Automatic Selection of Pattern Collections for Domain Independent Planning

MASTER THESIS PRESENTATION
Overview

• Introduction & Overview
• Background
• iPDB Heuristic
• PhO Heuristic
• Conclusion & Future Work
Background: Planning

• Planning task
• Variables
• States assign values to variables
• Initial state $s_0$ and goal state(s) $s_*$
• Operators have preconditions and effects
Background: Planning

- Heuristic search in state space
- Heuristic
  - Estimates cost to nearest goal
  - Admissible: never overestimates
- A* used as search algorithm
- IPC Benchmark & Coverage
Background: 15-Puzzle

\[ S_0 \]

\[
\begin{array}{cccc}
1 & 3 & 8 & 7 \\
5 & 6 & 2 & ✔ \\
12 & 13 & 14 & 4 \\
9 & 10 & 11 & 15 \\
\end{array}
\]

\[ S_* \]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & ✔ \\
\end{array}
\]
Background: Pattern Databases

- Pattern is a subset of variables $P \subseteq V$
- Simplified problem ignoring all variables not in the pattern $P$
- PDB contains cost of optimal plans for all states in the simplified problem
- Evaluating $h^P(s)$
Background: 15-Puzzle Abstraction

Pattern $P = \{T_1, T_2, T_3, T_4\}$

$S_0$

$h^P(s_0) = 5$

$S_*$
Background: Using multiple PDB heuristics

• Selection of a pattern collection
• Combination of heuristic values
  • Maximum of both heuristics
  • Sum of both heuristics if pattern are additive
• Two patterns are additive if there is no operator that affects both
• Canonical heuristic
  • Add where possible, maximum over sums
iPDB Heuristic

• Haslum et al. 2007
• Hill climbing algorithm in space of pattern collections
• Candidate patterns
• Samples
• Evaluated using the canonical heuristic
• Improvement
iPDB Heuristic: Hill Climbing

1. Add pattern for each goal variable to collection
2. Compute candidates

3. Repeat:
   1. Evaluate candidates
   2. Add best candidate to collection
   3. Compute new candidates
   4. Check stopping criterion
iPDB: Changes

- Time limit
- Improvement limit
- Ignoring candidates
- Changed sampling
- **Increasing neighbourhood**
iPDB: Improvement & Total Time

![Graph showing improvement and total time over iterations]

- **x-axis**: Iteration
- **y-axis** (left): Improvement
- **y-axis** (right): Time [s]

Legend:
- **Blue line**: Improvement
- **Red line**: Total time

**5.6.2014**

MSC THESIS SASCHA SCHERRER, UNIVERSITY BASEL
iPDB: Increasing the Neighbourhood

- Uses time limit of 900 seconds
- Add $n$ new variables when creating candidates
- Variables must influence each other
- Memory issues
iPDB: Evaluation

• iPDB 2var and 3var include a time limit

<table>
<thead>
<tr>
<th>Coverage</th>
<th>iPDB base</th>
<th>iPDB time</th>
<th>iPDB 2var</th>
<th>iPDB 3var</th>
</tr>
</thead>
<tbody>
<tr>
<td>664</td>
<td>684</td>
<td>706</td>
<td>698</td>
<td></td>
</tr>
</tbody>
</table>
PhO: Posthoc Optimization

- Pommerening et al. 2013
- Systematic pattern generation
- Evaluation: solves linear program (LP)
  - One variable per operator
  - One constraint per PDB
  - Depends on state
PhO: Changes

- Partial systematic sizes
- Dynamic systematic sizes
- Limiting number & total size of PDBs
- Limiting evaluation time
- Pruning unused constraints
PhO: Pruning Unused Constraints

• Samples
• Evaluate heuristic on samples
• Count how often each constraint was “useful”
• Remove constraints that were never “useful”
## PhO: Evaluation

<table>
<thead>
<tr>
<th>Coverage</th>
<th>PhO base</th>
<th>PhO prune</th>
</tr>
</thead>
<tbody>
<tr>
<td>590</td>
<td>598</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion & Future Work

- Viability of abstraction heuristics
- Further investigate
  - Better limit for iPDB
  - Better pruning of constraints for PhO

<table>
<thead>
<tr>
<th></th>
<th>iPDB 2var</th>
<th>PhO prune</th>
<th>LM-cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>706</td>
<td>598</td>
<td>763</td>
</tr>
<tr>
<td>Avg. coverage</td>
<td>54.38%</td>
<td>45.19%</td>
<td>53.07%</td>
</tr>
</tbody>
</table>
Additional Data: iPDB

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>dyn.</th>
<th>time</th>
<th>2var</th>
<th>3var</th>
<th>ign.</th>
<th>FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>finished</td>
<td>1134</td>
<td>1360</td>
<td>1360</td>
<td>1361</td>
<td>1345</td>
<td>1308</td>
<td>1388</td>
</tr>
<tr>
<td>coverage</td>
<td>664</td>
<td>675</td>
<td>684</td>
<td>706</td>
<td>698</td>
<td>705</td>
<td>703</td>
</tr>
</tbody>
</table>

![Graph showing improvement and total time over iterations]
Additional Data: PhO

<table>
<thead>
<tr>
<th></th>
<th>base (k=2)</th>
<th>eval.</th>
<th>PhO prune</th>
</tr>
</thead>
<tbody>
<tr>
<td>coverage</td>
<td>590</td>
<td>591</td>
<td>598</td>
</tr>
</tbody>
</table>

![Coverage Chart](chart.png)