Principles of AI Planning

February 7th, 2007 — Nondeterministic planning with partial observability

Principles of AI Planning

Nondeterministic planning with partial observability

Malte Helmert   Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

February 7th, 2007

Nondeterministic planning with partial observability

Planning with partial observability is harder than both the fully observable and unobservable cases:

- Memoryless plans (where the next action to take only depends on the current situation) as in the fully observable case are not sufficient.
  - Of course, we cannot define a memoryless plan based on individual states because limited observability makes some states indistinguishable.
  - It is also not sufficient to consider memoryless plans where the action to take is based on the current observation class.
- Conformant (i.e., non-branching) plans as in the unobservable case are also clearly not powerful enough.

Strong planning

- We will (mostly) consider the strong planning problem.
- Generalizations to the strong cyclic planning are similar to the fully observable case.

Algorithms

Similar to other variants of the planning problem, there are three major approaches to nondeterministic planning with partial observability:

- Reduction to another problem
- Forward search
- Backward search

We will consider one example for each of these.

Algorithms

Three approaches

Reduction to another problem:

- Reduce to planning with full observability.

Forward search (progression):

- Define the search space as an AND/OR tree.
- Define a heuristic function for such trees.
- Use a tree search algorithm such as AO$*$ or Proof Number Search.

Backward search (regression):

- Start from the set of goal states.
- Find state sets from which already generated state sets can be reached by applying operators and making observations.

Reduction to fully observable case

- Memoryless plans are not sufficient for the partially observable case because a plan must take into account the knowledge collected in previous observations etc.
- During plan execution, this knowledge is represented in the current belief state.
- One idea for solving a partially observable task $\mathcal{T}$ is to map it to a fully observable task $\mathcal{T}'$ where each belief state of $\mathcal{T}$ corresponds to a state of $\mathcal{T}'$.

Reduction to fully observable case

State variables

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

State variables

- For each state $s \in S$, there is one state variable $v_s \in A'$.
- Intuition: v_s is true in a state of $\mathcal{T}'$ iff it is possible that we are currently in s.
- Formally: $A' := \{ v_s \mid s \in S \}$

Reduction to fully observable case

Initial state formula

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

Initial state formula

- The initial state of $\mathcal{T}'$ is fully deterministic (in terms of A'), as there is only one possible initial belief state in $\mathcal{T}$.
- For all states s in the initial belief state of $\mathcal{T}$, variable v_s is initially true. Other variables are initially false.
- Formally: $I' := \bigwedge_{s \in S, s \models I} v_s \wedge \bigwedge_{s \in S, s \not\models I} \neg v_s$.

Reduction to fully observable case

Initial state formula

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

Goal formula

- A goal belief state of $\mathcal{T}$ is one where all possible states satisfy G.
- This is equivalent to saying that no state in the current belief state violates G.
- We can express that by saying that none of the variables v_s for states s violating G are true.
- Formally: $G' := \bigwedge_{s \in S, s \not\models G} \neg v_s$.

Reduction to fully observable case

Initial state formula

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

Operators (preconditions)

- Each operator $o = \langle c, e \rangle \in O$ is translated to an operator $o' = \langle c', e' \rangle \in O'$.
- To test whether operator o is applicable, we must verify that all states in the current belief state satisfy c.
- Again, this is equivalent to saying that no state in the current belief state violates c.
- Formally: $c' := \bigwedge_{s \in S, s \not\models c} \neg v_s$.

Reduction to fully observable case

Initial state formula

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

Operators (effects)

- Each operator $o = \langle c, e \rangle \in O$ is translated to an operator $o' = \langle c', e' \rangle \in O'$.
- After applying operator o, we can possibly be in state $s \in S$ iff we were previously in some state in which o is applicable and from which applying o can lead to s.
- This is modeled by an effect $((\bigvee_{t \in preimg_o(s)} v_t) \rhd v_s) \wedge (\neg(\bigvee_{t \in preimg_o(s)} v_t) \rhd \neg v_s)$.
- Formally: $e' := \bigwedge_{s \in S}(((\bigvee_{t \in preimg_o(s)} v_t) \rhd v_s) \wedge (\neg(\bigvee_{t \in preimg_o(s)} v_t) \rhd \neg v_s))$.

Reduction to fully observable case

Done?

- We have translated state variables, initial state formula, goal formula and operators.

Is that it?

- So far, our translation is independent of the set of observable variables V!
- Moreover, the resulting planning task is deterministic!

Is there an error in our modeling?

Reduction to fully observable case

Not done

Is there an error in our modeling?

- No, but it is not complete yet: There are solvable partially observable tasks $\mathcal{T}$ for which $\mathcal{T}'$ (as defined so far) is unsolvable.
- The reason for this is that he have not yet modeled the possibility of observing state variables.

Modeling observations requires introducing nondeterminism in $\mathcal{T}'$.

Reduction to fully observable case

Observations

Let $\mathcal{T} = \langle A, I, O, G, V \rangle$ be the input task with state set S.
We define the fully observable task $\mathcal{T}' = \langle A', I', O', G', A' \rangle$.

Observations

- In general, our formalism allows observations to be general formulas over V. However, it is sufficient to only consider atomic observations $u \in V$.
- If we observe u in a belief state b, we can end up in two different belief states: one containing exactly the states of b where u is true, and one containing exactly the states of b where u is false.
- In other words, either the belief states where u is false or the belief states where u is true are ruled out.
- Formally: Translate observation of $u \in V$ into an operator $\langle \top, e'_u \rangle \in O'$ with $e'_u := (\bigwedge_{s \in S, s \not\models u} \neg v_s) | (\bigwedge_{s \in S, s \models u} \neg v_s)$.

Reduction to fully observable case

Discussion

- Note that the reduction works both for strong and for strong cyclic planning.
- The reduction has a significant drawback: Since it introduces as many state variables as there are states in the original task, the resulting problem is exponentially larger than the original one.
- This will usually not be practical.
- On the other hand, there does not really exist any truly "practical" algorithm for nondeterministic planning with partial observability.

Reduction to fully observable case

Complexity result

- Using an exponential-time planning algorithm for fully observable planning, $\mathcal{T}'$ can be solved in time $O(c^{\|\mathcal{T}'\|})$, and $\|\mathcal{T}'\| = O(c^{\|\mathcal{T}\|})$.
- Thus, we have a double-exponential ($O(c^{c^{\|\mathcal{T}\|}})$) algorithm for nondeterministic planning for partial observability.
- We will later prove that this is worst-case optimal.

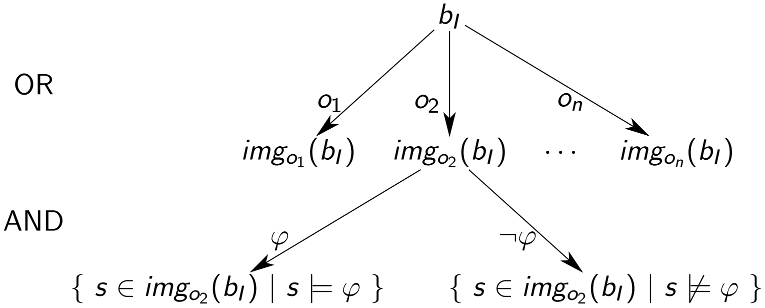
Search in AND/OR trees

In forward search, plans are represented as trees whose nodes represent the situations arising during plan execution.

- The root node represents the initial situation.
- OR nodes correspond to choosing and applying operators.
  - Note how these relate to operators in $\mathcal{T}'$ in the earlier reduction.
- AND nodes correspond to making observations.
  - Note how these relate to nondeterminism in $\mathcal{T}'$ in the earlier reduction.

Search in AND/OR trees

Example



AND/OR trees

Formal definition

Definition

An AND/OR tree is a labeled rooted tree where

- internal nodes are labeled with $(\wedge)$ or $(\vee)$ (AND nodes/OR nodes), and
- leaves are labeled with $(\top)$ or $(\bot)$ (true leaves/false leaves).

AND/OR trees

Truth value

Definition

An AND/OR tree evaluates to true iff

- it is a true leaf,
- it is an OR node with a child that evaluates to true, or
- it is an AND node whose children all evaluate to true.

Partial plan trees

Definition

A partial plan tree for a nondeterministic planning task $\langle A, I, O, G, V\rangle$ with state set S is an AND/OR tree with the following properties:

- Each node n has an associated belief state $b(n)$.
- If n is the root node, then $b(n) = \{\ s \in S \mid s \models I\ \}$.
- A leaf node n is labeled with $(\top)$ iff $b(n) \models G$. In this case it is called a goal node, otherwise an open node.
- ...

Partial plan trees

Definition (ctd.)

A partial plan tree for a nondeterministic planning task $\langle A, I, O, G, V\rangle$ with state set S is an AND/OR tree with the following properties:

- ...
- An OR node n (also called an operator node) has one child n_o for each operator $o \in O$ applicable in $b(n)$, with associated belief state $b(n_o) = app_o(b(n))$.
- An AND node n (also called an observation node) has an associated formula $\varphi(n)$ over V. It has two children:
  - $n^\top$ with $b(n^\top) = \{\ s \in b(n) \mid s \models \varphi\}$
  - $n^\perp$ with $b(n^\perp) = \{\ s \in b(n) \mid s \not\models \varphi\}$.

Forward planning as search in partial plan trees

- Clearly, a partial plan tree represents a strategy.
- This strategy is a strong plan iff the tree evaluates to true.

We thus obtain a (nondeterministic) forward search algorithm:

Forward search in partial plan trees

def expand-tree($\mathcal{T}$):
    Set T to the partial plan tree for $\mathcal{T}$ that consists
        of a single leaf, labeled with the initial belief state.
    while T evaluates to false:
        Choose some open leaf n in T.
        Replace n by an operator or observation node,
            adding the necessary children to T.

Search in AND/OR trees

Issues

- There is a conflict between plan size and observing:
  - With many observations, plans become very big.
  - With few observations, it may be impossible to find a plan.

  Trying out all possible ways to branch is not feasible.

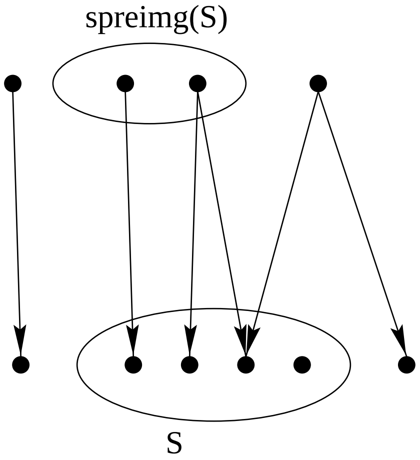
  No good general solutions to this problem exist.
- AND-OR search algorithms use heuristics for making branching decisions.
  - But they do not really work well...

Backward search algorithms

- Backward search algorithms are similar in flavour to the ones for fully observable problems.
- Backward steps with operator application:
  - Compute strong preimages.
- Backward steps with observations:
  - Compute union of belief states from disjoint observational classes.
  - Note: Can always take subsets of solved belief states to make them disjoint.

Backward search algorithms

Regression: strong preimages

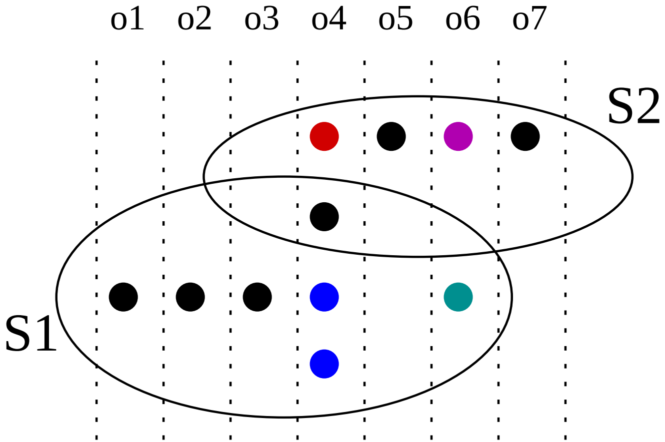


Observations in backward search

- Let $C_1, \ldots, C_n$ be different observational classes.
- Let $B_1, \ldots, B_n$ be belief states with $B_i \subseteq C_i$ for all $i = 1, \ldots, n$ for which we have a solution plan.
- Then we can find a plan for $B = B_1 \cup \cdots \cup B_n$ by first observing in which class C_i we are and then applying the corresponding plan for B_i.

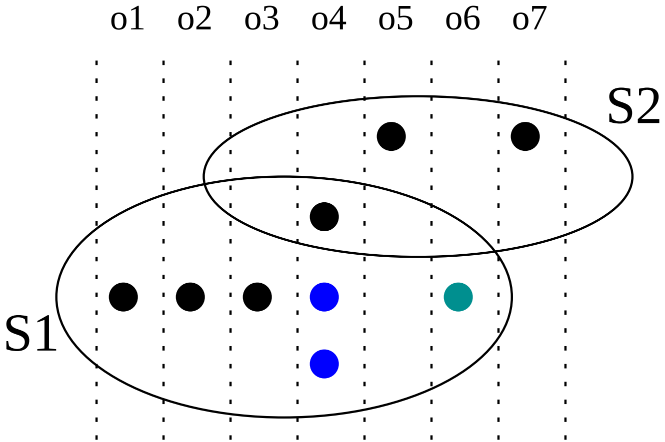
Observations in backward search

Example: Combining two belief states



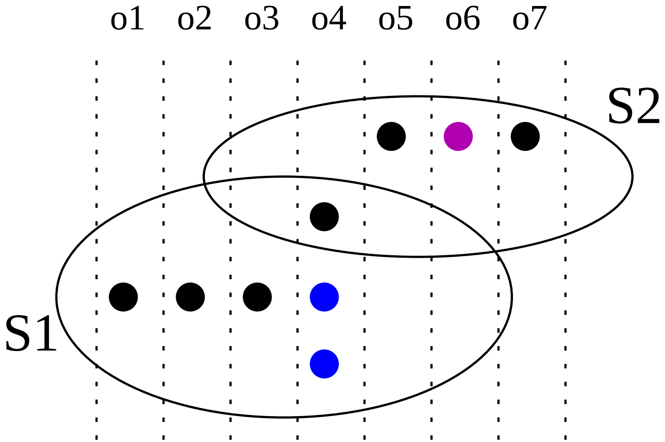
Observations in backward search

Example: Combining two belief states, option 1



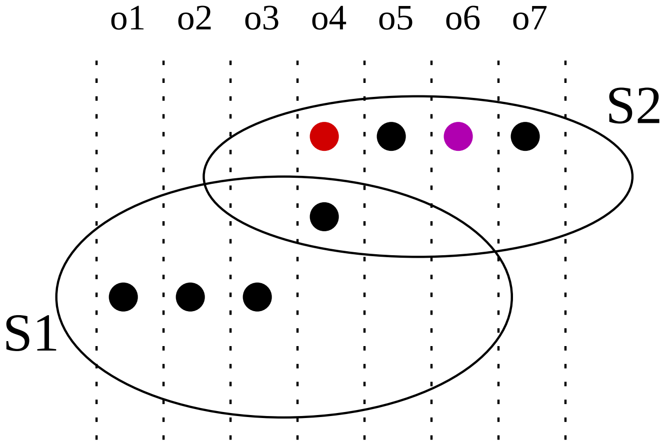
Observations in backward search

Example: Combining two belief states, option 2



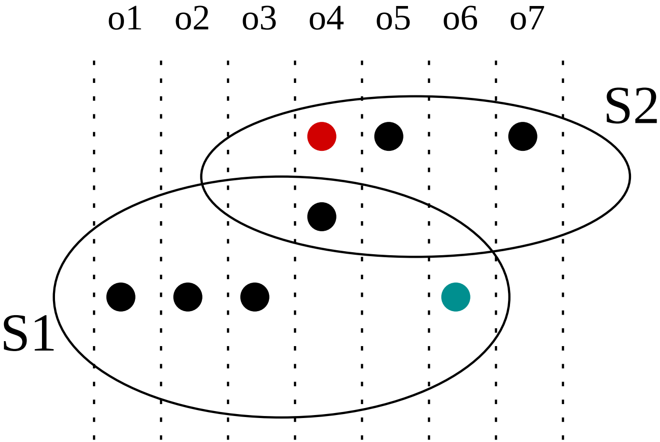
Observations in backward search

Example: Combining two belief states, option 3



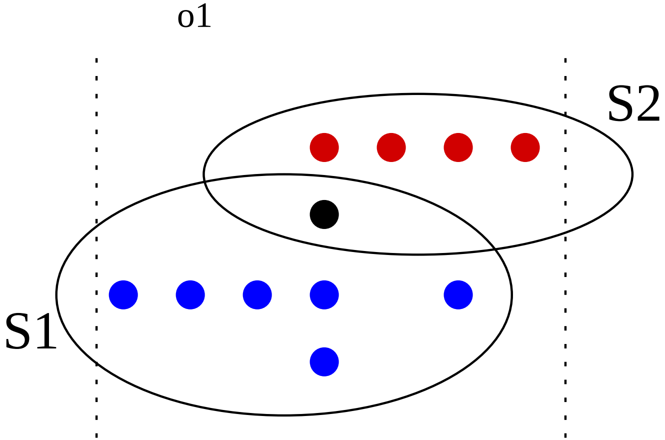
Observations in backward search

Example: Combining two belief states, option 4



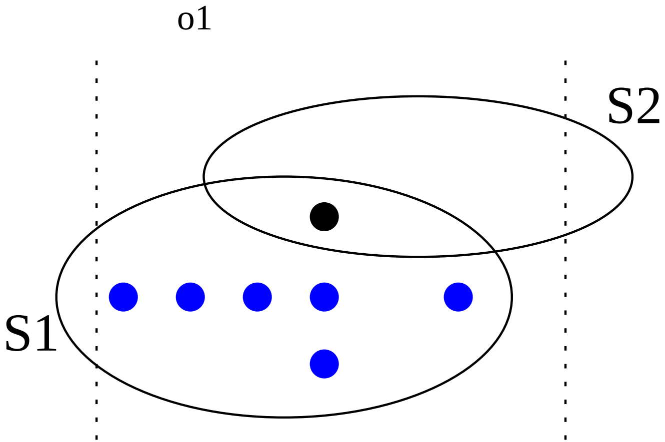
No observability ⇒ no branching

Only one observational class: no choice between subplans



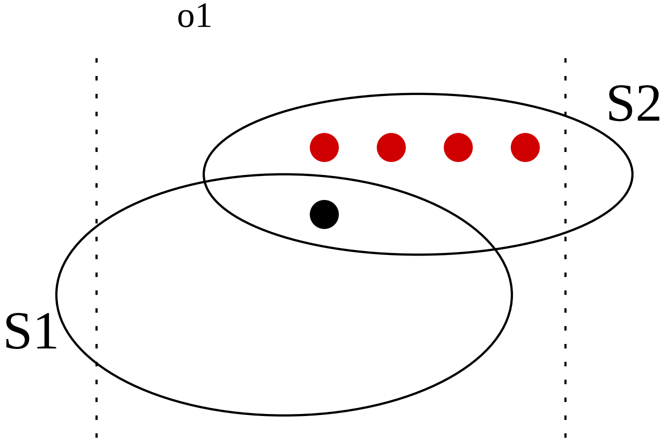
No observability ⇒ no branching

No choice between subplans during execution: option 1



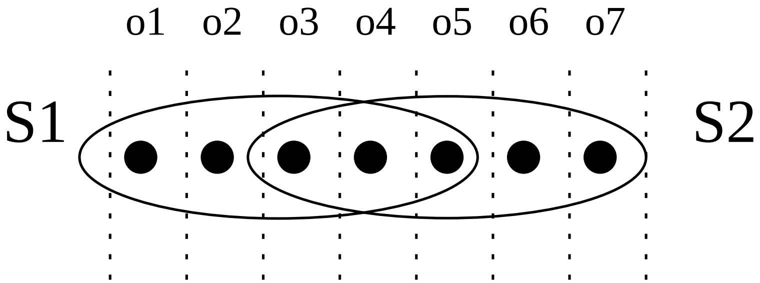
No observability ⇒ no branching

No choice between subplans during execution: option 2



Full observability ⇒ arbitrary branching

A different plan can be used for every state



A systematic backward algorithm

Idea: always split belief states into all observational classes.

Initially, the set of solved belief states includes the set $b_G \cap C_i$ for each observational class C_i, where b_G is the belief state containing all states satisfying the goal.

Then iterate the following steps:

1. Pick one belief state b_i for each observational class and compute their union b.
2. If b includes all initial states $\rightsquigarrow$ solution.
3. Otherwise, compute the strong preimage of b with respect to some operator o.
4. Split the resulting set of states to belief states for different observational classes and add them to the set of solved belief states.

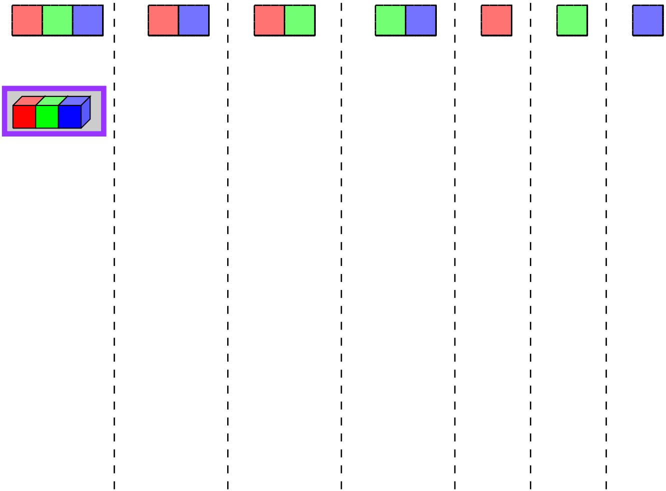
Backward search

Example

- Blocks world with three blocks
- Goal: all blocks are on the table
- Only the variables $\mathsf{clear}(X)$ are observable.
- A block can be moved onto the table if the block is clear.
- 8 observational classes corresponding to the 8 valuations of $\{\mathsf{clear}(A), \mathsf{clear}(B), \mathsf{clear}(C)\}$ (one of the valuations does not correspond to a blocks world state).

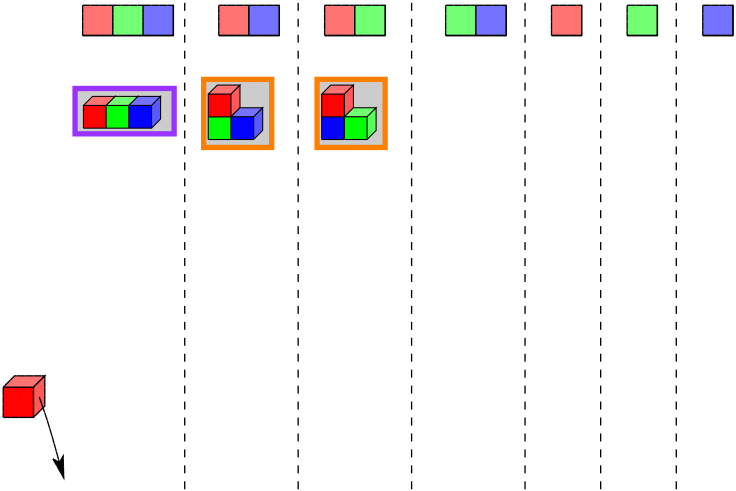
Plan construction by backward search

Example: goal belief state



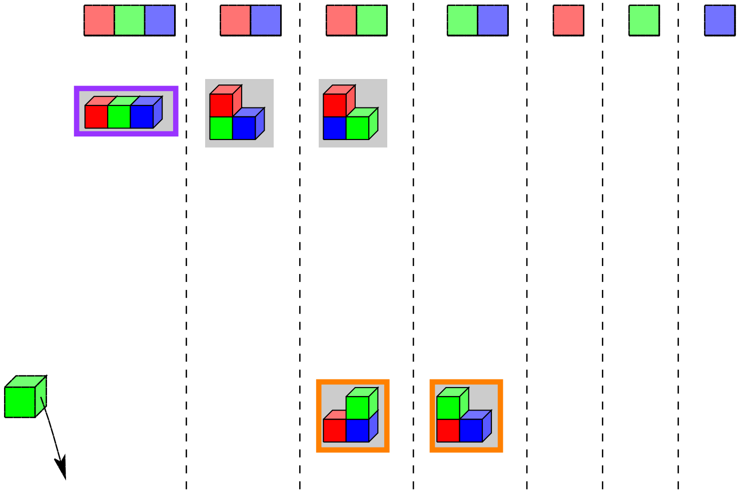
Plan construction by backward search

Example: backward step with red-block-onto-table



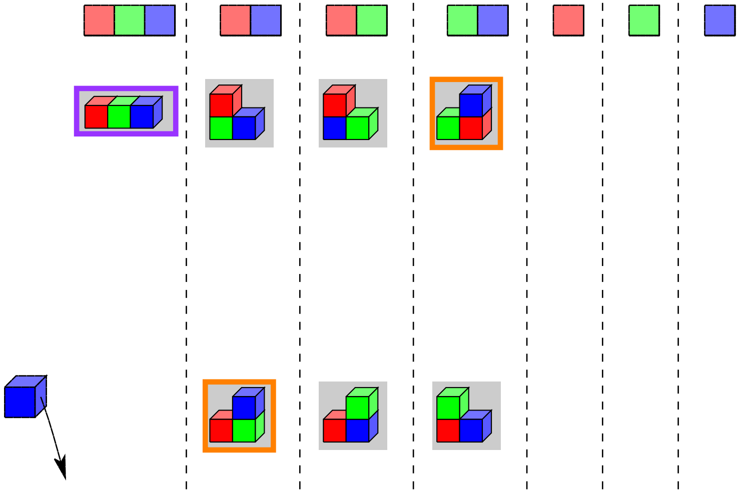
Plan construction by backward search

Example: backward step with green-block-onto-table



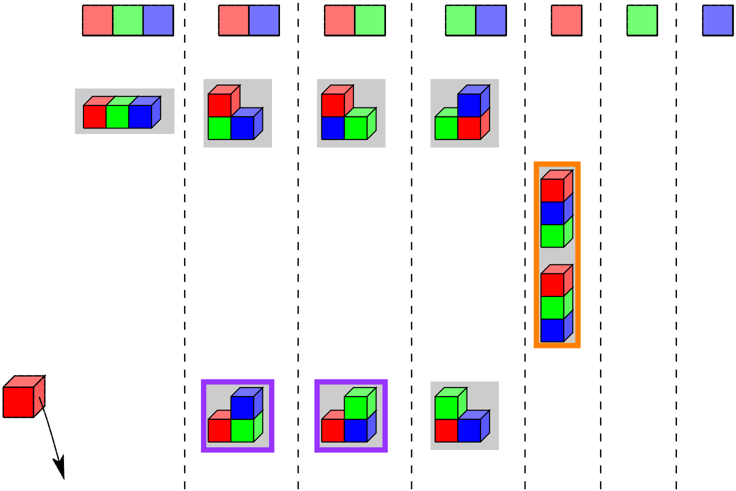
Plan construction by backward search

Example: backward step with blue-block-onto-table



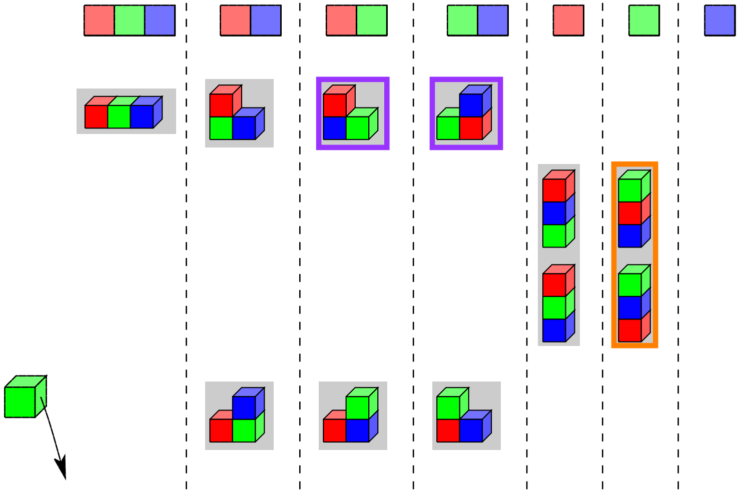
Plan construction by backward search

Example: backward step with red-block-onto-table



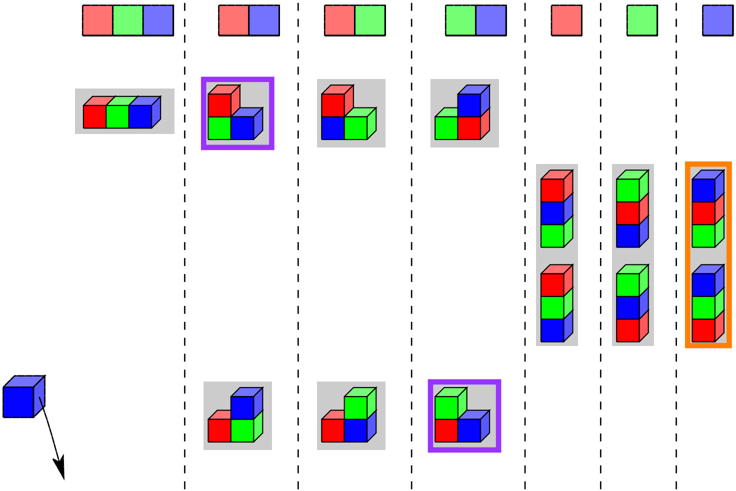
Plan construction by backward search

Example: backward step with green-block-onto-table



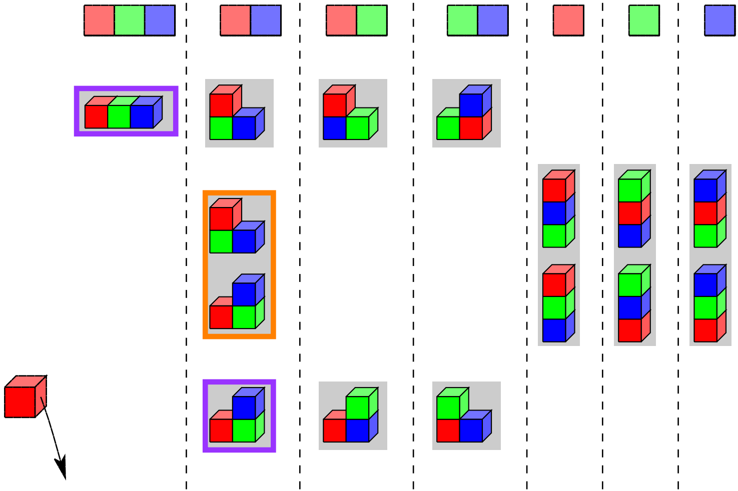
Plan construction by backward search

Example: backward step with blue-block-onto-table



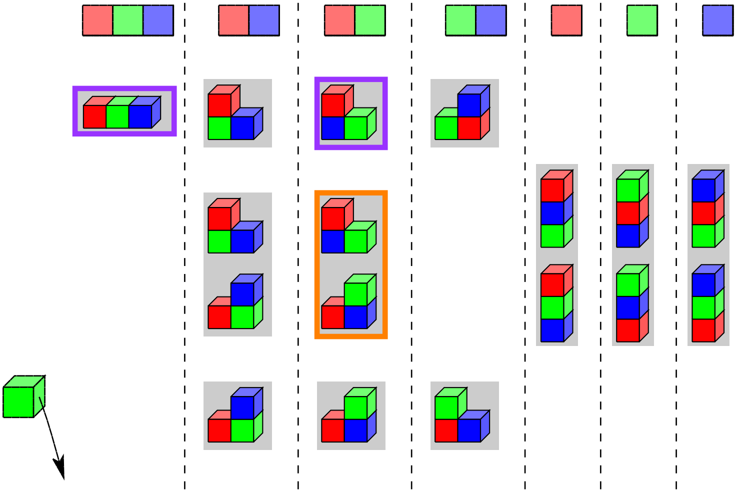
Plan construction by backward search

Example: backward step with red-block-onto-table



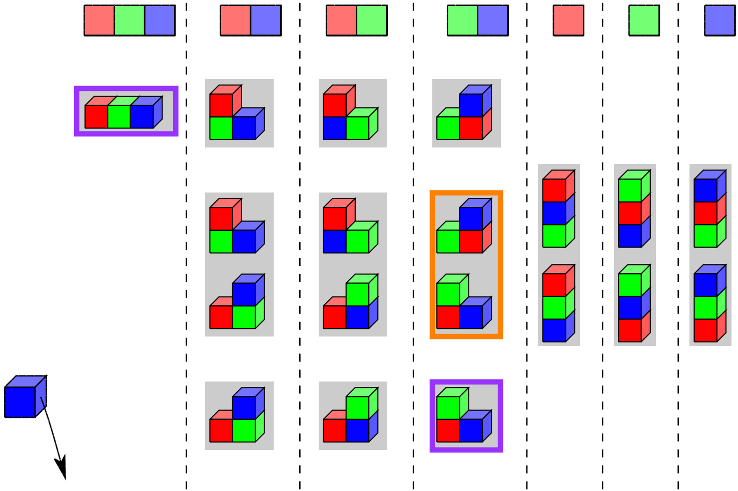
Plan construction by backward search

Example: backward step with green-block-onto-table



Plan construction by backward search

Example: backward step with blue-block-onto-table



Summary

- Planning with partial observability in general requires more general classes of plans than the fully observable and unobservable special cases.
- It appears to be significantly harder.
- Algorithmic ideas are similar to the simpler cases:
  - Reduction to full observability by viewing belief states as states.
  - Forward search in AND/OR trees.
  - Dynamic-programming style backward construction of solvable belief states, starting from goal belief states.