CS-E4800 Artificial Intelligence

Algorithms for Planning with Partial Observability

Jussi Rintanen

Department of Computer Science
Aalto University

March 18, 2021

Problem Definition

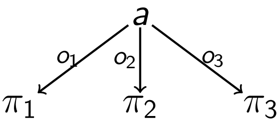
- Planning with partial observability, and without probabilities
- Objective: Reach one of the goal states (with probability 1)

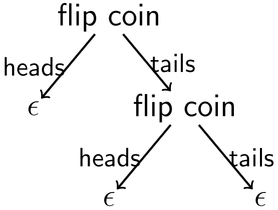
Problem Definition

- Planning with partial observability, and without probabilities
- Objective: Reach one of the goal states (with probability 1)
- Form of plans: tree-like branching programs
  - The empty program ϵ
  - Execute action a; Upon observation o_i execute program π_i
    notation: $a; \{(o_1, \pi_1), \ldots, (o_n, \pi_n)\}$

Problem Definition

- Planning with partial observability, and without probabilities
- Objective: Reach one of the goal states (with probability 1)
- Form of plans: tree-like branching programs
  - The empty program ϵ
  - Execute action a; Upon observation o_i execute program π_i
    notation: $a; \{(o_1, \pi_1), \ldots, (o_n, \pi_n)\}$





Approaches

Approaches to planning with partial observability (non-probabilistic):

- State-space search in belief space (instead of state space)
  - Algorithms for AND-OR tree search
  - Used with heuristic lower bound functions
  - Given a set of states, generate its alternative successors

Approaches

Approaches to planning with partial observability (non-probabilistic):

- State-space search in belief space (instead of state space)
  - Algorithms for AND-OR tree search
  - Used with heuristic lower bound functions
  - Given a set of states, generate its alternative successors
- Backward search starting from goal states
  - Breadth-first search style, brute force
  - Analogous to POMDP value iteration algorithms

Approaches

Approaches to planning with partial observability (non-probabilistic):

- State-space search in belief space (instead of state space)
  - Algorithms for AND-OR tree search
  - Used with heuristic lower bound functions
  - Given a set of states, generate its alternative successors
- Backward search starting from goal states
  - Breadth-first search style, brute force
  - Analogous to POMDP value iteration algorithms
- Others exist

Problem Definition

- Set S of states
- Set A of actions
- Set E of observations

Problem Definition

- Set S of states
- Set A of actions
- Set E of observations
- Transition relation $R_a \subseteq S \times S$ for every $a \in A$

Problem Definition

- Set S of states
- Set A of actions
- Set E of observations
- Transition relation $R_a \subseteq S \times S$ for every $a \in A$
- Observations $E_a \subseteq E$ for every $a \in A$
- Set $S_e \subseteq S$ for every observation $e \in E$

Problem Definition

- Set S of states
- Set A of actions
- Set E of observations
- Transition relation $R_a \subseteq S \times S$ for every $a \in A$
- Observations $E_a \subseteq E$ for every $a \in A$
- Set $S_e \subseteq S$ for every observation $e \in E$
- Set $I \subseteq S$ of initial states
- Set $G \subseteq S$ of goal states

The Observation Model

Example

We cannot observe which weekday it is, but we can observe

- $\text{DayOff} = \{\text{Saturday}, \text{Sunday}\}$
- $\text{WorkingDay} = \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}\}$

The Observation Model

Example

We cannot observe which weekday it is, but we can observe

- $\text{DayOff} = \{\text{Saturday}, \text{Sunday}\}$
- $\text{WorkingDay} = \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}\}$

- For every action a, the observations have to cover all successor states obtained by that action:
  For every $B \subseteq S$, $\text{img}_a(B) \subseteq \bigcup\{S_e | e \in E_a\}$

The Observation Model

Example

We cannot observe which weekday it is, but we can observe

- $\text{DayOff} = \{\text{Saturday}, \text{Sunday}\}$
- $\text{WorkingDay} = \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}\}$

- For every action a, the observations have to cover all successor states obtained by that action:
  For every $B \subseteq S$, $\text{img}_a(B) \subseteq \bigcup\{S_e | e \in E_a\}$
- "No observations" corresponds to $e_\top \in E$ such that $S_{e_\top} = S$

The Observation Model

Example

We cannot observe which weekday it is, but we can observe

- DayOff $= \{\text{Saturday}, \text{Sunday}\}$
- WorkingDay $= \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}\}$

- For every action a, the observations have to cover all successor states obtained by that action:
  For every $B \subseteq S$, $\text{img}_a(B) \subseteq \bigcup\{S_e | e \in E_a\}$
- "No observations" corresponds to $e_\top \in E$ such that $S_{e_\top} = S$
- We *do not assume* the disjointness of observations: $S_e \cap S_{e'} = \emptyset$ for $e \in E$ and $e' \in E$ such that $e \neq e'$

Image Operations

Successors of states w.r.t. an action:

Image

$\mathrm{img}_a(B) = \{s' | s \in B, sR_a s'\}$

Image Operations

Successors of states w.r.t. an action:

Image

$\text{img}_a(B) = \{s' | s \in B, sR_a s'\}$

Predecessors of states w.r.t. an action:

(Weak) Pre-Image

$\text{preimg}_a(B) = \{s | s' \in B, sR_a s'\}$

Strong Pre-Image

$\text{spreimg}_a(B) = \{s \in S | \emptyset \subset \text{img}_a(\{s\}) \subseteq B\}$

Successor Belief States

Successor Belief States

Given current belief state $B \subseteq S$ and action a, the successor belief states B' of B are all non-empty $\mathrm{succ}_a^o(B) = \mathrm{img}_a(B) \cap S_o$ for some $o \in E_a$

Successor Belief States

Successor Belief States

Given current belief state $B \subseteq S$ and action a, the successor belief states B' of B are all non-empty $\mathrm{succ}_a^o(B) = \mathrm{img}_a(B) \cap S_o$ for some $o \in E_a$

If $|E_a| = 1$, that is, action a has only one observation, then there is only one successor belief state.

Forward Search

- Image $\text{img}_a(B)$ for successors of states in B
- If observation o is made, then belief state is $\text{img}_a(B) \cap S_o$

Forward Search

- Image $\mathrm{img}_a(B)$ for successors of states in B
- If observation o is made, then belief state is $\mathrm{img}_a(B) \cap S_o$
- With multiple observations $o_1, \ldots, o_n$, multiple successors $\mathrm{img}_a(B) \cap S_{o_1}$, ..., $\mathrm{img}_a(B) \cap S_{o_n}$

Forward Search

- Image $\mathrm{img}_a(B)$ for successors of states in B
- If observation o is made, then belief state is $\mathrm{img}_a(B) \cap S_o$
- With multiple observations $o_1, \ldots, o_n$, multiple successors $\mathrm{img}_a(B) \cap S_{o_1}$, ..., $\mathrm{img}_a(B) \cap S_{o_n}$
- Search for plans by algorithms for AND-OR trees:
  - OR-nodes: choice of action
  - AND-nodes: choice of successor (observation)

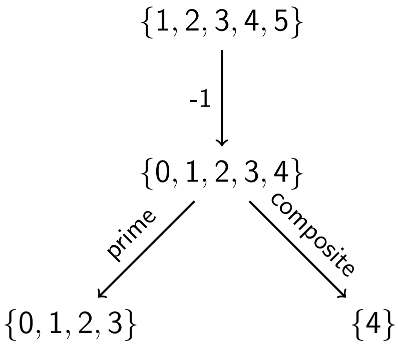
Forward Search

- Image $\mathrm{img}_a(B)$ for successors of states in B
- If observation o is made, then belief state is $\mathrm{img}_a(B) \cap S_o$
- With multiple observations $o_1, \ldots, o_n$, multiple successors $\mathrm{img}_a(B) \cap S_{o_1}$, ..., $\mathrm{img}_a(B) \cap S_{o_n}$
- Search for plans by algorithms for AND-OR trees:
  - OR-nodes: choice of action
  - AND-nodes: choice of successor (observation)
  - belief state for root node I
  - belief state for leaf nodes $\subseteq G$

Forward Search

- Image $\text{img}_a(B)$ for successors of states in B
- If observation o is made, then belief state is $\text{img}_a(B) \cap S_o$
- With multiple observations $o_1, \ldots, o_n$, multiple successors $\text{img}_a(B) \cap S_{o_1}$, ..., $\text{img}_a(B) \cap S_{o_n}$
- Search for plans by algorithms for AND-OR trees:
  - OR-nodes: choice of action
  - AND-nodes: choice of successor (observation)
  - belief state for root node I
  - belief state for leaf nodes $\subseteq G$
  - Algorithms:
    - Standard Depth First-Search (DFS) generalized to AND-OR trees
    - Informed algorithms: AO* (Nilsson 1980) and its various improvements

Example



Actions:

- Effect: -1, Observations: prime, composite
- Effect: $+2$, Observations: even, odd
- Effect: mod 2, Observations: -

Goal states: $\{1\}$

Example

$$\begin{tikzcd}
 & \{1, 2, 3, 4, 5\} \arrow[d, "+2"'] & \\
 & \{3, 4, 5, 6, 7\} \arrow[dl, "\text{odd}"] \arrow[dr, "\text{even}"] & \\
\{3, 5, 7\} & & \{4, 6\}
\end{tikzcd}$$

Actions:

- Effect: -1, Observations: prime, composite
- Effect: $+2$, Observations: even, odd
- Effect: mod 2, Observations: -

Goal states: $\{1\}$

Example

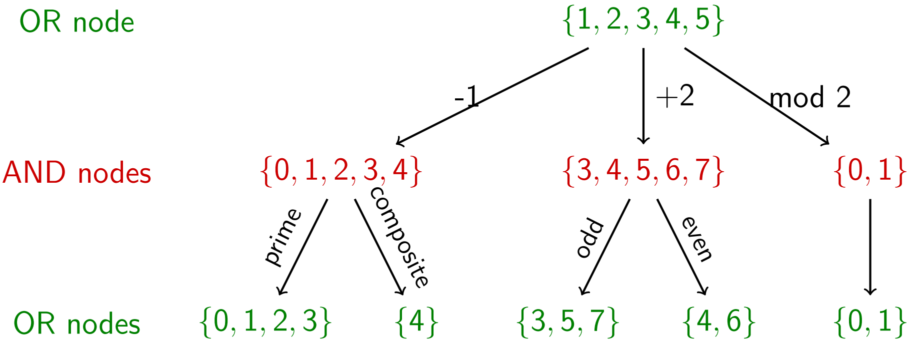
$$\begin{tikzcd}
\{1,2,3,4,5\} \arrow[d, "\text{mod } 2"'] \\
\{0,1\} \arrow[d] \\
\{0,1\}
\end{tikzcd}$$

Actions:

- Effect: -1, Observations: prime, composite
- Effect: $+2$, Observations: even, odd
- Effect: mod 2, Observations: -

Goal states: $\{1\}$

Example

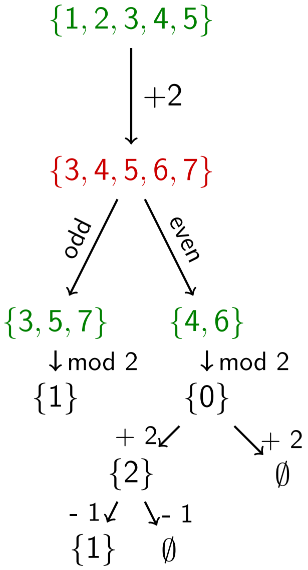

OR node
{1, 2, 3, 4, 5}
-1
+2
mod 2
AND nodes
{0, 1, 2, 3, 4}
{3, 4, 5, 6, 7}
{0, 1}
prime
composite
odd
even
OR nodes
{0, 1, 2, 3}
{4}
{3, 5, 7}
{4, 6}
{0, 1}

Example

OR node

AND nodes

OR nodes



Forward Search

procedure findplan(B,path,G)
if $B \subseteq G$ **then return** ϵ;
if $B_0 \subseteq B$ for some $B_0 \in$path **then return** NONE;
for each $a \in A$ **do**
    if a not applicable in some $s \in B$ **then** continue;
    success := true;
    subplans := $\emptyset$;
    for each $o \in E_a$
        subplan := findplan($\text{img}_a(B) \cap S_o$,path$\cup\{B\}$,G);
        if subplan = NONE **then** success := false; break;
        subplans := subplans$\cup\{(o, \text{subplan})\}$;
    if success **then return** (a;subplans);
return NONE;

Backward Search: Full Observability

Algorithm much simpler when problem fully observable

Algorithm (Full Observability)

1. $P := G$
2. $P_{\text{old}} := P$
3. $P := P \cup \bigcup_{a \in A} \text{spreimg}_a(P)$
4. If $I \subseteq P$, then return TRUE — Solution found.
5. If $P = P_{\text{old}}$ then return FALSE — No more states reached.
6. Go to 2

Backward Search: Idea

- Generate more and bigger belief states from which goals can be reached (with bigger/longer plans)

Backward Search: Idea

- Generate more and bigger belief states from which goals can be reached (with bigger/longer plans)
- Starting point: the set of goal states G (goals reachable by ϵ)

Backward Search: Idea

- Generate more and bigger belief states from which goals can be reached (with bigger/longer plans)
- Starting point: the set of goal states G (goals reachable by ϵ)
- Given belief states $\mathcal{B}$ from which goals reachable by $\leq i$ steps:
  Find state sets B and actions a such that
  - for every $o \in E$, $\text{succ}_a^o(B) \subseteq B_0$ for some $B_0 \in \mathcal{B}$, and
  - $B \not\subseteq B_0$ for all $B_0 \in \mathcal{B}$

  Goals reachable from B by $\leq i + 1$ steps

Backward Search: Backup Operation

backup(P): Find a plan that can reach goals from a new set of states

Backward Search: Backup Operation

backup(P): Find a plan that can reach goals from a new set of states

- Let $P = \{(\pi_1^*, B_1^*), \ldots, (\pi_n^*, B_n^*)\}$

Backward Search: Backup Operation

backup(P): Find a plan that can reach goals from a new set of states

- Let $P = \{(\pi_1^*, B_1^*), \ldots, (\pi_n^*, B_n^*)\}$
- Try every action $a \in A$
  1. Let $E_a = \{e_1, \ldots, e_m\}$ (observations possible after action a)
  2. Try every $\{(\pi_1, B_1), \ldots, (\pi_m, B_m)\} \subseteq P$
     1. Let $B = \text{spreimg}_a((S_{e_1} \cap B_1) \cup \cdots \cup (S_{e_m} \cap B_m))$
     2. If $B \not\subseteq B_i^*$ for every $i \in \{1, \ldots, n\}$, then return $(a; \{(e_1, \pi_1), \ldots, (e_m, \pi_m)\}, B)$

Backward Search: Backup Operation

backup(P): Find a plan that can reach goals from a new set of states

- Let $P = \{(\pi_1^*, B_1^*), \ldots, (\pi_n^*, B_n^*)\}$
- Try every action $a \in A$
  1. Let $E_a = \{e_1, \ldots, e_m\}$ (observations possible after action a)
  2. Try every $\{(\pi_1, B_1), \ldots, (\pi_m, B_m)\} \subseteq P$
     1. Let $B = \text{spreimg}_a((S_{e_1} \cap B_1) \cup \cdots \cup (S_{e_m} \cap B_m))$
     2. If $B \not\subseteq B_i^*$ for every $i \in \{1, \ldots, n\}$, then return $(a; \{(e_1, \pi_1), \ldots, (e_m, \pi_m)\}, B)$
- Return $\emptyset$ (No new plan could be found)

Example

For state space $\{0, 1, 2, 3, 4, 5, 6, 7\}$,
from $P = \{(\epsilon, \{1\})\}$, *backup* produces

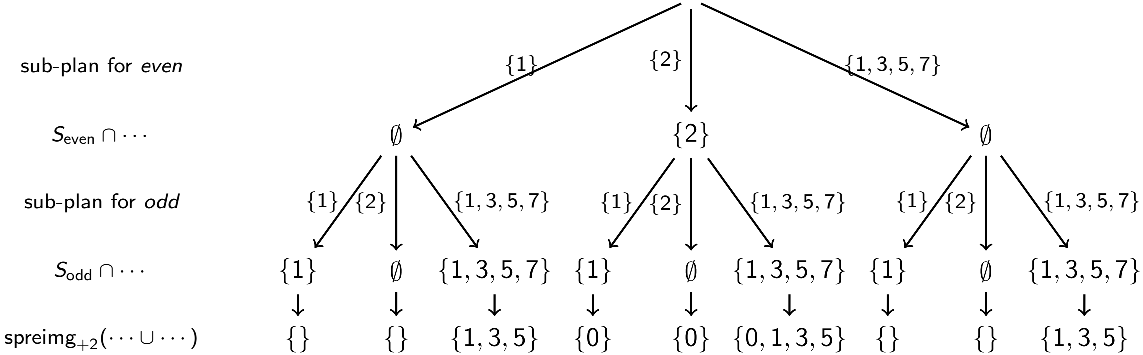
- $\{2\}$ by action -1,
- $\emptyset$ by action $+2$, and
- $\{1, 3, 5, 7\}$ by action mod 2.

This results in

$$P = \{(\epsilon, \{1\}), ((+1, \{(\mathit{prime}, \epsilon)\}), \{2\}), ((\mathit{mod}\ 2, \{(\mathit{odd}, \epsilon)\}), \{1, 3, 5, 7\})\}$$

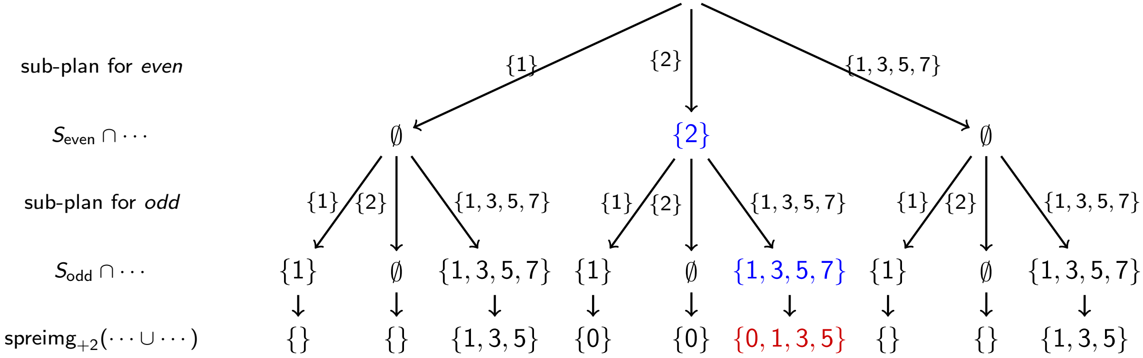
Example

Given $P = \{(\epsilon, \{1\}), ((-1, \{(prime, \epsilon)\}), \{2\}), ((mod\ 2, \{(odd, \epsilon)\}), \{1, 3, 5, 7\})\}$ and action $+2$, *backup* goes through the following search tree.



Example

Given $P = \{(\epsilon, \{1\}), ((-1, \{(prime, \epsilon)\}), \{2\}), ((mod\ 2, \{(odd, \epsilon)\}), \{1, 3, 5, 7\})\}$
and action $+2$, *backup* goes through the following search tree.



Only $\text{spreimg}_{+2}((S_{\text{even}} \cap \{2\}) \cup (S_{\text{odd}} \cap \{1,3,5,7\})) = \{0,1,3,5\}$ not already covered by P

It corresponds to plan $(+\ 2, \{(\text{even}, (-1, \{(\text{prime}, \epsilon)\})), (\text{odd}, (mod\ 2, \{(odd, \epsilon)\}))\})$

Backward Search: Algorithm

Algorithm

1. $P := \{(\epsilon, G)\}$
2. $(\pi, B_{\text{new}}) := \text{backup}(P)$
3. If $B_{\text{new}} = \emptyset$, stop — No new set could be found. No solution exists.
4. If $I \subseteq B_{\text{new}}$, then return π — Solution found.
5. $P := \{(\pi, B) \in P | B \not\subseteq B_{\text{new}}\} \cup \{(\pi, B_{\text{new}})\}$
6. Go to 2

Example

- Three packages 1, 2 and 3, of different unknown weights
- The heaviest package contains an expensive item
- Choose the package that contains the item

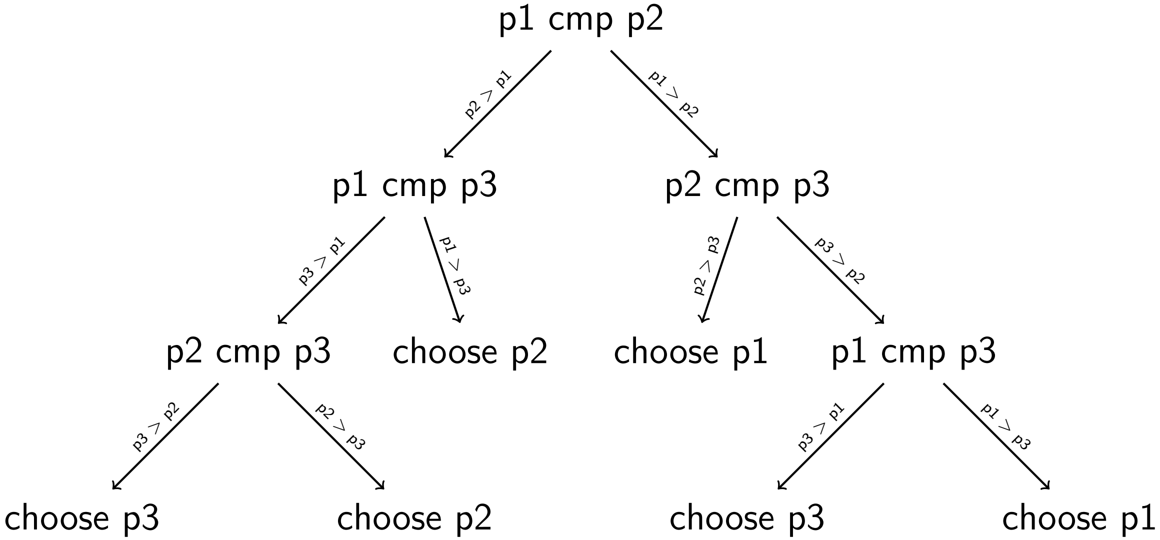
Example

- Three packages 1, 2 and 3, of different unknown weights
- The heaviest package contains an expensive item
- Choose the package that contains the item
- Actions:
  - Compare weights of packages i and j, with result $w(i) > w(j)$ or $w(j) > w(i)$
  - Choose package i, with $i \in \{1, 2, 3\}$

Example

- Three packages 1, 2 and 3, of different unknown weights
- The heaviest package contains an expensive item
- Choose the package that contains the item
- Actions:
  - Compare weights of packages i and j, with result $w(i) > w(j)$ or $w(j) > w(i)$
  - Choose package i, with $i \in \{1, 2, 3\}$
- Plan: Weigh packages to identify the heavy one, and choose it

Example



p1 cmp p2
p2 > p1
p1 > p2
p1 cmp p3
p2 cmp p3
p3 > p1
p1 > p3
p2 > p3
p3 > p2
p2 cmp p3
choose p2
choose p1
p1 cmp p3
p3 > p2
p2 > p3
p3 > p1
p1 > p3
choose p3
choose p2
choose p3
choose p1

Example: Problem Representation

- State space S consists of all (w_1, w_2, w_3, p) such that
  - $w_i \in \{1, 2, 3\}$ for $i \in \{1, 2, 3\}$ (weights of packages)
  - $w_1 \neq w_2$, $w_1 \neq w_3$, $w_2 \neq w_3$ (all of different weight)
  - $p \in \{1, 2, 3\}$ (the chosen package)

Example: Problem Representation

- State space S consists of all (w_1, w_2, w_3, p) such that
  - $w_i \in \{1, 2, 3\}$ for $i \in \{1, 2, 3\}$ (weights of packages)
  - $w_1 \neq w_2$, $w_1 \neq w_3$, $w_2 \neq w_3$ (all of different weight)
  - $p \in \{1, 2, 3\}$ (the chosen package)
- Initial states $I = S$ (weights are not known)
- Goal states are $(w_1, w_2, w_3, p) \in S$ such that $p = i$ and $w_i = 3$:
  $G = \{(1, 2, 3, 3), (1, 3, 2, 2), (2, 1, 3, 3), (2, 3, 1, 2), (3, 1, 2, 1), (3, 2, 1, 1)\}$

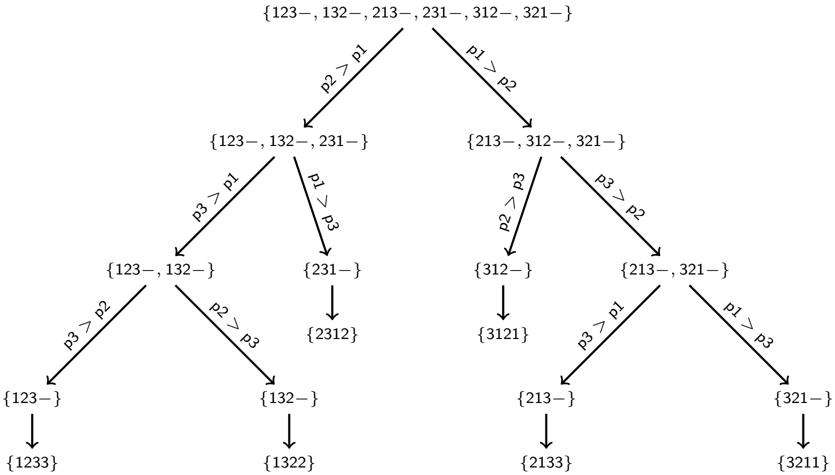
Example: Problem Representation

- State space S consists of all (w_1, w_2, w_3, p) such that
  - $w_i \in \{1, 2, 3\}$ for $i \in \{1, 2, 3\}$ (weights of packages)
  - $w_1 \neq w_2$, $w_1 \neq w_3$, $w_2 \neq w_3$ (all of different weight)
  - $p \in \{1, 2, 3\}$ (the chosen package)
- Initial states $I = S$ (weights are not known)
- Goal states are $(w_1, w_2, w_3, p) \in S$ such that $p = i$ and $w_i = 3$:

  $G = \{(1, 2, 3, 3), (1, 3, 2, 2), (2, 1, 3, 3), (2, 3, 1, 2), (3, 1, 2, 1), (3, 2, 1, 1)\}$
- Action: Choose i, turns state (w_1, w_2, w_3, p) to (w_1, w_2, w_3, i)
- Action: Compare i-j produces the observations
  - GT if $w_i > w_j$
  - LT if $w_i < w_j$

  from (w_1, w_2, w_3, p), and does not change the state

Example



There are 18 initial states. Above, $123-$ is all sequences with prefix 123, that is $\{1231, 1232, 1233\}$.

Example: Running the Algorithm

Goal states:

$$P_0 = \{(\epsilon, \{(1,2,3,3),(1,3,2,2),(2,1,3,3),(2,3,1,2),(3,1,2,1),(3,2,1,1)\})\}$$

Example: Running the Algorithm

Goal states:

$$P_0 = \{(\epsilon, \{(1,2,3,3),(1,3,2,2),(2,1,3,3),(2,3,1,2),(3,1,2,1),(3,2,1,1)\})\}$$

Add states from which goals reached with *Choose 3*:

$$P_1 = P_0 \cup \{((\text{Choose } 3; \{(e_\top, \epsilon)\}), \{(1,2,3,1),(2,1,3,1),(1,2,3,2),(2,1,3,2),(1,2,3,3),(2,1,3,3)\})\}$$

Example: Running the Algorithm

Goal states:

$$P_0 = \{(\epsilon, \{(1,2,3,3),(1,3,2,2),(2,1,3,3),(2,3,1,2),(3,1,2,1),(3,2,1,1)\})\}$$

Add states from which goals reached with *Choose 3*:

$$P_1 = P_0 \cup \{((\text{Choose } 3; \{(e_\top, \epsilon)\}), \{(1,2,3,1),(2,1,3,1),(1,2,3,2),(2,1,3,2),(1,2,3,3),(2,1,3,3)\})\}$$

Add states from which goals reached with *Choose 2*:

$$P_2 = P_1 \cup \{((\text{Choose } 2; \{(e_\top, \epsilon)\}), \{(1,3,2,1),(2,3,1,1),(1,3,2,2),(2,3,1,2),(1,3,2,3),(2,3,1,3)\})\}$$

Example: Running the Algorithm

Goal states:

$$P_0 = \{(\epsilon, \{(1,2,3,3),(1,3,2,2),(2,1,3,3),(2,3,1,2),(3,1,2,1),(3,2,1,1)\})\}$$

Add states from which goals reached with *Choose 3*:

$P_1 = P_0 \cup \{((\text{Choose 3}; \{(e_\top, \epsilon)\}), \{(1,2,3,1),(2,1,3,1),(1,2,3,2),(2,1,3,2),(1,2,3,3),(2,1,3,3)\})\}$

Add states from which goals reached with *Choose 2*:

$P_2 = P_1 \cup \{((\text{Choose 2}; \{(e_\top, \epsilon)\}), \{(1,3,2,1),(2,3,1,1),(1,3,2,2),(2,3,1,2),(1,3,2,3),(2,3,1,3)\})\}$

Add states from which goals reached with *Choose 1*:

$P_3 = P_2 \cup \{((\text{Choose 1}; \{(e_\top, \epsilon)\}), \{(3,1,2,1),(3,2,1,1),(3,1,2,2),(3,2,1,2),(3,1,2,3),(3,2,1,3)\})\}$

Example: Running the Algorithm

$$P_3 = \{(\epsilon, G), ((\text{Choose } 1; \pi_1), B_1), ((\text{Choose } 2; \pi_2), B_2), ((\text{Choose } 3; \pi_3), B_3)\}$$

Example: Running the Algorithm

$$P_3 = \{(\epsilon, G), ((\text{Choose } 1; \pi_1), B_1), ((\text{Choose } 2; \pi_2), B_2), ((\text{Choose } 3; \pi_3), B_3)\}$$

Add states from which belief states in P_3 reached with *Compare 2-3*:

$$P_4 = P_3 \cup \{((\text{Compare 2-3}; \{(e_{\text{GT}}, \pi_2), (e_{\text{LT}}, \pi_3)\}), B_2 \cup B_3)\}$$

Example: Running the Algorithm

$$P_3 = \{(\epsilon, G), ((\text{Choose } 1; \pi_1), B_1), ((\text{Choose } 2; \pi_2), B_2), ((\text{Choose } 3; \pi_3), B_3)\}$$

Add states from which belief states in P_3 reached with *Compare 2-3*:

$$P_4 = P_3 \cup \{((\text{Compare } 2\text{-}3; \{(e_{\text{GT}}, \pi_2), (e_{\text{LT}}, \pi_3)\}), B_2 \cup B_3)\}$$

Add states from which belief states in P_3 reached with *Compare 1-3*:

$$P_5 = P_4 \cup \{((\text{Compare } 1\text{-}3; \{(e_{\text{GT}}, \pi_1), (e_{\text{LT}}, \pi_3)\}), B_1 \cup B_3)\}$$

Example: Running the Algorithm

$$P_3 = \{(\epsilon, G), ((\text{Choose } 1; \pi_1), B_1), ((\text{Choose } 2; \pi_2), B_2), ((\text{Choose } 3; \pi_3), B_3)\}$$

Add states from which belief states in P_3 reached with *Compare 2-3*:

$$P_4 = P_3 \cup \{((\text{Compare 2-3}; \{(e_{\text{GT}}, \pi_2), (e_{\text{LT}}, \pi_3)\}), B_2 \cup B_3)\}$$

Add states from which belief states in P_3 reached with *Compare 1-3*:

$$P_5 = P_4 \cup \{((\text{Compare 1-3}; \{(e_{\text{GT}}, \pi_1), (e_{\text{LT}}, \pi_3)\}), B_1 \cup B_3)\}$$

This now includes the two bottom levels of the plan given earlier
Additional steps will construct the rest of that plan...

Conclusion

- We have discussed algorithm for planning with partial observability
- Based on three basic operations:
  - Compute successor state set with respect to a given action
  - Compute successor state set with respect to a given observation
  - Compute predecessor state set with respect to a given action

Conclusion

- We have discussed algorithm for planning with partial observability
- Based on three basic operations:
  - Compute successor state set with respect to a given action
  - Compute successor state set with respect to a given observation
  - Compute predecessor state set with respect to a given action
- Two main approaches:
  - Forward search: Plan constructed top-down
  - Backward search: Plan constructed bottom-up

Conclusion

- We have discussed algorithm for planning with partial observability
- Based on three basic operations:
  - Compute successor state set with respect to a given action
  - Compute successor state set with respect to a given observation
  - Compute predecessor state set with respect to a given action
- Two main approaches:
  - Forward search: Plan constructed top-down
  - Backward search: Plan constructed bottom-up
- Complexity exponentially higher than with full-observability
- Number of belief states exponentially higher than number of states