Artificial Intelligence

CS482, CS682, MW 1 – 2:15, SEM 201, MS 227

Prerequisites: 302, 365

Instructor: Sushil Louis, sushil@cse.unr.edu, http://www.cse.unr.edu/~sushil

Search Leftovers

- Non-determinism in search
- Solutions can be contingency plans (trees)
  - How do we handle non-determinism?
    - First: What types of determinism?
    - Second: For each type, how do we handle it?
- Partial Observability
  - What types of observability do we have?
  - How do we handle each?
  - Don’t forget Kriegspeil and partial observability in games
- Online-vs-Offline search and execution
  - Learning search algorithms

Non-determinism in Actions

- Erratic vacuum-cleaners
  - Bad suck actions example
  - And-Or trees
  - AND
    - You can end up in multiple states as the result of an action
    - You have to find a path from **all** of these states (AND)
  - OR
    - Try each action
    - **Any** one action can lead to the goal state(s) (OR)

Non-determinism in actions

- Slippery vacuum world
- If at first you don't succeed try, try again
- We need to add label to some portion of a plan and use the label to refer to that portion – rather than repeating the subplan → And-Or graphs with labels
- Plan: [Suck, L1: Right, if State == 5 then L1 else Suck]

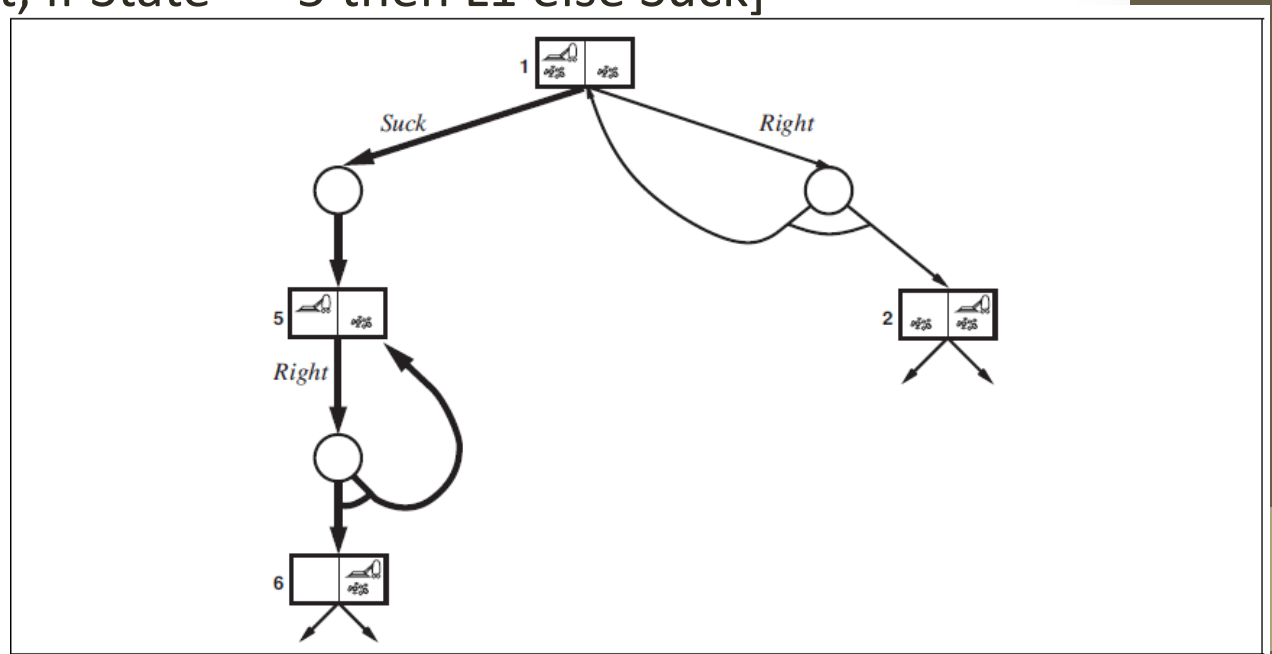


Figure 4.12 **FILES: figures/slippery-vacuum-loop-plan.eps (Tue Nov 3 13:48:56 2009).** Part of the search graph for the slippery vacuum world, where we have shown (some) cycles explicitly. All solutions for this problem are cyclic plans because there is no way to move reliably.

Searching with Partial observation

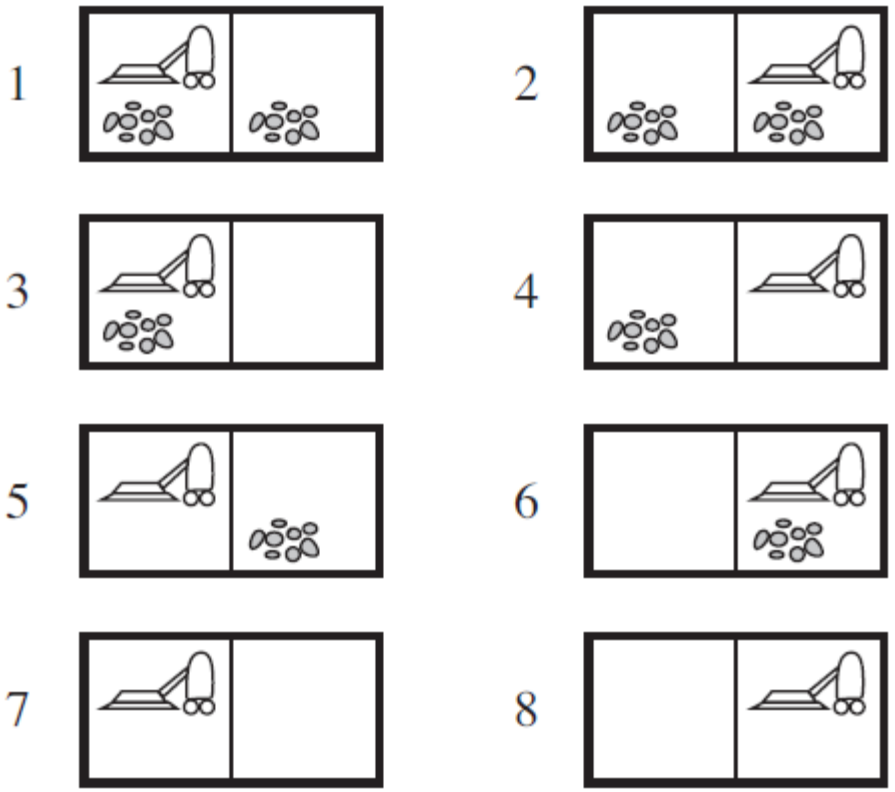
- Agents percepts cannot pin down the exact state the agent is in
- Let Agents have **Belief** states
  - Search for a sequence of belief states that leads to a goal
  - Search for a plan that leads to a goal

- First: NO percepts → sensor-less
- States? (Belief states)
- Initial State?
- Actions?
- Transition Model?
- Goal test?
- Path cost?

Consider sensor-less vacuum world

Sensor-less vacuum world

- Assume belief states are the same but no location or dust sensors
- Initial state = {1, 2, 3, 4, 5, 6, 7, 8}
- Action: Right
  - Result = {2, 4, 6, 8}
- Right, Suck
  - Result = {4, 8}
- Right, Suck, Left, Suck
  - Result = {7} guaranteed!



You do not need sensors to **COERCE** the world into a specific state!

Sensor-less search

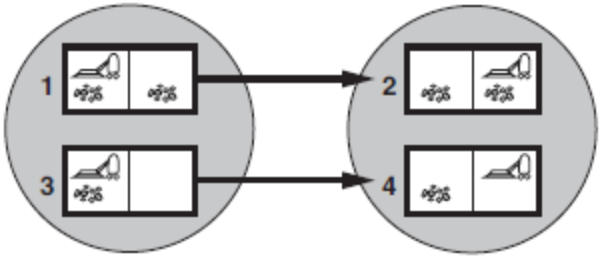
- Search in belief state space, where the problem is fully observable!
- Solution is a sequence, even if the environment is non-deterministic!
- Suppose the underlying problem (P) is
  - $\{Actions_p, Result_p, Goal-Test_p, Step-Cost_p\}$
  - What is the corresponding sensor-less problem
- States → Belief States: every possible set of physical states
  - If N physical states, number of belief states can be 2^N
- Initial State: Typically the set of all states in P
- Actions: Consider {s1, s2}
  - If $Actions_p$(s1) != $Actions_p$(s2) should we take the Union of both sets of actions or the Intersection?
  - Union if all actions are legal, intersection if not

Sensor-less search (cont'd)

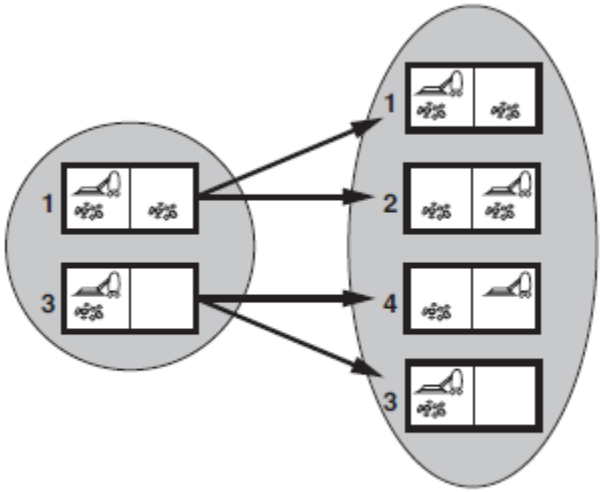
- Transition model
  - Union of all states that $Result_p$(s) returns for all states, s, in your current belief state
    - $b' = Result(b, a)$ = {$s' : s'$ = $Result_p$(s, a) and s ϵ b}
    - This is the prediction step, $Predict_p$(b, a)
- Goal-Test: If all physical states in belief state satisfy $Goal - Test_p$
- Path cost → Tricky in general. Consider what happens if actions in different physical states have different costs. For now assume cost of an action is the same in all states

Examples

- Erratic - *Right*



- Slippery – *Right*



- Action can increase the number of physical states in a belief state

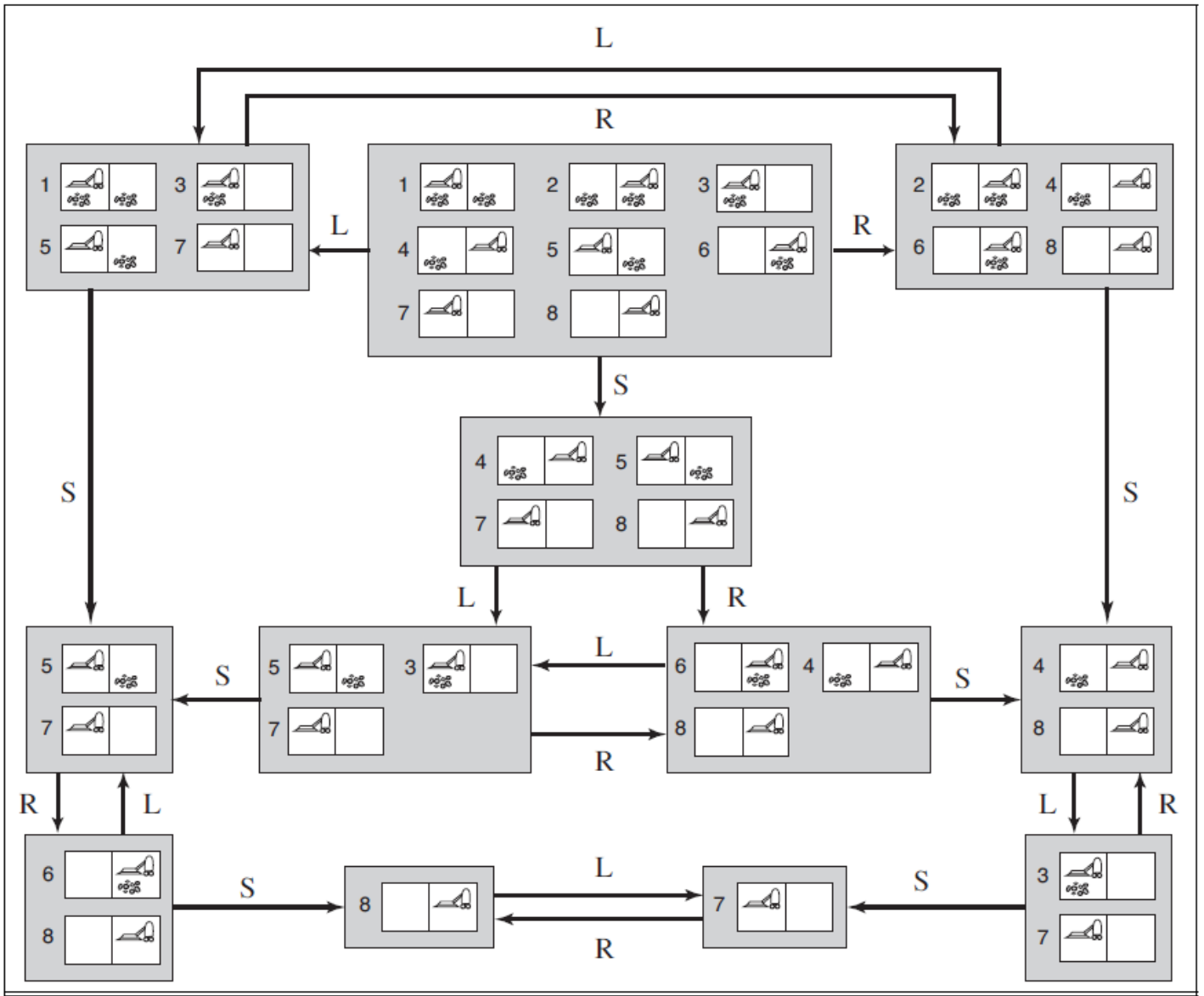
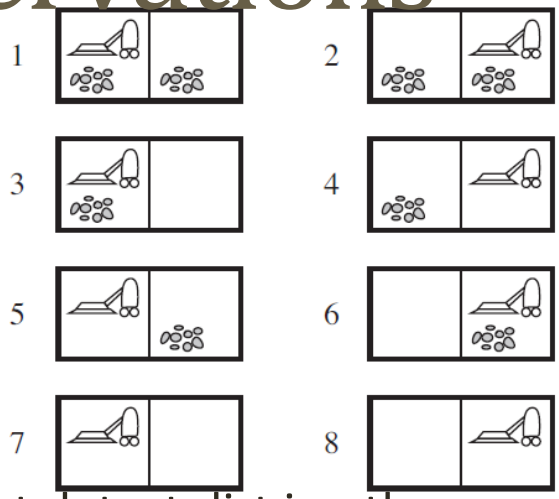


Figure 4.14 **FILES: figures/vacuum2-sets.eps (Tue Nov 3 16:24:01 2009).** The reachable portion of the belief-state space for the deterministic, sensorless vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know which physical state it is in. The initial belief state (complete ignorance) is the top center box. Actions are represented by labeled links. Self-loops are omitted for clarity.

Belief states synposis

- Search through belief state space is usually worse than physical state space (size)
- Alternatives:
  - Logic representations
  - Incremental belief-state search
    - For each physical state in belief state find a solution that will take you to goal
    - Fast failure but have to find one solution that works for all physical states in initial state

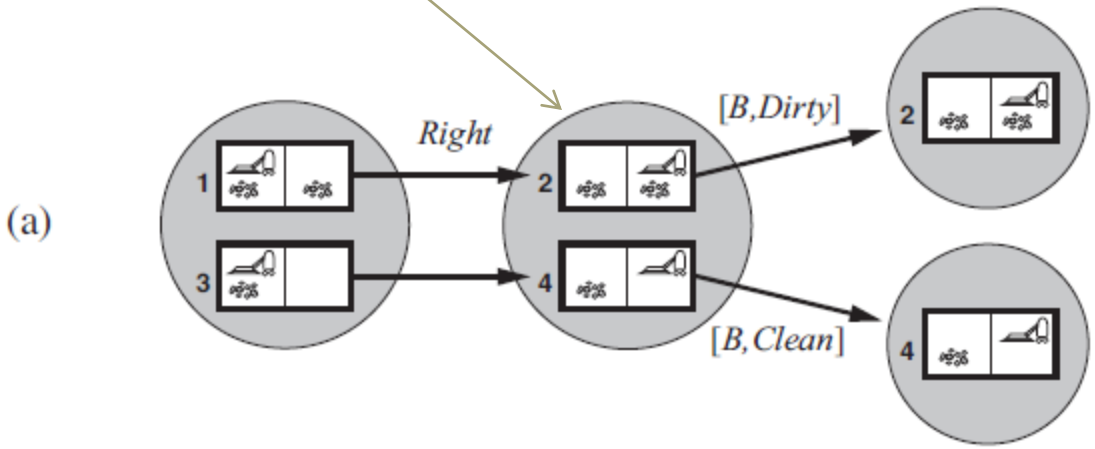
Searching with observations



- Many problems require sensors
  - Percept(s) or Percepts(s) function
- Vacuum world example
  - Location sensor
  - Current location dirt sensor. Cannot detect dirt in other square
  - Percept(s1) = [A, Dirty]
- Observability
  - Sensor-less problems → Percepts(s) = *Null* for all s
  - Fully observable → Percepts(s) = s for every s

Example

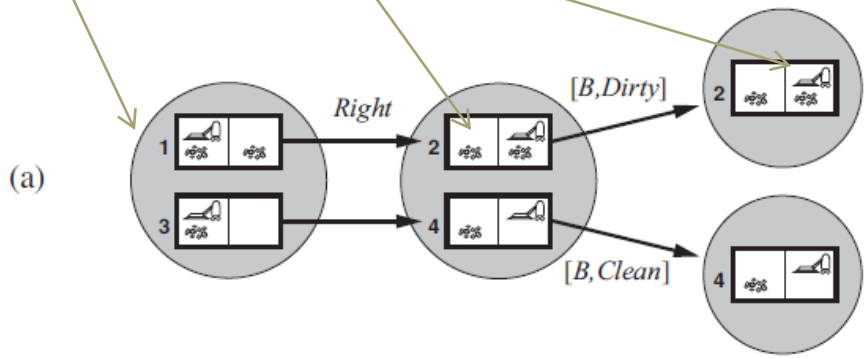
- If you get [A, Dirty] you could be in {1, 3}
- Result({1, 3}, *Right*) is



- **Now**
  - if you see (observe) [B, Dirty] you are in {2}
  - If you observe [b, Clean] you are in {4}
- **Transition Model** is more complicated, otherwise this is not very different from other search problems

3-stage transition model

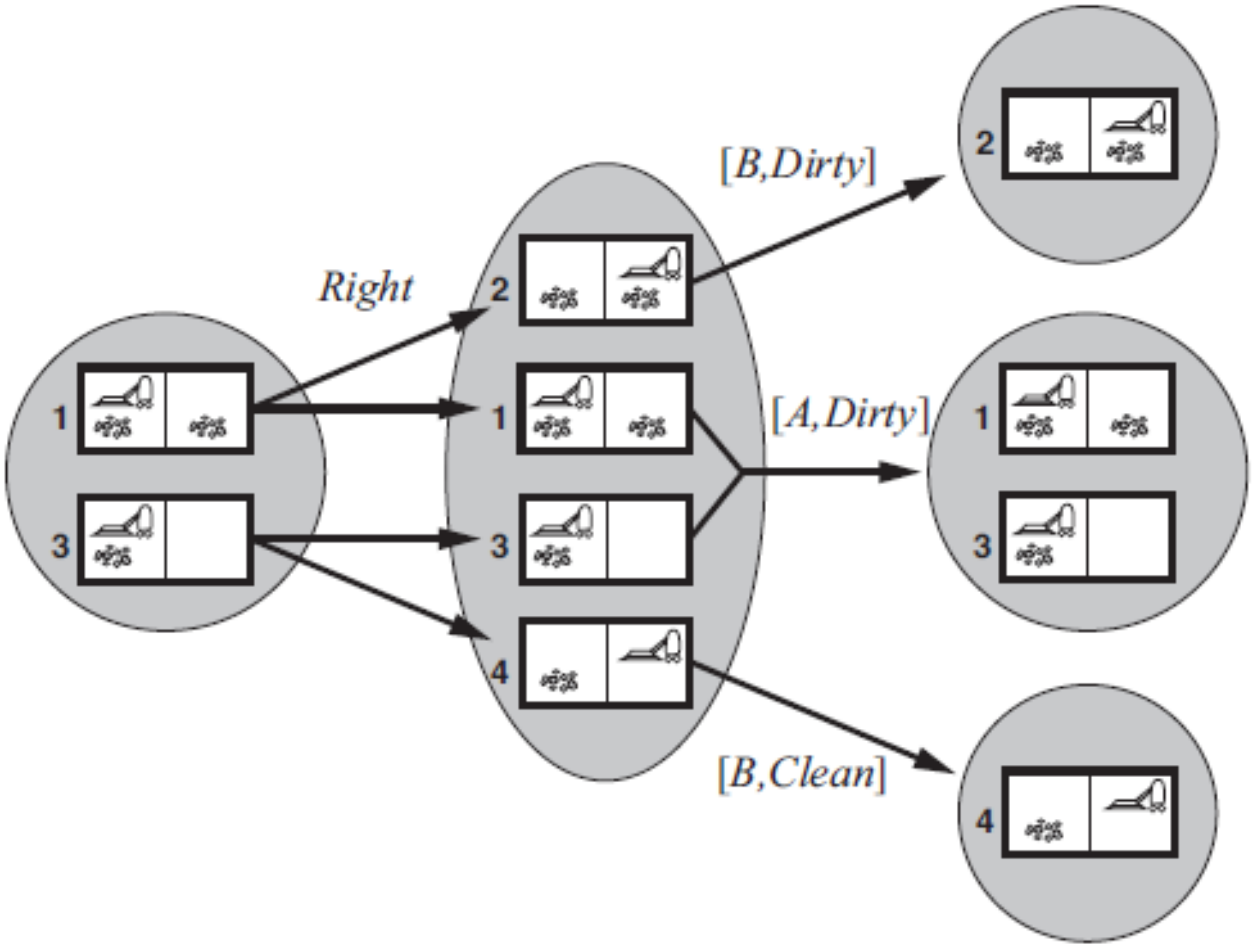
- Prediction stage
  - Predicted belief state is b^ = Predict(b, a)
- Observation prediction stage
  - Possible-Percepts(b^) = {o : o = Percept(s) and s ϵ b^}
- Update stage
  - b_o = Update(b^, o) = {s : o = Percept(s) and s ϵ b^}



- So, Results(b, a)
  - = {b_o : b_o = Update(Predict(b, a), o) and o ϵ Possible-Percepts(Predict(b, a))}

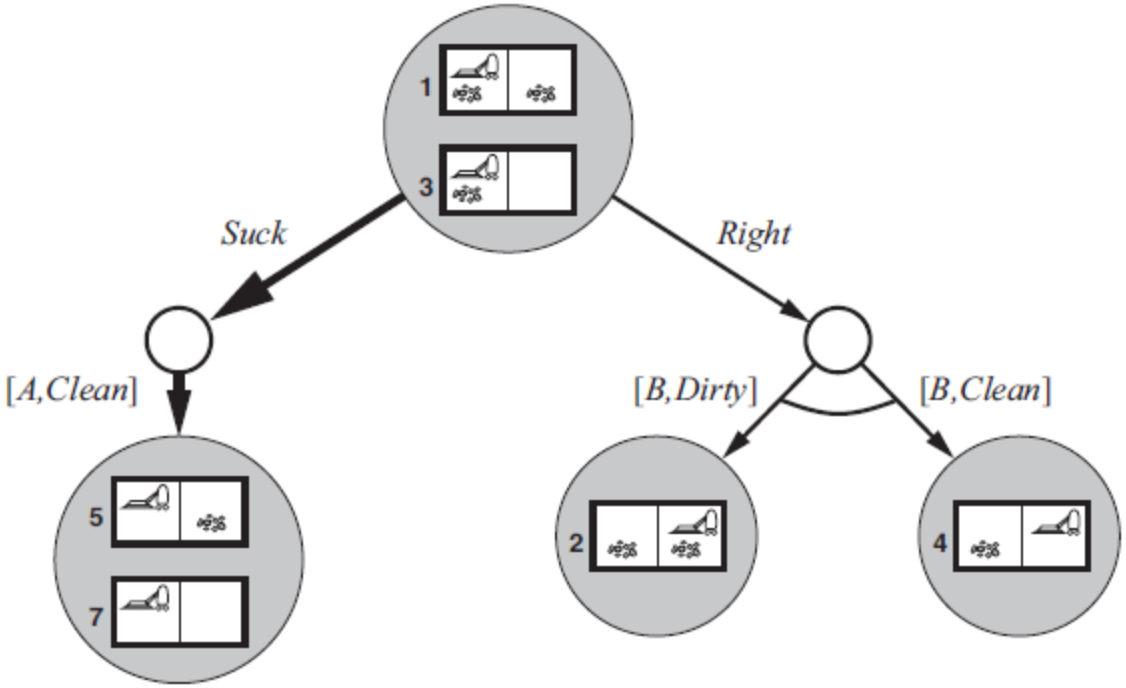
Example: Slippery vacuum

(b)



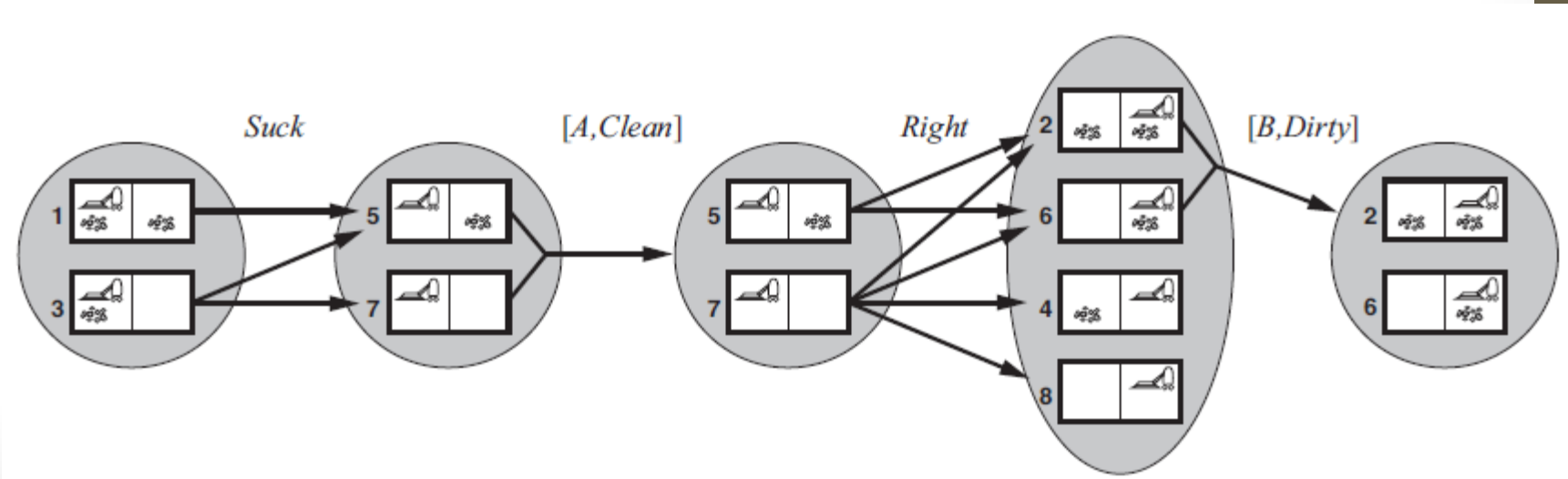
And-Or solution

- Given this problem formulation, we can use the And-Or search algorithm to come up with a plan to solve the problem
- Given [A, Dirty], Plan = {Suck, Right, if Bstate = {6} then Suck else []}



Partially observable environments

- An agent in a partially observable environment must update belief state from percept
  - b’ = Update(Predict(b, a), o)
  - So the agent is only looking at the current o (percept) not the entire history, as we considered earlier. This is recursive state estimation
  - Example: Kindergarten vacuum world



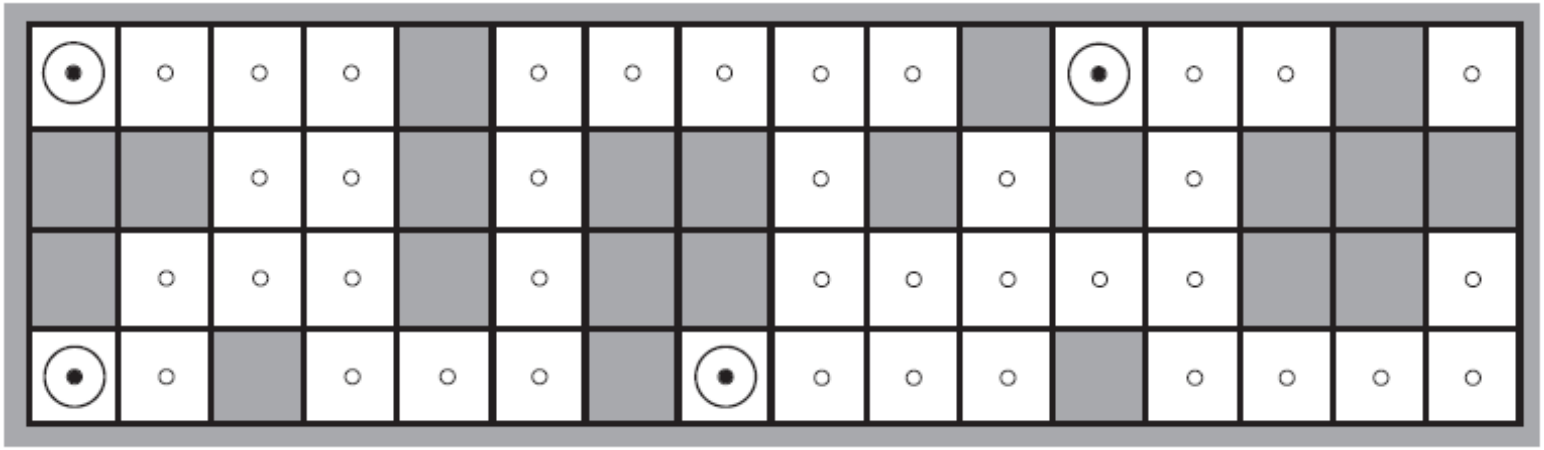
Localization in robotics

- Maintaining belief states is a core function of any Intelligent Agent
- Monitoring, filtering, state estimation
- Robot:
  - Four sonar sensors (NSWE) → give correct data
  - Robot has correct map of environment
  - Move is broken → Robot moves to random adjacent square
- Robot must determine current location

- Suppose it gets [NSW] → obstacles N, S, and W

Robot localization

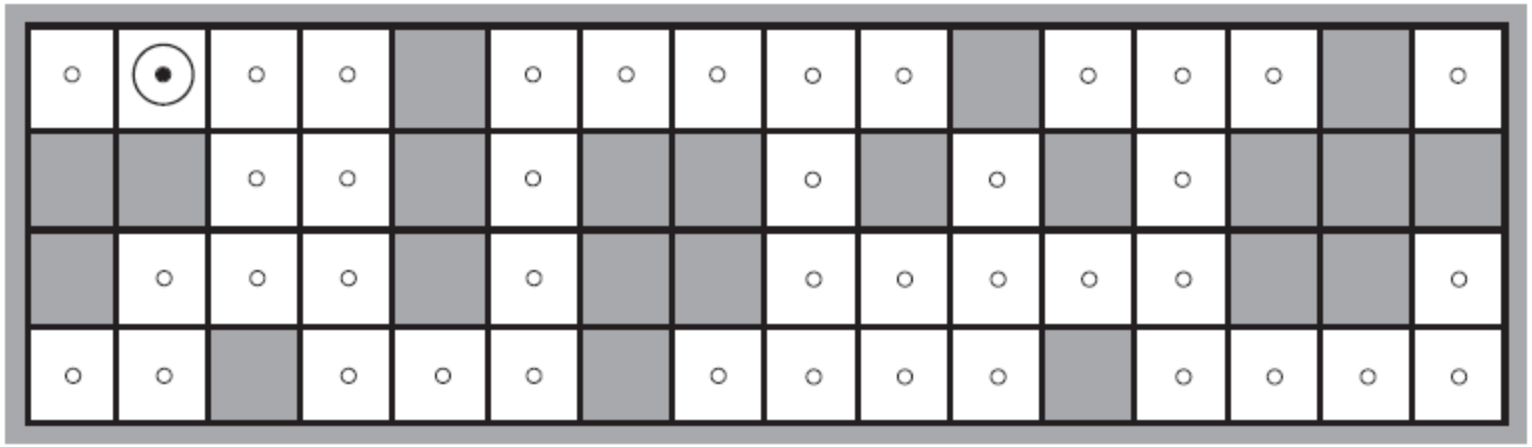
- It must be in one of the following squares after [NSW]



- Now it gets [NS], where can it be?

Robot localization

- Only one location possible

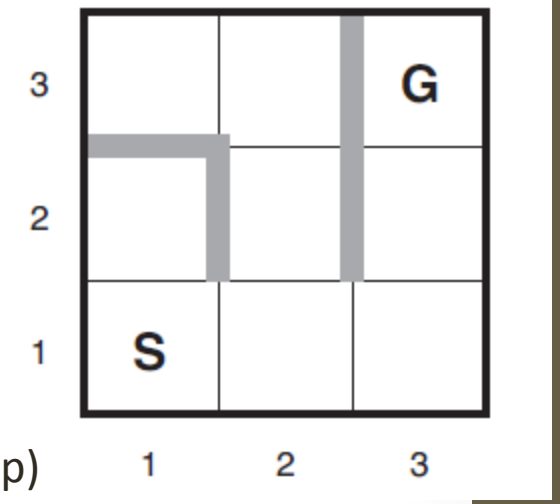


(b) Possible locations of robot After $E_1 = NSW, E_2 = NS$

Percepts usually reduce uncertainity

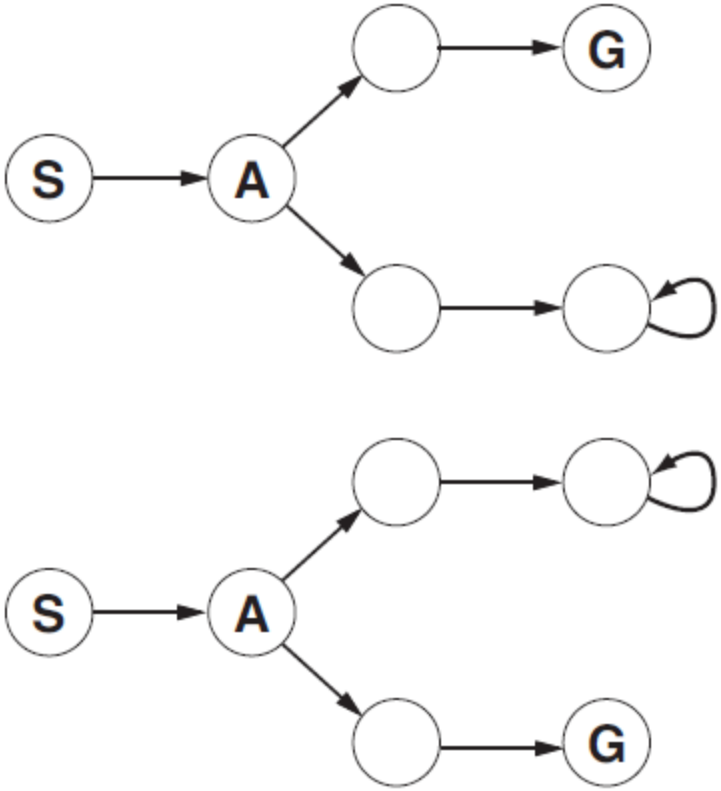
Online search

- Not find plan then execute then stop
- Compute, execute, observe, compute, execute, …
  - Interleave computation and action
  - Great for
    - dynamic domains
    - Non deterministic domains
  - Necessary in unknown environments
    - Robot localization in an unknown environment (no map)
    - Does not know about obstacles, where the goal is, that UP from (1,1) goes to (1, 2)
    - Once in (1, 2) does not know that down will go to (1, 1)
  - Some knowledge might be available
    - If location of goal is known, might use Manhattan distance heuristic
    - Competitive Ratio = Cost of shortest path without exploration/Cost of actual agent path
    - Irreversible actions can lead to dead ends and CR can become infinite

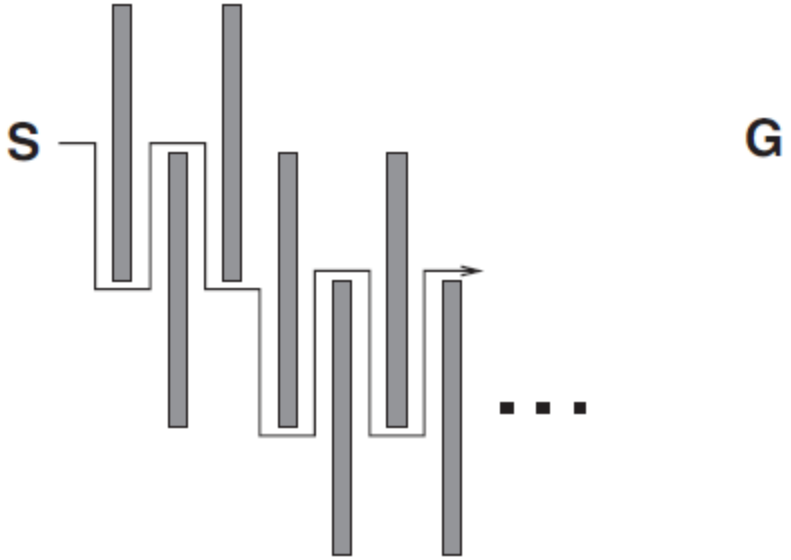


Examples

- Adversary argument



(a)



(b)

Online search algorithms

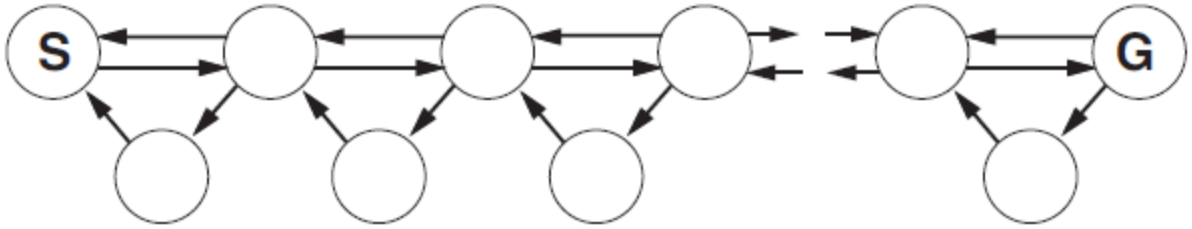
- Local search is better!
- Online-DFS

```
function ONLINE-DFS-AGENT(s') returns an action
  inputs: s', a percept that identifies the current state
  persistent: result, a table indexed by state and action, initially empty
              untried, a table that lists, for each state, the actions not yet tried
              unbacktracked, a table that lists, for each state, the backtracks not yet tried
              s, a, the previous state and action, initially null

  if GOAL-TEST(s') then return stop
  if s' is a new state (not in untried) then untried[s'] ← ACTIONS(s')
  if s is not null then
      result[s, a] ← s'
      add s to the front of unbacktracked[s']
  if untried[s'] is empty then
      if unbacktracked[s'] is empty then return stop
      else a ← an action b such that result[s', b] = POP(unbacktracked[s'])
  else a ← POP(untried[s'])
  s ← s'
  return a
```

Online local search

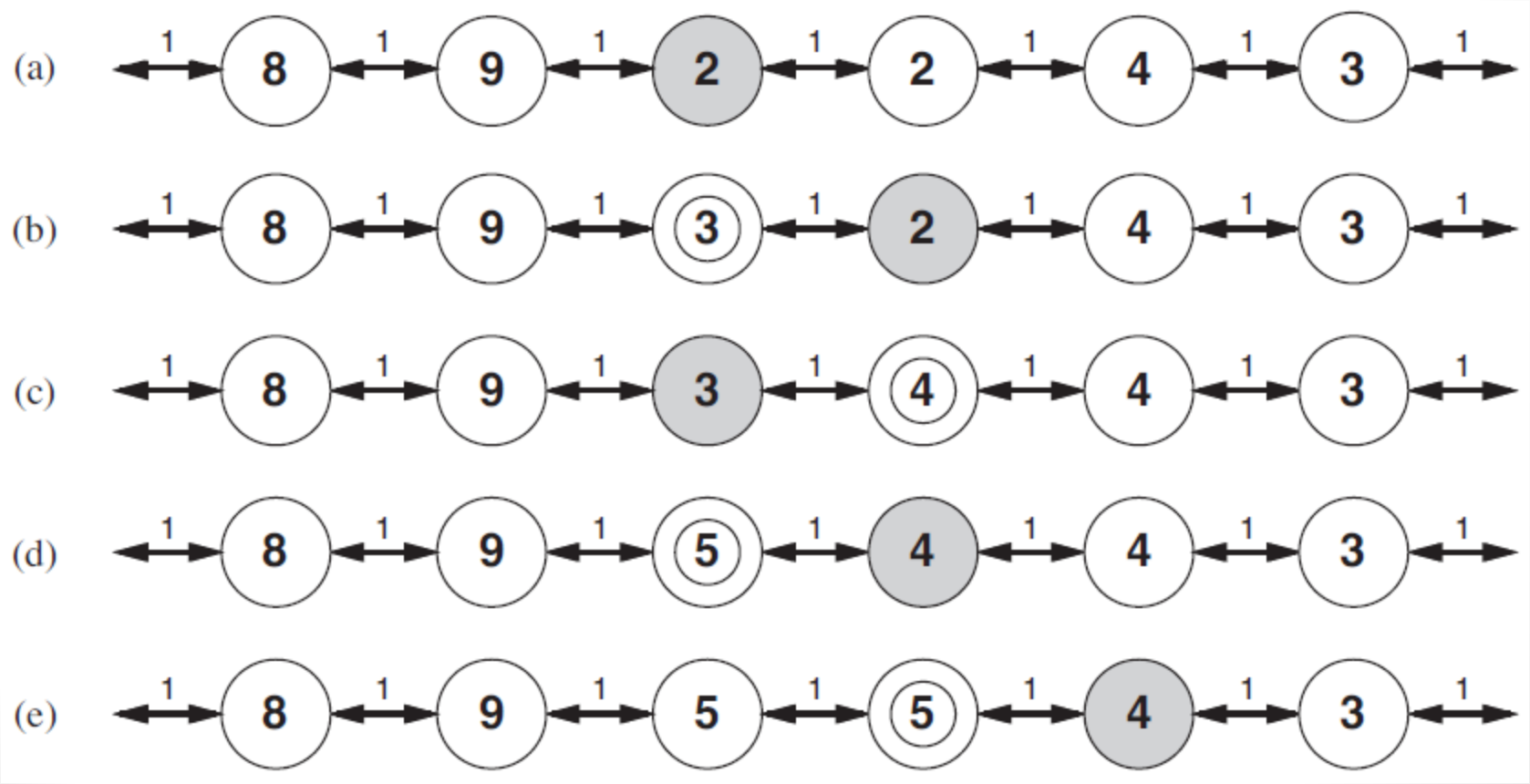
- Hill-climbing is already an online search algorithm but stops at local optimum. How about randomization?
  - Cannot do random restart (you can't teleport a robot)
  - How about just a random walk instead of hill-climbing?



  - Can be very bad (two ways back for every way forward above)
- Let's augment HC with memory
  - Learning real-time A* (LRTA*)
    - Updates cost estimates, g(s), for the state it leaves
    - Likes unexplored states
      - f(s) = h(s) not g(s) + h(s) for unexplored states

LRTA* Example

- We are in shaded state



LRTA* algorithm

function LRTA*-AGENT(s') **returns** an action
  inputs: s', a percept that identifies the current state
  persistent: *result*, a table, indexed by state and action, initially empty
    H, a table of cost estimates indexed by state, initially empty
    s, a, the previous state and action, initially null

  if GOAL-TEST(s') **then return** *stop*
  if s' is a new state (not in H) **then** $H[s'] \leftarrow h(s')$
  if s is not null
    $result[s, a] \leftarrow s'$
    $H[s] \leftarrow \min_{b \in \text{ACTIONS}(s)}$ LRTA*-COST($s, b, result[s, b], H$)
  $a \leftarrow$ an action b in ACTIONS(s') that minimizes LRTA*-COST($s', b, result[s', b], H$)
  $s \leftarrow s'$
  return a

function LRTA*-COST(s, a, s', H) **returns** a cost estimate
  if s' is undefined **then return** $h(s)$
  else return $c(s, a, s') + H[s']$

Questions

- DFS always expands at least as many nodes as A* with an admissable heuristic (True/False). Explain.
- H(n) = 0 is an admissible heuristic for the 8-puzzle
- BFS is complete even if 0 step costs are allowed

Types of task environments

Task Env	Observable	Agents	Deterministic	Episodic	Static	Discrete
Soccer						
Explore Titan						
Shopping for used AI books on the Net						
Playing tennis						
Playing tennis against a wall						
Performing a high jump						
Knitting a sweater						
Bidding on an item in an auction						