Multi-agent games

Player A tries to maximize chance of A winning

Player B tries to minimize chance of A winning

Agents move in alternating turns

Environment

- Fully observable
- Deterministic and strategic
- Static
- Discrete
- Multi-agent

Design a utility function for each outcome of a game

MAX nodes (you): try to maximize utility

MIN nodes (adversary): try to minimize utility
Minimax Algorithm

- Stop at a certain depth, using heuristic utility instead of expanding
- Incentive: Do not have enough time/space to search all the way to the terminal
- Problem
  - Horizon effect: Catastrophic move just past stopping point
  - Remedy: Continue searching until no more moves of big impact

Time complexity: $O(b^m)$

DFS Behavior

Static Analysis

Alpha-Beta Pruning

- If the path you evaluating is worse than previous branch, do not bother to evaluate any more.
- Alpha: Best case for MAX
- Beta: Best case for MIN
- Prune whenever worse than current Alpha or Beta
Randomness in game
- Stochastic environment
  - e.g. dice rolling

Chance node
- A chance node between agent’s move constraints their decisions
  - Calculation: \( \text{Sum} \ \text{all children} \ (\text{Prob}(\text{child}) \times \text{Utility}(\text{child})) \)
Problem formulation

- Variables
- Domains
- Constraints

Picking variables
- Pick one with smallest branching factor first (MRV: minimum remaining values)
- If MRV returns 0, fail immediately

Picking values
- Look for most promising value (LCV: least constrained values)

Forward checking vs. arc consistency
- FC: single direction
- AC: both direction

K-consistency
- For any consistent assignment of k-1 variables, there exists consistent value for any kth
- k = 1: node consistency
- k = 2: arc consistency