Pattern Selection for Optimal Classical Planning with Saturated Cost Partitioning

Jendrik Seipp
August 14, 2019

University of Basel, Switzerland
• optimal classical planning
• $A^*$ search + admissible heuristic
• pattern databases
How to select patterns?

• bin packing (Edelkamp 2001)
• genetic algorithms (Edelkamp 2006)
• hill climbing (Haslum et al. 2007)
• CPC (Franco et al. 2017)
• CEGAR (Rovner et al. 2019)
• systematic (Pommerening et al. 2013)
How to combine multiple PDB heuristics?

- maximize
- cost partitioning
- saturated cost partitioning
Saturated cost partitioning

Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$$SCP\langle h_1; h_2 \rangle(s_2) = 5 + 3 = 8$$
Saturated cost partitioning

Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$max(h_1(s_2), h_2(s_2)) = \max(5, 4) = 5$
Saturated cost partitioning

Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$$SCP\langle h_{1}, h_{2} \rangle(s_{2}) = 5 + 3 = 8$$
Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$$
\text{SCP} \langle h_1; h_2 \rangle (s_2) = 5 + 3 = 8
$$
Saturated cost partitioning

Saturated cost partitioning algorithm

• order heuristics, then for each heuristic $h$:
  • use minimum costs preserving all estimates of $h$
  • use remaining costs for subsequent heuristics
Saturated cost partitioning

Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$SCP\langle h_1; h_2\rangle(s_2) = 5 + 3 = 8$
Saturated cost partitioning

Saturated cost partitioning algorithm

- order heuristics, then for each heuristic $h$:
  - use minimum costs preserving all estimates of $h$
  - use remaining costs for subsequent heuristics

$$h^{SCP}_{\langle h_1, h_2 \rangle}(s_2) = 5 + 3 = 8$$
Diverse orders for saturated cost partitioning

Diversification algorithm

- sample 1000 states \( S \)
- start with empty set of orders
- for 200 seconds:
  - sample a new state \( s \)
  - find a greedy order for \( s \)
  - if a sample in \( S \) profits from it, keep it
  - otherwise, discard it
Idea

- select patterns
- compute diverse saturated cost partitionings over PDBs
Idea

- select patterns with saturated cost partitioning
- compute diverse saturated cost partitionings over PDBs
## One Sys-SCP iteration

- start with empty pattern sequence $\sigma$
- for each pattern $P \in \text{ORDER}(\text{SYS})$:
  - add $P$ to $\sigma$ if $h_{\sigma}^{SCP}(s) < h_{\sigma \oplus P}^{SCP}(s) < \infty$ for any state $s$
- repeat until hitting time limit
- return all selected patterns
One Sys-SCP iteration

- start with empty pattern sequence $\sigma$
- for each pattern $P \in \text{ORDER}($SYS$)$:
  - add $P$ to $\sigma$ if $h^{SCP}_\sigma(s) < h^{SCP}_{\sigma \oplus P}(s) < \infty$ for any state $s$
- repeat until hitting time limit
- return all selected patterns
- problem: testing every state is infeasible
Evaluating a pattern using its projection

**Theorem**

\[ \exists s \in S(T) : h_{\sigma}^{SCP}(\text{cost}, s) < h_{\sigma \oplus P}^{SCP}(\text{cost}, s) < \infty \]

\[ \iff \exists s' \in S(T_P) : \quad 0 < h_{\land P}^*(\text{rem}, s') < \infty \]
Using the theorem

• keep track of the remaining cost function
• select a PDB if it has positive finite goal distances
Pattern orders

order by increasing pattern size, break ties by:

- random
- states in projection
- active operators
- Fast Downward variable order:
  - up: [7, 5], [8, 2], [8, 5]
  - down: [8, 5], [8, 2], [7, 5]
Pattern orders

order by increasing pattern size, break ties by:

- random
- states in projection
- active operators
- **Fast Downward variable order:**
  - up: [7, 5], [8, 2], [8, 5]
  - down: [8, 5], [8, 2], [7, 5]
**Systematic patterns with limits**

**LIM**: 2M states per PDB, 20M states in collection, 100 seconds

<table>
<thead>
<tr>
<th>Max pattern size</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sys</strong></td>
<td>840</td>
<td>986</td>
<td>1057</td>
<td>922</td>
<td>731</td>
</tr>
<tr>
<td><strong>Sys-LIM</strong></td>
<td>840</td>
<td>985</td>
<td><strong>1088</strong></td>
<td>1050</td>
<td>1035</td>
</tr>
</tbody>
</table>
## Sys-SCP vs. other pattern selection algorithms

<table>
<thead>
<tr>
<th></th>
<th>HC</th>
<th>Sys-3-LIM</th>
<th>CPC</th>
<th>CEGAR</th>
<th>Sys-SCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>966</td>
<td>1088</td>
<td>1055</td>
<td>1098</td>
<td>1168</td>
</tr>
<tr>
<td>#domains Sys-SCP better</td>
<td>28</td>
<td>23</td>
<td>21</td>
<td>21</td>
<td>–</td>
</tr>
<tr>
<td>#domains Sys-SCP worse</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>–</td>
</tr>
</tbody>
</table>
Future work

• test patterns on samples
• new pattern selection algorithm based on saturated cost partitioning
• outperforms all previous pattern selection algorithms