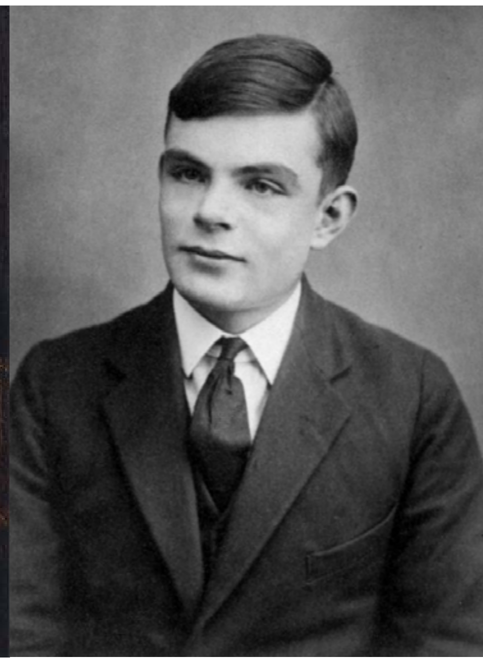
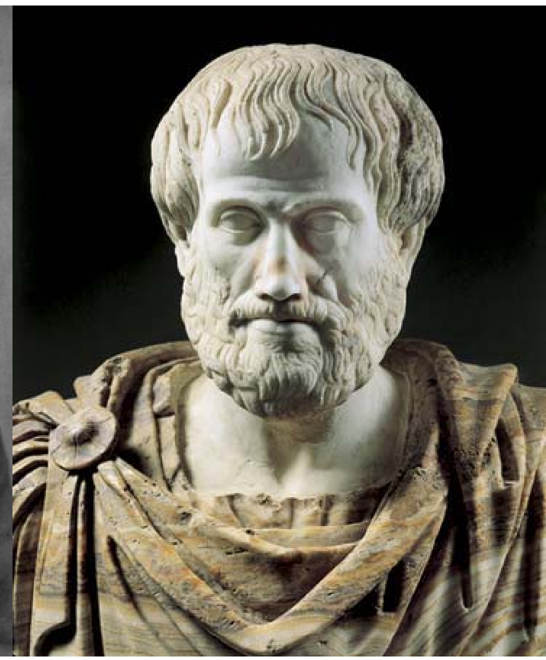
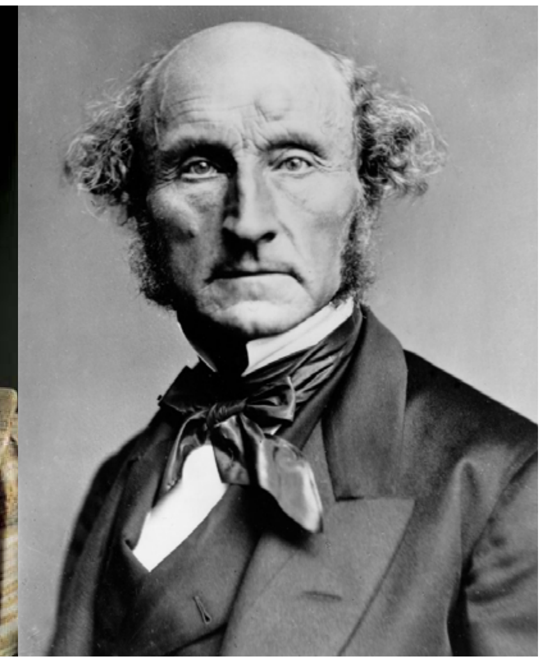
CS 440/ECE 448 Lecture 11: Exam 1 Review

Spring 2018

CS440/ECE448: Artificial Intelligence
Lecture 1: What is AI?

What is Artificial Intelligence?

- Candidate definitions from the textbook:

1. Thinking humanly	**2. Acting humanly**
3. Thinking rationally	**4. Acting rationally**

CS440/ECE 448 Lecture 3: Agents and Rationality

Slides by Svetlana Lazebnik, 9/2016
Modified by Mark Hasegawa-Johnson, 1/2018

Specifying the task environment

- **PEAS: Performance, Environment, Actions, Sensors**
- **P:** a function the agent is maximizing (or minimizing)
  - Assumed given
- **E:** a formal representation for *world states*
  - For concreteness, a tuple (var_1=val_1, var_2=val_2, … ,var_n=val_n)
- **A:** actions that change the state according to a *transition model*
  - Given a state and action, what is the successor state (or distribution over successor states)?
- **S:** observations that allow the agent to infer the world state
  - Often come in very different form than the state itself
  - E.g., in tracking, observations may be pixels and state variables 3D coordinates

Types of Agents

- Reflex agent: no concept of past, future, or value
  - Might still be Rational, if the environment is known to the designer with sufficient detail
- Internal-State agent: knows about the past
- Goal-Directed agent: knows about the past and future
- Utility-Directed agent: knows about past, future, and value

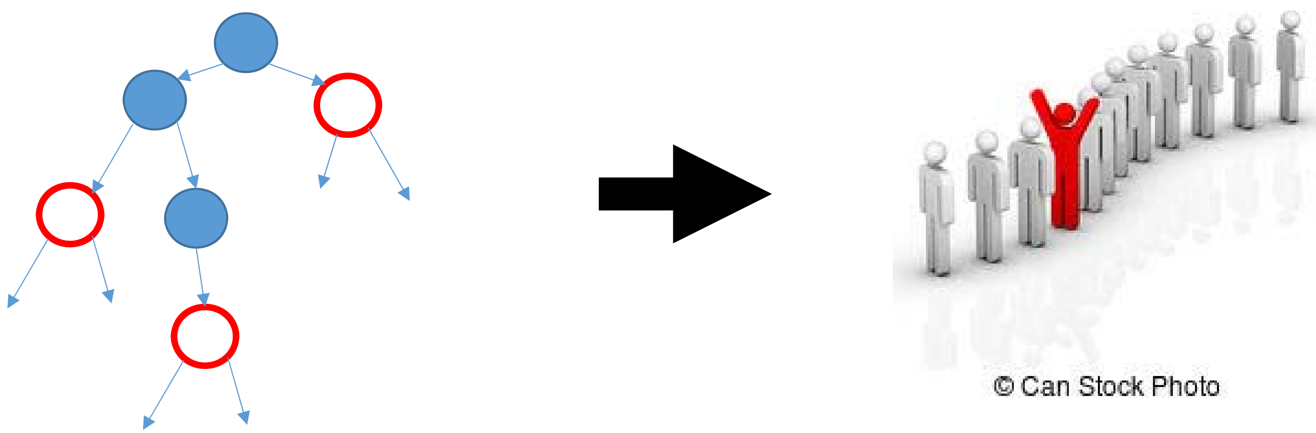
Properties of Environments

- Fully observable vs. partially observable
- Deterministic vs. stochastic
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Single agent vs. multi-agent
- Known vs. unknown

CS440/ECE448 Lectures 4-5: Search

Slides by Svetlana Lazebnik, 9/2016

Revised by Mark Hasegawa-Johnson, 1/2018



Tree Search Algorithm

- Initialize: Frontier = { startnode }
- While Frontier $\neq \emptyset$
  - Choose a node from the frontier, (add it to the visited list)
  - If it's the end node: terminate
  - If not, expand it: put its (non-visited) neighbors into the frontier

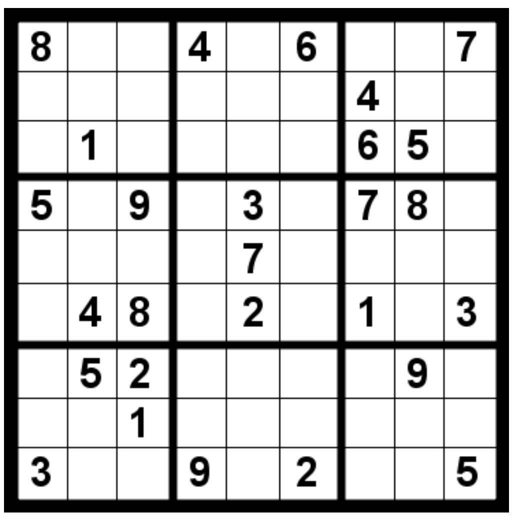
All search strategies

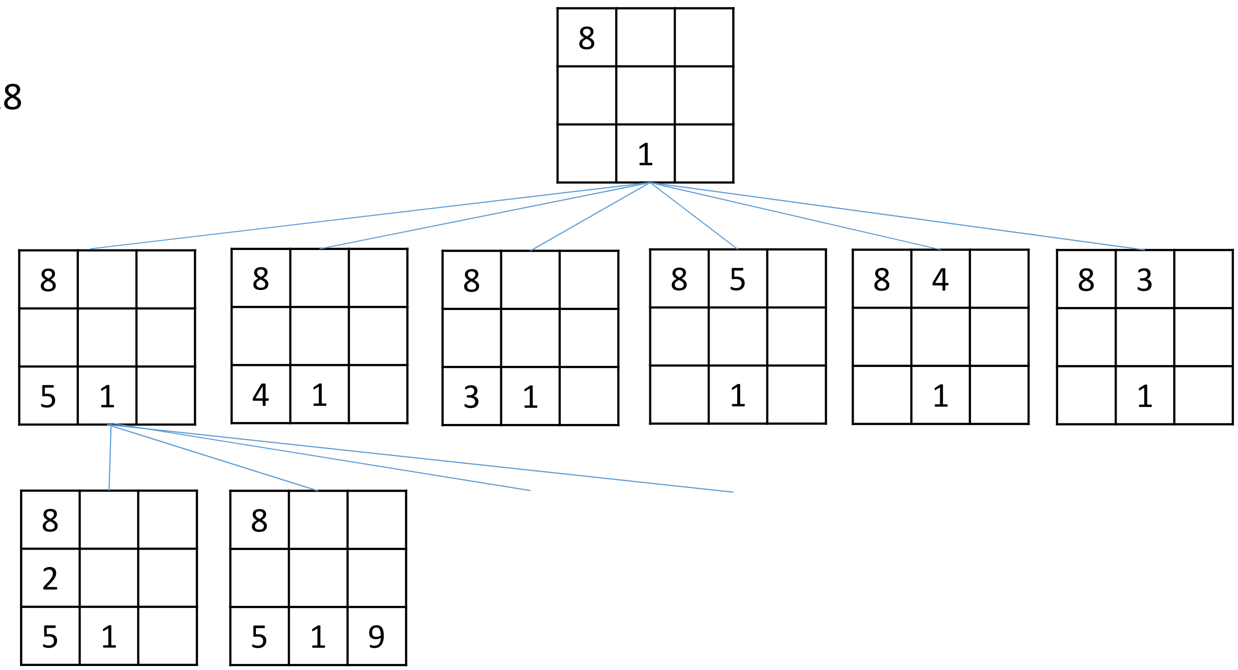
Algorithm	Complete?	Optimal?	Time complexity	Space complexity	Implement the Frontier as a…
BFS	Yes	If all step costs are equal	O(b^d)	O(b^d)	Queue
DFS	No	No	O(b^m)	O(bm)	Stack
IDS	Yes	If all step costs are equal	O(b^d)	O(bd)	Stack
UCS	Yes	Yes	Number of nodes w/ g(n) ≤ C*	Number of nodes w/ g(n) ≤ C*	Priority Queue sorted by g(n)
Greedy	No	No	Worst case: O(b^m) Best case: O(bd)	Worse case: O(b^m) Best case: O(bd)	Priority Queue sorted by h(n)
A*	Yes	Yes	Number of nodes w/ g(n)+h(n) ≤ C*	Number of nodes w/ g(n)+h(n) ≤ C*	Priority Queue sorted by h(n)+g(n)

CS440/ECE 448, Lecture 6: Constraint Satisfaction Problems

Slides by Svetlana Lazebnik, 9/2016

Modified by Mark Hasegawa-Johnson, 1/2018





Backtracking search

- In CSP's, variable assignments are **commutative**
  - For example, *[WA = red then NT = green]* is the same as *[NT = green then WA = red]*
- We only need to consider assignments to a single variable at each level (i.e., we fix the order of assignments)
  - Then there are only m^n leaves
- Depth-first search for CSPs with single-variable assignments is called **backtracking search**

Heuristics for making backtracking search more efficient

Still DFS, but we use heuristics to decide which child to expand first.
You could call it GDFS…

- Heuristics that choose the next variable to assign:
  - Minimum Remaining Values (MRV)
  - Most Constraining Variable (MCV)
- Heuristic that chooses a value for that variable:
  - Least Constraining Assignment (LCA)
- Early detection of failure:
  - Forward Checking
  - Arc Consistency

Planning (Chapter 10)

Slides by Svetlana Lazebnik, 9/2016
with modifications by Mark Hasegawa-Johnson, 1/2018

Planning as Search

Pre-specified set of possible actions

- Example: carry_left(beans), carry_right(goat)
- Action = function of one or more variables
- Result = variables changed in pre-defined way
  - With pre-defined cost

- This is not at all like CSP. Order of the actions is important.
  - Constraints apply not just to the goal state, but also to every intermediate state.

Complexity of planning

- Planning is *PSPACE-complete*
  - The length of a plan can be exponential in the number of "objects" in the problem!
  - So is game search
- Archetypal PSPACE-complete problem: *quantified boolean formula* (QBF)
  - Example: is this formula true?
    $\exists \mathbf{x_1} \forall \mathbf{x_2} \exists \mathbf{x_3} \forall \mathbf{x_4} (\mathbf{x_1} \vee \neg \mathbf{x_3} \vee \mathbf{x_4}) \wedge (\neg \mathbf{x_2} \vee \mathbf{x_3} \vee \neg \mathbf{x_4})$
- Compare to SAT:
    $\exists \mathbf{x_1} \exists \mathbf{x_2} \exists \mathbf{x_3} \exists \mathbf{x_4} (\mathbf{x_1} \vee \neg \mathbf{x_3} \vee \mathbf{x_4}) \wedge (\neg \mathbf{x_2} \vee \mathbf{x_3} \vee \neg \mathbf{x_4})$
- Relationship between SAT and QBF is akin to the relationship between puzzles and games

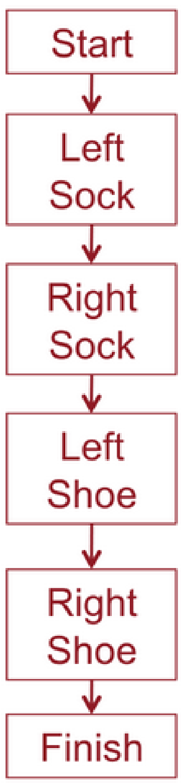
A* Heuristics by Constraint Relaxation

- Heuristics from Constraint Relaxation: The heuristic h(n) is the number of steps it would take to get from n to G, if problem constraints were relaxed --- this guarantees that h(n) is admissible
- $h_1(n)$ dominates $h_2(n)$ ($h_1(n) \geq h_2(n)$) if $h_1(n)$ is computed by relaxing fewer constraints.

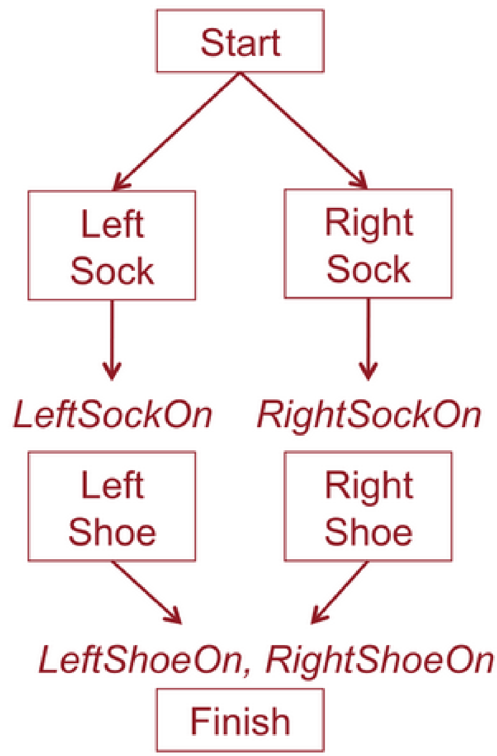
Partial order planning

- Task: put on socks and shoes

Total order (linear) plans



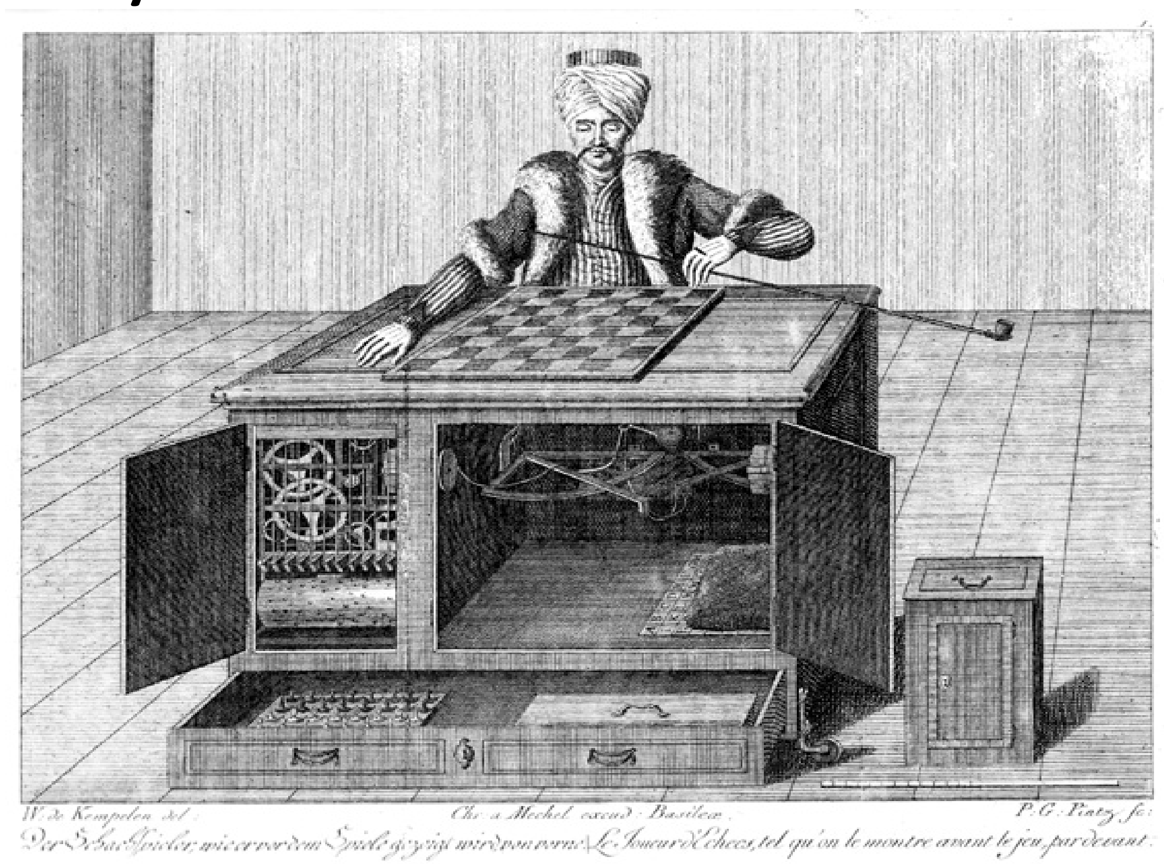
Partial order plan



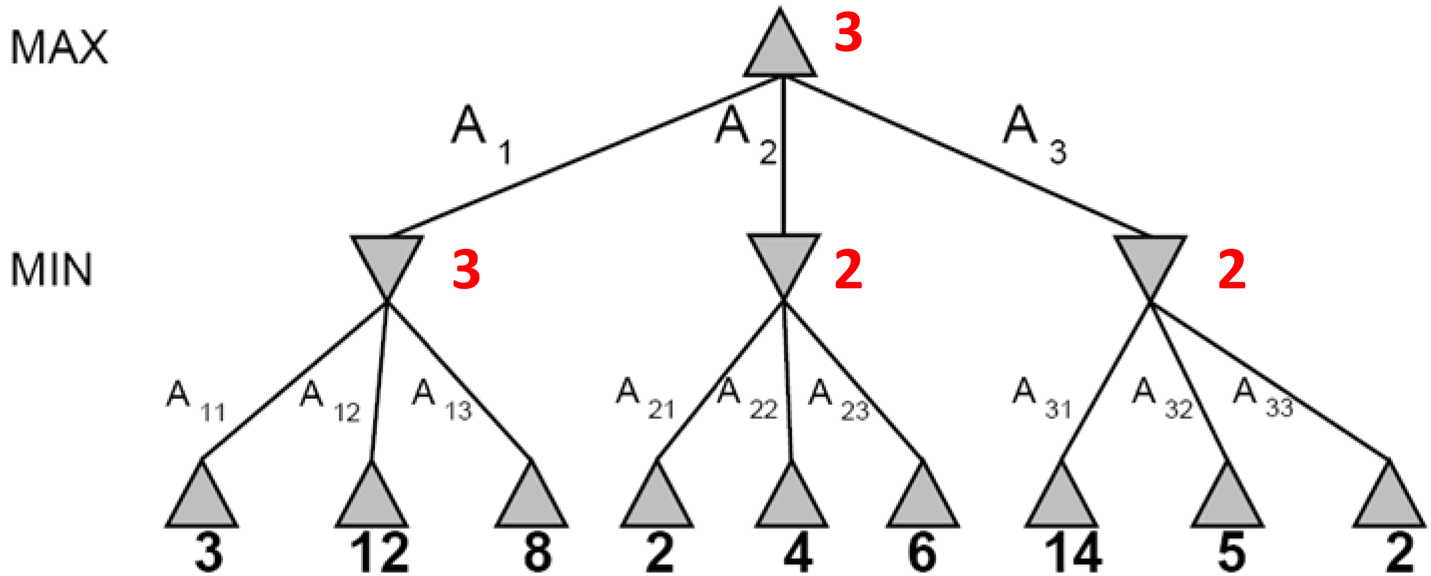
CS440/ECE448 Lecture 8: Two-Player Games

Slides by Svetlana Lazebnik 9/2016

Modified by Mark Hasegawa-Johnson 2/2018



Computing the minimax value of a node



- **Minimax**(*node*) =
  - Utility(*node*) if *node* is terminal
  - $\max_{action}$ **Minimax**(Succ(*node, action*)) if *player* = MAX
  - $\min_{action}$ **Minimax**(Succ(*node, action*)) if *player* = MIN

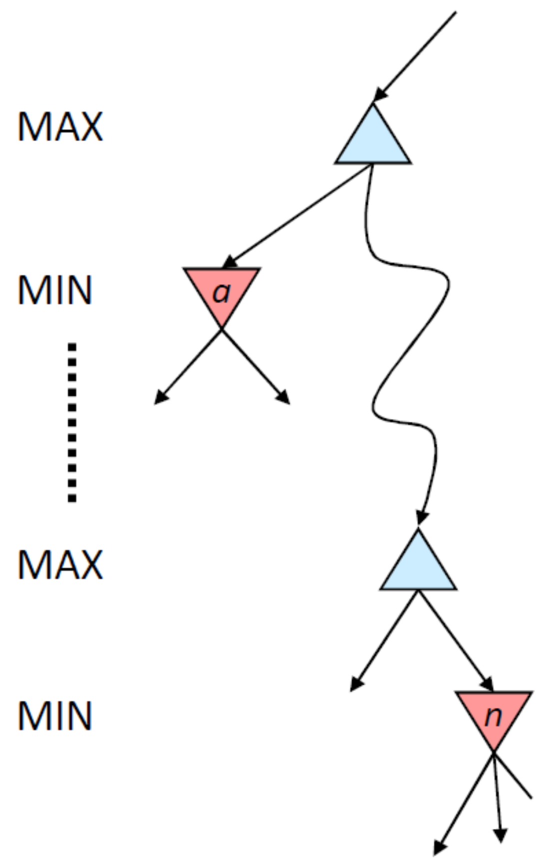
Alpha-Beta Pruning

Key point that I find most counter-intuitive:

- MIN needs to calculate which move MAX will make.
- MAX would never choose a suboptimal move.
- So if MIN discovers that, at a particular node in the tree, she can make a move that's REALLY REALLY GOOD for her…
- She can assume that MAX will never let her reach that node.
- … and she can prune it away from the search, and never consider it again.

Alpha-beta pruning

- **α** is the value of the best choice for the MAX player found so far at any choice point above node *n*
- More precisely: **α** is the highest number that MAX knows how to force MIN to accept
- We want to compute the MIN-value at *n*
- As we loop over *n*'s children, the MIN-value decreases
- If it drops below **α**, MAX will never choose *n*, so we can ignore *n*'s remaining children
- $\alpha \leq \beta$



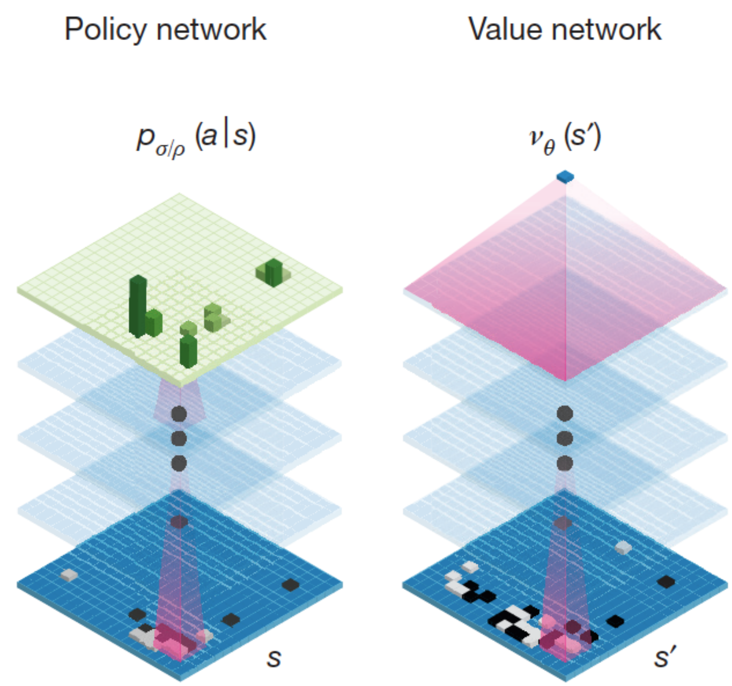
Cutting off search

- Cut off search at a certain depth and compute the value of an **evaluation function** for a state instead of its minimax value
- **Horizon effect:** you may incorrectly estimate the value of a state by overlooking an event that is just beyond the depth limit
  - For example, a damaging move by the opponent that can be delayed but not avoided
- Possible remedies
  - **Quiescence search:** do not cut off search at positions that are unstable – for example, are you about to lose an important piece?
  - **Singular extension:** a strong move that should be tried when the normal depth limit is reached

CS440/ECE448 Lecture 10: Stochastic Games, Stochastic Search, and Learned Evaluation Functions

Slides by Svetlana Lazebnik, 9/2016

Modified by Mark Hasegawa-Johnson, 1/2018



Minimax vs. Expectiminimax

- **Minimax:**
  - **Maximize** (over all possible moves I can make) the
  - **Minimum** (over all possible moves Min can make) of the
  - Reward

$$Value(node) = \max_{my\ moves}\left(\min_{Min's\ moves}(Reward)\right)$$

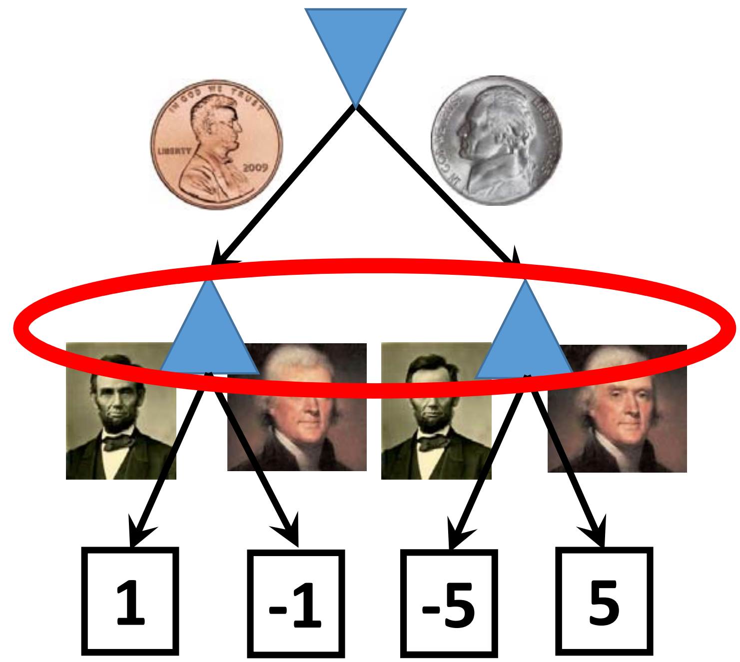
- **Expectiminimax:**
  - **Maximize** (over all possible moves I can make) the
  - **Minimum** (over all possible moves Min can make) of the
  - **Expected** reward

$$Value(node) = \max_{my\ moves}\left(\min_{Min's\ moves}(\mathbb{E}[Reward])\right)$$

$$\mathbb{E}[Reward] = \sum_{outcomes} Probability(outcome) \times Reward(outcome)$$

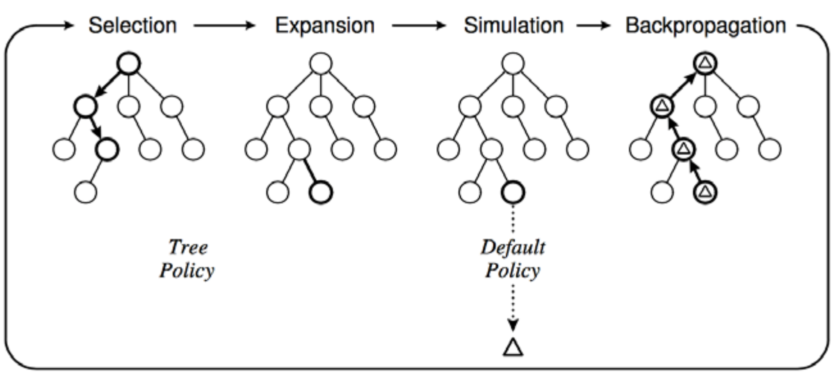
Imperfect information

- The problem: I don't know which state I'm in. I only know it's one of these two.
- This is called an "information set:" our information is sufficient to know that we're in one of these states, but we don't know which one



Monte Carlo Tree Search (Stochastic Search)

- What about ***<u>deterministic</u>*** games with deep trees, large branching factor, and no good heuristics – like Go?
- Instead of depth-limited search with an evaluation function, use randomized simulations
- Starting at the current state (root of search tree), iterate:
  - Select a leaf node for expansion using a *tree policy* (trading off *exploration* and *exploitation*)
  - Run a simulation using a *default policy* (e.g., random moves) until a terminal state is reached
  - Back-propagate the outcome to update the value estimates of internal tree nodes



C. Browne et al., [A survey of Monte Carlo Tree Search Methods](), 2012

CS 440/ECE448 Lecture 10: Game Theory

Slides by Svetlana Lazebnik, 9/2016

Modified by Mark Hasegawa-Johnson, 2/2018

Prisoner B / Prisoner A	Prisoner B stays silent (*cooperates*)	Prisoner B betrays (*defects*)
Prisoner A stays silent (*cooperates*)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
Prisoner A betrays (*defects*)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years

https://en.wikipedia.org/wiki/Prisoner’s_dilemma

Prisoner's dilemma

- **Nash equilibrium:** A pair of strategies such that no player can get a bigger payoff by switching strategies, provided the other player sticks with the same strategy
  - (Testify, Testify) is a Nash equilibrium
- **Dominant strategy:** A strategy whose outcome is better for the player regardless of the strategy chosen by the other player
  - Testify is dominant for Alice
  - Testify is dominant for Bob
- **Pareto optimal outcome:** There is no outcome that would make one of the players better off without making another one worse off
  - All outcomes except the Nash equilibrium are Pareto optimal

	Alice: Testify	**Alice: Refuse**
Bob: Testify	-5,-5	-10,0
Bob: Refuse	0,-10	-1,-1

Mixed strategy equilibrium

	P1: Choose S with prob. p	**P1:** Choose C with prob. $1-p$
P2: Choose S with prob. q	-10, -10	-1, 1
P2: Choose C with prob. $1-q$	1, -1	0, 0

- Expected payoffs for P1 given P2's strategy:

  P1 chooses S: $q(-10) + (1-q)1 = -11q + 1$

  P1 chooses C: $q(-1) + (1-q)0 = -q$

- In order for P2's strategy to be part of a Nash equilibrium, P1 has to be indifferent between its two actions:

  $-11q + 1 = -q$ or $q = 1/10$

  Similarly, $p = 1/10$

Repeated Games

If the game is repeated N times, then

- Nash equilibrium = neither player has any reason to change strategies, given knowledge of the other player's strategy.
  - Nash equilibrium for the sequence might not be Nash equilibrium for any individual game, it might even appear "moral," e.g., Ultimatum game
- Dominant strategy = strategy that's optimal regardless of what the other player does
  - Strategy might be different for the 1st, 2nd, 3rd, …, Nth game
  - Dominant strategy might require random choice, e.g., the monopolist game

http://en.wikipedia.org/wiki/Ultimatum_game