**Synthesis of Search Heuristics for Temporal Planning via Reinforcement Learning**

Andrea Micheli and Alessandro Valentini
Embedded Systems Unit, Fondazione Bruno Kessler, Italy

---

**Motivation**

Once deployed, a temporal planner will solve several different problems on the same domain

**Example**

- Organize the logistics of the same factory once a day
- Operate the same drone in the same area for different missions from different initial states

**Key intuition**

Instead of resorting to pure reasoning each time, can we learn characteristics of the domain and exploit them for efficiency?

Analogous to a worker that gets accustomed to a certain workplace and gains dexterity

---

**Pipeline**

We learn a specialized heuristic keeping a fully-functional planner online

---

**MDP for a bounded planning problem set**

A bounded planning problem set with at most \( k \) objects for a planning domain \( \mathcal{D} \) (written \( \mathcal{P}^k \)) is a finite set of planning problems \( \mathcal{P}^k \) for each having less than \( k \) objects.

Given a planning domain and a bounded planning problem set we define an MDP \( \mathcal{M}^k = (S, A, T, R, \gamma) \) s.t.

- \( S = \{\ldots\} \) all planner states for all instances
- \( A = \{\ldots\} \) all actions (events) for all instances

\( T(s, a, s') = 0 \) if \( s \rightarrow a \rightarrow s' \) \( \in \mathcal{P}^k \)

\( T(s, a, s') = 0 \) if \( s \not\rightarrow a \rightarrow s' \)

\( R(s, a, s') = 1 \) if \( s' \) is a goal state

\( R(s, a, s') = 0 \) otherwise.

---

**From the optimal value function \( (\gamma^*) \) to the optimal heuristic \( (h^*) \)**

For a bounded planning problem set \( \mathcal{P}^k \), the following equation holds.

\[
K^*_c(s) = \begin{cases} 
\log(V(s)) \text{ if } \text{if } V(s) > 0 \\
0 \text{ otherwise}
\end{cases}
\]

**Intuition**

\[
\begin{align*}
V^0 & = 1 \\
V^1 & = 1 \\
V^2 & = 1 \\
V^3 & = 1 \\
V^4 & = 1 \\
V^5 & = 1 \\
V^6 & = 1 \\
V^7 & = 1 \\
V^8 & = 0
\end{align*}
\]

**Neural Architecture: Predicting \( V^* \)**

---

**State Vectorization**

Given a planning state \( s \) we derive a vector \( z_i \) in \( \mathbb{R}^n \)

**RL-Based Heuristic Learner**

Basically Deep-Q-Learning on \( \mathcal{M}^k \)

with some adjustments:

- State value function, single output network instead of DQN
- Heuristic proportional random action selection
- Bias in problem selection
- Memory replay with positive bias
- Fixed max depth of episodes

**Learned Heuristic**

The learned heuristic \( h_{\text{Learn}} \) is an approximation of \( h^* \)

\[
h_{\text{Learn}}(s) = \begin{cases} 
\min(h, V(s)) & \text{if } V(s) > 0 \\
\Delta_{h} & \text{if } V(s) = 0 \\
\text{otherwise}
\end{cases}
\]

Where \( \Delta_{h} \) is bigger than the pre-fixed cutoff length of episodes set in learning

**Experimental Evaluation**

**Case Studies**

- MaJSP: A fleet of AGVs with logistics tasks in a warehouse.
- The problems differ for the number of items to be moved and the intermediate steps.
- Kitting: A single robot serving a continuous production line with kits of components taken from shelves.
- The problems require different sequences of kits to be delivered.

**Competitors**

- TAMER (\( h_{\text{Learn}} \)): our fully-symbolic state-of-the-art planning
- TAMER (\( h_{\text{Q}} \)): the learned RL policy executed without backtracking.
- TAMER (\( h_{\text{RL}} \)): our planner equipped with the learned heuristic \( h_{\text{Learn}} \)

**Results**

10-fold cross-validation and sensitivity analysis over 100k RL episodes.

---

**Conclusion**

- Take-Away Message
  - Strict correlation between planning heuristics and state value functions in RL
  - Use RL to automatically synthesize planning heuristics looks promising

- Future work
  - Extend the approach to overcome limitations
    - Fixed state size, Fixed network architecture, Bounded numeric values, Incomplete temporal information
  - Supervised learning from search spaces