Artificial Intelligence

Nondeterministic Search

Christopher Simpkins

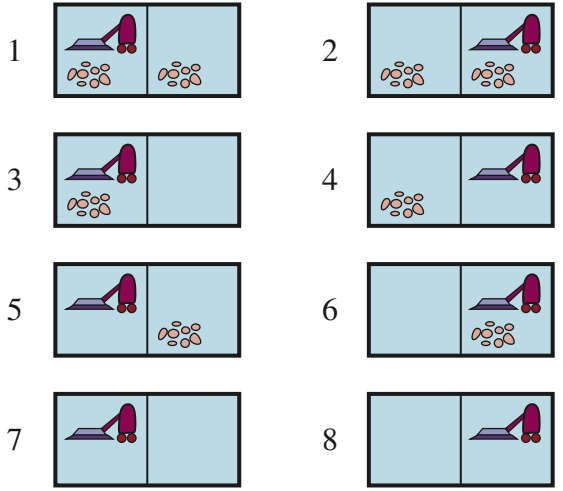
Nondeterministic Search (AIMA 4.3-4.5)

So far we've learned about search in fully observable, deterministic, known environments. In this lesson we consider:

- environments with nondeterministic actions,
- partially observable environments, and
- unknown environments.

States in the Vacuum World

Let's return to the vacuum world, whose states are:



Nondeterministic Actions: The Erratic Vacuum World

In the erratic vacuum world, the Suck action works as follows:

- When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.
- When applied to a clean square the action sometimes deposits dirt on the carpet.

So the result of each action is a set, e.g.:

$$\text{Results}(1, Suck) = \{5, 7\}$$

That set of states that the agent believes is possible, $\{5, 7\}$, is called a **belief state**.

A Factored Representation

Let's depart from the book for a few slides and, instead of using an index into a vector of states, create a factored representation for clarity.

- `left-condition` $\in$ {`CLEAN`, `DIRTY`}
- `right-condition` $\in$ {`CLEAN`, `DIRTY`}
- `vacuum-location` $\in$ {`LEFT`, `RIGHT`}
- State representation: `<vacuum-location, left-condition, right-condition>`

So

`Results(1,Suck)= {5, 7}`

becomes

`Results(<LEFT, DIRTY, DIRTY>, Suck)= {<LEFT, CLEAN, DIRTY>, <LEFT, CLEAN, CLEAN>}`

Note that the factored representation is easier for us to read (don't have to look up states in a table), but the search algorithms we're considering here treat these states as atomic.

Conditional Plans

A conditional plan, a.k.a. *contingency plan*, is a plan that specifies action selection based on the observed state while executing the plan.

- In a fully-observable, deterministic world contingencies are not necessary – a plan is just a sequence of actions.
- We need conditional/contingency plans in environments that are partially observable or nondeterministic.

Consider the start state, `<LEFT, DIRTY, DIRTY>`. Due to the environment's nondeterminism, not possible to find a sequence of actions guaranteed to solve the problem. But this simple conditional plan does:

```
[Suck, if State == <LEFT, CLEAN, DIRTY> then [Right, Suck] else []]
```

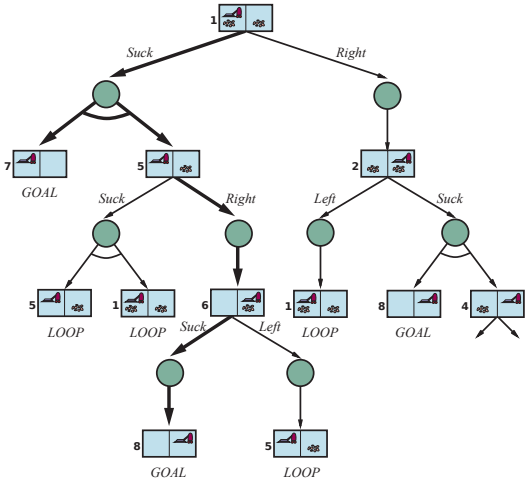
KENNESAW STATE UNIVERSITY

AND-OR Search Trees

- Branch on agent's action: **OR nodes**, shown as states.
- Branch on environment's outcome: **AND nodes**, shown as circles with arc linking branches to possible outcome states (when > 1).
- A plan includes actions for OR nodes, and conditional actions for AND nodes that contain more than one state.

Trace this conditional plan through the tree on the right.

```
[Suck,
if State == <LEFT, CLEAN, DIRTY> then [Right, Suck]
else []]
```



Slippery Vacuum World

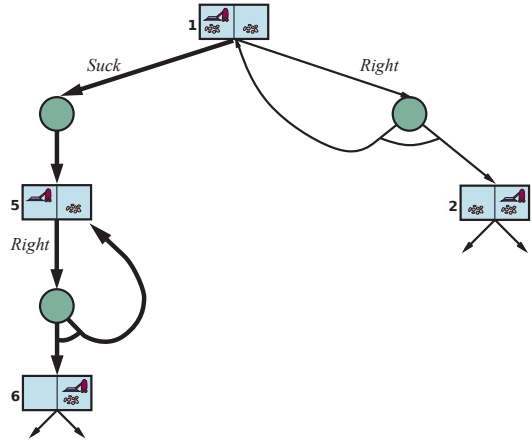
- Like deterministic vacuum world, but a movement action may result in no movement.

`Results(<LEFT, DIRTY, DIRTY>, Right)=`

`{<LEFT, DIRTY, DIRTY>, <RIGHT, DIRTY, DIRTY>}`

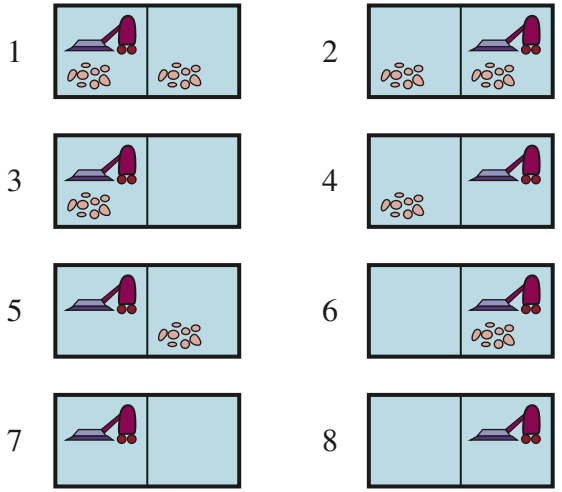
- Do deal with nondetermnistic movements we need cyclic plans. Use a **while** construct:

```
[Suck, while State == <LEFT, CLEAN, DIRTY> do Right,
Suck]
```



States in the Vacuum World

Recall the states of the vacuum world:



Search in Sensorless Environments

Now let's turn to uncertainty in the state observations, first with a sensorless world.

Sensorless, a.k.a. conformant, problems are surprisingly common.

- Manufacturing: orienting parts regardless of initial position.
- Medicine: applying broadly applicable treatments without running tests.

Consider a sensorless version of the (deterministic) vacuum world. Assume that the agent knows the geography of its world, but not its own location or the distribution of dirt.

Given an initial belief state is $\{1,2,3,4,5,6,7,8\}$.

- After [Right], belief state is $\{2,4,6,8\}$
- After [Right,Suck] belief state is $\{4,8\}$.
- After [Right,Suck,Left,Suck], belief state is $\{7\}$.

We say that the agent can **coerce** the world into state 7.

States in Sensorless Environments

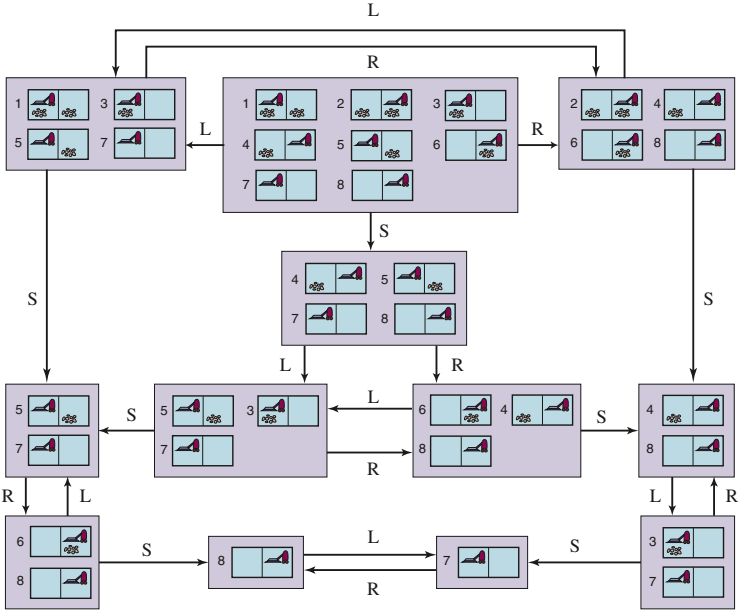
Instead of creating new algorithms, we transform the original problem into a belief state problem.

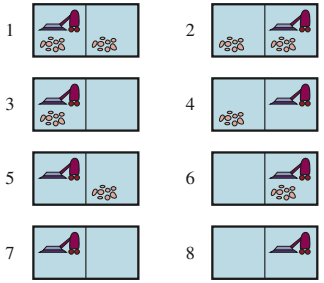
The original problem, P, has components $Actions_P$, $Result_P$ etc., and the belief-state problem has the following components:

- **States**: The belief-state space contains every possible subset of the physical states. If P has N states, then the belief-state problem has 2^N belief states, although many of those may be unreachable from the initial state (see next slide).
- **Initial state**: Typically the belief state consisting of all states in P, although in some cases the agent will have more knowledge than this.

Reachable States in Sensorless Vacuum World

Only 12 reachable belief states out of $2^8 = 256$ possible belief states.





Actions in Sensorless Environments

- **Actions**: If $b = \{s_1, s_2\}$, but $Actions_P(s_1) \neq Actions_P(s_2)$; then agent can't be sure which actions are legal. If illegal actions have no effect, safe to take union of all actions in the current belief state b:

$$Actions(b) = \bigcup_{s \in b} Actions_P(s)$$

If an illegal action might lead to catastrophe, safer to allow only the intersection – set of actions legal in all states. For the vacuum world, every state has the same legal actions, so both methods give the same result.

Transition Model in Sensorless Environments

- **Transition model**: For deterministic actions, the new belief state has one result state for each of the current possible states (although some result states may be the same):

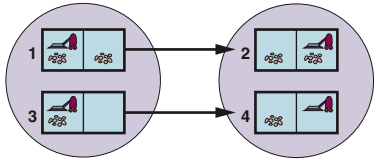
$$b' = Result(b, a) = \{s' : s' = Result_P(s, a) \text{ and } s \in b\}$$

With nondeterminism, the new belief state consists of all the possible results of applying the action to any of the states in the current belief state:

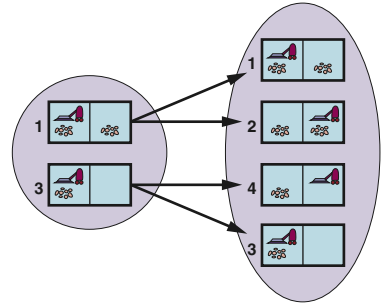
$$\begin{aligned} b' = Results(b, a) &= \{s' : s' \in Results_P(s, a) and s \in b\} \\ &= \bigcup_{s \in b} Results_P(s, a) \end{aligned}$$

The size of b' will be the same or smaller than b for deterministic actions, but may be larger than b with nondeterministic actions.

Predicting Belief States in Sensorless Vacuum World



(a)



(b)

Apply the action to all states in b to get b'.

- (a) Predicting the next belief state with the deterministic action, Right.
- (b) Prediction for the same belief state and action in the slippery sensorless vacuum world.

Goals and Action Costs in Sensorless Environments

- **Goal test**:
  - The agent possibly achieves the goal if $\exists s \in b : IsGoal_P(s)$.
  - The agent necessarily achieves the goal if $\forall s \in b : IsGoal_P(s)$.
  - We aim to necessarily achieve the goal.
- **Action cost**: If the same action can have different costs in different states, then the cost of taking an action in a given belief state could be one of several values. For now we assume that the cost of an action is the same in all states and so can be transferred directly from the underlying physical problem.

Search in Partially Observable Environments

Many problems cannot be solved without sensing, e.g., sensorless 8-puzzle is impossible.

We can solve 8-puzzles if we can see just the upper-left corner square by moving each tile in turn into the observable square and keeping track of its location from then on.

For a partially observable problem, the problem specification will specify a $Percept(s)$ function that returns the percept received by the agent in a given state.

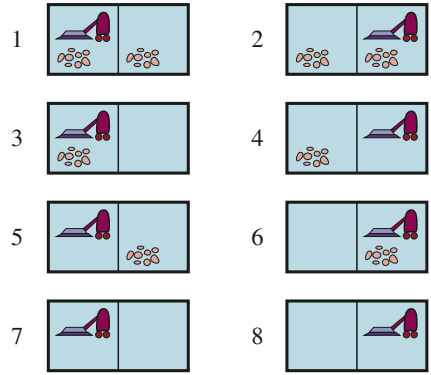
- For nondeterministic sensing, $Percepts(s) = \{s\}_{s \in S}$
- For fully observable problems, $\forall s, Percept(s) = s$
- For sensorless problems $Percept(s) =$ null.

Local-Sensing Vacuum World

The agent has a position sensor that yields the percept L in the left square, and R in the right square, and a dirt sensor that yields Dirty when the current square is dirty and Clean when it is clean – but does not sense the other square. This is nondeterministic sensing becuase the same percept can match more than one state:

- The PERCEPT in State 1 is [L,Dirty].
- State 3 will also produce [L,Dirty].
- Hence, the initial belief state will be {1,3}:
  - $Percepts(1) = \{1, 3\}$

How do we algorithmically get that belief state ...



Transition Model in Partially Observable Environments

After we apply an action in a given belief state, we can think of the transition model between belief states for partially observable problems as occurring in three stages, as depicted in the next slide:

- The **prediction** stage computes the belief state resulting from the action, Result(b,a), exactly as we did with sensorless problems. To emphasize that this is a prediction, we use the notation $\hat{b} = Result(b, a)$, where the hat over the b means "estimated," and we also use Predict(b,a) as a synonym for Result(b,a).
- The **possible percepts** stage computes the set of percepts that could be observed in the predicted belief state (using the letter o for observation):

$$PossiblePercepts(\hat{b}) = \{o : o = Percept(s) \text{ and } s \in \hat{b}\}$$

- The **update** stage computes, for each possible percept, the belief state that would result from the percept. The updated belief state b_o is the set of states in b that could have produced the percept:

$$b_o = Update(\hat{b}, o) = \{s : o = Percept(s) \text{ and } s \in \hat{b}\}$$

KENNESAW STATE UNIVERSITY

Planning Time State Estimation

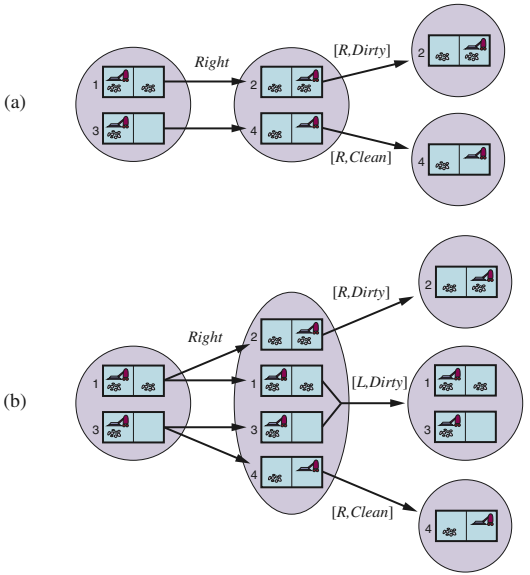
The agent needs to deal with possible percepts at planning time, because it won't know the actual percepts until it executes the plan.

- Nondeterminism in the physical environment can enlarge the belief state in the prediction stage, but each updated belief state b_o can be no larger than the predicted belief state $\hat{b}$; observations can only help reduce uncertainty.
- For deterministic sensing, the belief states for the different possible percepts will be disjoint, forming a partition of the original predicted belief state.

Putting the three stages from previous slide together, we obtain the possible belief states resulting from a given action and the subsequent possible percepts:

$$Results(b, a) = \{b_o : b_o = Update(Predict(b, a), o) \text{ and } o \in PossiblePercepts(Predict(b, a))\}.$$

State Transitions with Local Sensing



1. Predict $\hat{b} = Result(b, a)$

2. $PossiblePercepts(\hat{b}) =$
$$\{o : o = Percept(s) \text{ and } s \in \hat{b}\}$$

3. $b_o = Update(\hat{b}, o) =$
$$\{s : o = Percept(s) \text{ and } s \in \hat{b}\}$$

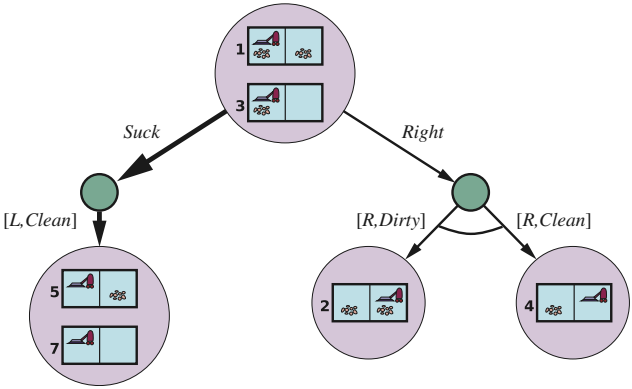
Final column on left shows input to Update step, which will then compute the final b_o based on percept/observation o.

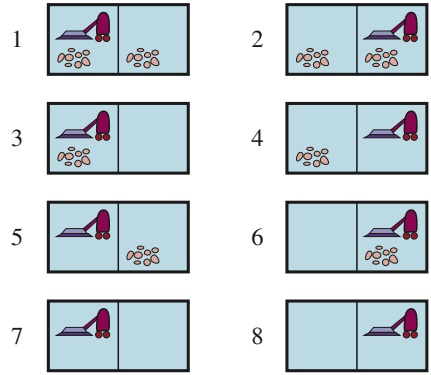
- (a) Deterministic world.
- (b) Slippery world. Input to Update step are 3 belief states, each of which is no larger than the belief state from which they were produced.

KENNESAW STATE UNIVERSITY

Local Sensing And-Or Trees

Given previous formulation of nondeterministic belief state problems, AND-OR can be used.
With initial percept [L, Dirty]:





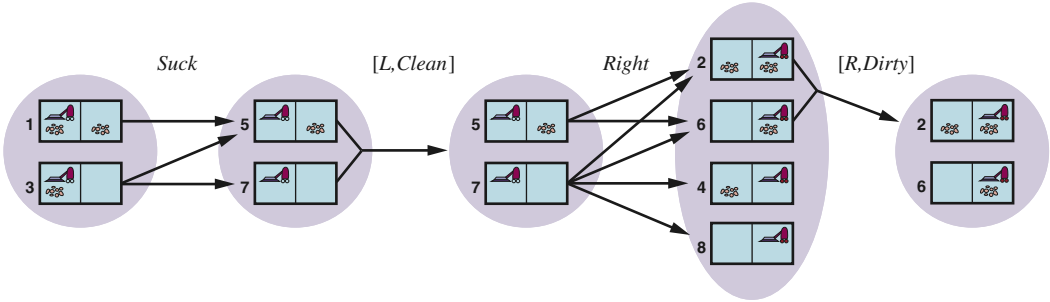
A partially observable problem can be solved by the AND-OR algorithm.

$$[Suck, Right, \textbf{ if } state = \{6\} \textbf{ then } Suck \textbf{ else } []]$$

KENNESAW STATE UNIVERSITY

Belief State Maintenance in Partially Observable Environments

Kindergarten world (square not being actively cleaned can become dirty):



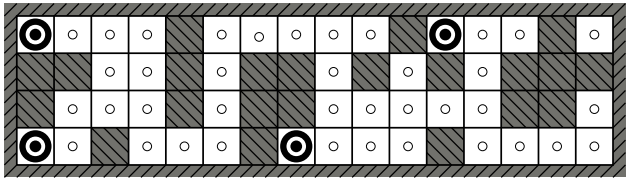
- Most real-world environments partially observable. Belief state maintenance is core task.
- Also known as **monitoring**, **filtering**, and **state estimation**.

$$b' = Update(Predict(b, a), o).$$

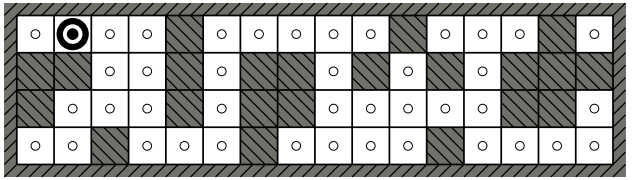
Equation above is called a recursive state estimator because it computes the new belief state from the previous one rather than by examining the entire percept sequence.

Robot Localization

Localization: typical robot state estimation problem in which the robot works out where it is, given a map of the world and a sequence of percepts and actions.



(a) Possible locations of robot after $E_1 = 1011$



(b) Possible locations of robot after $E_1 = 1011, E_2 = 1010$

Online Seach

- Agents we've studied so far use **offline search** algorithms, which compute a complete solution before taking first action.
- **Online search** agents interleave computation and action.
  - Good in dynamic environments where computation time must be limited so environment doesn't change while the agent computes an action.
  - Good in nondeterministic environments so agent can focus on contingencies that actually arise.
  - In unkown environments, agent must act in order to learn about the environment.
  - Tradeoff: more planning can prevent ending up in dead ends.

Canonical example of online search: mapping problem. Agent placed in unknown environment and must explore to build a map.

- The problem of doing localization and mapping at the same time is called **SLAM**: Simultaneous Localization and Mapping.