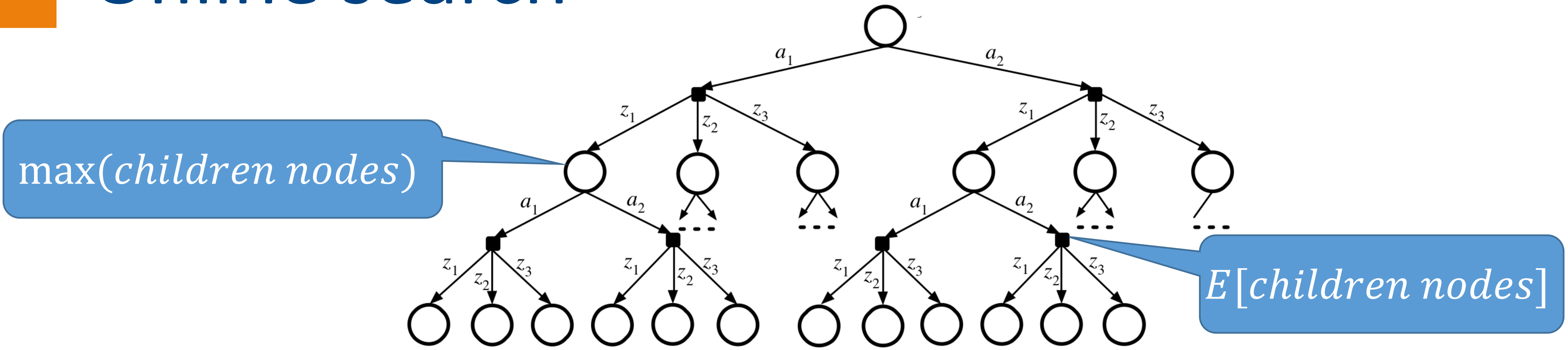
- VI/PI iterates through all states; so exponential with the number of variables
- To handle curse of dimensionality
  - Use function approximation
    - Linear function of features
    - Deep neural networks
  - Do online search with sampling

Online search



- At every step, construct a search tree
  - Up to a fixed depth D
  - Root is the current state
  - $|A|$children of the root (and other action nodes)
  - $|S|$ children of observation nodes

Online search

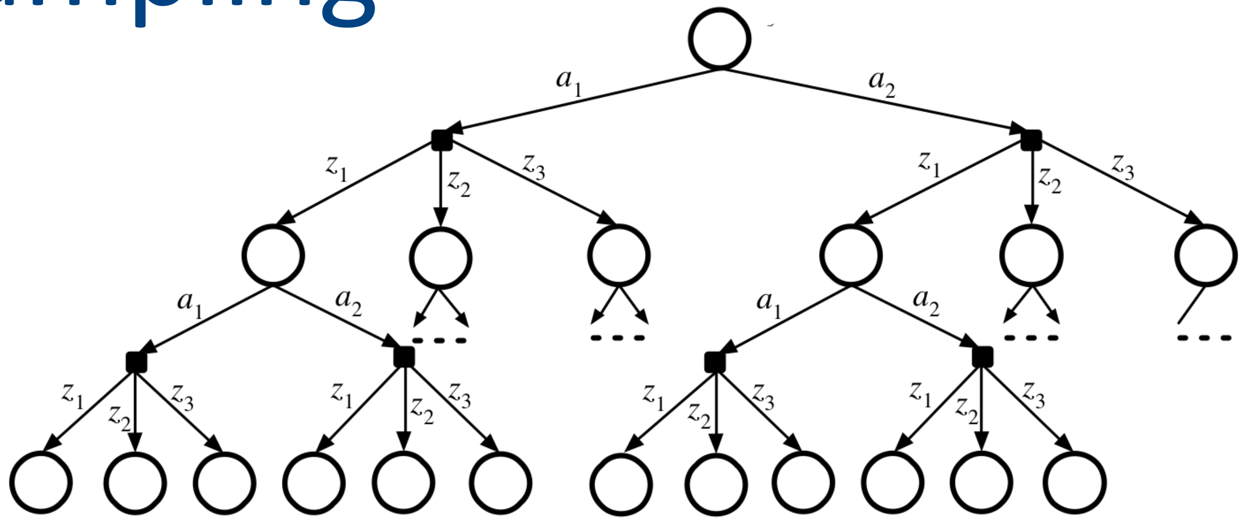


- To compute the value at the root:
  - Initialize leaf with value estimates (or zeros)
  - At observation node, compute the expected values of the children
  - At the action nodes, compute the max of the children

Q: Have we fixed the curse of dimensionality?

Sparse sampling

Don’t search the entire tree!

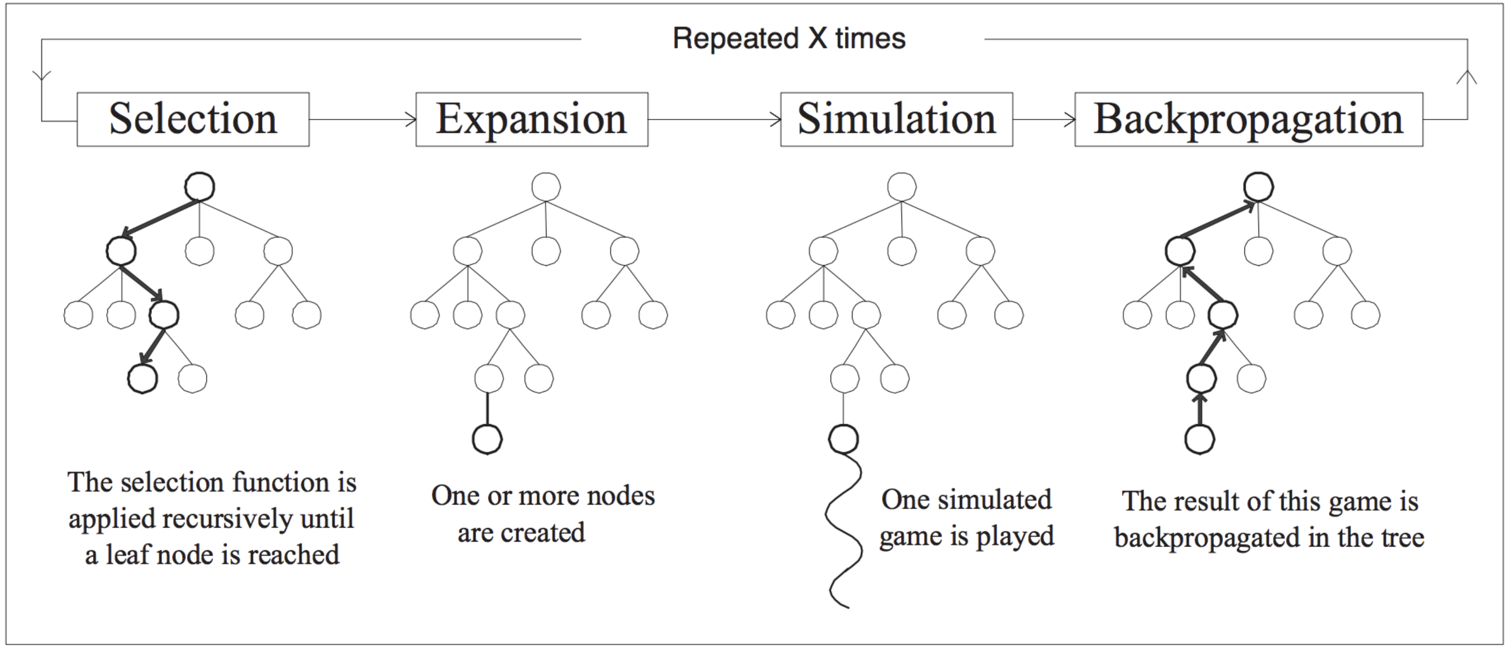


- Tree size: $|A|^D |S|^D$
- Sparse sampling[1]
  - Estimate by sampling k observations at observation nodes instead of using $|S|$ states as possible observations
    - Tree size now: $|A|^D k^D$
    - Curse of dimensionality is solved …
      … but still exponential with the search depth – curse of history

[1]Michael Kearns, Yishay Mansour, and Andrew Y Ng. “A sparse sampling algorithm for near-optimal planning in large Markov decision processes”. In: Machine learning 49.2-3 (2002), pp. 193-208.

Monte Carlo Tree Search

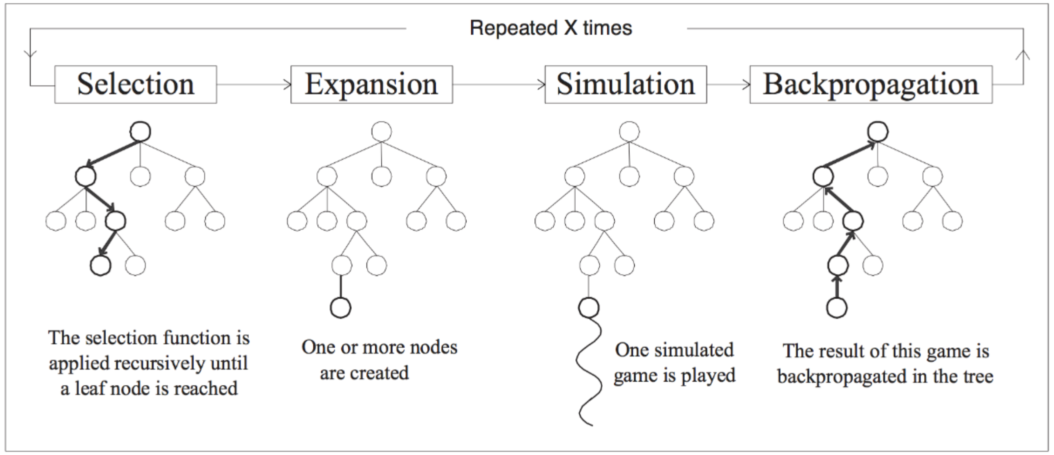
MCTS[2]



- Commonly used to solve MDP and games

[2]Guillaume Chaslot et al. “Monte-Carlo Tree Search: A New Framework for Game AI”. In: AIIDE. 2008.

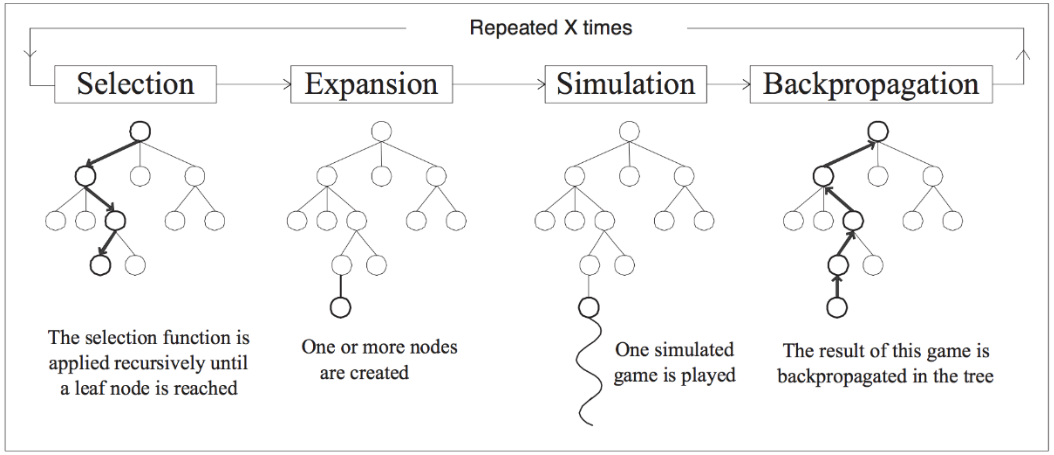
MCTS

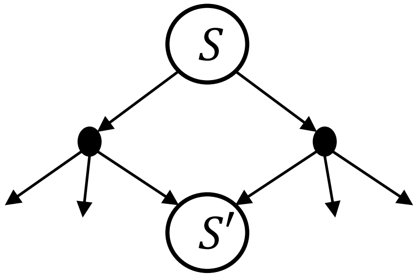


- Repeatedly run trials from the root (current state in online search)
- Trial:
  - Repeatedly select the node to go to at the next level until
    - Target depth is reached, or
    - Selected node has not been discovered – create a new node; run a simulation using a default policy till required depth
  - Backup the outcomes all the way to the root

- Anytime policy: When time is up, use the action that looks best at the root at that time

MCTS



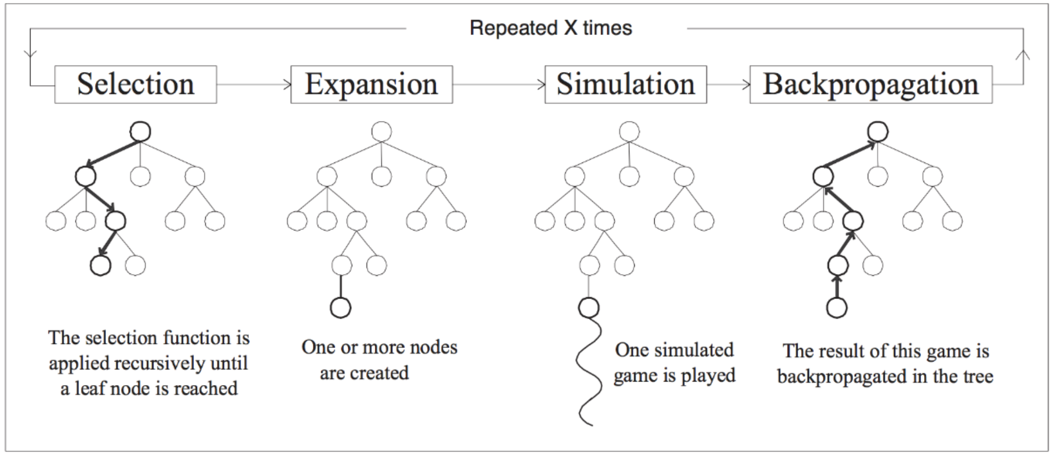


- For MDP, a tree (actually DAG) node n is associated with a state s

- A node n' at the next level is selected by applying an action a to s
  - Sample next state s' (corresponding to n') according to $P(s'|s, a)$

- Action a is selected to balance exploration with exploitation

MCTS



Estimated

- Estimated value $\hat{V}(n)$ at node n is the average return of all trials at n
  - Return $r_t(n)$ of trial t starting from n with state s and next node n' is
    $$r_t(n) = R(s) + \gamma r_t(n')$$
- Estimated Q-function at n, $\hat{Q}(n, a)$, is the average return of all trials at n that start with action a

Estimated

  - $\hat{Q}(r, a)$ at the root r is used to select the action to take at the root node
- All these are updated in the backup operation to the root

Upper confidence tree (UCT)

UCT[3]

Used for tuning the performance

Confidence interval

Being greedy

- Action selection is guided by this function:

$$\pi_{UCT}(n) = \arg\max_{a} \hat{Q}(n,a) + c\sqrt{\frac{\ln\big(N(n)\big)}{N(n,a)}}$$

  - Where, $N(n)$ and $N(n,a)$ count the number of trials through n and (n,a) respectively and c is a constant
- UCT will eventually converge to the optimal policy with enough trials
  - Worst case can be very bad[4] $\Omega(\overbrace{\exp(\exp(\ldots\exp}^{D-1\text{ times}}(1)\ldots)))$
  - Often works well in practice
    - E.g., PROST planner[5], which uses UCT, won the international probabilistic planning competition in 2011 & 2014

[3]Levente Kocsis and Csaba Szepesvari. “Bandit based monte-carlo planning". In: European conference on machine learning. Springer. 2006, pp. 282-293.

MCTS in practice

Player 2 uses MCTS

- Visualizing MCTS:
  - https://www.youtube.com/watch?v=FvRSxNLTg7U&ab_channel=DaveDyer

MCTS in practice – AlphaGo Zero

- AlphaGo Zero[6]
  - Uses MCTS + Approximate Policy Iteration
  - Play against Self to learn
  - Defeated AlphaGo (that beat Lee Sedol) 100-0!

- Go has a state space size of about 10^{170}
  - Need function approximation to represent value and policy functions
- AlphaGo Zero uses deep neural network with two heads (outputs)
  - Value head – outputs real value estimate of the value function (board position)
  - Policy head – outputs a vector of size 19×19 (maps board position to action)
    - Each item represents the probability that the policy will play that board position

Incorrect in the video

[6]David Silver et al. "Mastering the game of Go without human Knowledge". In: Nature 550.7676 (2017), p. 354.

AlphaGo Zero – MCTS

- Variant of UCT that exploits the policy head output: $P(s,a)$

$$\pi_{UCT}(s) = \arg\max_a \hat{Q}(s,a) + c\,P(s,a)\sqrt{\frac{\sum_b N(s,b)}{1+N(s,a)}}$$

  - When leaf node is reached, the value head output is used to evaluate the state instead of doing a roll-out (simulation)

- Go is a zero-sum, turn taking game instead of MDP
  - Search alternates between:
    - selecting action that maximizes when it is first player's turn
    - selecting the action that minimizes for second player's turn
  - At termination: reward $+1$ for the first player win and -1 for second player win

AlphaGo Zero – Approximate PI

- Recollect: Policy iteration has 2 stages, policy evaluation and policy improvement
- AlphaGo Zero does both using supervised learning
- With the current value function, MCTS can be viewed as a policy improvement operator – gives improved policy values for the evaluated states
- Self-play with search gives the policy evaluation for the evaluated states
- Supervised learning is used to interpolate the values and the policy over the whole domain using data from a set of states

Reading

- Sutton and Barto [Section 8.11]
- Sutton and Barto [Section 2.7]
- [RN] 16.2.4, 6.4 (Online algorithms, MCTS)

Thank you!