Generalized Potential Heuristics for Classical Planning

Guillem Francès, Augusto B. Corrêa, Cedric Geissmann, and Florian Pommerening
University of Basel, Switzerland

Generalized Planning

- Classical planners compute plans that work for a single instance.
- Generalized planning aims instead at policies that solve entire *classes of instances* that share an underlying structure.

Objective of this work

- Learn interpretable generalized policies from small instances.
- The target policies solve any instance in time linear in problem size.

Running Example: Blocksworld

Initial state   Goal

Description Logics for Planning ($\mathcal{SOI}$ with equality role-value-maps)

Primitive Concepts & Roles

- $ontable = \{$■, ■$\}$
- $on = \{($■, ■$)\}$
- $holding = \emptyset$
- $clear = \{$■, ■$\}$

Goal Concepts & Roles

- $on_G = \{($■, ■$), ($■, ■$)\}$

Complex Concepts & Roles

- C_1: Blocks that are correctly placed on the table
  $ontable_G \sqcap ontable = \emptyset$
- C_2: Blocks that are placed on their target block
  $(\exists on_G.\top) \sqcap (on = on_G) = \{$■$\}$
- C_3: Blocks with no target in the goal
  $\neg(ontable_G \sqcup \exists on_G.\top) = \{$■$\}$
- C_4: Blocks on object consistent with the goal
  $C_1 \sqcup C_2 \sqcup C_3 = \{$■, ■$\}$
- C_5: Blocks where the block above is consistent with the goal
  $\forall on_G^{-1}.(on = on_G) = \{$■, ■$\}$
- C_6: Blocks that are well-placed
  $C_4 \sqcap \forall on^+.(C_4 \sqcap C_5) = \{$■$\}$
- C_7: Blocks held while their target block is clear and well-placed
  $holding \sqcap \exists on_G.(clear \sqcap C_6) = \emptyset$

Generalized Potential Heuristics

Generalized potential heuristics are linear combinations of first-order features well defined over all instances:

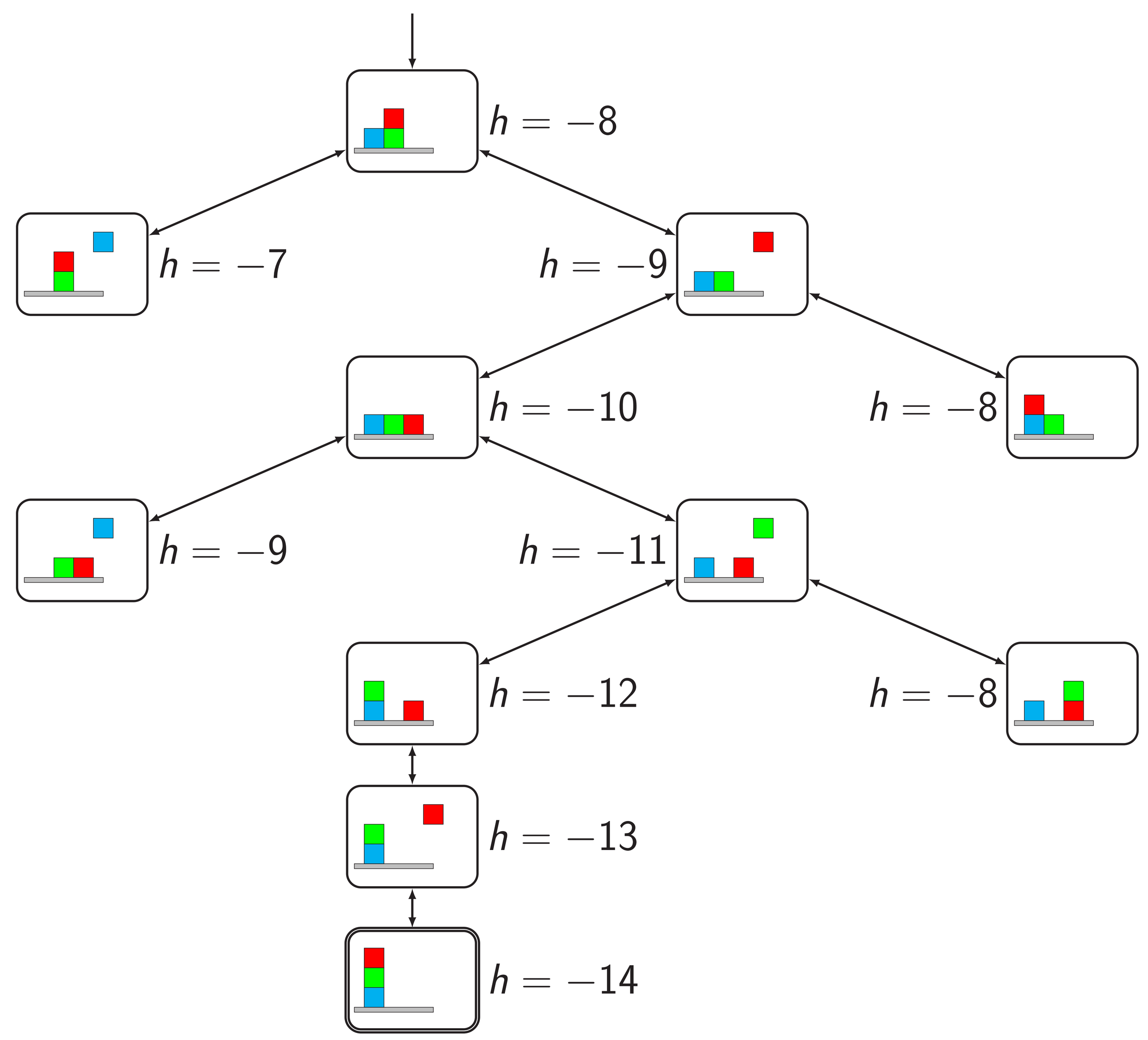
$$h(s) = \sum_{f \in \mathcal{F}} w_f f(s)$$

- Two types of *features*:
  - cardinality features $|C|$
  - distance features $dist(C, R, C')$
- Blocksworld example:

$$h_{\mathrm{bw}}(s) = -4|C_6| - |holding| - 2|ontable| - 2|C_7|$$

Good Heuristic Properties for Satisficing Greedy Search

- **descending**: all alive (reachable, solvable and non-goal) states have an improving successor.
- **dead-end avoiding**: all improving successors of alive states are solvable.



- **Key property**: Hill climbing with a descending, dead-end avoiding heuristic solves a problem in a *linear number of steps*.

Theoretical Results

We show that descending, dead-end avoiding generalized heuristics exist for a number of standard domains:

- Blocksworld, Gripper, Spanner, Miconic, Logistics.

Learning a Generalized Heuristic from Examples

1. Fully expand a set of small instances
2. Generate a pool of features $\mathcal{F}$
3. Synthesize weights for a descending and dead-end avoiding heuristic using features from $\mathcal{F}$
   $\rightarrow$ if not possible: add features to $\mathcal{F}$ and continue with (3)
4. Test on unseen instances
   $\rightarrow$ if not solved: add *refinement constraint* and continue with (3)

Synthesizing Weights

Synthesizing weights for fully expanded state spaces

MIP Model

$$\min_{w} \sum_{f \in \mathcal{F}} [w_f \neq 0]\mathcal{K}(f) \quad \text{subject to}$$
$$\bigvee_{s' \in succ(s)} h(s') + 1 \leq h(s) \qquad \text{for } s \in \mathcal{S}_A$$
$$h(s') \geq h(s) \qquad \text{for } (s, s') \in \mathcal{T}, s' \text{ unsolvable,}$$

Search fails on unseen instance

- heuristic is either not descending or not dead-end avoiding

Refinement Constraint

$$\left(\bigvee_{i=0}^{n-1} h(s_i) \leq h(s_{i+1})\right) \vee \left(\bigvee_{s' \in succ(s)} h(s') + 1 \leq h(s)\right)$$

Empirical Results

	Gripper	Miconic	Spanner	VisitAll
# of training instances	8	12	11	9
$\lvert\mathcal{F}\rvert$	469	2105	904	330
Complexity of h	8 (18)	6 (14)	8 (20)	5 (8)
# of features in h	5	4	5	3
Total time	8h	32m	178s	87s
Total MIP time	7.4h	26m	6.8s	2.1s

Conclusions & Future Work

- We can learn interpretable, linear solving mechanisms that work for infinite classes of problems from small instances.
- First-order theorem proving could be used to prove deductively the correctness of the learned heuristics.