Learning Generalized Unsolvability Heuristics for Classical Planning

Simon Ståhlberg\textsuperscript{1}, Guillem Francès\textsuperscript{2} and Jendrik Seipp\textsuperscript{1}

\textsuperscript{1}Linköping University, Sweden
\textsuperscript{2}Universitat Pompeu Fabra, Spain

IJCAI 2021
Introduction

• Recently, interest in unsolvability heuristics for classical planning
• Tailored heuristics are successful for individual problems
• Learn generalized formulas for detecting unsolvable states
• Many unsolvable states are detectable in polynomial time
• We use Boolean features based on description logic to describe such states in a concise and generalized way
• Three different methods for learning such formulas
Description Logic

• Use grammar defined by Description Logic (DL) in classical planning
• We are interested in the grammar, not inference
• DL separates schema (TBox) and data (ABox), similar to planning
  • TBox corresponds to the domain
    • Unary predicates in the domain are concepts
    • Binary predicates in the domain are roles
  • ABox corresponds to the problem
    • The universe is the set of objects in the problem
Spanner

• Spanner as a running example
  • A single agent, Bob, must *tighten* all *loose nuts* at the gate with *spanners*
  • Bob lives in a shed and must walk to the gate
  • Along the one-way path there are *usable* spanners that he can *pick up*
  • All spanners becomes unusable after use
• Once Bob leaves a location, he cannot return
• Bob is in an unsolvable state if he leaves behind too many spanners
• Predicates: location/1, locatable/1, man/1, nut/1, spanner/1, at/2, carrying/2, useable/1, link/2, tightened/1, loose/1
• Actions: walk, pickup_spanner, tighten_nut
Example

![Diagram of a sequence of locations with items]

<table>
<thead>
<tr>
<th>man</th>
<th>loose</th>
<th>usable</th>
<th>link</th>
<th>at</th>
<th>link^+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>{bob}</td>
<td>{nut1}</td>
<td>{(shed, location1),</td>
<td>{bob, location1),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(location1, location2),</td>
<td>(shed, location2),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spanner1, spanner2</td>
<td></td>
<td>(location2, gate)}</td>
<td>(shed, gate),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(location1, gate),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(location2, gate)}</td>
</tr>
</tbody>
</table>
Example

\[\exists t^{-1}.\text{man} \quad \exists \text{link}^+. (\exists t^{-1}.\text{man}) \quad \exists t. (\exists \text{link}^+. (\exists t^{-1}.\text{man}))\]

<table>
<thead>
<tr>
<th>4</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>{\text{location2}}</td>
<td>{\text{location1, shed}}</td>
<td>{\text{spanner1}}</td>
</tr>
</tbody>
</table>
Features

• Numerical features are composed by taking the cardinality, e.g.
  • $|\text{usable}|$
  • $|\text{loose} \sqcup \exists \text{at.}(\exists \text{link}^+. (\exists \text{at}^{-1}. \text{man}))|$

• Boolean features are composed by comparing numerical features, e.g.
  • $|\text{loose} \sqcup \exists \text{at.}(\exists \text{link}^+. (\exists \text{at}^{-1}. \text{man}))| > |\text{usable}|$
  • $|\exists \text{at-person.}(\exists \text{at-car}^{-1}. \top)| = 0$
  • Also, greater than zero
Spanner

\[ |\text{loose} \sqsupset \exists \text{at. } (\exists \text{link}^+. (\exists \text{at}^{-1}. \text{man})) | > |\text{usable}| \]

• The formula generalizes to the class of problems that can be produced by the generator

• The formula exploits certain properties of the generator:
  • The traversal graph from Bob's shed and the gate is a path graph
  • All spanners are initially usable
  • There is only one agent

• When Bob tightens a nut, both left- and righthand side decrements
Pipeline

• **State labeling**
  • Input: a PDDL domain and a collection of PDDL problems
  • Output: a set of states labeled as solvable or unsolvable

• **Feature generation**
  • Input: a PDDL domain and the output from the previous step
  • Output: a set of Boolean features and their evaluations on all states (and their label)

• **Formula construction**
  • Input: the output from the previous step
  • Output: a DNF of Boolean features, we consider the three criteria
    • Perfect: holds if and only if state is an unsolvable state (not always possible)
    • Safe: holds if state is an unsolvable state
    • DecisionTree: maximizes F1 score
<table>
<thead>
<tr>
<th>Task</th>
<th>$h^{CG}$</th>
<th>$h^{CEA}$</th>
<th>$h^{SEQ}$</th>
<th>$h^1$</th>
<th>$h^2$</th>
<th>$h^3$</th>
<th>$k$-consistency</th>
<th>$\mathcal{T}$-Perfect</th>
<th>$\mathcal{T}$-Safe</th>
<th>DecisionTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barman</td>
<td>0.00</td>
<td>0.88</td>
<td>0.39</td>
<td>0.88</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.97</td>
</tr>
<tr>
<td>Childsnack</td>
<td>0.58</td>
<td>0.58</td>
<td>0.09</td>
<td>0.58</td>
<td>0.94</td>
<td>1.00</td>
<td>0.00</td>
<td>0.27</td>
<td>0.27</td>
<td>1.00</td>
</tr>
<tr>
<td>Hiking</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Nomystery</td>
<td>0.00</td>
<td>0.53</td>
<td>0.00</td>
<td>0.31</td>
<td>0.91</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.83</td>
<td>0.87</td>
</tr>
<tr>
<td>Spanner</td>
<td>0.05</td>
<td>0.05</td>
<td>0.00</td>
<td>0.05</td>
<td>0.13</td>
<td>0.31</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>1.00</td>
</tr>
<tr>
<td>Woodworking</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thanks for listening!