RePOP: Reviving Partial Order Planning

XuanLong Nguyen & Subbarao Kambhampati
{xuanlong,rao}@asu.edu

Yochan Group:
http://rakaposhi.eas.asu.edu/yochan
In the beginning it was all POP.

Then it was cruelly UnPOPped.

The good times return with Re(vived)POP.

In the beginning it was all POP.
A recent (turbulent) history of planning

1970s-1995

- UCPOP [Penberthy & Weld]
- IxTeT [Ghallab et al]

The whole world believed in POP and was happy to stack 6 blocks!

1995

- Advent of CSP style compilation approach:
  - Graphplan [Blum & Furst]
  - SATPLAN [Kautz & Selman]

1997

- Domination of heuristic state search approach:
  - HSP/R [Bonet & Geffner]
  - UNPOP [McDermott]: POP is dead!

- Importance of good Domain-independent heuristics

UCPOP

2000 -

- Hoffman’s FF – a state search planer swept through AIPS-00 competition!

- NASA’s highly publicized RAX still a POP dinosaur

- POP believed to be good framework to handle temporal and resource planning [Smith et al, 2000]

UNPOP

RePOP
Outline

RePOP: A revival for partial order planning

• To show that POP can be made very efficient by exploiting the same ideas that scaled up state search and Graphplan planners
  – Effective heuristic search control
  – Use of reachability analysis
  – Handling of disjunctive constraints

• RePOP, implemented on top of UCPOP
  – Dramatically better than all known partial-order planners
  – Outperforms Graphplan and competitive with state search planners in many (parallel) domains
**POP background**

Partial plan representation

\[ P = (A, O, L, OC, UL) \]

- **A:** set of action steps in the plan
  \[ S_0, S_1, S_2, \ldots, S_{\infty} \]
- **O:** set of action ordering \( S_i < S_j, \ldots \)
- **L:** set of causal links \( S_i \xrightarrow{p} S_j \)
- **OC:** set of open conditions
  (subgoals remain to be satisfied)
- **UL:** set of unsafe links \( S_i \xrightarrow{p} S_j \)
  where \( p \) is deleted by some action \( S_k \)

**Flaw:** Open condition OR unsafe link

**Solution plan:** A partial plan with no remaining flaw

- Every open condition must be satisfied by some action
- No unsafe links should exist (i.e. the plan is consistent)
1. Let $P$ be an initial plan
2. **Flaw Selection**: Choose a flaw $f$ (either open condition or unsafe link)
3. **Flaw resolution**:
   - If $f$ is an open condition, choose an action $S$ that achieves $f$
   - If $f$ is an unsafe link, choose promotion or demotion
   - Update $P$
   - Return NULL if no resolution exist
4. If there is no flaw left, return $P$ else go to 2.

**Choice points**
- Flaw selection (*open condition? unsafe link?*)
- Flaw resolution (*how to select (rank) partial plan?*)
  - Action selection (backtrack point)
  - Unsafe link selection (backtrack point)
Our approach (main ideas)

1. **Ranking partial plans:**
   - use an effective distance-based heuristic estimator

2. **Exploit reachability analysis:**
   - use invariants to discover implicit conflicts in the plan.

3. **Unsafe links are resolved by posting disjunctive ordering constraints into the partial plan:**
   - avoid unnecessary and exponential multiplication of failures due to promotion/demotion splitting

State-space idea of distance heuristic

CSP ideas of consistency enforcement
1. Ranking partial plans using distance-based heuristic

**1. Ranking Function** \( f(P) = g(P) + w h(P) \)

- \( g(P) \): number of actions in \( P \)
- \( h(P) \): estimate of number of new actions needed to refine \( P \) to become a solution plan
- \( w \): increase the greediness of the heuristic search

**2. Estimating \( h(P) \)**

\( h(P) \) is estimated by relaxing some constraints present in the partial plan \( P \)

Negative effects of actions are relaxed

- \( P \) has no unsafe link flaws
- \( h(P) \) becomes the number of actions (\( \text{cost}(S) \)) needed to achieve the set of open condition \( S \) from the initial state
Distance-based heuristic estimate

Estimate cost(S)
1. Build a planning graph PG from the initial state.
2. Cost(S) := 0 if all subgoals in S are in level 0.
3. Let p be a subgoal in S that appears last in PG.
4. Pick an action a in the graph that first achieves p
5. Update
   cost(S) := 1 + cost(S+Prec(a) – Eff(a))
6. Replace S = S+Prec(a) – Eff(a), goto 2

+ Any state-space heuristic can be adapted
+ Relaxing negative effects makes the estimate inaccurate in serial domains.
2. Handling unsafe link flaws

1. For each unsafe link \( S_i \xrightarrow{p} S_j \) threatened by another step \( S_k \):
   Add disjunctive constraint to \( O \)
   \[ S_k < S_i \lor S_i < S_j \]

2. Whenever a new ordering constraint is introduced to \( O \), perform the constraint propagations:

   \[
   S_1 < S_2 \lor S_3 < S_4 \land S_4 < S_3 \Rightarrow S_1 < S_2 \\
   S_1 < S_2 \land S_2 < S_3 \Rightarrow S_1 < S_3 \\
   S_1 < S_2 \land S_2 < S_1 \Rightarrow \text{False}
   \]

   - Avoid the unnecessary exponential multiplication of failing partial plans.
3. Detecting indirect conflicts using reachability analysis

1. Reachability analysis to detect invariant:
   - \textit{on}(a,b) and clear(b)
   - \textit{How to get state information in a partial plan}

2. Cutset: Set of literals that must be true at some point during execution of plan
   For each action a, $S_i \xrightarrow{p} S_j$

   \[ \text{pre-C}(S_k) = \text{Prec}(S_k) \cup \{p \mid S_i \xrightarrow{p} S_j, \text{is a link and } S_i < S_k < S_j \} \]

   \[ \text{post-C}(S_k) = \text{Eff}(S_k) \cup \{p \mid S_i < S_k < S_j, \text{is a link and } S_i < S_k < S_j \} \]

4. If exists a cutset that violates of a variant, the partial plan is invalid and should be pruned

\[ \text{Disadvantage:} \]
   - Inconsistency checking is passive and maybe expensive
Detecting indirect conflicts using reachability analysis

1. Generalizing unsafe link: $S_k$ threatens $S_i \xrightarrow{p} S_j$ iff $p$ is mutually exclusive (mutex) with either $\text{Prec}(S_k)$ or $\text{Eff}(S_k)$

2. Unsafe link is resolved by posting disjunctive constraints (as before)
   $S_k < S_i \vee S_i < S_j$

• Detects indirect conflicts early
• Derives more disjunctive constraints to be propagated
Experiments on RePOP

• RePOP is implemented on top of UCPOP planner using the three presented ideas
  – Written in Lisp, runs on Linux, 500MHz, 250MB
• Compare RePOP against UCPOP, Graphplan and AltAlt in a number of benchmark domains
  – Time
  – Solution quality
Comparing planning time  

1. RePOP is very good in parallel domains (gripper, logistics, rocket, parallel blocks world)
   • Outperforms Graphplan in many domains
   • Competitive with AltAlt
   • Completely dominates UCPOP
2. RePOP still inefficient in serial domains: Travel, Grid, 8-puzzle
## Comparing planning time (time in seconds)

<table>
<thead>
<tr>
<th>Problem</th>
<th>UCPOP</th>
<th>RePOP</th>
<th>Graphplan</th>
<th>AltAlt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gripper-8</td>
<td>-</td>
<td>1.01</td>
<td>66.82</td>
<td>.43</td>
</tr>
<tr>
<td>Gripper-10</td>
<td>-</td>
<td>2.72</td>
<td>47min</td>
<td>1.15</td>
</tr>
<tr>
<td><strong>Gripper-20</strong></td>
<td>-</td>
<td><strong>81.86</strong></td>
<td>-</td>
<td><strong>15.42</strong></td>
</tr>
<tr>
<td>Rocket-a</td>
<td>-</td>
<td>8.36</td>
<td>75.12</td>
<td>1.02</td>
</tr>
<tr>
<td><strong>Rocket-b</strong></td>
<td>-</td>
<td><strong>8.17</strong></td>
<td><strong>77.48</strong></td>
<td><strong>1.29</strong></td>
</tr>
<tr>
<td>Logistics-a</td>
<td>-</td>
<td>3.16</td>
<td>306.12</td>
<td>1.59</td>
</tr>
<tr>
<td>Logistics-b</td>
<td>-</td>
<td>2.31</td>
<td>262.64</td>
<td>1.18</td>
</tr>
<tr>
<td>Logistics-c</td>
<td>-</td>
<td>22.54</td>
<td>-</td>
<td>4.52</td>
</tr>
<tr>
<td><strong>Logistics-d</strong></td>
<td>-</td>
<td><strong>91.53</strong></td>
<td>-</td>
<td><strong>20.62</strong></td>
</tr>
<tr>
<td>Bw-large-a</td>
<td>45.78</td>
<td>(5.23)</td>
<td>-</td>
<td>14.67</td>
</tr>
<tr>
<td>Bw-large-b</td>
<td>-</td>
<td>(18.86)</td>
<td>-</td>
<td>122.56</td>
</tr>
<tr>
<td><strong>Bw-large-c</strong></td>
<td>-</td>
<td>(137.84)</td>
<td>-</td>
<td><strong>116.34</strong></td>
</tr>
</tbody>
</table>
Some solution quality metrics

1. **Number of actions**

2. **Makespan**: minimum completion time (number of time steps)

3. **Flexibility**: Average number of actions that do not have ordering constraints with other actions
RePOP generates partially ordered plans

- **Number of actions**: RePOP typically returns shortest plans
- **Number of time steps (makespan)**: Graphplan produces optimal number of time steps, RePOP comes close
- **Flexibility**: RePOP typically returns the most flexible plans
Comparing solution quality

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of actions/ time steps</th>
<th>Flexibility degree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RePOP</td>
<td>Graphplan</td>
</tr>
<tr>
<td>Gripper-8</td>
<td>21/15</td>
<td>23/15</td>
</tr>
<tr>
<td>Gripper-10</td>
<td>27/19</td>
<td>29/19</td>
</tr>
<tr>
<td>Gripper-20</td>
<td>59/39</td>
<td>-</td>
</tr>
<tr>
<td>Rocket-a</td>
<td>35/16</td>
<td>40/7</td>
</tr>
<tr>
<td>Rocket-b</td>
<td>34/15</td>
<td>30/7</td>
</tr>
<tr>
<td>Logistics-a</td>
<td>52/13</td>
<td>80/11</td>
</tr>
<tr>
<td>Logistics-b</td>
<td>42/13</td>
<td>79/13</td>
</tr>
<tr>
<td>Logistics-c</td>
<td>50/15</td>
<td>-</td>
</tr>
<tr>
<td>Logistics-d</td>
<td>69/33</td>
<td>-</td>
</tr>
<tr>
<td>Bw-large-a</td>
<td>(8/5)</td>
<td>-</td>
</tr>
<tr>
<td>Bw-large-b</td>
<td>(11/8)</td>
<td>-</td>
</tr>
<tr>
<td>Bw-large-c</td>
<td>(17/10)</td>
<td>-</td>
</tr>
</tbody>
</table>
Ablation studies

CE: Consistency enforcement techniques (reachability analysis and disjunctive constraint handling)
HP: Distance-based heuristic

<table>
<thead>
<tr>
<th>Problem</th>
<th>UCPOP</th>
<th>+ CE</th>
<th>+ HP</th>
<th>+CE+HP (RePOP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gripper-8</td>
<td>*</td>
<td>6557/3881</td>
<td>*</td>
<td>1299/698</td>
</tr>
<tr>
<td>Gripper-10</td>
<td>*</td>
<td>11407/6642</td>
<td>*</td>
<td>2215/1175</td>
</tr>
<tr>
<td>Gripper-12</td>
<td>*</td>
<td>17628/10147</td>
<td>*</td>
<td>3380/1776</td>
</tr>
<tr>
<td><strong>Gripper-20</strong></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td><strong>11097/5675</strong></td>
</tr>
<tr>
<td>Rocket-a</td>
<td>*</td>
<td>*</td>
<td><strong>30110/17768</strong></td>
<td><strong>7638/4261</strong></td>
</tr>
<tr>
<td>Rocket-b</td>
<td>*</td>
<td>*</td>
<td>85316/51540</td>
<td>28282/16324</td>
</tr>
<tr>
<td>Logistics-a</td>
<td>*</td>
<td>*</td>
<td>411/191</td>
<td>847/436</td>
</tr>
<tr>
<td>Logistics-b</td>
<td>*</td>
<td>*</td>
<td>920/436</td>
<td>542/271</td>
</tr>
<tr>
<td>Logistics-c</td>
<td>*</td>
<td>*</td>
<td>4939/2468</td>
<td>7424/4796</td>
</tr>
<tr>
<td><strong>Logistics-d</strong></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td><strong>16572/10512</strong></td>
</tr>
</tbody>
</table>
Conclusion

• Developed effective techniques for improving partial-order planners:
  – Ranking partial plan heuristics,
  – Disjunctive representation for unsafe links,
  – Use of reachability analysis

• Presented and evaluated RePOP
  – Brings POP to the realm of effective planning algorithms
  – Can now exploit the flexibility of POP without too much efficiency penalty
Future Work

- Extend RePOP to deal with time and resource constraints
- Extend RePOP to deal with partially instantiated actions
- Improve the efficiency of RePOP in serial domains
  - Serial domains inherent weakness of POP?
  - Real-world domains tend to admit partially ordered plans
- Devising effective admissible heuristics for POP