Minimizing Necessary Observations for Nondeterministic Planning

Robert Mattmüller  Manuela Ortlieb  Erik Wacker
Research Group Foundations of Artificial Intelligence
Department of Computer Science
University of Freiburg, Germany

September 25th, 2014
Motivation

Example: Partially Observable BLOCKSWORLD Domain with Nondeterministic PUT-DOWN Operator

observe $\text{CLEAR}(\text{red })$
Motivation

Example: Partially Observable BLOCKSWORLD Domain with Nondeterministic PUT-DOWN Operator

Observing \texttt{CLEAR} predicate sufficient to find solutions.

Initial state known $\Rightarrow$ overhead camera sufficient as a sensor.

**Question:** How to find minimal sets of variables sufficient for solution existence in arbitrary POND planning tasks?

**Remark:** Here, “solution” means \textit{strong cyclic plan}:
- \texttt{closed}: defined for all belief states it may reach, and
- \texttt{proper}: no dead ends
Problem **ObserveInclMin**: 

- **Input**: POND planning task $\Pi = \langle V, B_0, B_*, A, W \rangle$ with
  - state variables $V$
  - initial belief state $B_0$
  - goal description $B_*$
  - nondeterministic actions $A$
  - possibly observable variables $W \subseteq V$

- **Output**: Inclusion-minimal set of variables $O \subseteq W$ such that there exists a strong cyclic plan for $\Pi$ observing only variables from $O$, or `None` if no such set $O$ exists.
Theory

Hardness Result

**Theorem (Rintanen, 2004)**

*The strong cyclic plan existence problem for POND planning, PlanExPOND, is 2-Exptime-complete.*

**Theorem**

*ObserveInclMin is 2-Exptime-complete.*

**Proof.**

- Trivial reduction from PlanExPOND
  \[ \Rightarrow \text{ObserveInclMin is 2-Exptime-hard.} \]
- Naive algorithm iterating over all subsets of \( \mathcal{W} \)
  \[ \Rightarrow \text{ObserveInclMin} \in \text{2-Exptime}. \]
**Question:** Can we improve over the naive algorithm from the proof?

**Assumption:** No obviously irrelevant variables in $\mathcal{W}$. Ignore variables known in $B_0$ and never made unknown by any action.
Baseline Algorithm
Simple Greedy Algorithm

\begin{function} \textsc{simpleGreedySearch}(\Pi):\n  \textbf{if} \ \Pi \text{ is unsolvable} \ \textbf{then} \\
  \ \ \ \ \ \textbf{return} \ \text{None} \\
  \text{Compute some plan } \pi \text{ for } \Pi \\
  \text{Let } \mathcal{O} \text{ be the set of variables actually observed in } \pi \\
  \textbf{while} \ \Pi \text{ still solvable with some } o \text{ removed from } \mathcal{O} \ \textbf{do} \\
  \ \ \ \text{Remove } o \text{ from } \mathcal{O} \\
  \textbf{return} \ \mathcal{O}
\end{function}

\textbf{Theorem}

\textit{Function} \textsc{simpleGreedySearch}

- \textit{runs in } 2-\text{Exptime},
- \textit{correctly solves } \textsc{ObserveInclMin}.
Plan Reuse

Motivation

✓ **simpleGreedySearch** correct

✓ **simpleGreedySearch** asymptotically optimal

✗ **simpleGreedySearch** naive and inefficient

→ Look for ways to speed up **simpleGreedySearch**!

→ Reuse portions of plan not affected by dropping a variable.
Plan Reuse

Example, $\mathcal{W} = \{x, y\}$

Plan observing $x$ and $y$:
Plan Reuse

Example, \( \mathcal{W} = \{x, y\} \)

Reusable plan fragment if \( y \) unobservable:

- \( y \) known or
- never needs to be observed before goal

\[ W = \{x, y\} \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]

\[ \text{observe } x \]
Plan Reuse

Example, $\mathcal{W} = \{x, y\}$

Gap states that need replanning

Gap 1

Gap 2

observe $x$
Plan Reuse

Example, $\mathcal{W} = \{x, y\}$

Plan to fill gap 1
Plan Reuse

Example, $\mathcal{W} = \{x, y\}$
Plan Reuse

Soundness

To eliminate variable \( y \):

- Let \( \pi \) be the old plan still observing \( y \).
- Identify gaps \( 1, \ldots, n \) in \( \pi \).
- Let \( \pi_y \) be the reusable fragment of \( \pi \).
- Let \( \pi_j, j = 1, \ldots, n \), be the new sub-plans filling the gaps.
- Let \( \pi' = \pi_y \oplus \pi_1 \oplus \cdots \oplus \pi_n \) \( (\oplus = \text{function overriding}) \).

**Lemma**

If \( \pi \) and all \( \pi_j \) are strong cyclic plans, then so is \( \pi' \).

**Proof sketch.**

In \( \pi' \), “last” subplan “wins”. Thus, closedness and properness of \( \pi \) and all \( \pi_j \) carry over to \( \pi' \).
Plan Reuse

Soundness

To minimize set of observation variables:
- Eliminate variables one by one, if possible.

Theorem

Plan \( \pi \) resulting from successive elimination of variables is strong cyclic plan.

Proof sketch.

Base case + inductive application of previous lemma.

Remarks:
- In induction, skip gaps filled/circumvented by chance when filling earlier gap.
- In elimination step, existence of \( \pi_j \) not guaranteed. Resulting \( \pi' \) not necessarily with inclusion minimal set \( \mathcal{O} \).
Remark:
Observation sets found with plan reuse can be suboptimal.

Example: Let $\Pi$ with

- Propositional variables $\mathcal{V} = \{a, b, c\}$,
- Initial belief state $B_0 = \overline{a}\overline{b}\overline{c}$,
- Goal belief state $B_* = c$,
- Observable variables $\mathcal{W} = \{b\}$,
- Actions $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, where
  - $a_1 = \langle \overline{a} \rightarrow a \rangle$,
  - $a_2 = \langle \overline{b} \rightarrow (b \text{ or } \top) \rangle$,
  - $a_3 = \langle b \rightarrow c \rangle$,
  - $a_4 = \langle \overline{a}\overline{b}\overline{c} \rightarrow c \rangle$. 
Plan Reuse

Suboptimality

Example (ctd.): Possible plan for Π:

With observation of $b$. 

\[ \overline{abc} \]
\[ \rightarrow \quad a_1 \]
\[ \overline{abc} \]
\[ \rightarrow \quad a_2 \]
\[ \overline{abc} \quad \text{obs } b \]
\[ \overline{abc} \]
\[ \rightarrow \quad \text{obs } b \]
\[ \overline{abc} \quad \text{obs } b \]
\[ \overline{abc} \]
\[ \rightarrow \quad a_3 \]
\[ abc \]
Example (ctd.): Possible plan for $\Pi$:

With observation of $b$.

Only gap state.
Example (ctd.): Possible plan for $\Pi$:

With observation of $b$.

No plan for $\overline{ab}c$ without observing $b$:

- only applicable action $a_2$ makes $b$ unknown
- then all actions inapplicable
Example (ctd.): Possible plan for $\Pi$:

With observation of $b$.

No plan for $a\overline{b}c$ without observing $b$:
- only applicable action $a_2$ makes $b$ unknown
- then all actions inapplicable

Variable $b$ cannot be removed
$\Rightarrow$ solution $\mathcal{O} = \{b\}$
Example (ctd.): Possible plan for $\Pi$:

Plan for $\overline{abc}$ without observation of $b$ exists.
Example (ctd.): Possible plan for $\Pi$:

Plan for $\overline{abc}$ without observation of $b$ exists.

$\Rightarrow$ optimal solution $O^* = \emptyset$
Example (ctd.): Possible plan for \( \Pi \):

Plan for \( \overline{abc} \) without observation of \( b \) exists.

\[ \Rightarrow \] optimal solution \( O^* = \emptyset \)

\[ \Rightarrow \] solution \( O = \{b\} \) found with plan reuse was suboptimal
Further enhancement: functional dependencies

- **Idea:** If value of variable $o$ can be inferred from values of observed variables $o^1, \ldots, o^n$, need not observe $o$.
- Identify such functional dependencies in plan $\pi$.
- Replace observations of $o$ in $\pi$ by observation of $o^1, \ldots, o^n$.

- **Remark:** functional dependencies only have to hold in states reachable following $\pi$, not necessarily in all reachable states.

- **Implemented:** only exactly-one mutexes between propositional variables.
Empirical Evaluation

Runtimes

- Implementation on top of myND planner\(^1\).
- Domains:
  - BLOCKSWORLD
  - FIRSTRESPONDERS
  - TIDYUP
- Legend:
  - Gr = greedy
  - PR = plan reuse
  - FD = functional dependencies

\(^1\)https://bitbucket.org/robertmattmueller/mynd
Overall runtime needed for finding final observation set.

![Graph showing runtimes for different observation sets.](chart.png)
Empirical Evaluation

Observation Set Cardinalities

Cardinalities of the observation sets before/after minimization.
Empirical Evaluation
Domain-specific Observations

Variables in resulting observation sets:

- **BLOCKSWORLD**: mostly
  - **OnTable**
  - **Clear**

  (either of them alone is sufficient.)

- **FIRSTRESPONDERS**: 
  - **Fire** (in all tasks, for relevant locations)
  - In one instance without road to hospital:
    - **VictimStatus** – needs to be observed for applicability of **TreatVictimOnScene**.

- **TIDYUP**: relevant instances of
  - **GripperStatus**
  - **TableClean**
  - **DoorState**
  - **RobotLocation**
  - **CupLocation**
Conclusion and Future Work

Conclusion:

- **Theory:** ObserveInclMin is 2-Exptime-complete.
- Presented asymptotically optimal baseline greedy top-down algorithm for ObserveInclMin.
- Extended it with
  - plan reuse (pays off) and
  - functional dependencies (do not really pay off).

Future work:

- Complement top-down with bottom-up procedure.
- Investigate variable ordering heuristics for the iteration over candidate variables for removal.
- Study problem on domain instead of planning task level.