Generalized Potential Heuristics for Classical Planning

Guillem Francès, Augusto B. Corrêa, Cedric Geissmann, Florian Pommerening

University of Basel, Switzerland

August 15, 2019
Introduction
Motivation

- Problems of a classical planning domain share a common structure.
- Generalized planning tries to find a solution for a whole domain.
In this Work

- For some domains a solution that solves the whole domain can be described easily
- We try to **learn these solutions from small instances**
Background
Representing Progress with Heuristic Functions

Descending & dead-end avoiding heuristics (Seipp et al., 2016)

- **descending:**
  all alive states have an improving successor
- **dead-end avoiding:**
  all improving successors of alive states are solvable

A state is *alive* if it is reachable, solvable and not a goal.

Descending, dead-end avoiding heuristics

- guide greedy search to a goal
- use at most $h(s_0) - h(s_g)$ steps
- encode a *measure of progress* (Parmar, 2002)
Description Logics

Description Logic $\mathcal{SOL}$ with Role Value Maps

- **Primitive concepts**
  - represent set of objects with some property
- **Primitive roles**
  - represent relation between objects

Complex concepts:
- $\bot, \top$
- $\neg C$
- $C_1 \sqcup C_2, C_1 \sqcap C_2$
- $\forall R.C, \exists R.C$
- $R_1 = R_2$
- $\{a_1, \ldots, a_n\}$

Complex roles:
- $R^{-1}$
- $R^+$
- $R_1 \circ R_2$
Description Logic for Planning Domain

Description logic for a planning domain

- Interpretation for every state of any instance

Concepts and roles

- Primitive concept for each unary predicate
  - Example: *clear*

- Primitive role for each binary predicate
  - Example: *on*

- Primitive concepts and roles for predicates in the goal
  - Example: *on\_G*
Generalized Potential Heuristics
Generalized Potential Heuristics

Definition (Generalized Potential Heuristic)

Linear combination of features well-defined over all instances:

\[ h(s) = \sum_{f \in \mathcal{F}} w(f) \cdot f(s) \]

- We use two types of features based on description logics:
  - cardinality features \(|C|\)
  - distance features (see paper)
Example: Clearing a Block

- Consider the subset of Blocksworld problems where the goal is to clear a given subset of blocks.

**Descending and Dead-end Avoiding Generalized Potential Heuristic**

\[ h(s) = 2 \cdot |C_1| + |C_2| \]

- \( C_1 \equiv \exists on^+ . clear_G \): “Set of blocks above some block that needs to be cleared”
- \( C_2 \equiv holding \): “Set of blocks being held”
We prove that descending, dead-end avoiding generalized heuristics exist for a number of standard domains:
- Blocksworld
- Gripper
- Spanner
- Miconic
- Logistics

Greedy search solves all instances in linear time

→ The challenge: can we obtain these heuristics automatically?
Learning the Heuristic
Overview of our inductive approach:

1. Fully expand small instances to generate training set $S$.
2. Generate set of generalized features $\mathcal{F}$ with all features under a certain syntactic complexity.
3. Compute simplest potential heuristic on $\mathcal{F}$ that is descending and dead-end avoiding on states in $S$.
   - If no such $h$ exists, try with larger set $\mathcal{F}$.
   - If it does exist, test $h$ on unseen instances.
Computing the Weights

Mixed Integer Linear Program

\[
\min_w \sum_{f \in \mathcal{F}} [w_f \neq 0] \mathcal{K}(f) \quad \text{subject to}
\]

\[
\bigvee_{s' \in \text{succ}(s)} h(s') + 1 \leq h(s) \quad \text{for alive states } s
\]

\[
h(s') \geq h(s) \quad \text{for transitions } (s, s')
\]

where \( s \) is alive and \( s' \) is unsolvable

- Solutions map to heuristics that are descending and dead-end avoiding on all states in \( S \)...
Computing the Weights

Mixed Integer Linear Program

\[
\min_w \sum_{f \in F} [w_f \neq 0]K(f) \quad \text{subject to}
\]
\[
\bigvee_{s' \in \text{succ}(s)} h(s') + 1 \leq h(s) \quad \text{for alive states } s
\]
\[
h(s') \geq h(s) \quad \text{for transitions } (s, s')
\]

where \( s \) is alive and \( s' \) is unsolvable

- Solutions map to heuristics that are descending and dead-end avoiding on all states in \( S \)...
- ... and have minimum complexity.
Our approach learns generalized heuristics on standard domains such as Gripper, Miconic, Spanner, VisitAll.

We have (manually) checked that they are descending and dead-end avoiding on all instances of the domain.
  - Exception VisitAll: No linear solution possible

Steepest-ascent hill-climbing with these heuristics solves any instances of these domains in linear time.

Other domains such as Blocksworld appear to need better feature exploration strategies.
Conclusion
Contributions

- General descending and dead-end avoiding heuristics exist for several planning domains.
- These solve any instance in linear time.
- We can learn them automatically from a suitable logical model and small instances.
Discussion and Future Work

- The learned heuristic can be easily interpreted.
- The learned heuristic has only inductive guarantees, but
  - We have shown how it can be refined in an online fashion whenever it doesn’t generalize correctly.
  - One could attempt to prove the correctness of the heuristic deductively with an automatic theorem prover.
- Better feature generation methods are necessary to scale up to more complex problems.
Bonus Slides
Example: unrestricted Blocksworld instances

\[ h_{bw}(s) = -4|C_6| - |\text{holding}| - 2|\text{ontable}| - 2|C_7|, \]

- **C_1**: \(\text{ontable}_G \sqcap \text{ontable}\)
  Blocks that are correctly placed on the table

- **C_2**: \((\exists \text{on}_G. \top) \sqcap (\text{on} = \text{on}_G)\)
  Blocks that are placed on their target block

- **C_3**: \(\neg(\text{ontable}_G \sqcup \exists \text{on}_G. \top)\)
  Blocks that are not mentioned in the goal

- **C_4**: \(C_1 \sqcup C_2 \sqcup C_3\)
  Blocks where block (or table) below is consistent with the goal

- **C_5**: \(\forall \text{on}_G.\neg^1.(\text{on} = \text{on}_G)\)
  Blocks where the block above is consistent with the goal

- **C_6**: \(C_4 \sqcap \forall \text{on}^+.\neg \neg^1.(C_4 \sqcap C_5)\)
  Blocks that are well-placed.

- **C_7**: \(\text{holding} \sqcap \exists \text{on}_G.(\text{clear} \sqcap C_6)\)
  Blocks held while their target block is clear and well-placed.
<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>M</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td># of training instances</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td># of iterations</td>
<td>2.0</td>
<td>2.7</td>
<td>1.0</td>
<td>1.7</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{F}</td>
<td>$</td>
<td>469</td>
<td>2105</td>
</tr>
<tr>
<td># of MIP variables</td>
<td>2017</td>
<td>7273</td>
<td>3381</td>
<td>1039</td>
</tr>
<tr>
<td># of MIP constraints</td>
<td>2238</td>
<td>7331</td>
<td>3370</td>
<td>1190</td>
</tr>
<tr>
<td>Complexity of $h$</td>
<td>8 (18)</td>
<td>6 (14)</td>
<td>8 (20)</td>
<td>5 (8)</td>
</tr>
<tr>
<td># of features in $h$</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Total time</td>
<td>8h</td>
<td>32m</td>
<td>178s</td>
<td>87s</td>
</tr>
<tr>
<td>Total MIP time</td>
<td>7.4h</td>
<td>26m</td>
<td>6.8s</td>
<td>2.1s</td>
</tr>
</tbody>
</table>