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

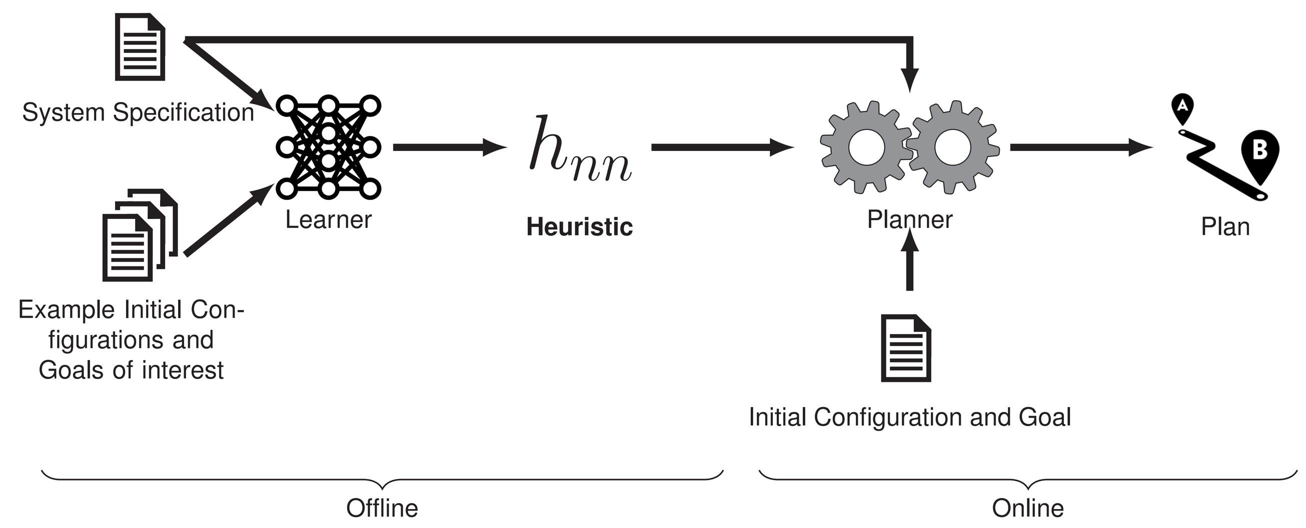
Example

- 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

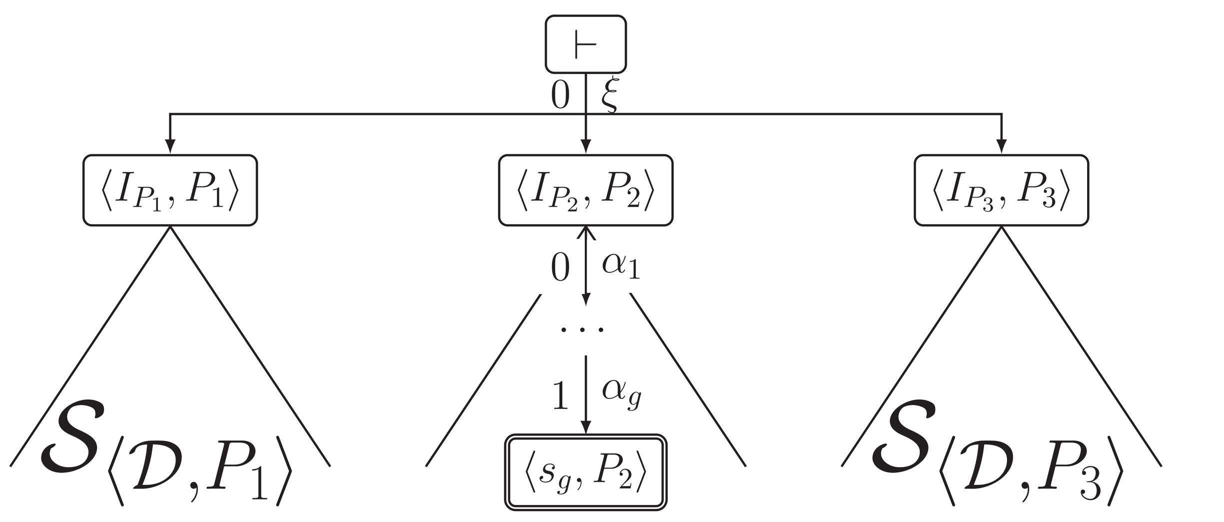
Bounded Planning Problem Set

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

Given a planning domain and a bounded planning problem set we define an MDP $\mathcal{M}_{\mathcal{P}^k_{\mathcal{D}}} \doteq \langle S, A, T, R, \vdash \rangle$ is s.t.
$S \doteq \{\vdash\} \cup$ all planner states for all instances
$A \doteq \{\xi\} \cup$ all actions (events) for all instances;

$$T(s,a) \doteq \begin{cases} \{\langle I_{P_i}, \frac{1}{|\mathcal{P}^k_{\mathcal{D}}|}\rangle \mid P_i \in \mathcal{P}^k_{\mathcal{D}}\} & \text{if } s = \vdash, a = \xi \\ \{\langle a[s], 1\rangle\} & \text{if } s \neq \vdash \end{cases}$$

$$R(s,a,s') \doteq \begin{cases} 1 & \text{if } s' \text{ is a goal state} \\ -1 & \text{if } s' \text{ is a dead-end} \\ 0 & \text{otherwise.} \end{cases}$$

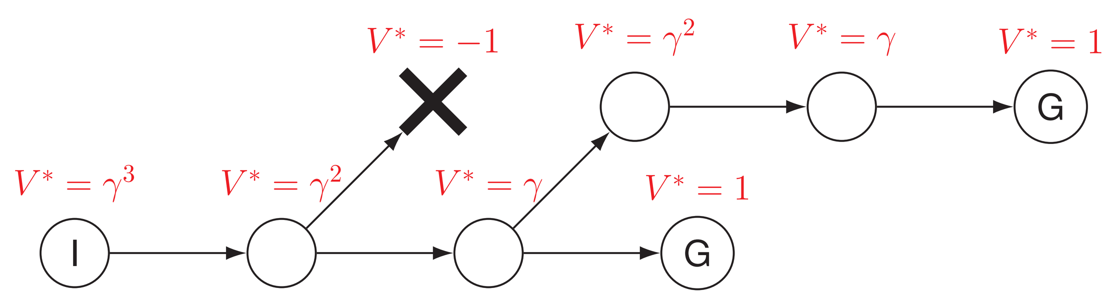


FROM THE OPTIMAL VALUE FUNCTION (V^*) TO THE OPTIMAL HEURISTIC (h^*)

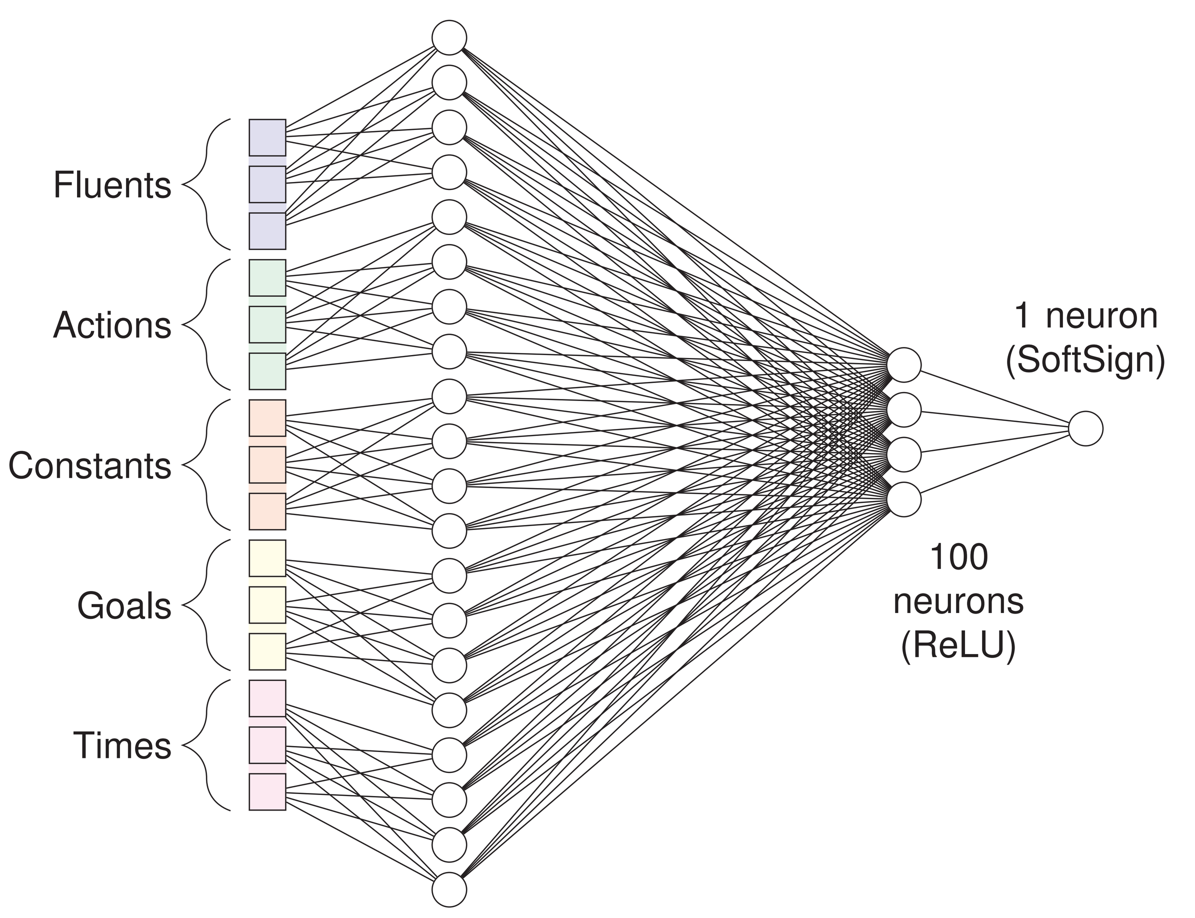
For a bounded planning problem set $\mathcal{P}^k_{\mathcal{D}}$ the following equation holds.

$$h^*_{\mathcal{P}^k_{\mathcal{D}}}(s) = \begin{cases} \log_\gamma(V^*_{\mathcal{M}_{\mathcal{P}^k_{\mathcal{D}}}}(s)) & \text{if } V^*_{\mathcal{M}_{\mathcal{P}^k_{\mathcal{D}}}}(s) > 0 \\ \infty & \text{otherwise} \end{cases}$$

Intuition

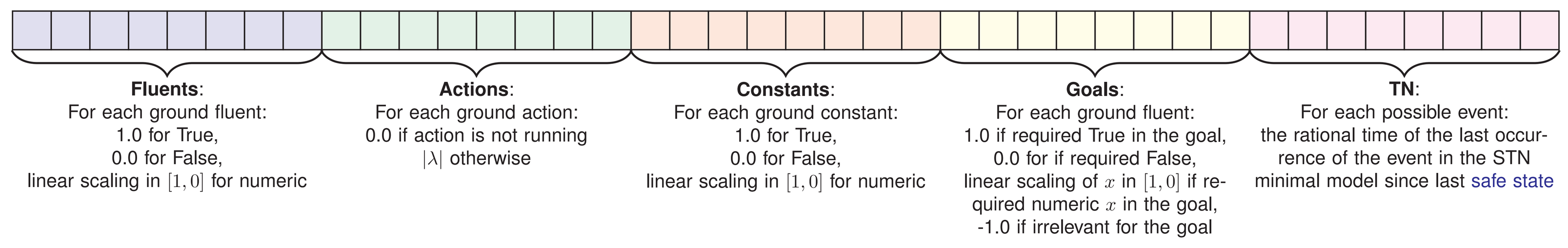


NEURAL ARCHITECTURE: PREDICTING V^*



STATE VECTORIZATION

Given a planning state s we derive a vector $\vec{s}$ in $\mathbb{R}^x$



RL-BASED HEURISTIC LEARNER

Basically, Deep-Q-Learning on $\mathcal{M}_{\mathcal{P}^k_{\mathcal{D}}}$ with some adjustments:

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

procedure RL2PLANHEURISTIC(tis, $N_{episodes}$)
  $V_{nn} \leftarrow$ INITNN( )
  $mem \leftarrow$ LIST( )
  $i2s \leftarrow \{i \to 0 \mid i \in tis\}$
  for $i \in 1, \dots, N_{episodes}$ **do**
    $\langle s, goals\rangle = inst \leftarrow$ PICKKEYINVPROPORTIONALLYTOVALUE($i2s$)
    $\langle done, solved\rangle \leftarrow \langle False, False\rangle$
    $\pi \leftarrow \langle s\rangle$
    while not $done$ **do**
      $\epsilon \leftarrow \epsilon_{max} \times e^{(\frac{\ln(\epsilon_{min}/\epsilon_{max})}{N_{episodes}} \times i)}$
      if RANDOM( ) $< \epsilon$ **then**
        $\alpha \leftarrow$ SELECTACTIONUSINGHEURISTIC(s)
      else
        $\alpha \leftarrow$ SELECTACTIONUSINGPOLICY(V_{nn}, s)
      $\langle s', done, \rho\rangle \leftarrow$ DOSTEP(π, s, α, $inst$)
      APPEND(mem, $\langle s, \rho\rangle$)
      if $\rho[\alpha] = 1$ **then**
        $solved \leftarrow True$
      APPEND(π, $\langle s'\rangle$)
      $s \leftarrow s'$
    $V_{nn} \leftarrow$ REPLAY(V_{nn}, mem)
    if $solved$ **then**
      $i2s[inst] \leftarrow i2s[inst] + 1$
  return V_{nn}

Learned heuristic

The learned heuristic h_{nn} is an approximation of h^*

$$h_{nn}(s) \doteq \begin{cases} \min(\log_\gamma(V_{nn}(\vec{s})), \Delta_h) & \text{if } V_{nn}(\vec{s}) > 0 \\ \Delta_h & \text{if } V_{nn}(\vec{s}) = 0 \\ 2\Delta_h - \min(log_\gamma(-V_{nn}(\vec{s})), \Delta_h) & \text{otherwise} \end{cases}$$

Where Δ_h is bigger than the pre-fixed cutoff length of episodes set in learning

Soundness: h_{nn} never returns ∞ because learning can be imperfect and we do not want unsound pruning.

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

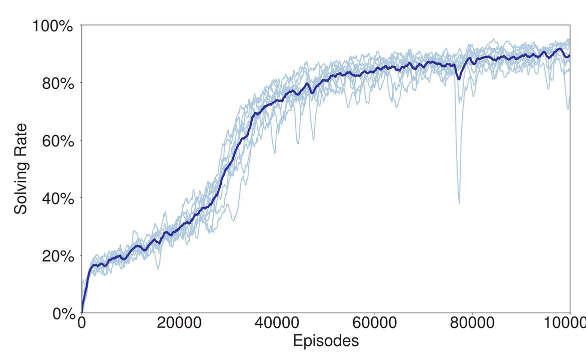
- **TAMER** (h_{add}): our fully-symbolic state-of-the-art planning
- π_{nn}: The learned RL policy executed without backtracking
- **TAMER** (h_{nn}): our planner equipped with the learned heuristic h_{nn}

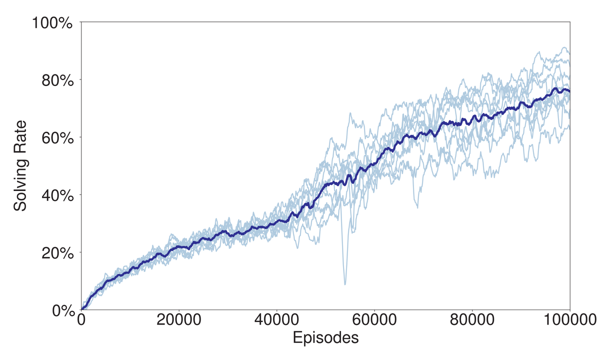
Results

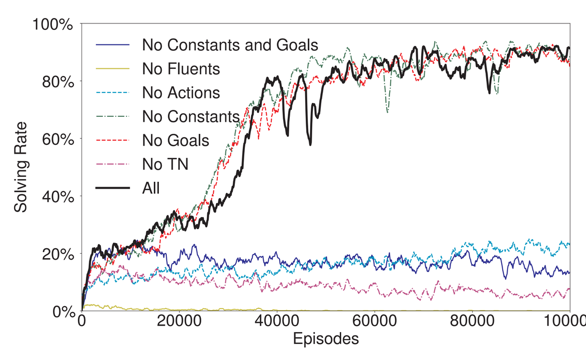
10-fold cross-validation and sensitivity analysis over 100K RL episodes

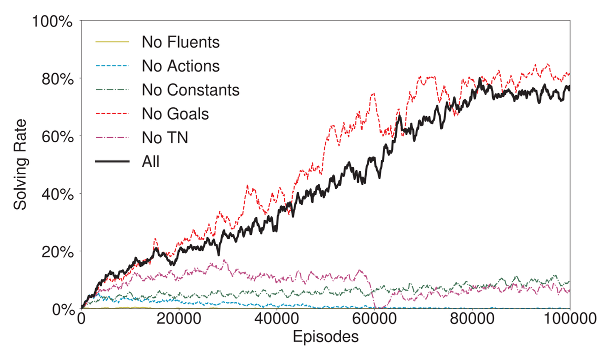
MaJSP							
fold (size: 77)	TAMER (h_{add}) solved	TAMER (h_{add}) avg plan size	# episodes	π_{nn} solved	π_{nn} avg plan size	TAMER (h_{nn}) solved	TAMER (h_{nn}) avg plan size
1	52	14	50k 100k	66 71	25 22	**73** **73**	18 18
2	58	14	50k 100k	70 70	22 19	**75** 72	17 17
3	58	14	50k 100k	70 73	21 19	73 **75**	17 17
4	57	13	50k 100k	66 68	21 20	72 **76**	17 17
5	55	15	50k 100k	66 69	25 21	**75** 69	19 19
6	60	14	50k 100k	66 69	23 17	76 **77**	17 17
7	54	14	50k 100k	68 75	21 21	**76** 73	18 18
8	57	14	50k 100k	61 **73**	23 20	**73** 69	18 18
9	57	14	50k 100k	71 66	25 21	**74** 70	18 18
10	52	14	50k 100k	72 65	21 22	**77** 54	19 16
all	560	14	50k 100k	676 699	23 20	**744** 708	18 18

Kitting							
fold (size: 109)	TAMER (h_{add}) solved	TAMER (h_{add}) avg plan size	# episodes	π_{nn} solved	π_{nn} avg plan size	TAMER (h_{nn}) solved	TAMER (h_{nn}) avg plan size
1	44	15	50k 100k	66 97	18 21	99 **107**	21 21
2	35	15	50k 100k	66 82	21 21	95 **97**	22 21
3	38	15	50k 100k	55 97	18 20	83 **99**	20 20
4	45	15	50k 100k	68 88	20 22	98 **100**	19 21
5	47	15	50k 100k	85 88	19 19	**101** **101**	19 19
6	38	15	50k 100k	53 78	20 22	85 **108**	20 23
7	30	15	50k 100k	44 90	18 24	75 **106**	19 23
8	42	15	50k 100k	65 95	18 21	95 **104**	20 21
9	36	15	50k 100k	44 89	15 22	70 **91**	17 20
10	40	14	50k 100k	71 92	19 21	95 **102**	21 21
all	395	15	50k 100k	617 896	19 21	896 **1015**	20 21









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