Contingent Planning

The slides that follow describe the following publication:


Slides largely prepared by Christian Muise.
Overview

**PRP (2012)**

- Planner for Fully Observable Non-Deterministic (FOND).
- Powerful techniques for deadend detection, state relevance, etc.
- Recently updated to handle conditional effects.

**PO-PRP (2014)**

- Computes conditional plans for *simple* contingent problems.
- Modifies the computational core of PRP to solve encodings.
- Novel approach for constructing compact conditional plans.
### Overview

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Planner for Fully Observable Non-Deterministic (FOND).</td>
<td>Computes conditional plans for <em>simple</em> contingent problems.</td>
</tr>
<tr>
<td>Powerful techniques for deadend detection, state relevance, etc.</td>
<td>Modifies the computational core of PRP to solve encodings.</td>
</tr>
<tr>
<td>Recently updated to handle conditional effects.</td>
<td>Novel approach for constructing compact conditional plans.</td>
</tr>
</tbody>
</table>

### Main Contribution

Leverage modern FOND planners and POND encodings to effectively compile a compact conditional plan offline.
Overview

PRP (2012)
- Planner for Fully Observable Non-Deterministic (FOND).
- Powerful techniques for deadend detection, state relevance, etc.
- Recently updated to handle conditional effects.

PO-PRP (2014)
- Computes conditional plans for simple contingent problems.
- Modifies the computational core of PRP to solve encodings.
- Novel approach for constructing compact conditional plans.

Main Contribution
Leverage modern FOND planners and POND encodings to effectively compile a compact conditional plan offline.
Example Plan: doors-5
Example Plan: doors-5
Outline

1. Background & Notation

2. PO-PRP
   - Encoding
   - Computing
   - Representing

3. Evaluation

4. Conclusion
Outline

1 Background & Notation
2 PO-PRP
3 Evaluation
4 Conclusion
Contingent Planning (1/2)

Planning Task: $\langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{O}, \mathcal{A} \rangle$

- $\mathcal{F}$: Finite set of fluents.
- $\mathcal{I}$: Set of clauses over $\mathcal{F}$ that determines the initial state.
- $\mathcal{G}$: Conjunction of atoms over $\mathcal{F}$ that determines the goal.
- $\mathcal{O}$: Set of observations.
- $\mathcal{A}$: Set of actions.

States

- Belief state $b$ represents a set of states.
- $\mathcal{I}$ corresponds to a belief state $b_{\mathcal{I}}$.
- $b$ satisfies a formula $\phi$ if $\phi$ holds in every state $s \in b$. 
## Actions

- **\( \text{PRE}(a) \):** Conjunction of atoms that must hold to execute \( a \).
- **\( \text{EFF}(a) \):** Set of pairs \( \langle C, L \rangle \) to capture conditional effects.
- **Action progression:** \( b_a = \{ \text{Prog}(s, a) \mid s \in b \} \)

## Observations

- **Observation** \( o \) consists of the condition \( \text{PRE}(o) \) that the agent must know before observing the truth of fluent \( \text{SENSE}(o) \).
- **Belief update with observation:** \( b_a^o = \{ s \mid s \in b_a \text{ and } L \in s \} \), where \( o = \langle C, L \rangle \) and every state in \( b_a \) satisfies \( C \).
Contingent Planning (2/2)

**Actions**

- $\text{PRE}(a)$: Conjunction of atoms that must hold to execute $a$.
- $\text{EFF}(a)$: Set of pairs $\langle C, L \rangle$ to capture conditional effects.
- Action progression: $b_a = \{ \text{Prog}(s, a) \mid s \in b \}$

**Observations**

- Observation $o$ consists of the condition $\text{PRE}(o)$ that the agent must know before observing the truth of fluent $\text{ENSE}(o)$.
- Belief update with observation: $b^o_a = \{ s \mid s \in b_a \text{ and } L \in s \}$, where $o = \langle C, L \rangle$ and every state in $b_a$ satisfies $C$. 
Effective Characterization

The non-unary clauses in $I$ are invariant. E.g.:
- Location of the agent (always in exactly one spot).
- Uncertainty of a fluent (can only ever be true or false).

Monotonicity

Uncertainty of a fluent cannot propagate to other fluents. E.g.:
- Ok: The effect dropping a vase when we know its status:
  \[\langle \{\text{fragile vase}\}, \text{broken vase}\rangle\]
- Not Ok: Unknowingly moving into a trap:
  \[\langle \{\text{at wumpus loc 3}\}, \text{dead} \rangle\]

Corollary: Once known, a fluent remains known.
Effective Characterization

The non-unary clauses in $\mathcal{I}$ are invariant. E.g.,:

- Location of the agent (always in exactly one spot).
- Uncertainty of a fluent (can only ever be true or false).
Effective Characterization

The non-unary clauses in $\mathcal{I}$ are invariant. E.g.,:

- Location of the agent (always in exactly one spot).
- Uncertainty of a fluent (can only ever be true or false).

Monotonicity

Uncertainty of a fluent cannot propagate to other fluents. E.g.,:

- Ok: The effect dropping a vase when we know its status:
  $\langle \{\text{fragile_vase}\}, \text{broken_vase} \rangle$

- Not Ok: Unknowingly moving into a trap:
  $\langle \{\text{at}_\text{wumpus}_\text{loc3}\}, \text{dead} \rangle$

- Corollary: Once known, a fluent remains known.
Planning Task: $\langle F, I, G, A \rangle$

- $F$: Finite set of fluents.
- $I$: Complete setting of the fluents for the initial state.
- $G$: Conjunction of atoms over $F$ that determines the goal.
- $A$: Set of (possibly non-deterministic) actions.
Planning Task: \( \langle \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle \)

- \( \mathcal{F} \): Finite set of fluents.
- \( \mathcal{I} \): Complete setting of the fluents for the initial state.
- \( \mathcal{G} \): Conjunction of atoms over \( \mathcal{F} \) that determines the goal.
- \( \mathcal{A} \): Set of (possibly non-deterministic) actions.

Actions

- \( \text{PRE}(a) \) is the set of fluents that must be true to execute \( a \).
- \( \text{EFF}(a) \) is a finite set of non-deterministic effects.
- A non-deterministic effect consists of a set of conditional effects, each being a pair \( \langle C, L \rangle \) (\( C \) possibly being \( \top \))
FOND Solutions and Representation

Weak Plan

Init → X → Goal
FOND Solutions and Representation

**Weak Plan**

Init → X

**Strong Plan**

Init → Goal

Goal
FOND Solutions and Representation

Weak Plan

```
Init • ———> X
```

Strong Plan

```
Init • ———> • ———> • ———> Goal
```

Strong Cyclic Plan

```
Init • ———> • ———> • ———> • ———> Goal
```

**Policy**

\( P(s) \): Given a set of tuples 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 tuple with the lowest cost \( c \) such that \( s \models p \).
1. Background & Notation
2. PO-PRP
3. Evaluation
4. Conclusion
Approach

1. Convert the contingent problem to a FOND one.
2. Solve with a tailored version of PRP.
3. Compile the final conditional plan.
Outline

2 PO-PRP
   • Encoding
     • Computing
     • Representing
### Planning Task: \( \langle \mathcal{F}', \mathcal{I}', \mathcal{G}', \mathcal{A}' \rangle \)

- \( \mathcal{F}' \): \( \{KL \mid L \in F\} \cup \{K\neg L \mid L \in F\} \)
- \( \mathcal{I}' \): \( \{KL \mid L \in I\} \)
- \( \mathcal{G}' \): \( \{KL \mid L \in G\} \)
- \( \mathcal{A}' \): \( \mathcal{A}_A' \cup \mathcal{A}_O' \cup \mathcal{A}_V' \)
FOND Encoding (based on Bonet & Geffner, 2011)

Planning Task: $\langle F', I', G', A' \rangle$

- $F'$: $\{KL \mid L \in F\} \cup \{K\neg L \mid L \in F\}$
- $I'$: $\{KL \mid L \in I\}$
- $G'$: $\{KL \mid L \in G\}$
- $A'$: $A'_A \cup A'_O \cup A'_V$
  - $A'_A$: $a'$ for every $a \in A$ where, $\text{PRE}(a') = \{KL \mid L \in \text{PRE}(a)\}$ and $\text{EFF}(a') = \{\langle KC, KL \rangle \mid \langle C, L \rangle \in \text{EFF}(a)\}$.

**Note:** Do not need to include $\{\neg K\neg C, \neg K\neg L \mid \langle C, L \rangle \in \text{EFF}(a)\}$.
### Planning Task: \( \langle F', I', G', A' \rangle \)

- **\( F' \):** \( \{ KL \mid L \in F \} \cup \{ K \neg L \mid L \in F \} \)
- **\( I' \):** \( \{ KL \mid L \in I \} \)
- **\( G' \):** \( \{ KL \mid L \in G \} \)
- **\( A' \):** \( A'_A \cup A'_O \cup A'_V \)
  - **\( A'_A \):** \( a' \) for every \( a \in A \) where, \( \text{PRE}(a') = \{ KL \mid L \in \text{PRE}(a) \} \) and \( \text{EFF}(a') = \{ \langle KC, KL \rangle \mid \langle C, L \rangle \in \text{EFF}(a) \} \).
  - **\( A'_O \):** for \( o = \langle C, L \rangle \in O \), there is an action \( a' \in A'_O \) such that \( \text{PRE}(a') = KC \land \neg KL \land \neg K \neg L \) with two possible non-deterministic effects: \( \{ \langle T, KL \rangle \} \) and \( \{ \langle T, K \neg L \rangle \} \).

**Note:** Can only observe a fluent once due to monotonicity.
Planning Task: $\langle F', I', G', A' \rangle$

- $F'$: $\{KL \mid L \in F\} \cup \{K\neg L \mid L \in F\}$
- $I'$: $\{KL \mid L \in I\}$
- $G'$: $\{KL \mid L \in G\}$
- $A'$: $A'_A \cup A'_O \cup A'_V$
  - $A'_A$: $a'$ for every $a \in A$ where, $\text{PRE}(a') = \{KL \mid L \in \text{PRE}(a)\}$ and $\text{EFF}(a') = \{\langle KC, KL \rangle \mid \langle C, L \rangle \in \text{EFF}(a)\}$.
  - $A'_O$: for $o = \langle C, L \rangle \in O$, there is an action $a' \in A'_O$ such that $\text{PRE}(a') = KC \land \neg KL \land \neg K\neg L$ with two possible non-deterministic effects: $\{\langle \top, KL \rangle\}$ and $\{\langle \top, K\neg L \rangle\}$.
  - $A'_V$: for every $(C \supset L) \in I^*$, there is an action $a' \in A'_V$ such that $\text{PRE}(a') = KC$ and $\text{EFF}(a') = \{\langle \top, KL \rangle\}$. 
<table>
<thead>
<tr>
<th>Key Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fairness Assumption</strong></td>
</tr>
<tr>
<td>How do we get away with assuming that the non-deterministic outcomes of a sensing action are “fair”?</td>
</tr>
<tr>
<td><strong>Reachable Incoherent States</strong></td>
</tr>
<tr>
<td>How do we handle belief states that will never occur in practice? (e.g., considering a sensing outcome that is not possible)</td>
</tr>
<tr>
<td><strong>Solution Correspondence</strong></td>
</tr>
<tr>
<td>How do we interpret the policy that PRP generates for the original contingent planning problem?</td>
</tr>
</tbody>
</table>
**Key Considerations**

**Fairness Assumption**
How do we get away with assuming that the non-deterministic outcomes of a sensing action are “fair”? **Answer:** Monotonicity!

**Reachable Incoherent States**
How do we handle belief states that will never occur in practice? (e.g., considering a sensing outcome that is not possible)

**Solution Correspondence**
How do we interpret the policy that PRP generates for the original contingent planning problem?
Outline

2 PO-PRP
   • Encoding
   • Computing
   • Representing
Simple Contingent Problem $\rightarrow$ FOND Problem $\rightarrow$ Contingent Plan
General PO-PRP Algorithm

Simple Contingent Problem $\rightarrow$ FOND Problem $\Rightarrow$ Contingent Plan

1. Initialize Open and Closed lists of states handled by the incumbent policy $P$ (Open initially contains only $I'$);

2. Select and move a state $s$ from Open to Closed such that,
   1. If $P(s) = \bot$, run $\text{UPDATEPOLICY}((\mathcal{F}', s, \mathcal{G}', \mathcal{A'}), P)$;
   2. If $P(s) = a$, and $a \neq \bot$, add to Open every state in $\{\text{Progs}(s, a, \text{Eff}_i) | a = \langle \text{Pre}_a, \text{Eff}_1, \ldots, \text{Eff}_k \rangle \} \setminus \text{Closed}$;
   3. If $\text{UPDATEPOLICY}$ failed in 2.2, process $s$ as a deadend;

3. If Open is empty, return $P$. Else, repeat from step 2;
General PO-PRP Algorithm

Simple Contingent Problem → FOND Problem ⇒ Contingent Plan

1. Initialize Open and Closed lists of states handled by the incumbent policy \( P \) (Open initially contains only \( I' \));

2. Select and move a state \( s \) from Open to Closed such that,
   1. If \( P(s) = \bot \), run \( \text{UPDATE_POLICY}(⟨F', s, G', A'⟩, P) \);
   2. If \( P(s) = a \), and \( a \neq \bot \), add to Open every state in \( \{\text{Prog}(s, a, \text{Eff}_i) \mid a = ⟨\text{PRE}_a, \text{Eff}_1, \ldots, \text{Eff}_k⟩\} \backslash \text{Closed} \);
   3. If \( \text{UPDATE_POLICY} \) failed in 2.2, process \( s \) as a deadend;

3. If Open is empty, return \( P \). Else, repeat from step 2;
\textbf{UpdatePolicy}\((\langle F', s, G', A' \rangle, P)\)

1. \(A'' = \text{Determinize}(A'); // \text{Using all-outcomes}\)

2. \([a_1, \cdots a_n] = \text{ComputePlan}(\langle F', s, G', A'' \rangle);\)

3. For every suffix \([a_i, \cdots a_n]\) of the plan,
   1. \(p_i = \text{Regess}(G', [a_i, \cdots a_n]);\)
   2. \(c_i = \text{cost}([a_i, \cdots a_n]);\)
   3. Add \(\langle p_i, a_i, c_i \rangle\) to \(P;\)
**UpdatePolicy**((⟨F', s, G', A'⟩, P))

1. \( A'' = \text{Determineize}(A'); \) // Using all-outcomes

2. \([a_1, \cdots a_n] = \text{ComputePlan}(⟨F', s, G', A''⟩);\)

3. For every suffix \([a_i, \cdots a_n]\) of the plan,
   1. \( p_i = \text{Regress}(G', [a_i, \cdots a_n]);\)
   2. \( c_i = \text{cost}([a_i, \cdots a_n]);\)
   3. Add \(⟨p_i, a_i, c_i⟩\) to \(P;\)

- Invariants left as actions (and not axioms).
- Priority is given to *useful* invariants.
- Arbitrary ordering of invariants enforced.
Outline

2 PO-PRP
- Encoding
- Computing
- Representing
## Contingent Plan Representations

<table>
<thead>
<tr>
<th>Option 1: State-Action Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses the default output of PRP.</td>
</tr>
<tr>
<td>Requires maintaining the compiled belief.</td>
</tr>
<tr>
<td>Difficult to see the “bigger picture”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option 2: Conditional Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressed in terms of the original problem.</td>
</tr>
<tr>
<td>Standard for (offline) contingent planning.</td>
</tr>
<tr>
<td>Typically takes the form of an exponentially large tree.</td>
</tr>
</tbody>
</table>
Contingent Plan Representations

<table>
<thead>
<tr>
<th>Option 1: State-Action Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Uses the default output of PRP.</td>
</tr>
<tr>
<td>- Requires maintaining the compiled belief.</td>
</tr>
<tr>
<td>- Difficult to see the “bigger picture”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option 2: Conditional Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Expressed in terms of the original problem.</td>
</tr>
<tr>
<td>- Standard for (offline) contingent planning.</td>
</tr>
<tr>
<td>- Typically takes the form of an exponentially large tree.</td>
</tr>
</tbody>
</table>
(PO-)PRP identifies a subset of the tuples in a policy $P$ that are determined to be *marked as strong cyclic*. 
(PO-)PRP identifies a subset of the tuples in a policy $P$ that are determined to be *marked as strong cyclic*.

**Strong Cyclic Property 1**

If a tuple $\langle p, a, c \rangle \in P$ is marked as strong cyclic, then for any state $s$ such that $s \models p$, the policy $P$ is a strong cyclic plan for $s$. 
(PO-)PRP identifies a subset of the tuples in a policy $P$ that are determined to be *marked as strong cyclic*.

**Strong Cyclic Property 1**

If a tuple $\langle p, a, c \rangle \in P$ is marked as strong cyclic, then for any state $s$ such that $s \models p$, the policy $P$ is a strong cyclic plan for $s$.

**Strong Cyclic Property 2**

If a tuple $\langle p, a, c \rangle \in P$ is marked as strong cyclic, then for any state $s$ such that $s \models p$, $s' = \text{Prog}(s, a)$, and $P(s') = \langle p', a', c' \rangle$, either $p'$ is $G'$ or the tuple $\langle p', a', c' \rangle$ is marked as strong cyclic.
Exporting a Conditional Plan

General Intuition

Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.
Exporting a Conditional Plan

**General Intuition**

Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.

1. Let $\text{Open} = \{I'\}$, $\text{Cls} = \emptyset$, and $\langle N, E \rangle = \langle \{I'\}, \emptyset \rangle$;
Exporting a Conditional Plan

**General Intuition**

Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.

1. Let $Open = \{I'\}$, $Cls = \emptyset$, and $\langle N, E \rangle = \langle \{I'\}, \emptyset \rangle$;
2. Select and move a state $s$ from $Open$ to $Cls$ such that,
Exporting a Conditional Plan

General Intuition
Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.

1. Let $Open = \{I'\}$, $Cls = \emptyset$, and $\langle N, E \rangle = \langle \{I'\}, \emptyset \rangle$;
2. Select and move a state $s$ from $Open$ to $Cls$ such that,
   (i) Let $P(s) = \langle p, a, c \rangle$ and if the tuple is marked as being strong cyclic, let $s' = p$, otherwise let $s' = s$;
   (ii) Assuming that $\text{EFF}(a) = \{\text{EFF}_1 \cdots \text{EFF}_n\}$, let $Succ = \{\text{Prog}(s', a, \text{EFF}_1), \cdots, \text{Prog}(s', a, \text{EFF}_n)\}$;
Exporting a Conditional Plan

General Intuition

Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.

1. Let $Open = \{I\}'$, $Cls = \emptyset$, and $\langle N, E \rangle = \langle \{I\}' \rangle, \emptyset \rangle$;
2. Select and move a state $s$ from $Open$ to $Cls$ such that,
   (i) Let $P(s) = \langle p, a, c \rangle$ and if the tuple is marked as being strong cyclic, let $s' = p$, otherwise let $s' = s$;
   (ii) Assuming that $\text{Eff}(a) = \{ \text{Eff}_1 \cdots \text{Eff}_n \}$, let $Succ = \{ \text{Prog}(s', a, \text{Eff}_1), \cdots, \text{Prog}(s', a, \text{Eff}_n) \}$;
   (iii) Add $Succ \setminus Cls$ to $N$ and $Open$;
   (iv) Add $(s, s'')$ to $E$ for every $s''$ in $Succ$;
Exporting a Conditional Plan

**General Intuition**

Enumerate the reachable state space of the policy $P$, and use the tuples in $P$ marked as strong cyclic to replace complete states.

1. Let $Open = \{I'\}$, $Cls = \emptyset$, and $\langle N, E \rangle = \langle \{I'\}, \emptyset \rangle$;

2. Select and move a state $s$ from $Open$ to $Cls$ such that,
   
   (i) Let $P(s) = \langle p, a, c \rangle$ and if the tuple is marked as being strong cyclic, let $s' = p$, otherwise let $s' = s$;
   
   (ii) Assuming that $Eff(a) = \{Eff_1, \ldots, Eff_n\}$, let $Succ = \{Prog(s', a, Eff_1), \ldots, Prog(s', a, Eff_n)\}$;
   
   (iii) Add $Succ \setminus Cls$ to $N$ and $Open$;
   
   (iv) Add $(s, s'')$ to $E$ for every $s''$ in $Succ$;

3. If $Open$ is empty, return $\langle N, E \rangle$. Else repeat from 2.
Outline

1 Background & Notation
2 PO-PRP
3 Evaluation
4 Conclusion
Setup

- Used one fabricated and three existing domains to compare PO-PRP with CLG (Albore, Palacios, and Geffner 2009).
- All domains are simple contingent planning problems.
- Resources given: 30min and 1Gb time and memory limit.
- Uses a modified version of the kreplanner translator.
- Fast Downward’s invariant synthesis limited to 3 seconds.
## Testbed

### Domains
- Doors (doors)
- Coloured Balls (cballs)
- Simple Wumpus (wumpus)
- Canadian Traveller Problem (ctp)

### Metrics
- Size of contingent plan (action + sensing nodes)
- Time to compute offline plan (sec)
- PO-PRP Policy size (# of tuples)
<table>
<thead>
<tr>
<th>Problem</th>
<th>Time (seconds)</th>
<th>Size (actions + sensing)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLG</td>
<td>PO-PRP</td>
</tr>
<tr>
<td></td>
<td>Time (seconds)</td>
<td>Size (actions + sensing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cballs-4-1</td>
<td>0.20</td>
<td>343</td>
</tr>
<tr>
<td>cballs-4-2</td>
<td>19.86</td>
<td>22354</td>
</tr>
<tr>
<td>cballs-4-3</td>
<td>1693.02</td>
<td>1247512</td>
</tr>
<tr>
<td>cballs-10-1</td>
<td>211.66</td>
<td>4829</td>
</tr>
<tr>
<td>cballs-10-2</td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>doors-5</td>
<td>0.12</td>
<td>169</td>
</tr>
<tr>
<td>doors-7</td>
<td>3.50</td>
<td>2492</td>
</tr>
<tr>
<td>doors-9</td>
<td>187.60</td>
<td>50961</td>
</tr>
<tr>
<td>doors-11</td>
<td>T</td>
<td>M</td>
</tr>
</tbody>
</table>
## Results: Time and Size (2/2)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time (seconds)</th>
<th>Size (actions + sensing)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLG</td>
<td>PO-PRP</td>
</tr>
<tr>
<td>wumpus-5</td>
<td>0.44</td>
<td>0.16</td>
</tr>
<tr>
<td>wumpus-7</td>
<td>9.28</td>
<td>1.54</td>
</tr>
<tr>
<td>wumpus-10</td>
<td>1379.62</td>
<td>11.17</td>
</tr>
<tr>
<td>wumpus-15</td>
<td>T</td>
<td>86.16</td>
</tr>
<tr>
<td>wumpus-20</td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>ctp-ch-1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ctp-ch-5</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>ctp-ch-10</td>
<td>2.2</td>
<td>0.02</td>
</tr>
<tr>
<td>ctp-ch-15</td>
<td>133.24</td>
<td>0.07</td>
</tr>
<tr>
<td>ctp-ch-20</td>
<td>T</td>
<td>0.22</td>
</tr>
</tbody>
</table>
Results: PO-PRP Policy -vs- Contingent Plan Size

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size</th>
<th>PO-PRP Policy Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLG</td>
<td>PO-PRP</td>
</tr>
<tr>
<td>cballs-4-1</td>
<td>343</td>
<td>261</td>
</tr>
<tr>
<td>cballs-4-2</td>
<td>22354</td>
<td>13887</td>
</tr>
<tr>
<td>cballs-4-3</td>
<td>1247512</td>
<td>671988</td>
</tr>
<tr>
<td>cballs-10-1</td>
<td>4829</td>
<td>4170</td>
</tr>
<tr>
<td>wumpus-5</td>
<td>854</td>
<td>233</td>
</tr>
<tr>
<td>wumpus-7</td>
<td>7423</td>
<td>770</td>
</tr>
<tr>
<td>wumpus-10</td>
<td>362615</td>
<td>2669</td>
</tr>
</tbody>
</table>
1  Background & Notation

2  PO-PRP

3  Evaluation

4  Conclusion
Summary

- Introduced PO-PRP: an extension to the FOND planner, PRP, capable of solving simple contingent planning problems.
- Identified some of the key considerations involved when using the POND-as-FOND methodology.
- Demonstrated the performance gains to be had over the previous state of the art in offline contingent planning.
Future Work

- Extend PO-PRP to handle width-1 problems.
- Handle a richer set of invariants for new compilations.
- Improve the strong cyclic detection for more compact policies.
Future Work

- Extend PO-PRP to handle width-1 problems.
- Handle a richer set of invariants for new compilations.
- Improve the strong cyclic detection for more compact policies.

Session PM 2b: Planning Under Uncertainty (Tues. 2:50pm)

- **Non-Deterministic Planning With Conditional Effects** Muise, C.; McIlraith, S. A.; and Bell, V. In the 24th International Conference on Automated Planning and Scheduling, 2014.
Benchmarks, code, and slides available (in July) at:
http://www.haz.ca/research/poprp/
KEEP CALM AND MAKE A PLAN