Non-Deterministic Planning With Conditional Effects

Christian Muise Sheila A. McIlraith Vaishak Belle

Department of Computer Science,
University of Toronto, Toronto, Canada.
{cjmuise,sheila,vaishak}@cs.toronto.edu

June 24, 2014

Computer Science
UNIVERSITY OF TORONTO

Motivation

- PRP enabled us to scale FOND planning significantly.
- Many domains (including non-deterministic ones) can be encoded naturally with conditional effects.
- Regression is at the heart of PRP's success – adapting it for conditional effects poses an interesting challenge.

Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

SAS$^+$ FOND Planning

Planning Task: $\langle \mathcal{V}, s_0, s_*, \mathcal{A} \rangle$

- $\mathcal{V}$: Finite set of variables v, each having the finite domain D_v
- s_0: Initial state of the planning problem.
- s_*: Goal state of the planning problem.
- $\mathcal{A}$: Finite set of non-deterministic actions.

SAS$^+$ FOND Planning

Planning Task: $\langle \mathcal{V}, s_0, s_*, \mathcal{A} \rangle$

- $\mathcal{V}$: Finite set of variables v, each having the finite domain D_v^+
- s_0: Initial (complete) state of the planning problem.
- s_*: Goal (partial) state of the planning problem.
- $\mathcal{A}$: Finite set of non-deterministic actions.

$$v_1 = 2$$
$$v_2 = \perp$$
$$v_3 = 0$$
$$\ldots$$

SAS$^+$ FOND Planning

Planning Task: $\langle \mathcal{V}, s_0, s_*, \mathcal{A} \rangle$

- $\mathcal{V}$: Finite set of variables v, each having the finite domain D_v^+
- s_0: Initial (complete) state of the planning problem.
- s_*: Goal (partial) state of the planning problem.
- $\mathcal{A}$: Finite set of non-deterministic actions.

Action

- Each action $a \in \mathcal{A}$ is of the form $\langle pre_a, Eff_1, \ldots, Eff_n \rangle$
- pre_a is a partial state that must hold prior to execution
- Eff_i is an *effect* which is a set of the form

$$\{\langle cond_1, v_1, d_1 \rangle, \ldots, \langle cond_k, v_k, d_k \rangle\}$$

Regression (*without* conditional effects)

Definition for SAS$^+$: $\mathcal{R}(s, a)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold.

$$a = \langle pre_a, \langle v_1, d_1 \rangle, \ldots, \langle v_n, d_n \rangle \rangle$$

$$p(v) = \begin{cases} pre_a(v) & \text{if } v \text{ is defined in } pre_a \\ \perp & \text{if } pre_a(v) = \perp \text{ and } \exists i, v_i = v \\ s(v) & \text{if } pre_a(v) = \perp \text{ and } \forall i, v_i \neq v \end{cases}$$

Regression (*without* conditional effects)

Definition for SAS$^+$: $\mathcal{R}(s, a)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold.

$$a = \langle pre_a, \langle v_1, d_1 \rangle, \ldots, \langle v_n, d_n \rangle \rangle$$

$$p(v) = \begin{cases} pre_a(v) & \text{if } v \text{ is defined in } pre_a \\ \bot & \text{if } pre_a(v) = \bot \text{ and } \exists i, v_i = v \\ s(v) & \text{if } pre_a(v) = \bot \text{ and } \forall i, v_i \neq v \end{cases}$$

$$\begin{array}{ccccccccc} & a_1 & \cdots & & move(L1, L2) & & \cdots & a_n & \\ s_0 & s_1 & \cdots & s_i \nearrow & & \searrow s_{i+1} & \cdots & & s_n \end{array}$$

Regression (*without* conditional effects)

Definition for SAS$^+$: $\mathcal{R}(s, a)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold.

$$a = \langle pre_a, \langle v_1, d_1 \rangle, \ldots, \langle v_n, d_n \rangle \rangle$$

$$p(v) = \begin{cases} pre_a(v) & \text{if } v \text{ is defined in } pre_a \\ \perp & \text{if } pre_a(v) = \perp \text{ and } \exists i, v_i = v \\ s(v) & \text{if } pre_a(v) = \perp \text{ and } \forall i, v_i \neq v \end{cases}$$

$loc = L1$ $\quad$ $loc = L2$ $\quad$ s_*

$rich = y$ $\nwarrow$ $\quad$ $\swarrow$ $rich = y$

$$a_1 \quad \cdots \quad move(L1, L2) \quad \cdots \quad a_n$$

$$s_0 \quad s_1 \quad \cdots \quad s_i \nearrow \qquad \searrow s_{i+1} \quad \cdots \quad s_n$$

Regression (*without* conditional effects)

Definition for SAS$^+$: $\mathcal{R}(s, a)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold.

$$a = \langle pre_a, \langle v_1, d_1 \rangle, \ldots, \langle v_n, d_n \rangle \rangle$$

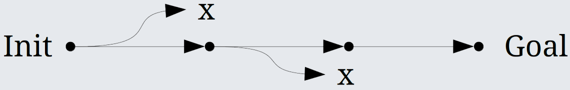
$$p(v) = \begin{cases} pre_a(v) & \text{if } v \text{ is defined in } pre_a \\ \bot & \text{if } pre_a(v) = \bot \text{ and } \exists i, v_i = v \\ s(v) & \text{if } pre_a(v) = \bot \text{ and } \forall i, v_i \neq v \end{cases}$$

Shorthand for Repeated Regression: $\mathcal{R}^*(s, [a_1, \cdots, a_n])$

$$\begin{aligned} \mathcal{R}^*(s, [a]) &= \mathcal{R}(s, a) \\ \mathcal{R}^*(s, [a_1, \cdots, a_{n-1}, a_n]) &= \mathcal{R}^*(\mathcal{R}(s, a_n), [a_1, \cdots, a_{n-1}]) \end{aligned}$$

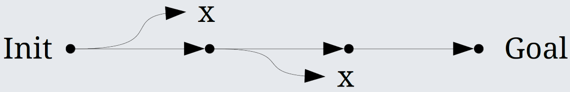
FOND Solutions and Representation

Weak Plan

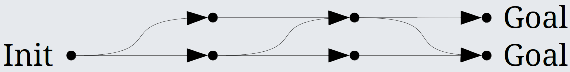


FOND Solutions and Representation

Weak Plan

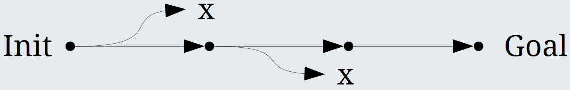


Strong Plan

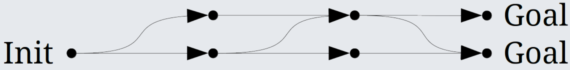


FOND Solutions and Representation

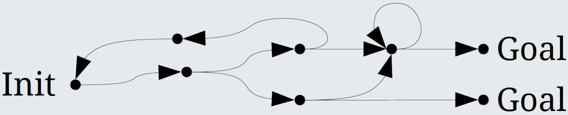
Weak Plan



Strong Plan



Strong Cyclic Plan

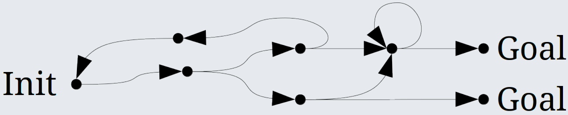


FOND Solutions and Representation

Policy

$P(s)$: Given a set of pairs of the form $\langle p, a, c \rangle$ where p is a partial state, a an action, and c a cost, return the action a in state s from the pair with the lowest cost c such that $s \models p$.

Strong Cyclic Plan



Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

General Algorithm

1. Initialize *Open* and *Closed* lists of states handled by the incumbent policy P (*Open* initially contains only s_0);
2. Select and move a state s from *Open* to *Closed* such that,
    1. If $P(s) = \bot$, run $\textsc{UpdatePolicy}(\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle, P)$;
    2. If $P(s) = a$, and $a \neq \bot$, add to *Open* every state in $\{Prog(s, a, Eff_i) \mid a = \langle pre_a, Eff_1, \ldots, Eff_k \rangle\} \setminus Closed$;
    3. If $\textsc{UpdatePolicy}$ failed in 2.2, process s as a *deadend*;
3. If *Open* is empty, return P. Else, repeat from step 2;

General Algorithm

1. Initialize *Open* and *Closed* lists of states handled by the incumbent policy P (*Open* initially contains only s_0);
2. Select and move a state s from *Open* to *Closed* such that,
    1. If $P(s) = \perp$, run $\textsc{UpdatePolicy}(\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle, P)$;
    2. If $P(s) = a$, and $a \neq \perp$, add to *Open* every state in $\{Prog(s, a, Eff_i) \mid a = \langle pre_a, Eff_1, \ldots, Eff_k \rangle\} \setminus Closed$;
    3. If $\textsc{UpdatePolicy}$ failed in 2.2, process s as a *deadend*;
3. If *Open* is empty, return P. Else, repeat from step 2;

$\textsc{UpdatePolicy}(\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle, P)$

1. $\mathcal{A}' = \textsc{Determinize}(\mathcal{A})$; // Using all-outcomes
2. $[a_1, \cdots a_n] = \textsc{ComputePlan}(\langle \mathcal{V}, s, s_*, \mathcal{A}' \rangle)$;
3. For every suffix $[a_i, \cdots a_n]$ of the plan,
    1. $p_i = \mathcal{R}^*(s_*, [a_i, \cdots a_n])$;
    2. $c_i = cost([a_i, \cdots a_n])$;
    3. Add $\langle p_i, a_i, c_i \rangle$ to P;

$\textsc{UpdatePolicy}(\langle \mathcal{V}, s, s_*, \mathcal{A} \rangle, P)$

1. $\mathcal{A}' = \textsc{Determinize}(\mathcal{A})$; // Using all-outcomes
2. $[a_1, \cdots a_n] = \textsc{ComputePlan}(\langle \mathcal{V}, s, s_*, \mathcal{A}' \rangle)$;
3. For every suffix $[a_i, \cdots a_n]$ of the plan,
    1. $p_i = \mathcal{R}^*(s_*, [a_i, \cdots a_n])$;
    2. $c_i = cost([a_i, \cdots a_n])$;
    3. Add $\langle p_i, a_i, c_i \rangle$ to P;

Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

Formalism for **Pre-Im**age **F**iltering

$\textsc{Primf}(s, a, \mathit{Eff}, s_c)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold assuming that a was executed in the context of state s_c.

Formalism for **Pre-Image Filtering**

$\textsc{Primf}(s, a, \mathit{Eff}, s_c)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold assuming that a was executed in the context of state s_c.

$$\begin{aligned} \mathit{Added} &= \{v \mid \langle \mathit{cond}, v, d\rangle \in \mathit{Eff} \text{ and } s_c \models \mathit{cond}\} \\ \mathit{Support} &= \mathit{pre}_a \cup \bigcup_{i=1\ldots n} \bigcup_{\langle \mathit{cond}, v, d\rangle \in \mathit{Eff}_i} \mathit{cond} \end{aligned}$$

Computer Science
UNIVERSITY OF TORONTO

Formalism for **Pre-Image Filtering**

$\textsc{Primf}(s, a, \mathit{Eff}, s_c)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold assuming that a was executed in the context of state s_c.

$$\begin{aligned}
\mathit{Added} &= \{v \mid \langle \mathit{cond}, v, d\rangle \in \mathit{Eff} \text{ and } s_c \models \mathit{cond}\} \\
\mathit{Support} &= \mathit{pre}_a \cup \bigcup_{i=1\ldots n} \; \bigcup_{\langle \mathit{cond}, v, d\rangle \in \mathit{Eff}_i} \mathit{cond}
\end{aligned}$$

Formalism for **Pre-Image Filtering**

$\textsc{Primf}(s, a, \mathit{Eff}, s_c)$

Intuition: The partial state p characterizing what must hold just prior to executing a in order for the partial state s to hold assuming that a was executed in the context of state s_c.

$$\mathit{Added} = \{v \mid \langle \mathit{cond}, v, d\rangle \in \mathit{Eff} \text{ and } s_c \models \mathit{cond}\}$$

$$\mathit{Support} = \mathit{pre}_a \cup \bigcup_{i=1\ldots n} \bigcup_{\langle \mathit{cond}, v, d\rangle \in \mathit{Eff}_i} \mathit{cond}$$

$$p(v) = \begin{cases} s_c(v) & \text{if } v \in \mathit{Support} \\ \perp & \text{if } v \in \mathit{Added} \setminus \mathit{Support} \\ s(v) & \text{otherwise} \end{cases}$$

Properties

Following the definition of $\textsc{Primf}$...

1. $s_c \models \textsc{Primf}(s, a, \mathit{Eff}, s_c)$

Properties

Following the definition of $\textsc{Primf}$...

1. $s_c \models \textsc{Primf}(s, a, \mathit{Eff}, s_c)$
2. $\textsc{Primf}(s, a, \mathit{Eff}, s_c) \models \mathcal{R}(s, (a, \mathit{Eff}))$

Properties

Following the definition of $\textsc{Primf}$...

1. $s_c \models \textsc{Primf}(s, a, \mathit{Eff}, s_c)$
2. $\textsc{Primf}(s, a, \mathit{Eff}, s_c) \models \mathcal{R}(s, (a, \mathit{Eff}))$
3. When there are no conditional effects:
$$\textsc{Primf}(s, a, \mathit{Eff}, s_c) = \mathcal{R}(s, (a, \mathit{Eff}))$$

Properties

Following the definition of $\textsc{Primf}$...

1. $s_c \models \textsc{Primf}(s, a, \mathit{Eff}, s_c)$
2. $\textsc{Primf}(s, a, \mathit{Eff}, s_c) \models \mathcal{R}(s, (a, \mathit{Eff}))$
3. When there are no conditional effects:
$$\textsc{Primf}(s, a, \mathit{Eff}, s_c) = \mathcal{R}(s, (a, \mathit{Eff}))$$

The Bottom Line

The $\textsc{Primf}$ operator gives us a single partial state that, while less general than regression, allows us to reason efficiently about the relevant portion of the state that held prior to executing an action.

Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

Setup

- Created four non-deterministic domains with conditional effects to test PRP augmented with PRIMF.
- Ran the newly augmented version of PRP to test the scalability in terms of policy size and runtime.
- Resources given: 30min and 1Gb time and memory limit.

Domains (1/2)

Search and Rescue (coverage: 15/15)

- Requires the agent to fly to various locations searching for a human. Non-determinism in whether or not a human is found.
- From the IPC-6 Probabilistic Track (probabilities stripped).
- No strong cyclic solutions, but "high probability" ones exist.

Sloppy Schedule (coverage: 108/150)

- Classical domain with conditional effects to run a workshop.
- Non-determinism introduced as (1) damaging side-effects of some actions, and (2) machines randomly becoming occupied.
- Domain selected for the complex conditional effects present.

Domains (1/2)

Search and Rescue (coverage: 15/15)

- Requires the agent to fly to various locations searching for a human. Non-determinism in whether or not a human is found.
- From the IPC-6 Probabilistic Track (probabilities stripped).
- No strong cyclic solutions, but “high probability” ones exist.

Sloppy Schedule (coverage: 108/150)

- Classical domain with conditional effects to run a workshop.
- Non-determinism introduced as (1) damaging side-effects of some actions, and (2) machines randomly becoming occupied.
- Domain selected for the complex conditional effects present.

Domains (2/2)

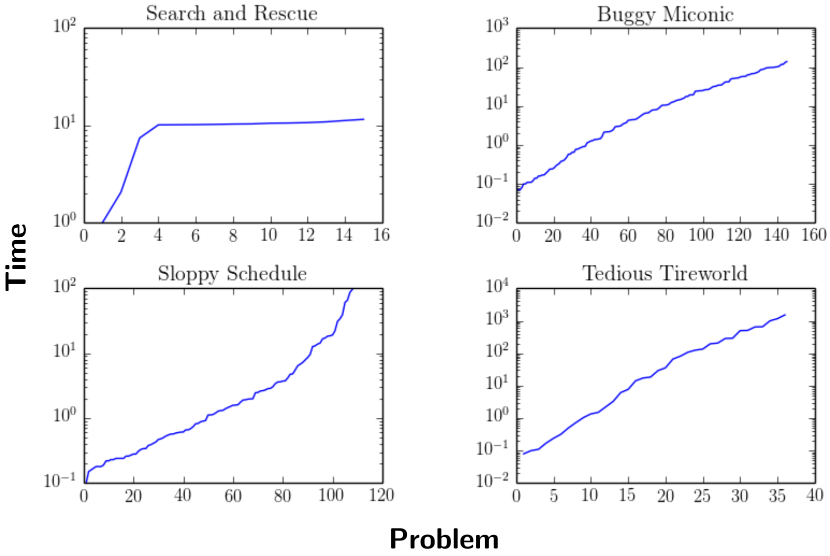
Buggy Miconic (coverage: 145/145)

- Classical domain to serve elevator customers.
- Non-determinism introduced as the elevator stopping at unintended floors on the way to the destination.
- Contains a large number of conditional effects.

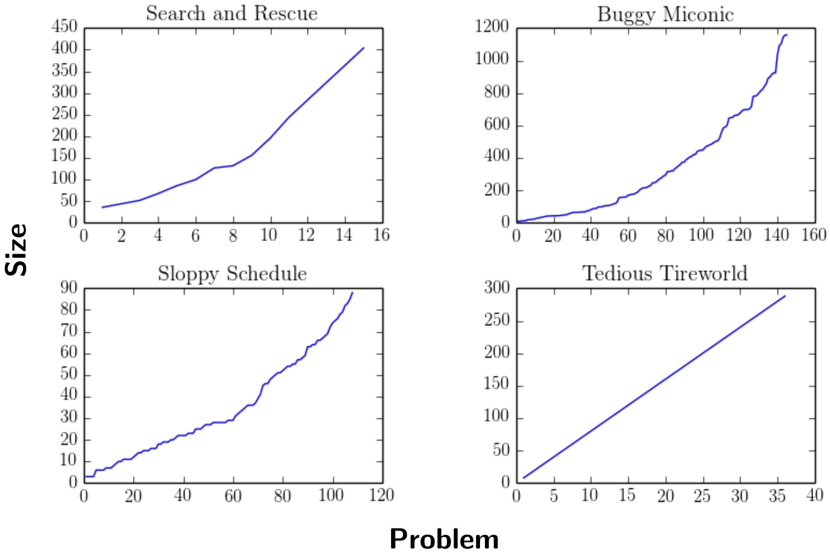
Tedious Tireworld (coverage: 36/40)

- Traditional non-deterministic tireworld with the goal of reaching the goal given the possibility of a tire becoming flat.
- Preconditions were moved into conditional effects.
- Domain selected to show little impact in modelling technique.

Results: **Runtime** (seconds)



Results: **Policy Size** (state-action pairs)



Outline

1. Notation
2. PRP
3. PRIMF
4. Evaluation
5. Conclusion

Summary

- Extended PRP with the PRIMF operator to enable the solving of non-deterministic problems with conditional effects
- Identified the strengths and weaknesses of the new approach
- Introduced a new suite of benchmarks

Future Work

- Apply PRP+PRIMF to Contingent Planning encodings
- Revisit the PRIMF definition to generalize further
- Consider full regression for conditional effects

Future Work

- ~~Apply PRP+PRIMF to Contingent Planning encodings~~
- Revisit the PRIMF definition to generalize further
- Consider full regression for conditional effects

Poster Session: 5:00 - 6:00

Computing Contingent Plans via Fully Observable Non-Deterministic Planning. Muise, C.; Bell, V.; and McIlraith, S. A. In The 28th AAAI Conference on Artificial Intelligence, 2014.

Any Questions?

Benchmarks, code, and slides available at:

`http://www.haz.ca/research/prp/`