CSC520 - Artificial Intelligence
Lecture 7

Dr. Scott N. Gerard

North Carolina State University

Jan 30, 2025

Agenda

- Nondeterministic actions
- Partial observability
- Online search

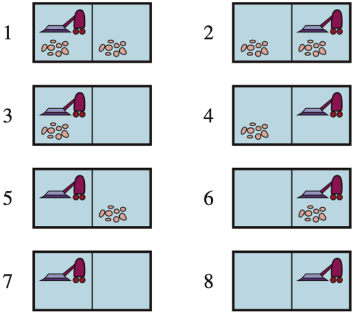
Nondeterministic Actions

- Earlier search algorithms assumed a deterministic environment
- We now consider tasks in which actions are non-deterministic
  - Agent doesn't know what state is reached after taking an action
  - E.g. if agent executes a in s_1, the agent may transition to s_2, s_4 or s_5
- Instead of a sequential plan, the solution is a conditional plan (or contingency plan)

Erratic Vacuum World

- Earlier solution: [*suck*, *right*, *suck*]
- Suppose *suck* action is non-deterministic
  - On a dirty square, it cleans the square and sometimes cleans up adjacent square, too
  - On a clean square, it sometimes deposits dirt
- Generalize the transition model
  - Earlier: $\textsc{Results}(1, \mathit{Suck}) = 5$
  - Now: $\textsc{Results}(1, \mathit{Suck}) = \{5, 7\}$ which is a **belief state**
- Solution is a conditional plan

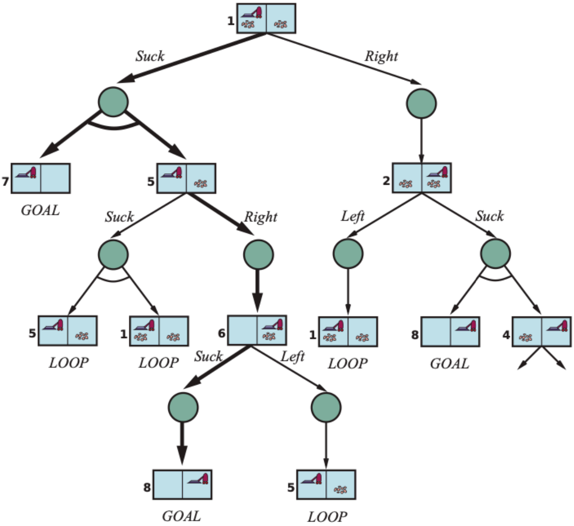
  [*Suck*, **if** *State* = 5 **then** [*Right*, *Suck*] **else**[]]



AND-OR Search Trees

- Solutions of non-deterministic problems are *trees* rather than sequence of actions
- Agent's choices are represented as OR nodes
- Environment's choice of outcome of an action are represented as AND nodes.

Erratic Vacuum AND-OR Search Tree



$[Suck, \textbf{if } State = 5 \textbf{ then } [Right, Suck] \textbf{ else}[]]$

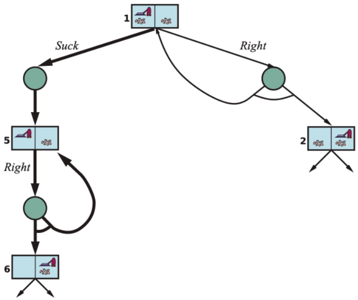
AND-OR Search Tree

- Solution is a subtree that:
  - Has a goal node at **every** leaf
  - Specifies one action at each OR node
  - Includes every outcome branch at each AND node
- Variations of BFS, DFS, A^*, etc. techniques can be used for AND-OR graph search

Slippery Vacuum World

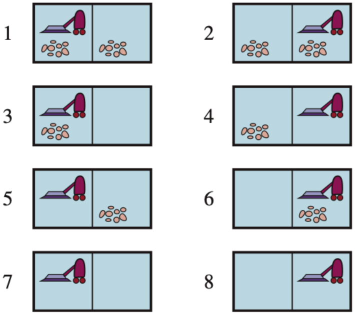
- Right (left) action may fail leaving agent in the same location
- Keep trying the action until it succeeds
- Assumption is that each possible outcome of a non-deterministic action eventually occurs
- A cyclic solution is defined using **while** construct

  $[Suck, \textbf{while } State = 5 \textbf{ do } Right, Suck]$



Searching With No Observations

- Sensorless problem: agent's percepts provide no information
- **Belief state** is a set of physical states that an agent believes are possible
- Suppose agent knows geography but doesn't know its own location
  - Initial belief state $= \{1, 2, 3, 4, 5, 6, 7, 8\}$
  - After [*Right*]: $\{2, 4, 6, 8\}$
  - After [*Right*, *Suck*]: $\{4, 8\}$
  - After [*Right*, *Suck*, *Left*, *Suck*]: $\{7\}$
- Belief states are fully observable. Agent always knows what it believes



Searching in Partially Observable Environments

- Earlier algorithms assumed full observability
- Searching with no observations
- Searching with partial observations

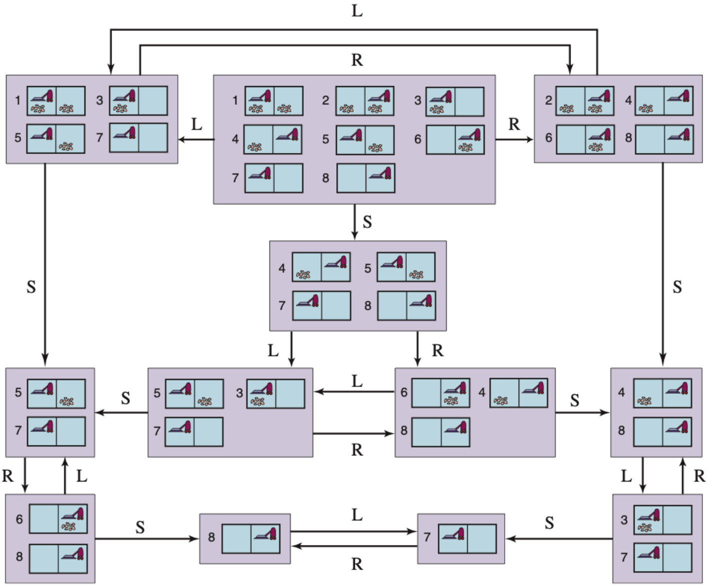
Searching With No Observations

- Search in the space of *belief states* rather than physical states
- Fully observable in belief-state space
- Solution is a sequence of actions
- Existing algorithms can be used if we formulate the search problem in terms of belief-state

Belief-state Problem

- *State space*: every possible subset of physical states; with N physical states, there can be 2^N belief states
- *Initial state*: typically a set with all physical states
- *Actions*: union of legal actions for all physical states in belief-state (Or intersection if illegal actions are possible)
- *Transition model*: belief-state resulting from applying an action in a given belief-state
  $b' = \text{RESULT}(b, a) = \{s' : s' = \text{RESULT}(s, a) \text{ and } s \in b\}$
- *Cost function*: (tricky) cost of executing an action can be calculated from the cost of applying an action in a physical state
- *Solution* is a *path* from the start state to a goal state

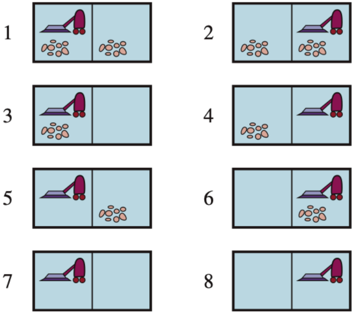
Deterministic Sensorless Vacuum World



Searching In Partially Observable Environments

- Problem specification includes a $\text{PERCEPT}(s)$ function that returns percepts for a given state

- Suppose vacuum cleaner can only sense dirt its tile it (partial observability)
- Then, percept $[A, \mathit{Dirty}]$ can be observed in states 1 and 3, that is, the belief-state $= \{1, 3\}$
- Percept $[B, \mathit{Clean}]$ can be observed in states 4 and 8, that is, the belief-state $= \{4, 8\}$



Transition Model in Partially Observable Search

- Three stages of the transition model
  - Prediction stage computes belief state resulting from an action
    $\hat{b} = \text{PREDICT}(b, a)$

Transition Model in Partially Observable Search

- Three stages of the transition model
  - Prediction stage computes belief state resulting from an action
    $\hat{b} = \textsc{Predict}(b, a)$
  - Possible percepts stage computes the percepts that could be observed in the predicted belief state
    $\textsc{Possible-Percepts}(\hat{b}) = \{o : o = \textsc{Percept}(s), s \in \hat{b}\}$

Transition Model in Partially Observable Search

- Three stages of the transition model
  - Prediction stage computes belief state resulting from an action
    $\hat{b} = \textsc{predict}(b, a)$
  - Possible percepts stage computes the percepts that could be observed in the predicted belief state
    $\textsc{possible-percepts}(\hat{b}) = \{o : o = \textsc{percept}(s), s \in \hat{b}\}$
  - Update stage computes the belief state that would result from the possible percepts
    $b_o = \textsc{update}(\hat{b}, o) = \{s : o = \textsc{percept}(s), s \in \hat{b}\}$

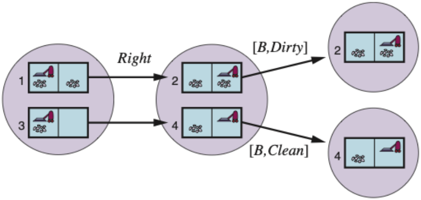
Transition Model in Partially Observable Search

- Three stages of the transition model
  - Prediction stage computes belief state resulting from an action
    $\hat{b} = \text{PREDICT}(b, a)$
  - Possible percepts stage computes the percepts that could be observed in the predicted belief state
    $\text{POSSIBLE-PERCEPTS}(\hat{b}) = \{o : o = \text{PERCEPT}(s), s \in \hat{b}\}$
  - Update stage computes the belief state that would result from the possible percepts
    $b_o = \text{UPDATE}(\hat{b}, o) = \{s : o = \text{PERCEPT}(s), s \in \hat{b}\}$
- Combining the above stages, we get
  $\text{RESULTS}(b, a) = \{b_o : b_o = \text{UPDATE}(\text{PREDICT}(b, a), o),$
  $o \in \text{POSSIBLE-PERCEPTS}(\text{PREDICT}(b, a))\}$

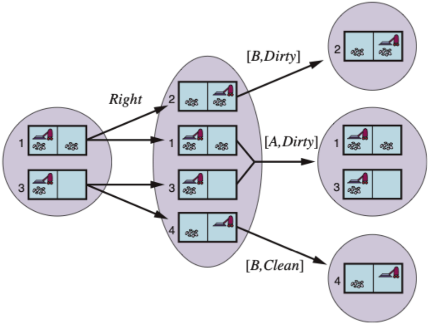
Transition Model in Partially Observable Search

- Three stages of the transition model
  - Prediction stage computes belief state resulting from an action
    $\hat{b} = \text{PREDICT}(b, a)$
  - Possible percepts stage computes the percepts that could be observed in the predicted belief state
    $\text{POSSIBLE-PERCEPTS}(\hat{b}) = \{o : o = \text{PERCEPT}(s), s \in \hat{b}\}$
  - Update stage computes the belief state that would result from the possible percepts
    $b_o = \text{UPDATE}(\hat{b}, o) = \{s : o = \text{PERCEPT}(s), s \in \hat{b}\}$
- Combining the above stages, we get
  $\text{RESULTS}(b, a) = \{b_o : b_o = \text{UPDATE}(\text{PREDICT}(b, a), o),$
  $o \in \text{POSSIBLE-PERCEPTS}(\text{PREDICT}(b, a))\}$
- PREDICT can enlarge belief state size.
  UPDATE can't enlarge and may reduce belief state size.

Example



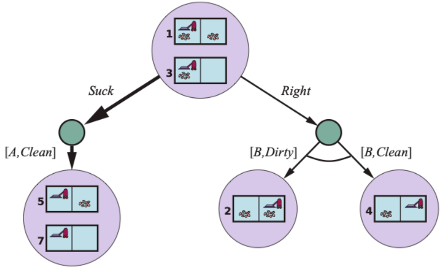
Deterministic Actions



Non-deterministic Actions

Solving Partially Observable Problems

- Use AND-OR search tree
- Solution is a conditional plan
- [*Suck*, *Right*, if *Bstate* = {6} then *Suck* else []]



Online Search

- We studied offline search algorithms so far
  - Compute a complete solution before taking the first action
- Online search interleaves *computation* and *action*

Online Search

- We studied offline search algorithms so far
  - Compute a complete solution before taking the first action
- Online search interleaves *computation* and *action*
- Useful for dynamic and semi-dynamic environments with penalty for delays

Online Search

- We studied offline search algorithms so far
  - Compute a complete solution before taking the first action
- Online search interleaves *computation* and *action*
- Useful for dynamic and semi-dynamic environments with penalty for delays
- Useful for non-deterministic environments since the agent can focus on contingencies that actually arise rather than those that might arise

Online Search

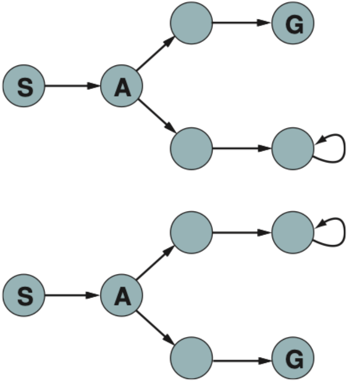
- We studied offline search algorithms so far
  - Compute a complete solution before taking the first action
- Online search interleaves *computation* and *action*
- Useful for dynamic and semi-dynamic environments with penalty for delays
- Useful for non-deterministic environments since the agent can focus on contingencies that actually arise rather than those that might arise
- Necessary in unknown environments
  - Agent does not know the states or effects of actions
  - Agent must learn from experience
  - E.g. a vacuum cleaner robot must learn the floor-plan of a home as it vacuums

Online Search Problem

- Agent knows the following:
  - ACTION(s): the legal actions in state s
  - $c(s, a, s')$: the cost of applying action a; agent cannot use this until it knows that s' is the outcome
  - GOAL-TEST(s): the goal test
  - A heuristic function that estimates the path cost to a goal state
- Agent's objective is to reach the goal state while minimizing cost
- *Competitive ratio* is the ratio of the *actual* cost with *optimal* cost in known environment

Safely Explorable Environment

- Online search is vulnerable to *dead* ends
- No algorithm can avoid dead ends in all state spaces
- We usually assume that state space is *safely explorable*: a goal state is reachable from every state



Online Search Algorithms

- Agent can only explore the successors of the current state
  - Agent cannot bounce to a distant state in another path like A^*

Online Search Algorithms

- Agent can only explore the successors of the current state
  - Agent cannot bounce to a distant state in another path like A^*
- Depth-first search can be used assuming the actions are reversible

Online Search Algorithms

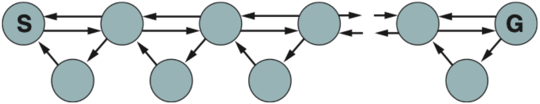
- Agent can only explore the successors of the current state
    - Agent cannot bounce to a distant state in another path like A^*
- Depth-first search can be used assuming the actions are reversible
- Can an online search agent use Hill-climbing algorithm?

Online Search Algorithms

- Agent can only explore the successors of the current state
  - Agent cannot bounce to a distant state in another path like A^*
- Depth-first search can be used assuming the actions are reversible
- Can an online search agent use Hill-climbing algorithm?
  - What about random restart Hill-climbing?

Online Search Algorithms

- Agent can only explore the successors of the current state
  - Agent cannot bounce to a distant state in another path like A^*
- Depth-first search can be used assuming the actions are reversible
- Can an online search agent use Hill-climbing algorithm?
  - What about random restart Hill-climbing?
- Random-walk hill-climbing can be used
  - Will eventually find a goal in a finite safely explorable state space
  - But can be very slow



Class Exercise

- Trace the execution of a DFS algorithm on the below simple maze problem.