Lifted Successor Generation using Query Optimization Techniques

Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès

University of Basel, Switzerland
Universitat Pompeu Fabra, Spain
Lifted Successor Generation using Query Optimization Techniques

Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès

University of Basel, Switzerland
Universitat Pompeu Fabra, Spain
Lifted Successor Generation using Query Optimization Techniques

Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès

University of Basel, Switzerland
Universitat Pompeu Fabra, Spain
(:action stack
  :parameters (?x ?y)
  :precondition (and (holding ?x)
                    (clear ?y))
  :effect (and (not (holding ?x))
           (not (clear ?y))
           (clear ?x)
           (handempty)
           (on ?x ?y)))

(:objects block1, block2, ..., block1000)
(:action stack
 :parameters (?x ?y)
 :precondition (and (holding ?x)
                    (clear ?y))
 :effect (and (not (holding ?x))
           (not (clear ?y))
           (clear ?x)
           (handempty)
           (on ?x ?y)))

(:objects block1, block2, ..., block100)
(stack block1 block2)

(stack block1 block3)

...

(stack block1 block100)

...

(stack block100 block99)
Almost 10,000 ground actions is still fine.
But grounding is not always fine.
Organic Synthesis domain, instance #11:
almost 71,000,000,000,000 ground actions.

Guess the optimal plan length.
Grounding is usually fine.

But sometimes it requires 35 trillion times more effort than we need.

What can we do about it?
Lifted Planning: Ground States + Action Schemas
Lifted Planning: **Ground States** + Action Schemas
(:predicates (at ?x ?y) (path ?x ?y))

(:init (at obj1 l1)
  (at obj2 l1)
  (at obj3 l3)
  (at obj4 l2)
  (path l1 l2)
  (path l1 l3)
  (path l2 l3)
  (path l3 l4))
<table>
<thead>
<tr>
<th></th>
<th>at</th>
<th></th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj1</td>
<td>l1</td>
<td></td>
<td>l1</td>
</tr>
<tr>
<td>obj2</td>
<td>l1</td>
<td></td>
<td>l1</td>
</tr>
<tr>
<td>obj3</td>
<td>l3</td>
<td></td>
<td>l2</td>
</tr>
<tr>
<td>obj4</td>
<td>l2</td>
<td></td>
<td>l3</td>
</tr>
</tbody>
</table>

@ICAPS 2020
Lifted Planning: Ground States + Action Schemas
Lifted Planning: Ground States + Action Schemas
:precondition
(and (at ?X ?Y)
  (path ?Y ?W)
  (path ?W ?Z)))
(precondition
 (and (at ?X ?Y)
      (path ?Y ?W)
      (path ?W ?Z)))

at(X,Y) ▷ path(Y,W) ▷ path(W,Z)

These are conjunctive queries.
<table>
<thead>
<tr>
<th>at</th>
<th>path</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Z</td>
</tr>
<tr>
<td>obj1</td>
<td>l1</td>
<td>l1</td>
</tr>
<tr>
<td>obj2</td>
<td>l1</td>
<td>l1</td>
</tr>
<tr>
<td>obj3</td>
<td>l3</td>
<td>l2</td>
</tr>
<tr>
<td>obj4</td>
<td>l2</td>
<td>l3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>l3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>l4</td>
</tr>
</tbody>
</table>
\( \text{at}(X, Y) \bowtie \text{path}(Y, W) \bowtie \text{path}(W, Z) \)

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>W</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj1</td>
<td>l1</td>
<td>l2</td>
<td>l3</td>
<td></td>
</tr>
<tr>
<td>obj1</td>
<td>l1</td>
<td>l3</td>
<td>l4</td>
<td></td>
</tr>
<tr>
<td>obj2</td>
<td>l1</td>
<td>l2</td>
<td>l3</td>
<td></td>
</tr>
<tr>
<td>obj2</td>
<td>l1</td>
<td>l3</td>
<td>l4</td>
<td></td>
</tr>
<tr>
<td>obj4</td>
<td>l2</td>
<td>l3</td>
<td>l4</td>
<td></td>
</tr>
</tbody>
</table>
Conjunctive queries are NP-hard.

But there’s a significant island of tractability.
\[ \text{at}(X, Y) \]
\[ \text{at}(S, F) \]
\[ \text{move-dir}(Y, F, D) \]
\[ \text{move-dir}(F, T, D) \]
Conjunctive queries with join-trees have acyclic hypergraphs.

They are solvable in output-polynomial time.
Almost 87% of the action schemas in IPC have preconditions with acyclic hypergraphs.

If we focus on hard-to-ground domains, then it is only 21%.
Great part of this is due to inequality constraints.

Processing inequalities:
80% in hard-to-ground domains.
Is this good in practice?
Time (s)

unsolved

Our Planner

Fast Downward

GED

GED Split

Org.Synt. MIT

Org.Synt. Alkene

Org.Synt. Orig.

Pipesworld Tank.

@ICAPS 2020
Conclusions

- Lifted planning can help in hard-to-ground domains.
- Most planning action schemas have acyclic preconditions.
- Much faster than previous state-of-the-art lifted planners.