Searching for Markovian Subproblems to Address Partially Observable Reinforcement Learning

Rodrigo Toro Icarte[†‡] Ethan Waldie[†] Toryn Q. Klassen[†] Richard Valenzano[§] Margarita P. Castro[II] Sheila A. McIlraith[†‡]

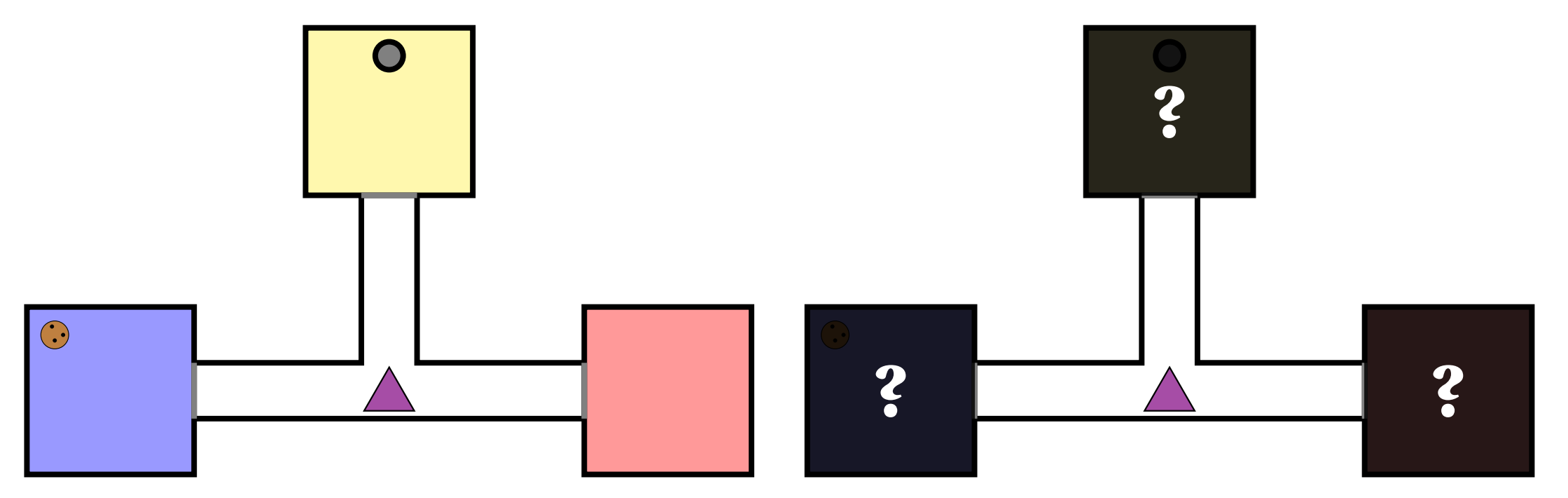
[†]Dept of Computer Science, University of Toronto [II]Dept of Mechanical and Industrial Engineering, University of Toronto [‡]Vector Institute [§]Element AI

[†]{rntoro, ewaldie96, toryn, sheila}@cs.toronto.edu, [II]mpcastro@mie.utoronto.ca, [§]rick.valenzano@elementai.com

Abstract

Reward Machines (RMs), originally proposed for specifying problems in RL, provide a structured, automata-based representation of a reward function that allows an agent to decompose problems into subproblems that can be efficiently learned using off-policy learning. Here we show that RMs can be learned from experience, instead of being specified by the user, and that the resulting problem decomposition can be used to effectively solve partially observable RL problems. We pose the task of learning RMs as a discrete optimization problem where the objective is to find an RM that decomposes the problem into a set of subproblems such that the combination of their optimal memoryless policies is an optimal policy for the original problem. We show the effectiveness of this approach on three partially observable domains, where it significantly outperforms A3C, PPO, and ACER.

The Cookie Domain



Task: Eat as many cookies as possible before the timeout.

- The agent (▲) can move in the four cardinal directions.
- The button (●) causes a cookie (🍪) to randomly appear.
- The cookie might appear in the red (🟥) or blue (🟦) room.
- The agent receives a +1 reward for eating a cookie.
- Pressing ● before reaching 🍪 will randomly move it.
- There is no cookie at the beginning of the episode.
- The agent can only see what it is in the current room.

Note: A3C, PPO, and ACER with LSTMs could not solve this task.

Motivation

Why is the cookie domain so challenging for Deep RL agents?

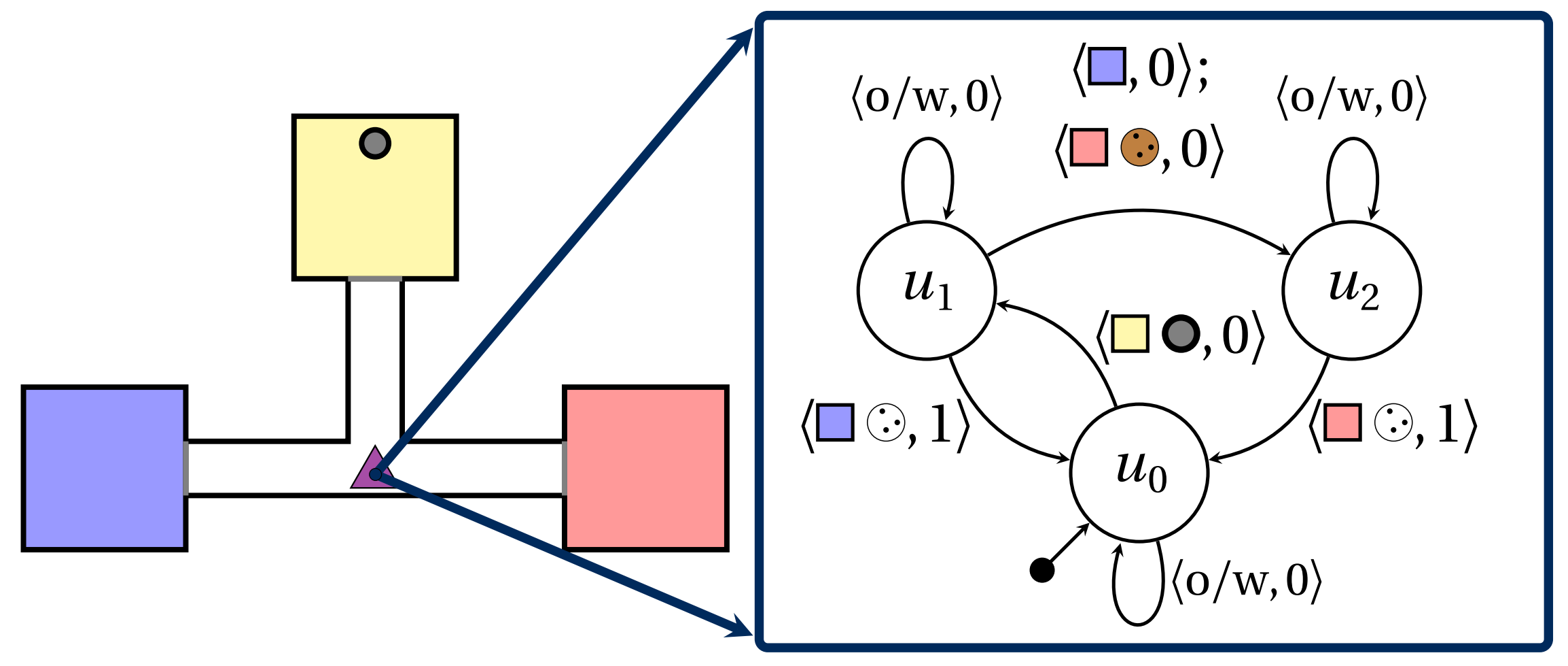
- It requires understanding of long-term dependencies.
- Solving this problems requires two levels of reasoning.
  Low-level: learning policies for navigating the map.
  High-level: reasoning about memory at the level of objects.

Idea: Let's give the agent a set of high-level binary classifiers to detect—from the current observation—the color of the room (🟥, ⬜, 🟦, 🟨), whether it sees a cookie (🍪), and whether it just ate a cookie (◌) or pressed the button (●).

Can we do better now?

Reward Machines (RMs)

RMs are automata-based representations of a reward function that allow RL agents to learn policies faster [1].



RMs start at state u_0 and moves according to the edges. Each edge has an `if-like` condition over high-level detectors and a reward.

Learning policies given an RM:

- **Standard**: Learn a policy $\pi(a|o,u)$ that selects the action a considering the current observation o and RM state u.
- **QRM**: Learn one policy per RM state and improve each policy in parallel using off-policy learning.

Main contributions

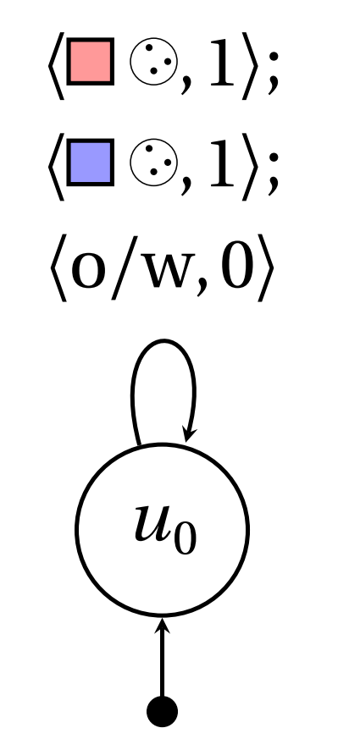
1. First approach for learning RMs from experience.
2. Extending RMs to work under partial observability.
3. Developing a theory for 1 and 2.

Learning Reward Machines

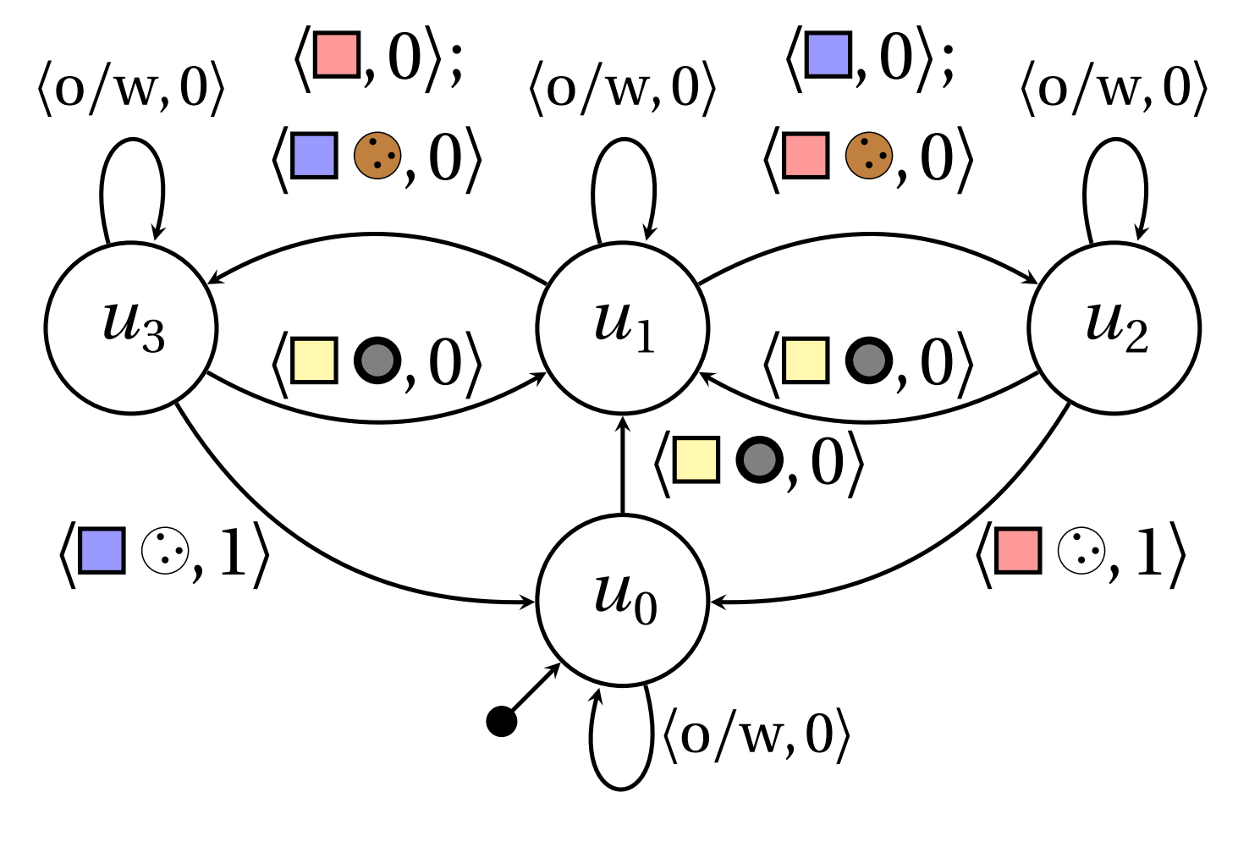
Problem setting: Find a policy that maximizes the collected external reward given by a partially observable environment $\mathcal{E}$.

Assumptions: The agent has access to a set of high-level binary classifiers (e.g., $\mathcal{P}=\{$🟥, ⬜, 🟦, 🟨, 🍪, ◌, ●$\}$ for the cookie domain).

Key insight: Learn an RM such that its internal state can be effectively used as external memory by the agent to solve the task.



(a) Naive RM.



(b) Perfect RM.

Example: The states from the "*perfect RM*" keep track of the possible location of the cookies—which is relevant for solving the task.

Perfect Reward Machines

Perfect RMs make the environment Markovian w.r.t. $O \times U$, i.e.:

$$\Pr(o_{t+1}, r_{t+1}|o_0, a_0, \ldots, o_t, a_t) = \Pr(o_{t+1}, r_{t+1}|o_t, u_t, a_t)$$

for every possible trace $o_0, a_0, \ldots, o_t, a_t$ generated by any policy.

Properties: Given an environment $\mathcal{E}$: (i) If the set of belief states of $\mathcal{E}$ is finite, then there exists a perfect RM for $\mathcal{E}$, and (ii) Optimal policies over $O \times U$ for perfect RMs are also optimal for $\mathcal{E}$.

How to Learn Perfect Reward Machines

Ideally: For every possible RM; collect a set of traces, fit a predictive model for $\Pr(o', r|o, u, a)$, and pick the RM that makes better predictions. **This is prohibitively expensive**.

Alternative: Optimize over the necessary condition that perfect RMs must correctly predict—at least—what is possible and impossible in the environment at the level of the binary classifiers.

time

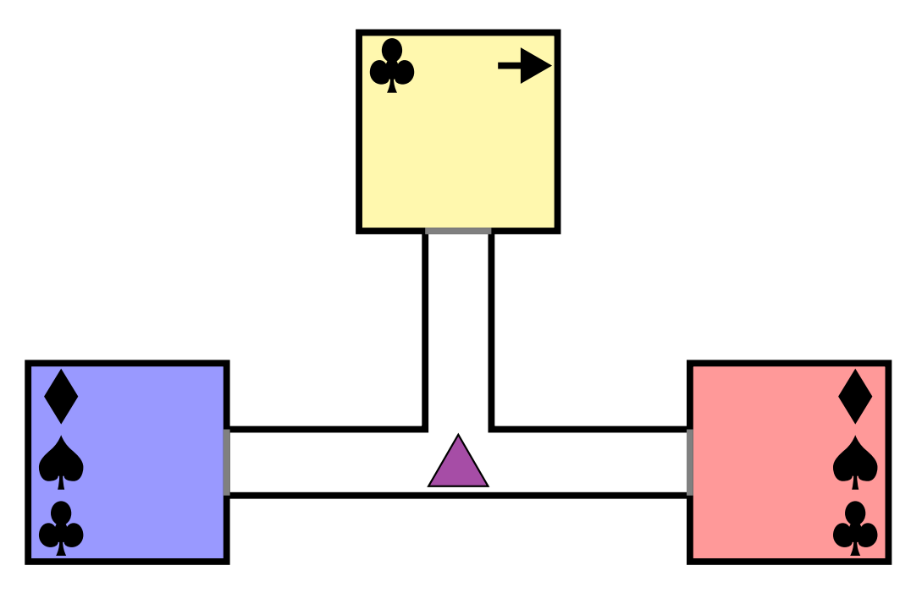
Trace	Naive RM	Perfect RM
⬜	u_0: (🟨),(🟥),(🟥🍪),(🟦),(🟦🍪)	u_0: (🟨),(🟥),(🟦)
🟨	u_0: (🟨●),(⬜)	u_0: (🟨●),(⬜)
🟨●	u_0: (🟨)	$u_0 \rightarrow u_1$: (🟨)
🟨	u_0: (🟨●),(⬜)	u_1: (🟨●),(⬜)
⬜	u_0: (🟨),(🟥),(🟥🍪),(🟦),(🟦🍪)	u_1: (🟨),(🟥),(🟥🍪),(🟦),(🟦🍪)
🟦	u_0: (⬜)	$u_1 \rightarrow u_2$: (⬜)
⬜	u_0: (🟨),(🟥),(🟥🍪),(🟦),(🟦🍪)	u_2: (🟨),(🟥🍪),(🟦)
🟥🍪	u_0: (⬜),(🟥◌)	u_2: (⬜),(🟥◌)
🟥◌	u_0: (🟥)	$u_2 \rightarrow u_0$: (🟥)
🟥	u_0: (⬜)	u_0: (⬜)
⬜	u_0: (🟨),(🟥),(🟥🍪),(🟦),(🟦🍪)	u_0: (🟨),(🟥),(🟦)
score:	30	**24**

In the paper: We formalized the previous idea as a combinatorial optimization problem and solved it using **Tabu search**.

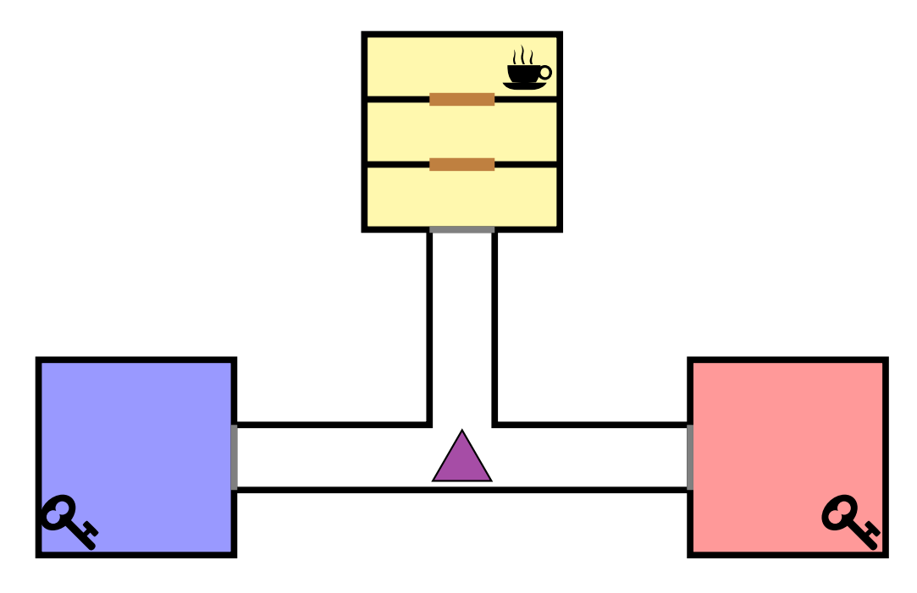
Learning RMs and Policies

1. Collect a training set T composed of random traces.
2. Learn an RM using T.
3. Learn policies for the learned RM using RL.
4. If the RM predictions are incorrect, augment T and goto 2.

Experiments



(a) Symbol domain.



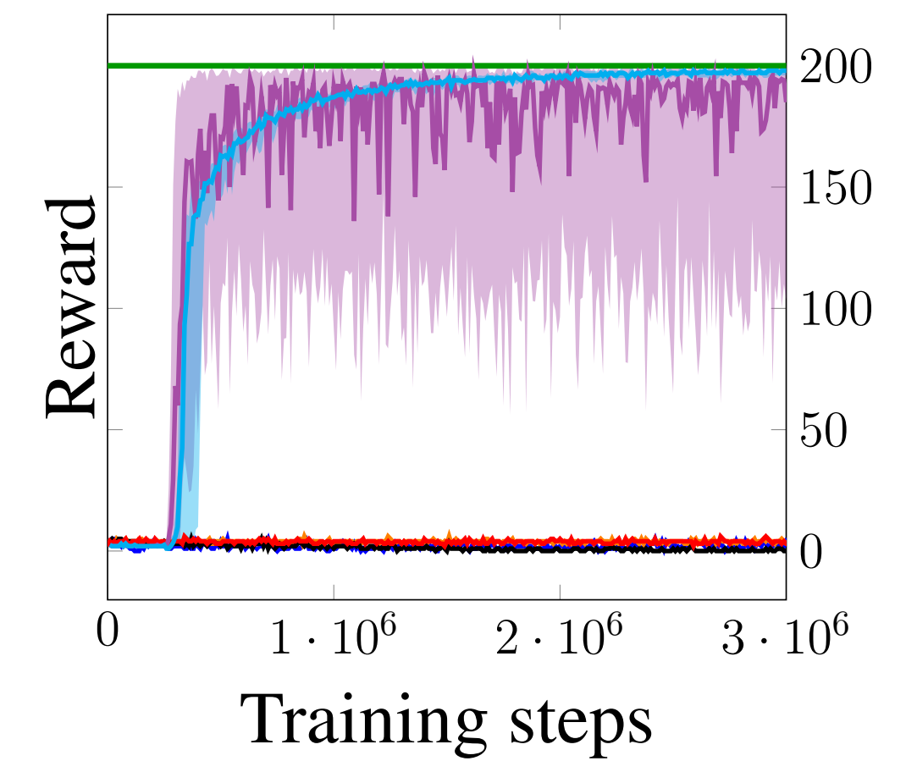
(b) 2-keys domain.

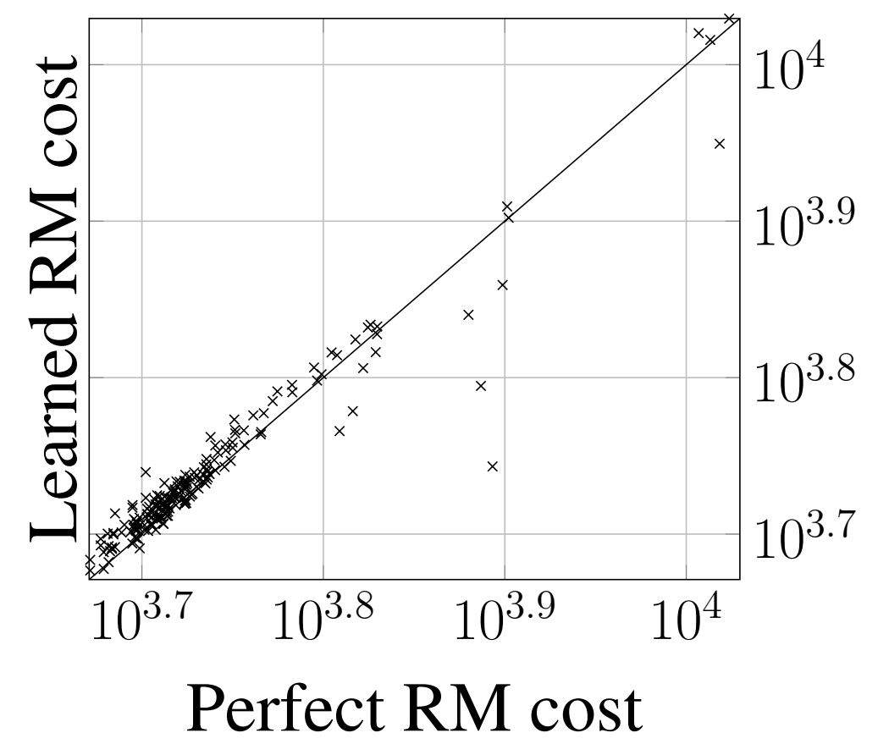
Approaches

LRM + DQRM (ours) — DDQN — A3C — Optimal
LRM + DDQN (ours) — ACER — PPO

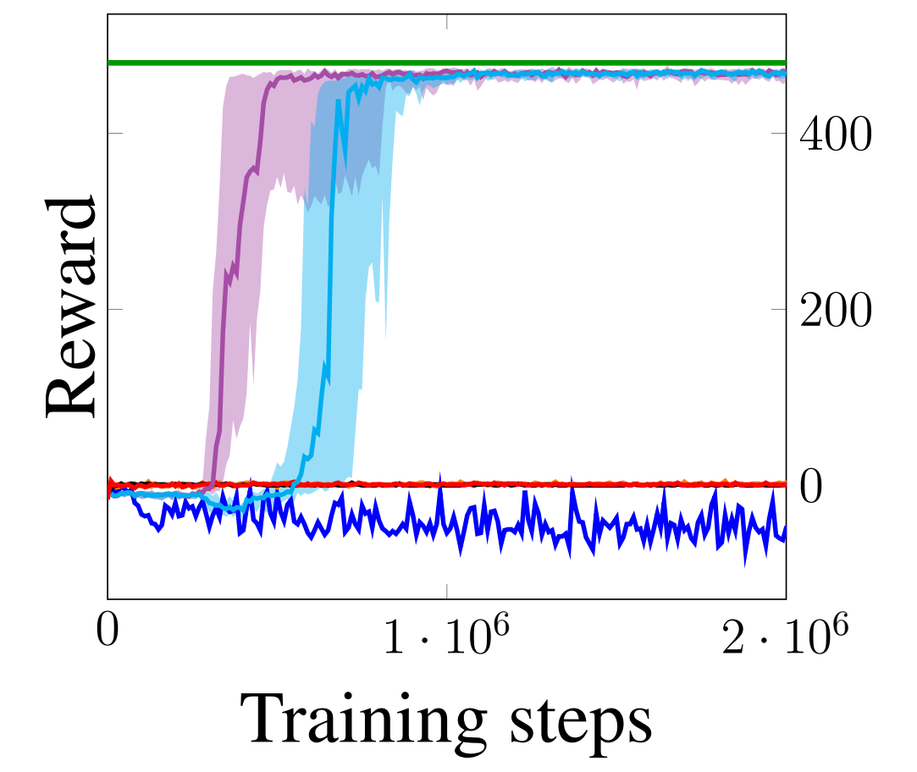
Note: The binary classifiers were used by all the approaches.

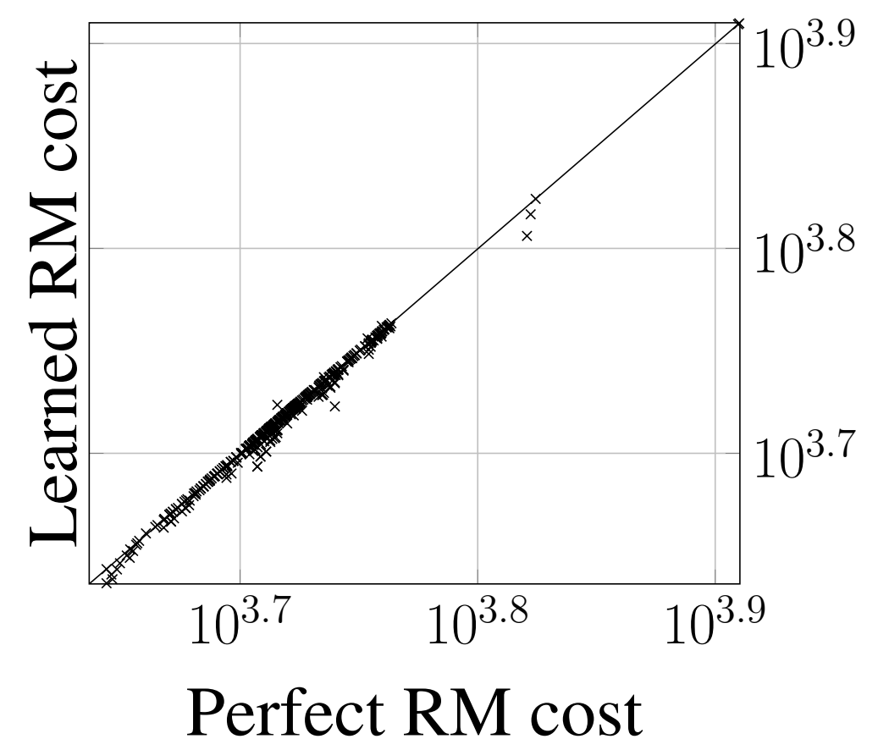
Cookie Domain Results



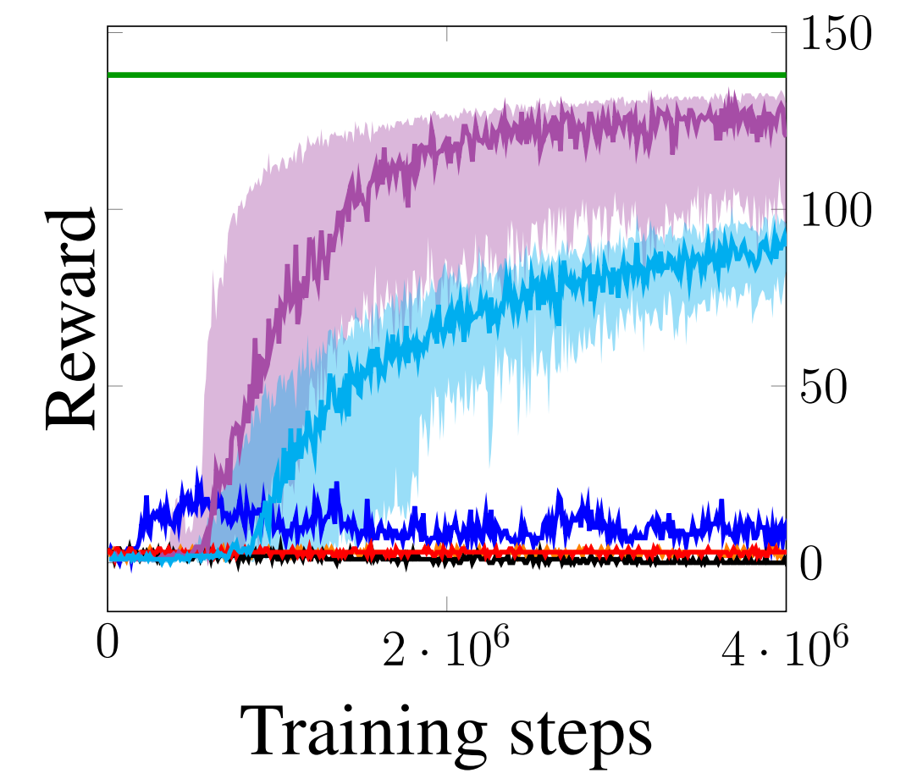


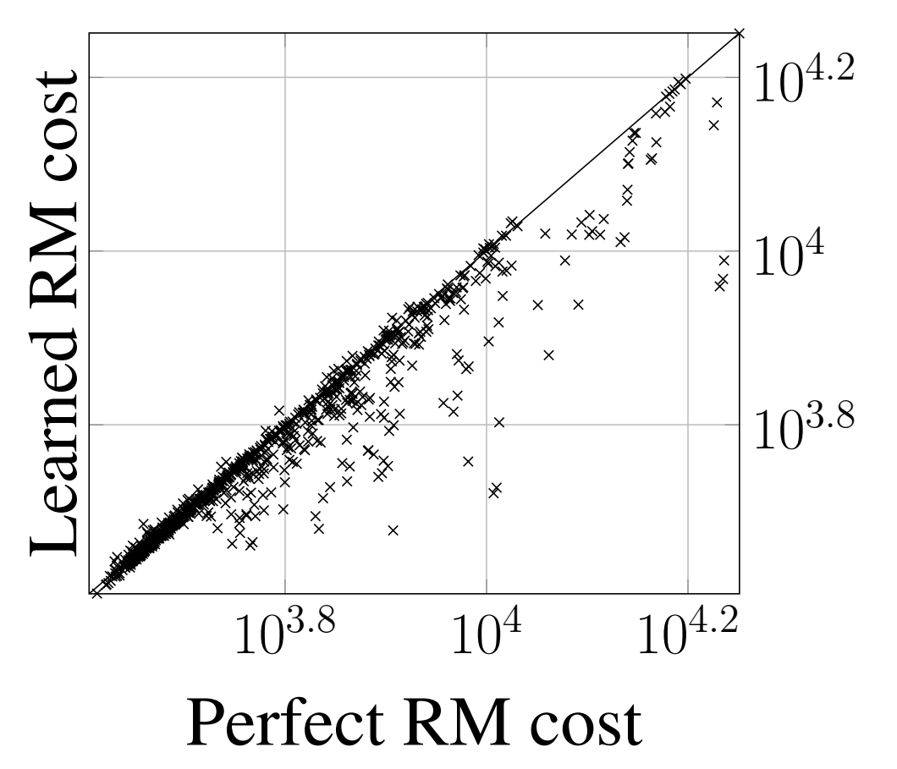
Symbol Domain Results





2 Keys Domain Results





Summary of the Results

- Our LRM approaches outperform the baselines.
- DQRM pays off in domains with sparse rewards.
- LRM works because Tabu search finds high-quality RMs.

Discussion

This work represents a building block for creating RL agents that can solve cognitively challenging partially observable tasks. RM learning provided the agent with memory, but more importantly with discrete reasoning capabilities that operated at a higher level of abstraction (i.e., Tabu search) while leveraging deep RL's ability to learn policies from low-level inputs (i.e., DQRM).

References

[1] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML18*.