Fast Strong Planning for FOND Problems with Multi-Root Directed Acyclic Graphs

Jicheng Fu\textsuperscript{1}, Andres Calderon Jaramillo\textsuperscript{1}, Vincent Ng\textsuperscript{2}, Farokh Bastani\textsuperscript{2}, and I-Ling Yen\textsuperscript{2}

\textsuperscript{1}The University of Central Oklahoma
\textsuperscript{2}The University of Texas at Dallas
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

- A planning problem is a triple $\langle s_0, g, \Sigma \rangle$, where
  - $s_0$ is the initial state,
  - $g$ is the goal condition, and
  - $\Sigma$ is the planning domain
Goal

- To solve strong planning problems from a Fully-Observable Non-deterministic planning domain
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
- Informally, in a nondeterministic planning domain,
  - an action may generate multiple effects

\[
\text{act} \rightarrow \text{effects}
\]
Goal

- To solve strong planning problems from a Fully-Observable **Nondeterministic planning domain**
- Informally, in a nondeterministic planning domain,
  - an action may generate multiple effects

Formally, a nondeterministic domain
- is a 4-tuple \( \Sigma = (P, S, A, \gamma) \)
  - \( P \) is a finite set of propositions;
  - \( S \subseteq 2^P \) is a finite set of states in the system;
  - \( A \) is a finite set of actions; and
  - \( \gamma: S \times A \to 2^S \) is the state-transition function
Goal

- To solve strong planning problems from a *Fully-Observable Nondeterministic* planning domain
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

- Full observability
  - The states of the world are fully observable
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

Strong planning

- refers to a particular type of solutions to nondeterministic problems
- different from so-called weak planning and strong cyclic planning
Weak Planning Solutions

- Solutions where there is a chance to achieve the goal

In fact, non-goal leaf states are not part of the weak plan!

In the weak plan, there is no path from a non-goal leaf state to the goal

Non-deterministic actions
Strong Cyclic Planning Solutions

- prescribe actions for all possible non-goal leaf states
  - find a path for each non-goal leaf state to the goal state
  - May loop indefinitely
  - But contain no dead-ends
  - More difficult than finding weak planning solutions

Then a strong cyclic plan is found!
Strong Planning Solutions

- prescribe actions for all possible non-goal leaf states
  - find a path for each non-goal leaf state to the goal state
  - Contain no cycles
  - Contain no dead-ends

Then a strong plan is found!
Representing a Plan

- Regardless of whether a plan is weak, strong cyclic, or strong, we can represent it as a **policy** $\pi$
  - a partial function mapping states to actions

- More formally, policy $\pi : S_\pi \rightarrow A$
  - consists of state action pairs $(s, a)$ such that $\pi(s) = a$
  - defines which action to take under state $s$
How to Generate a Strong Plan

△ Choice 1:

- Upgrade a state-of-the-art strong cyclic planner
  - Such as our FIP [Fu et al., 2011] or PRP [Muise et al., 2012]
  - 3 orders of magnitude faster than other state-of-the-art planners, such as Gamer and MBP
How to Generate a Strong Plan

- State-of-the-art strong cyclic planner tries to
  - find a path for each non-goal leaf state to the goal state
    - Using a classical planner

**Issue:**
- Lack of control over planning efficiency
  - If the classical planner runs longer than expected
  - Hard to tell whether
    - It needs more time; or
    - It is stuck in some hopeless situation
Desirable Characteristics

- Has full control over planning
- Has heuristics to ensure planning towards the relevant search direction
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack: make action $a$ inapplicable to $s$
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack: make action $a$ inapplicable to $s$
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack: make action $a$ inapplicable to $s$
  - If state $s$ only has one applicable action
    - It becomes a dead-end now
    - Backtrack continues to $s'$
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack: make action $a$ inapplicable to $s$
  - If state $s$ only has one applicable action
    - It is a dead-end now
    - Backtrack continues to $s'$

![Diagram showing states and actions](image)
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack: make action $a$ inapplicable to $s$
  - If state $s$ only has one applicable action
    - It is a dead-end now
    - Backtrack continues to $s'$
    - If $s'$ only has one applicable action
    - Backtrack continues
An Observation

- Applying action $a$ to state $s$ leads to a cycle
  - Backtrack continues until
    - It reaches a state $s''$ that has more than one applicable action

To handle cycles efficiently, we should distinguish states with one applicable action from those with more than ones!
States with One Applicable Action

- Very common
  - 25% of the states have only one applicable action
    - Based on benchmark problems in the International Planning Competition 2008 (IPC 2008)
  - More states will become those with only one applicable action as planning goes on
    - Actions are made inapplicable if they lead to cycles or dead-ends
A MRDAG $M = \{S_{Mr}, \pi_M\}$ consists of two elements, namely, a rootset $S_{Mr}$ and a policy $\pi_M$.

- $S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\} \subseteq S_{\pi_M}$ consists of a set of states
- States not in $S_{Mr}$ have only one applicable action
A state $s$ is called an outsider of a MRDAG $M = \{S_{Mr}, \pi_M\}$ if one of the following two conditions is satisfied:

- $s$ is a goal; or
- there exists $(s', a') \in \pi_M$ such that $s \in \gamma(s', a')$; in addition, $|A(s)| > 1$ and $s$ does not belong to any of $M$’s ancestry MRDAGs (i.e., MRDAGs constructed prior to $M$)
A MRDAG $M_c$ rooted at $S_{Mcr}$ is a child of MRDAG $M_p$ if $S_{Mcr}$ is the set of all non-goal outsiders of $M_p$. $M_p$ is called the parent of $M_c$. 

**Child MRDAG**

- **Parent MRDAG**

- **Initial State**
A Feasible MRDAG

- A MRDAG $M = \{S_M, \pi_M\}$ is feasible if the following three conditions are satisfied:
  - $\forall (s, a) \in \pi_M$, applying $a$ to $s$ does not lead to a cycle in $G_{\pi}(s_0)$;
  - $\forall (s, a) \in \pi_M$, applying $a$ to $s$ does not lead to a dead-end;
  - the child of $M$, if any, is also feasible
A Strong Solution

- **A strong solution** is \( \pi = \pi_{M1} \cup \pi_{M2} \cup \ldots \cup \pi_{Mn} \), where \( \pi_{M1}, \pi_{M2}, \ldots, \pi_{Mn} \) are the policies of a sequence of MRDAGs \( M_1, M_2, \ldots, M_n \), if the following three conditions are satisfied:
  - \( M_1 \) is rooted at \( s_0 \), i.e., the initial state;
  - \( M_i \) is the parent of \( M_{i+1} \) for \( i = 1, 2, 3, \ldots, n - 1 \); and
  - all the outsiders of \( M_n \) are goal states
Example: Simplified Blocksworld Domain

- Deterministic action *put-down*(B)
  - puts block B onto the table
- Two nondeterministic actions
  - *pick-up*(A, B)
  - *put-on*(A, B)
  - Both actions may drop the held block A onto the table.

Initial state

```
C  B  A
```

Goal state

```
C  B  A
```
Blocksworld Example – The First Weak Plan

Initial state $s_0$

$\text{PICK-UP (B A)}$

$S_1$

Goal

$\text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\}\rangle$
Blocksworld Example – The First Weak Plan

Initial state

\[
\text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP}(B\ A))\} \rangle
\]

\[
\text{MRDAG}_2 = \langle \{s_1\}, \{(s_1, \text{PUT-ON}(B\ C))\}, \{s_2, \text{PICK-UP}(B\ C)\} \rangle
\]
Blocksworld Example – The First Weak Plan

Initial state

MRDAG_1 = ⟨{s_0}, {(s_0, PICK-UP(B A))}⟩

MRDAG_2 = ⟨{s_1}, {(s_1, PUT-DOWN(B))}⟩
Outline of the Strong Planning Algorithm

Global Variables: \( \pi, \langle s_0, g, \Sigma \rangle \)

Function STRONG_PLANNING

\( R \leftarrow \{s_0\}; \pi \leftarrow \phi \quad /\!*R\ is\ the\ rootset\ of\ the\ MRDAG*/! \)

while \( R \neq \phi \) do

\[ \pi_M \leftarrow \text{GET-NEXT-SET-OF-ACTIONS}(R) \]

if \( \pi_M = \phi \) then

if \( R = \{s_0\} \) then return FAILURE else

BACKTRACK(R)

endif

else

if BUILD-MRDAG(\( \pi_M \)) \( \neq \) FAILURE then

\( \pi \leftarrow \pi \cup \pi_M \)

if All-GOAL-OUTSIDERS(\( R, \pi_M \)) then

return \( \pi \)

else

\( R \leftarrow \text{GET-OUTSIDERS}(R, \pi_M) \)

endif

endif

endif

endwhile
Blocksworld Example – The First Weak Plan

Initial state

\( s_0 \)  \( s_1 \)

\( \text{PICK-UP (B A)} \)

\( \text{PUT-ON (B C)} \)

Goal

\[ \text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\} \rangle \]

\[ \text{MRDAG}_2 = \langle \{s_1\}, \{(s_1, \text{PUT-ON(B C)})\} \rangle \]
Building a Feasible MRDAG

Function EXPAND-MRDAG (\(\pi_M, s, a\))

foreach \(s' \in \gamma(s, a) \& NOT-\text{GOAL}(s')\) do

if \(s' \in S_\pi \) or \(s' \in S_{\pi_M}\) then

if \(\text{DETECT-CYCLE}(\pi \cup \pi_M) = \text{TRUE}\) then

return FAILURE

endif

elseif \(|A(s')| = 1\) then

\(\pi_M \leftarrow \pi_M \cup \{(s', a')\}\) with \(a' \in A(s')\)

if \(\text{EXPAND-MRDAG}(\pi_M, s', a') = \text{FAILURE}\) then

return FAILURE

endif

elseif \(|A(s')| = 0\) then /*dead-end*/

return FAILURE

endif

endfor

return SUCCESS
Blocksworld Example – The First Weak Plan

$MRDAG_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\} \rangle$

$MRDAG_2 = \langle \{s_1\}, \{\{s_1, \text{PUT-ON(B C)}\} \rangle \langle s_2, \text{PICK-UP(B C)}\rangle$
Two Heuristics

- Try to answer
  - How to impose an ordering on the states to be expanded in the same rootset?
  - How to impose an ordering on the actions to be chosen for a state in the rootset?
Most Constrained State (MCS) Heuristic

- Assume that the rootset of a MRDAG is $S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\}$.
- Sort the states in $S_{Mr}$ in increasing order of the number of actions applicable to a state.

<table>
<thead>
<tr>
<th>$s_{r1}$</th>
<th>$s_{r2}$</th>
<th>$\ldots$</th>
<th>$s_{rk}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{11}$</td>
<td>$a_{21}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td>$a_{12}$</td>
<td>$a_{21}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td>$a_{13}$</td>
<td>$a_{21}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_{1&lt;m1&gt;}$</td>
<td>$a_{21}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td>$a_{11}$</td>
<td>$a_{22}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td>$a_{12}$</td>
<td>$a_{22}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td>$a_{13}$</td>
<td>$a_{22}$</td>
<td>$\ldots$</td>
<td>$a_{k1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_{1&lt;m1&gt;}$</td>
<td>$a_{2&lt;m2&gt;}$</td>
<td>$\ldots$</td>
<td>$a_{k&lt;mk&gt;}$</td>
</tr>
</tbody>
</table>
Least Heuristic Distance (LHD)

- For each state \( s_{ri} \in S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\} \) (1\( \leq \) \(i\) \(\leq\) \(k\)), we sort its applicable actions in increasing order of the heuristic distance to the goal.
Evaluation

- All problem instances were derived from the benchmark domains of the IPC2008 FOND track
  - Blocksworld, Tireworld, Faults, and First-responders

- Goal
  - For comparison, we implemented four versions
    - SP uses both heuristics,
    - MCS uses only the MCS heuristic,
    - LHD uses only the LHD heuristic, and
    - NOH uses none of the heuristics.
  - Two state-of-the-art strong planners: Gamer and MBP
  - give each planner 1200 seconds to solve each problem instance
Evaluation 1: Problem Coverage

<table>
<thead>
<tr>
<th>Domain</th>
<th>Gamer</th>
<th>MBP</th>
<th>SP</th>
<th>LHD</th>
<th>MCS</th>
<th>NOH</th>
</tr>
</thead>
<tbody>
<tr>
<td>scbw (30)</td>
<td>10</td>
<td>10</td>
<td>29</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>bw(30)</td>
<td>10</td>
<td>0</td>
<td>30</td>
<td>30</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>ft (10)</td>
<td>6</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>tw (12)</td>
<td>11</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>fr (50)</td>
<td>20</td>
<td>10</td>
<td>49</td>
<td>49</td>
<td>46</td>
<td>45</td>
</tr>
<tr>
<td>Total (132)</td>
<td>57</td>
<td>24</td>
<td>130</td>
<td>131</td>
<td>94</td>
<td>92</td>
</tr>
</tbody>
</table>

Our planners solve more problems than Gamer and MBP within the time limit.
Evaluation 2: Efficiency

Comparing with Gamer and MBP
- SP and LHD are about 4 orders of magnitude faster on strong blocksworld, first-responders, and tiresworld,
- about 3 orders of magnitude faster than Gamer on faults, and
- 2 orders of magnitude faster on strong cyclic blocksworld.

In terms of the contributions made by the two heuristics
- LHD is on average 5 times faster on first-responders, and up to 2 orders of magnitude faster on tireworld and 3 orders of magnitude faster on faults than MCS.
- MCS is about 3 times faster than LHD on strong and strong cyclic blocksworld domains.
- In terms of plan size, LHD consistently generates much compact plans than MCS.

| tw-10   | 234.021 | 1 | --- | 0.001 | 1 | --- | --- | 0.770 | 868 |
| tw-11   | 241.141 | 5 | --- | 0.001 | 5 | --- | --- | 0.016 | 448 |
| tw-12   | 242.036 | 1 | --- | 0.001 | 1 | --- | --- | 0.005 | 47  |
| tw-14   | 95.095  | 21 | --- | 0.009 | 34 | --- | --- | 0.009 | 32  |
| fr-1-8  | 10.046  | 10 | 55.377 | 0.002 | 10 | 0.003 | 10  | 0.010 | 328 |
| fr-1-9  | 52.265  | 11 | 296.332 | 0.003 | 11 | --- | --- | 0.016 | 448 |
| fr-1-10 | 721.715 | 12 | --- | 0.004 | 12 | 0.004 | 12  | 0.044 | 1037|
| fr-10-1 | 0.754   | 3 | --- | 0.012 | 3 | --- | --- | 0.070 | 289 |
| fr-10-2 | ---     | --- | --- | --- | --- | --- | --- | --- | --- |
Summary

- Proposed a novel data structure, MRDAG (multi-root directed acyclic graph)
- Conducted extensive experiments to evaluate how planning performance is affected by
  - the order in which the actions applicable to a state are chosen and
  - the order in which the states in the rootset of a MRDAG are expanded via the proposal of two heuristics, MCS and LHD.
Summary

- Experimental results showed that
  - the use of MRDAG indeed made cycle handling easier and more efficient, and
  - the use of the LHD heuristic significantly improved planning performance.
  - our planner significantly outperformed two state-of-the-art planners, Gamer and MBP, by solving more problems in less time.


