Reinforcement Learning for Classical Planning: Viewing Heuristics as Dense Reward Generators

Clement Gehring*, Masataro Asai*, Rohan Chitnis, Tom Silver, Leslie Pack Kaelbling, Shirin Sohrabi, Michael Katz
Corresponding Author: clement@gehring.io

MIT-IBM Watson AI Lab

Abstract

Recent advances in reinforcement learning (RL) have led to a growing interest in applying RL to classical planning domains or applying classical planning methods to some complex RL domains. However, the long-horizon goal-based problems found in classical planning lead to sparse rewards for RL, making direct application inefficient. In this paper, we propose to leverage domain-independent heuristic functions commonly used in the classical planning literature to improve the sample efficiency of RL. These classical heuristics act as dense reward generators to alleviate the sparse-rewards issue and enable our RL agent to learn domain-specific value functions as residuals on these heuristics, making learning easier. Correct application of this technique requires consolidating the discounted metric used in RL and the non-discounted metric used in heuristics. We implement the value functions using Neural Logic Machines, a neural network architecture designed for grounded first-order logic inputs. We demonstrate on several classical planning domains that using classical heuristics for RL allows for good sample efficiency compared to sparse-reward RL. We further show that our learned value functions generalize to novel problem instances in the same domain. The source code is available at github.com/ibm/pddlrl

Coverage Results

domain (total)	Baselines h^{blind}	Baselines h^{add}	Baselines h^{FF}	Ours (mean (max) of 20 runs) H^{blind}	Ours H^{add}	Ours H^{FF}	GBFS -HGN [4]	GBFS -H [3]
blocks (250)	0	126	87	**73.1 (94)**	**186.6 (229)**	**104 (114)**	3	208
ferry (250)	0	138	250	**40.4 (62)**	**233.9 (249)**	250 (250)	27	240
gripper (250)	0	250	250	**47.5 (85)**	250 (250)	250 (250)	63	139
logistics (250)	0	106	243	0 (0)	54.1 6.8(**115**)	79.8 12.9(189)	-	0
miconic (442)	171	442	442	143.3 (**246**)	442 (442)	440.8 (442)	-	0
parking (700)	0	607	700	**0.9 (3)**	**619 (689)**	696.9 (700)	-	333
satellite (250)	0	249	222	**26.5 (99)**	233.3 (250)	163.2 (205)	-	9
visitall (252)	252	252	252	207.6 (238)	251.9 (252)	252 (252)	-	101

Table 1: A comparison of the number of test instances solved. **Note:** the method labelled GBFS-HGN [4] was designed in the context of A^*but was used with GBFS in our work. Therefore, these results are not indicative of the capabilities of this method when using other search algorithms.

Our Goal

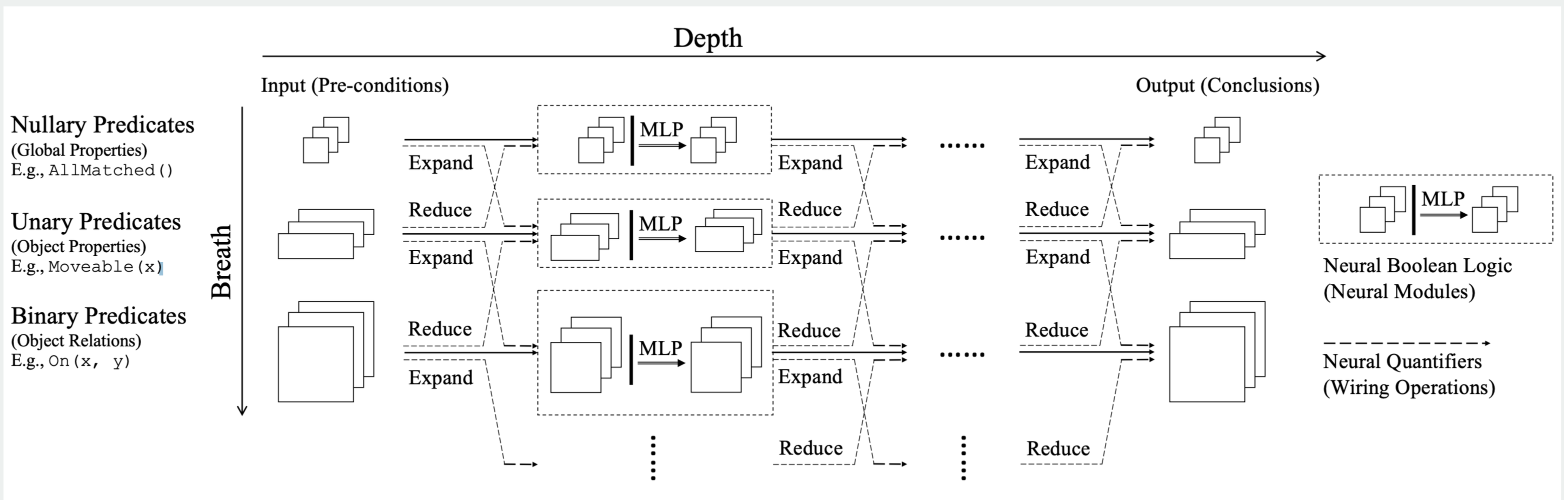
We want to combine RL and domain **independent** heuristics to efficiently learn domain **dependent heuristics**. By learning a specialized heuristic, we hope to plan more efficiently for various problem instances of targeted domain.

Contributions

1. Generalizing over problem instances by learning **goal** and **problem** conditioned heuristic using neural logic machines [1], and
2. using potential-based reward shaping to efficiently learn a domain **dependent** heuristic as a correction to a domain **independent** heuristic.

Encoding Domain Dependent Heuristics

We encode the grounded **state** and **goal** predicates with binary N-d arrays. We represent the heuristic with a **neural logic machine** [1].



Potential-Based Reward Shaping

Using a potential function h, define a **new reward function**[2]:

$$\hat{r}(s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) + h(s_t) - \gamma h(s_{t+1})$$

Theoretically nice approach:

- Optimal policies under the shaped rewards are optimal under the original rewards (and vice versa)

Evaluation Methodology

- Train on **small** instances (fast and easy), e.g., 2-6 blocks.
- Evaluate on **unseen** and **larger** instances, e.g., 10-50 blocks.
- Evaluate quality of the learned heuristic based on the number of node evaluations.
- Limited to no more than **100,000** node evaluations using **greedy best-first search (GBFS)**.

Improving Baseline Heuristics

domain (total)	Baselines h^{blind}	Baselines h^{add}	Baselines h^{FF}	Ours H^{blind}	Ours H^{add}	Ours H^{FF}
blocks (250)	0	5	9	94	224	109
ferry (250)	0	0	0	62	249	250
gripper (250)	0	0	50	85	250	200
logistics (250)	0	14	77	0	108	167
miconic (442)	17	61	0	234	381	442
parking (700)	0	105	173	2	508	484
satellite (250)	0	110	63	93	115	155
visitall (252)	60	36	59	192	216	192

Table 2: The number of test instances solved by the learned heuristic but not the baseline and vice versa.

Acknowledgements

We thank Shen, Trevizan, and Thiebaux and Rivlin, Hazan, and Karpas for kindly providing us their code.

References

[1] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. “Neural Logic Machines”. In: *ICLR*. 2018.

[2] Andrew Y Ng, Daishi Harada, and Stuart J Russell. “Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping”. In: *ICML*. 1999.

[3] Or Rivlin, Tamir Hazan, and Erez Karpas. “Generalized planning with deep reinforcement learning”. In: *arXiv preprint arXiv:2005.02305* (2020).

[4] William Shen, Felipe Trevizan, and Sylvie Thiébaux. “Learning Domain-Independent Planning Heuristics with Hypergraph Networks”. In: vol. 30. 2020, pp. 574–584.