Towards learning domain-independent planning heuristics

Paweł Gomoluch[*], Dalal Alrajeh[*], Alessandra Russo[*], Antonio Bucchiarone[†]

[*]Imperial College London, UK
[†]Fondazione Bruno Kessler, Trento, Italy

August 19, 2017

Presentation overview:

1. Planning as heuristic forward search.
2. Learning planning heuristics.
3. Preliminary results.
4. Future work.

Classical planning

Planning is composing sequences of actions which lead to satisfaction of a given goal when executed from a given initial state. In *classical* planning, the environment is fully-observable and the actions are deterministic.

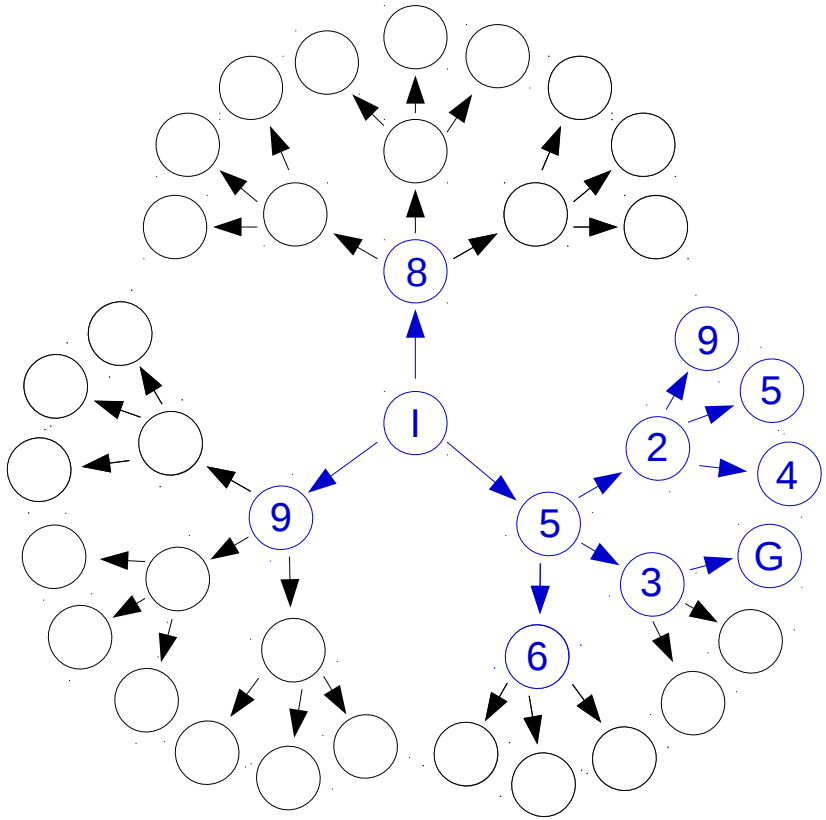
A classical planning task is defined by:

- Set of discrete variables (propositional or multi-valued)
- Initial state - an assignment over the variables
- Goal - a formula which must be satisfied by the final state
- Set of *operators* with specified *preconditions* and *effects*

Planning as heuristic forward search

Exhaustive search is impossible for all but the simplest problems. Search procedure needs to be guided towards more promising regions of search space and ignore the others.

A planning heuristic is a function estimating the cost of reaching the goal from the given state.

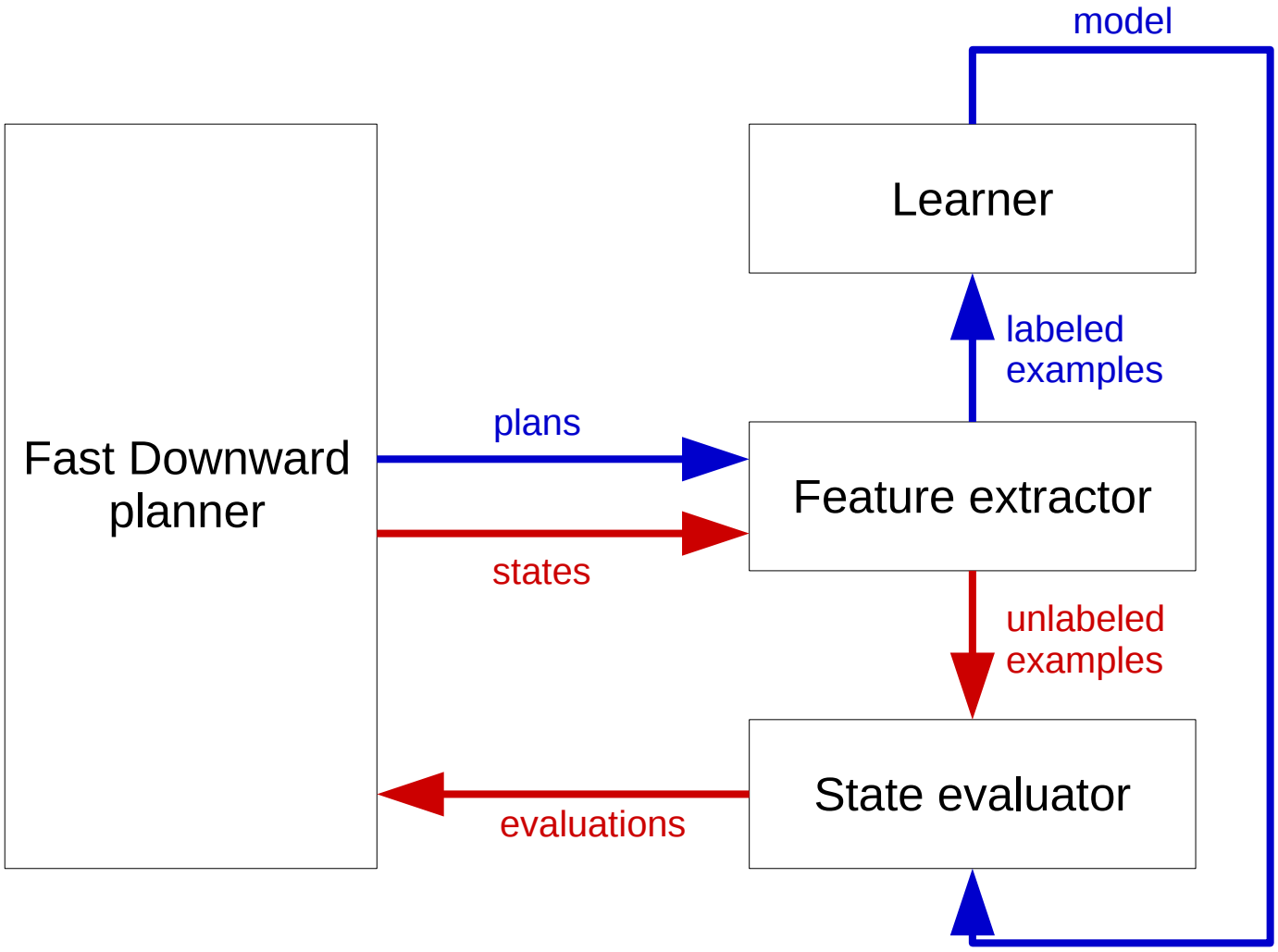


Learning heuristics

Existing domain-independent heuristics are based on hand-coded algorithms, for example relaxed planners. **Can the heuristics be learned instead? Can we (incrementally) improve on hand-coded algorithms?**

Domain-independence of the learned knowledge is essential for the **generality** of the solution. For example, in dynamic systems it may be impossible to know domain before planning is necessary, let alone gather data and train a heuristic.

System architecture



How can we represent a planning state?

- Naive approach: include values of all the variables (this is essentially *problem*-dependent).
- Domain-dependent approach: base the features concepts from the domain description (e.g. the number of trucks, measures of city graph connectedness).
- Domain-dependent features computed in a domain-independent manner (e.g. occurrences of particular actions or action pairs)[1]
- **Full domain-independence: features do not depend on any domain knowledge.**

[1] Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Learning to Rank for Synthesizing Planning Heuristics". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*. 2016, pp. 3089–3095.

What features can we use?

Learning heuristic functions from scratch means only using low-level features like for example the number of unsatisfied goal propositions.

When the learned mechanism makes extensive use of features based on existing heuristics, the task effectively changes to learning **corrections for existing heuristics** or learning **combinations of heuristics**.

Data collection

For training we need examples of state-goal pairs, labelled with the cost of achieving the goal from the state. To know the cost, we need to have solved the planning problem for the state-goal pair.

We can obtain such data from:

- Exhaustive solutions to sufficiently small planning problems.
- High-quality heuristic solutions to a bit larger problems (e.g. obtained with portfolio planners using conservative search routines, established heuristics and large timeouts).

Preliminary results

First experiments have been conducted on IPC domains (*Transport*, *Woodworking*, *Planning*) using ridge regression and a simple neural network.

- 3956 examples from optimal solutions of small problems representing 3 domains.
- 2 feature sets: one using only simple features and one based on FF[2] computation.
- 2 learning algorithms: ridge regression and a fully-connected feed-forward network with 2 hidden layers.

[2] Jörg Hoffmann and Bernhard Nebel. "The FF Planning System: Fast Plan Generation Through Heuristic Search". In: *Journal of Artificial Intelligence Research* 14 (2001), pp. 263–312.

Preliminary results - coverage

The number if problems solved out of 30 test problems for every domain.

	Transport	Woodworking	Parking
FF	10	26	19
Goal count	22	14	14
RR	22	13	11
NN	18	6	3
RR-FF	12	23	5
NN-FF	12	23	21

Preliminary results - number of generated states (*Parking*)

For the first 10 *Parking* problems, NN-FF generated on average 3.5 times less states[3] than the original FF. However, such advantage was only displayed on this particular problem subset.

problem	1	2	3	4	5
FF	34	153	1194	818	1876
NN-FF	34	52	482	260	470

problem	6	7	8	9	10
FF	2207	4068	13671	5232	42964
NN-FF	631	2228	1198	2428	2191

[3] geometric mean

Preliminary results - summary

- The heuristics learned from scratch have performed much worse than the *corrections*, which in turn struggle to outperform their original counterparts.
- Under specific conditions the corrections can be consistently better informed than the originals.

Future work

- More data (including high quality heuristic solutions).
- Better feature representation.
- Different learning mechanisms and loss functions.
- Preferred operators and rollout exploration.