Elective AI&R : Reasoning Agents

A.A. 2019/20

SAPIENZA
UNIVERSITÀ DI ROMA

Learning Reward Machines for Partially Observable Reinforcement Learning

Riccardo Gozzovelli *Mario Vetrini* *Francesco Caputo*

REINFORCEMENT LEARNING

SETTING

Markovian Decision Process (MDP), $M = \{S, A, r, p, \gamma\}$

PROBLEM

Maximize the future expected discounted reward for every state in S

SOLUTION

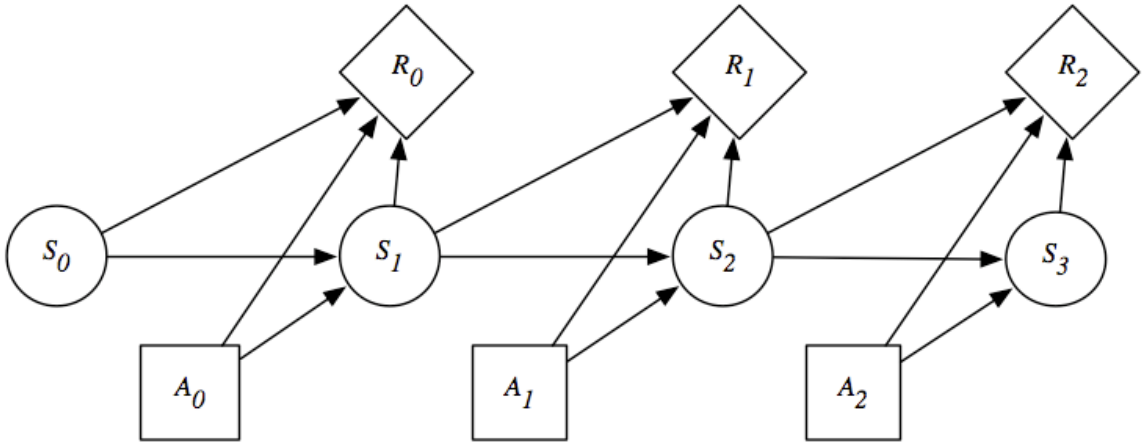
Optimal policy: $\pi^*(a_t|s_t)$

ALGHORITMS

Q-Learning, SARSA, A3C, PPO

DEEP REINFORCEMENT LEARNING

RL algorithms combined with DL principles.



WHY LEARN RMs

SETTING

Partially Observable Markovian Decision Process (POMDP), $M = \{S, O, A, r, p, w, \gamma\}$

PROBLEM

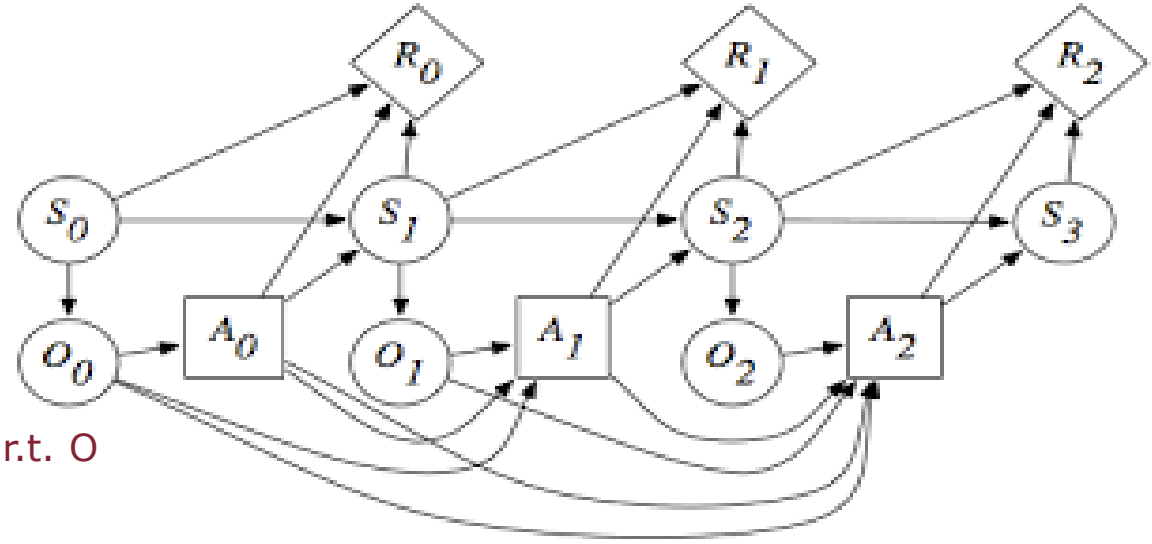
Maximize the future expected discounted reward for every state in S

SOLUTION

Optimal policy: $\pi^*(a_t|s_t)$

ALGHORITMS

Transition probabilities and reward function not necessarily Markovian w.r.t. O



DEEP REINFORCEMENT LEARNING

RL algorithms combined with DL principles.

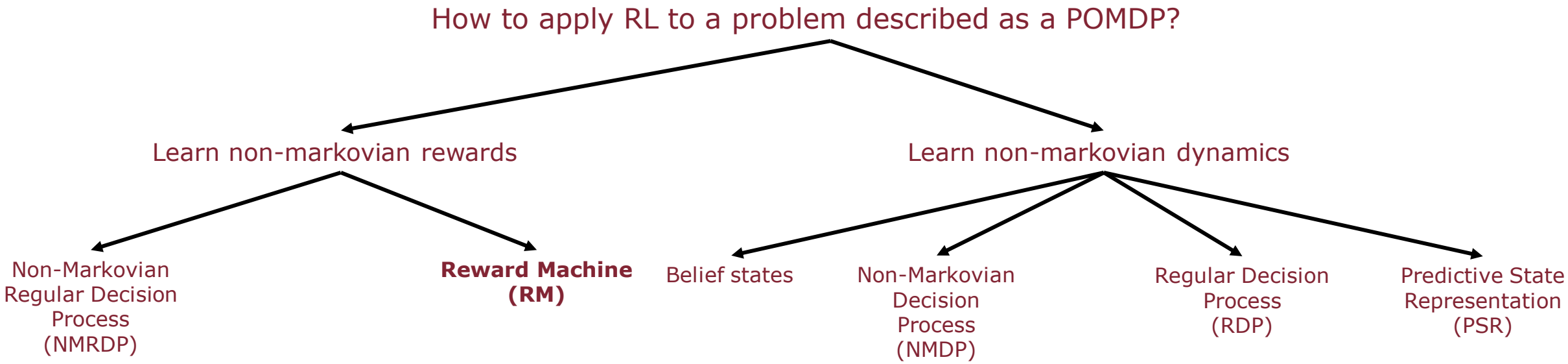
PARTIALLY OBSERVABLE DOMAINS

Many real-world applications have partially observable domains

A partially observable domain can have:

- *non-markovian dynamics* or
- *non-markovian rewards* or
- both

RL & POMDPs



We learn policies and automata!

REWARD MACHINEs

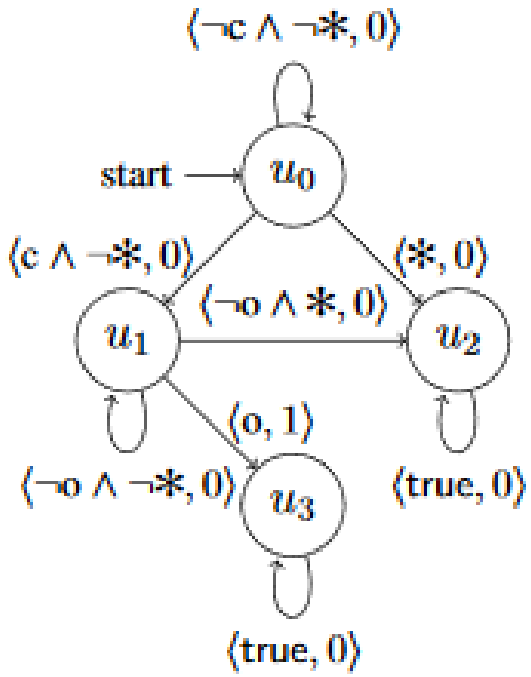
Structured Finite State Automata (FSA) representation.

Expose the reward function and speed up learning.

Provide «memory» to a learning agent.

Variant of Q-Learning algorithm used (QRM).

Initially handcrafted and suitable only for fully observable environments.



RMs AND POMDPs

RMs defined over set of propositional symbols *P* (binary properties).

Elements of *P* are detected by a labelling function $L : O_\phi \times A_\phi \times O \rightarrow 2^P$

L assigns truth values to symbols in *P* given an experience $e = (o, a, o')$

A reward machine is a tuple $R_p = \{U, u_0, \delta_u, \delta_r\}$

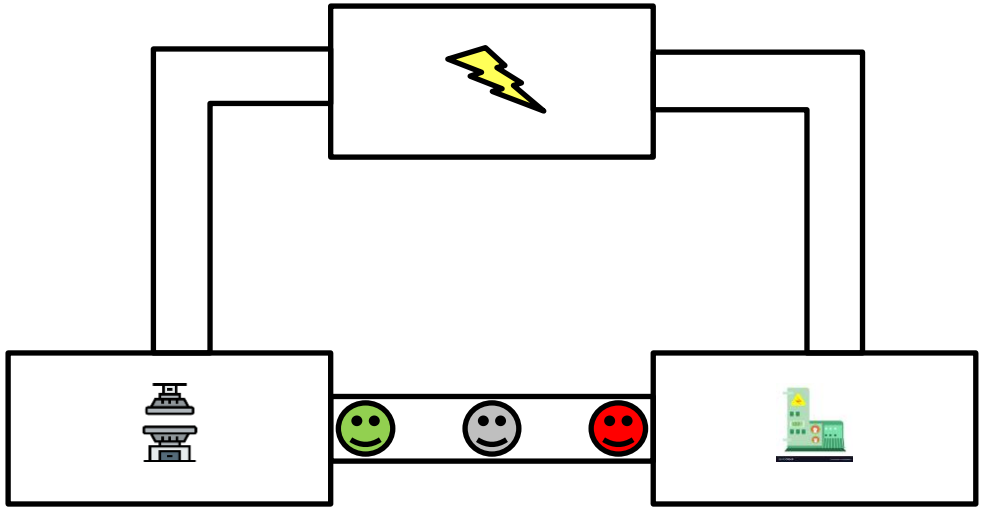
state-transition function

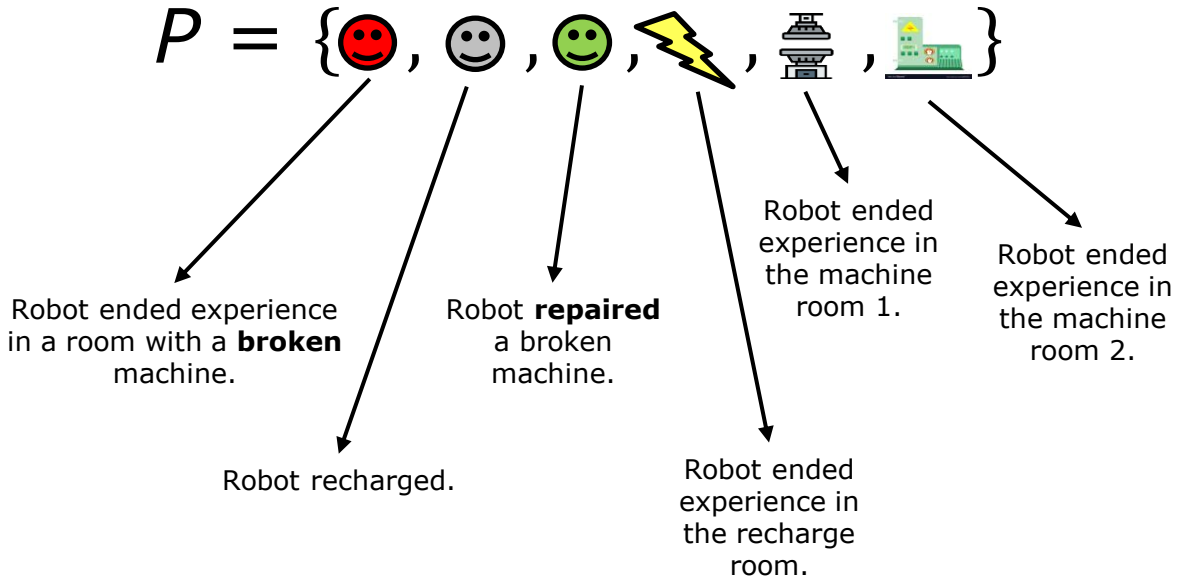
$\delta_u : U \times 2^P \rightarrow U$

reward-transition function

$\delta_u : U \times 2^P \rightarrow R$

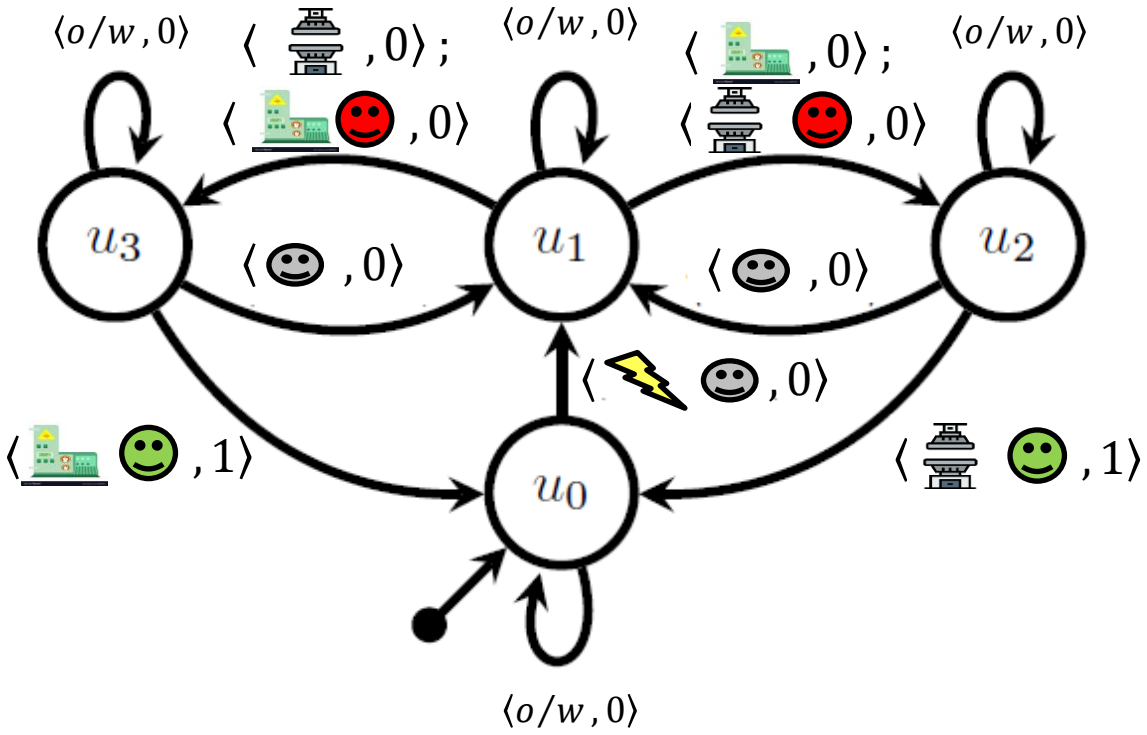
RM – MACHINE MAINTENANCE EXAMPLE





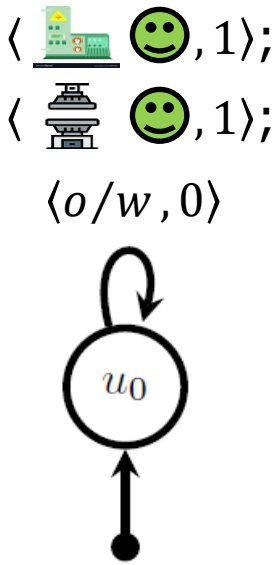
Assumptions: only one machine is broken, the robot can only repair it if it's charged

RM – MACHINE MAINTENANCE EXAMPLE



- Each RM starts in the initial state u0
- Edge label provides a visual representation of δ_u and δ_r
- Multiple labels separated by a semicolon used to describe different conditions for transitioning between the RM states
- The label "otherwise" on an edge means that that transition will be made if none of the other transitions from *u* can be taken

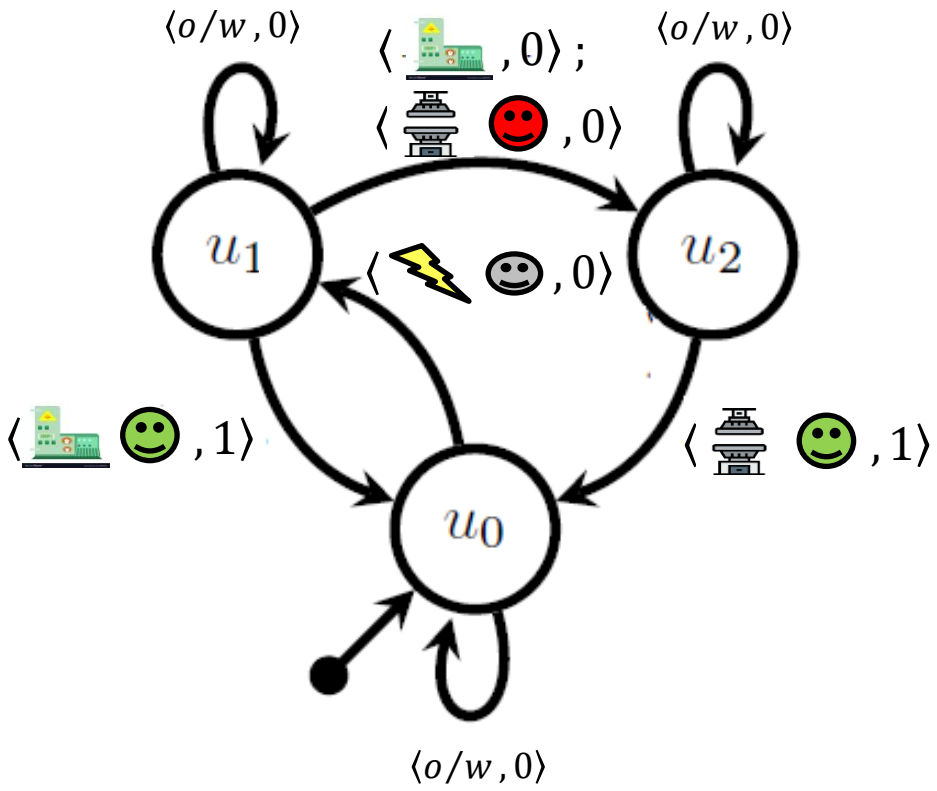
SEARCHING FOR AN RM – CASE 1



NAIVE RM

- Learning the smallest RM that correctly mimics the external reward signal given by the environment
- This naive RM correctly predicts reward in the domain but provides no memory in support of solving the task

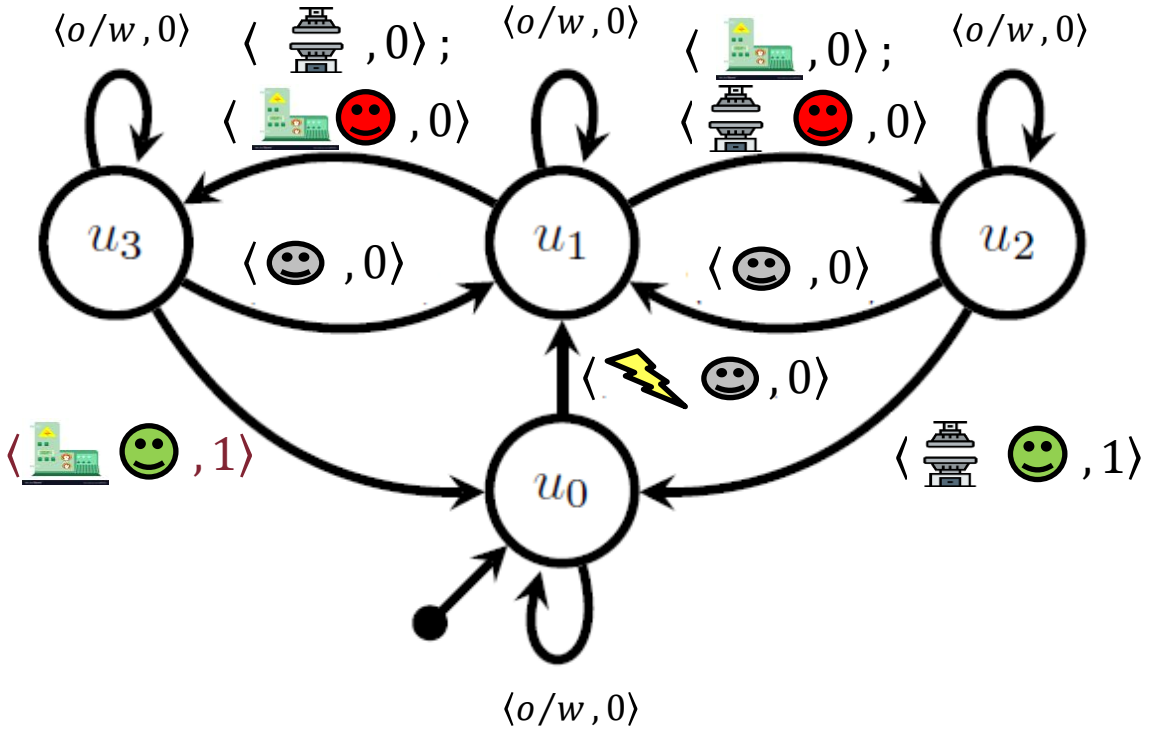
SEARCHING FOR AN RM – CASE 2



OPTIMAL RM

- Looking for the RM whose optimal policy receives the most reward
- Requires computing optimal policies in order to compare the relative quality of RMs, which seems prohibitively expensive

SEARCHING FOR AN RM - CASE 3



PERFECT RM

- Learning the RM that remembers sufficient information about the history to make accurate Markovian predictions about the next observation (w.r.t. O x U)
- Since keeping track of more information will not result in better predictions, this RM is perfect

PERFECT RM - DEFINITIONS AND THEOREMS

DEFINITION: An RM Rp is considered perfect for a POMDP Po with respect to a labelling function *L* if and only if for every trace generated by any policy over Po, the following holds:

$$\Pr(o_{t+1}, r_t | o_0, a_0, \ldots, o_t, a_t) = \Pr(o_{t+1}, r_t | o_t, x_t, a_t)$$
$$\text{where } x_0 = u_0 \text{ and } x_t = \delta_u(x_{t-1}, L(o_{t-1}, a_{t-1}, o_t))$$

THEOREM: Given any POMDP Po with a finite reachable belief space, there will always exists at least one perfect RM Rp for Po with respect to some labelling function L.

THEOREM: Let Rp be a perfect RM for a POMDP PO w.r.t. a labelling function L, then any optimal policy for Rp w.r.t. the environmental reward is also optimal for Po.

PERFECT RM - LEARNING

- Learning a perfect RM from traces, assuming one exists w.r.t. the given labelling function *L*
- One solution can be to fit a predictive model for the previously probability and picking the RM that makes better predictions but it's very expensive
- Alternative that focuses on a necessary condition for a perfect RM: the RM must predict what is possible and impossible in the environment at the abstract level

PERFECT RM - LEARNING

- Let $\mathcal{T} = \{\mathcal{T}_0, \ldots, \mathcal{T}_n\}$ be a set of traces with $\mathcal{T}_i = (o_{i,0}, a_{i,0}, r_{i,0}, \ldots, a_{i,t_i-1}, r_{i,t_i-1}, o_{i,t_i})$

- Look for an RM $\{U, u_0, \delta_u, \delta_r\}$ that predicts $L(e_{i,t+1})$ from $L(e_{i,t})$ and $x_{i,t}$

- Model parameters:
  - $\mathcal{T}$ (set of traces)
  - P (set of propositiobal symbols)
  - L (labelling function)
  - $u_{\max}$ (maximum number of states)
  - $I = \{0, \ldots, n\}$ (the index of the traces)
  - $\mathrm{T}i = \{0, \ldots, t_1 - 1\}$ (time steps of trace $\mathcal{T}_i$)
  - $N_{u,l} \subseteq 2^{2^p}$ (set of all the next abstract observations seen from the RM state u and the abstract observations l at some point in T)

LRM OPTIMIZATION PROBLEM

$$\underset{\langle U,u_0,\delta_u,\delta_r\rangle}{\text{minimize}} \sum_{i\in I}\sum_{t\in T_i} \log(|N_{x_{i,t},L(e_{i,t})}|) \qquad \text{(LRM)}$$

$$s.t.\ \langle U,u_0,\delta_u,\delta_r\rangle \in \mathcal{R}_{\mathcal{P}} \qquad (3)$$

$$|U| \leq u_{\max} \qquad (4)$$

$$x_{i,t} \in U \qquad \forall i \in I, t \in T_i \cup \{t_i\} \qquad (5)$$

$$x_{i,0} = u_0 \qquad \forall i \in I \qquad (6)$$

$$x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) \qquad \forall i \in I, t \in T_i \qquad (7)$$

$$N_{u,l} \subseteq 2^{2^{\mathcal{P}}} \qquad \forall u \in U, l \in 2^{\mathcal{P}} \qquad (8)$$

$$L(e_{i,t+1}) \in N_{x_{i,t},L(e_{i,t})} \qquad \forall i \in I, t \in T_i \qquad (9)$$

PERFECT RM – SOLVING THE MINIMIZATION PROBLEM

- Local search was the most effective category of methods
- Tabu search guarantees convergence
- Start from a random RM and iteratively evaluates *neighbouring* RMs
- The new RM will be the one with the minimum value of the objective function
- Pruning and methods to avoid local minima also used

PERFECT RM – SIMULTANEOUS LEARNING

In order to learn both an RM and a policy:

1. Collect a training set of traces *T* generated by a policy during t_w *'warmup' steps.*
2. Use T to find an initial RM using tabu search.
3. Initialize policy π, set the initial state of RM to u_0 and set the current label for the initial observation $L(\emptyset, \emptyset, o)$.
4. Repeat until convergence:
    - 4.1 Select action *a* following $\pi\ (\mathrm{o}, u)$
    - 4.2 Get observation *o'* and reward *r*
    - 4.3 Update the RM state $u' = \delta_u\big(u, L(o, a, o')\big)$
    - 4.4 Update π by using the last experience $(\langle o, u\rangle, a, r, \langle o', u'\rangle)$
5. If in any step of 4 there is an evidence that the current RM might not be the best one, a new one is learned.

PERFECT RM – LEARNING A NEW RM

What can be an evidence that a new RM must be learned?

RM *R* is selected in order to: $\underset{\langle U, u_0, \delta_u, \delta_r \rangle}{\text{minimize}} \sum_{i \in I} \sum_{t \in T_i} \log(|N_{x_{i,t}, L(e_{i,t})}|)$

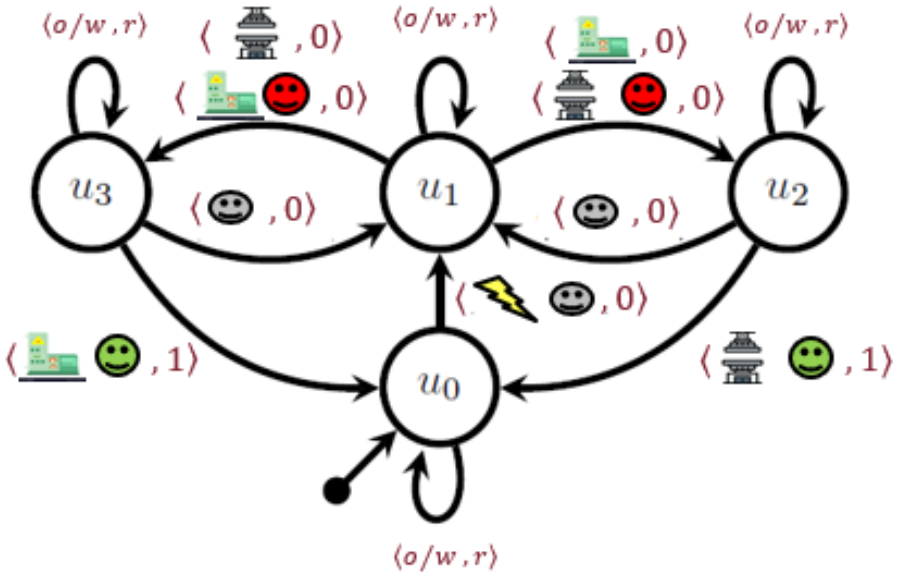
If the current abstract observation *l'* is not in $N_{u,l}$ then the current trace will increase the size of $N_{u,l}$ and the cost of *R*.

So the trace will be add to *T* and a new RM is found using the Tabu search.

If the new RM is better than *R* then a new policy must be learned for scratch.

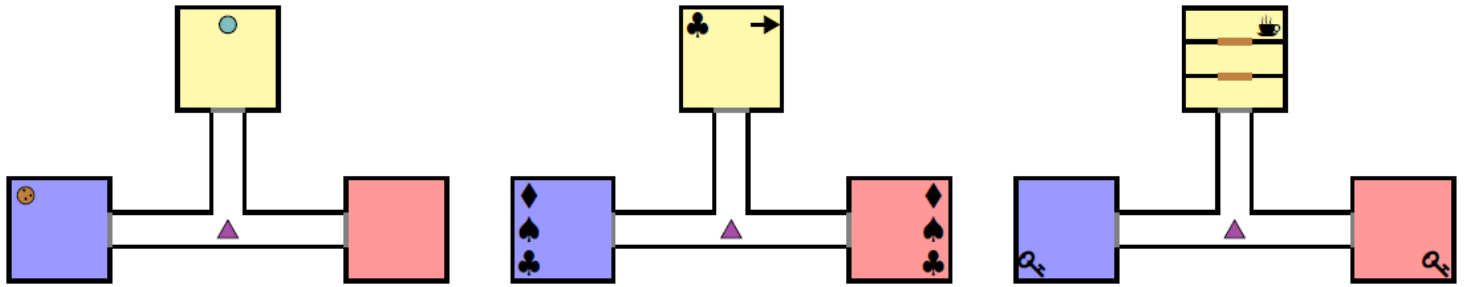
PERFECT RM – DQRM UNDER PARTIAL OBSERVABILITY

An experience e=(*o,a,o'*) can be more or less likely depending on the RM state where the experience was collected.



Update q_u using *(o,a,o')* if and only if $L(o,a,o') \in N_{u,l}$ with *l* abstract observation that generated *e*.

EVALUTION



(a) Cookie domain. (b) Symbol domain. (c) 2-keys domain.

Stochastic partially observable domains.

Tested two versions of LRM:

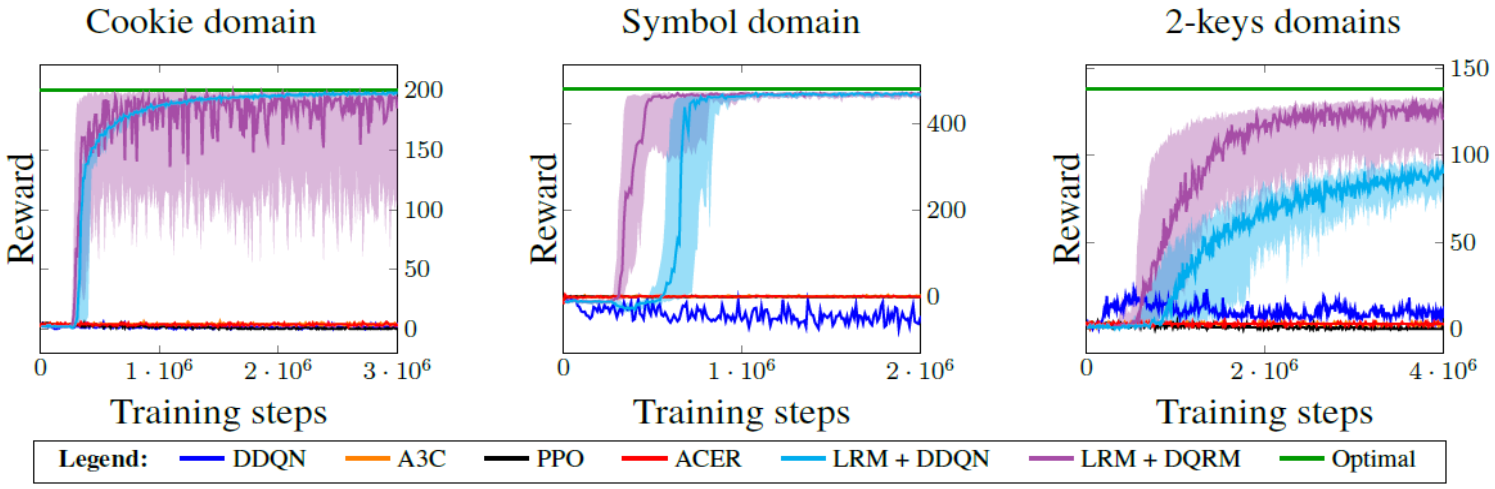
- LRM + DDQN
- LRM + DQRM

Cumulative reward every 10,000 training steps.

Median of 30 training runs per domain.

The LRM versions outperform every baseline.

LRM-DQRM faster but more unstable than LRM-DDQN.



ADVANTAGES AND LIMITATIONS

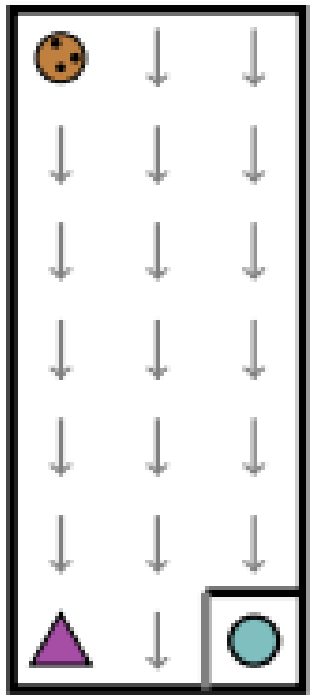
- RMs proved to be able to predict possible and impossible future observations.
- Defining high-level properties might not be trivial for more complex environments.
- Tabu search is the main bottleneck of the entire procedure.
- RMs might ignore (relevant) low-level informations.
- Unclear how to handle noise over the L detectors and how to apply transfer learning from previously learned policies when a new RM is learned.

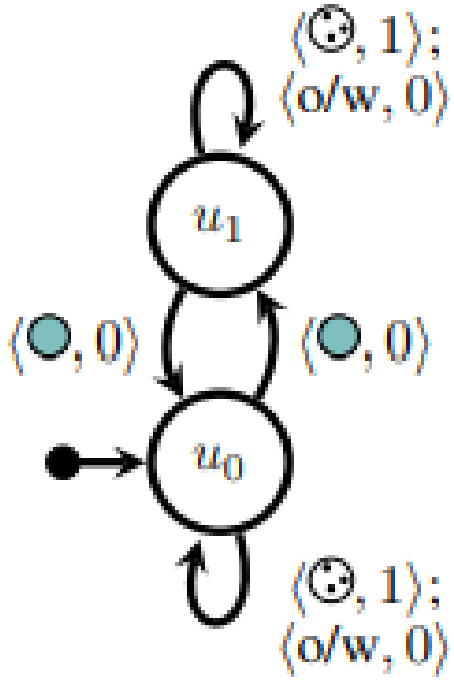
RM LIMITATION - EXAMPLE

Given $\mathcal{P} = \{$  $\}$ a perfect RM is very simple.

LRM might not find it because the button changes only the low level probability and nothing on the abstract level.

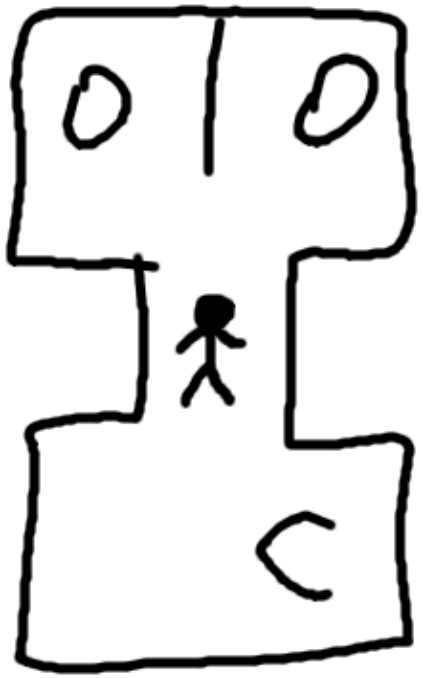
The heuristic in QRM would not work since the experience with the force on will be used to learn a policy in both RM states.

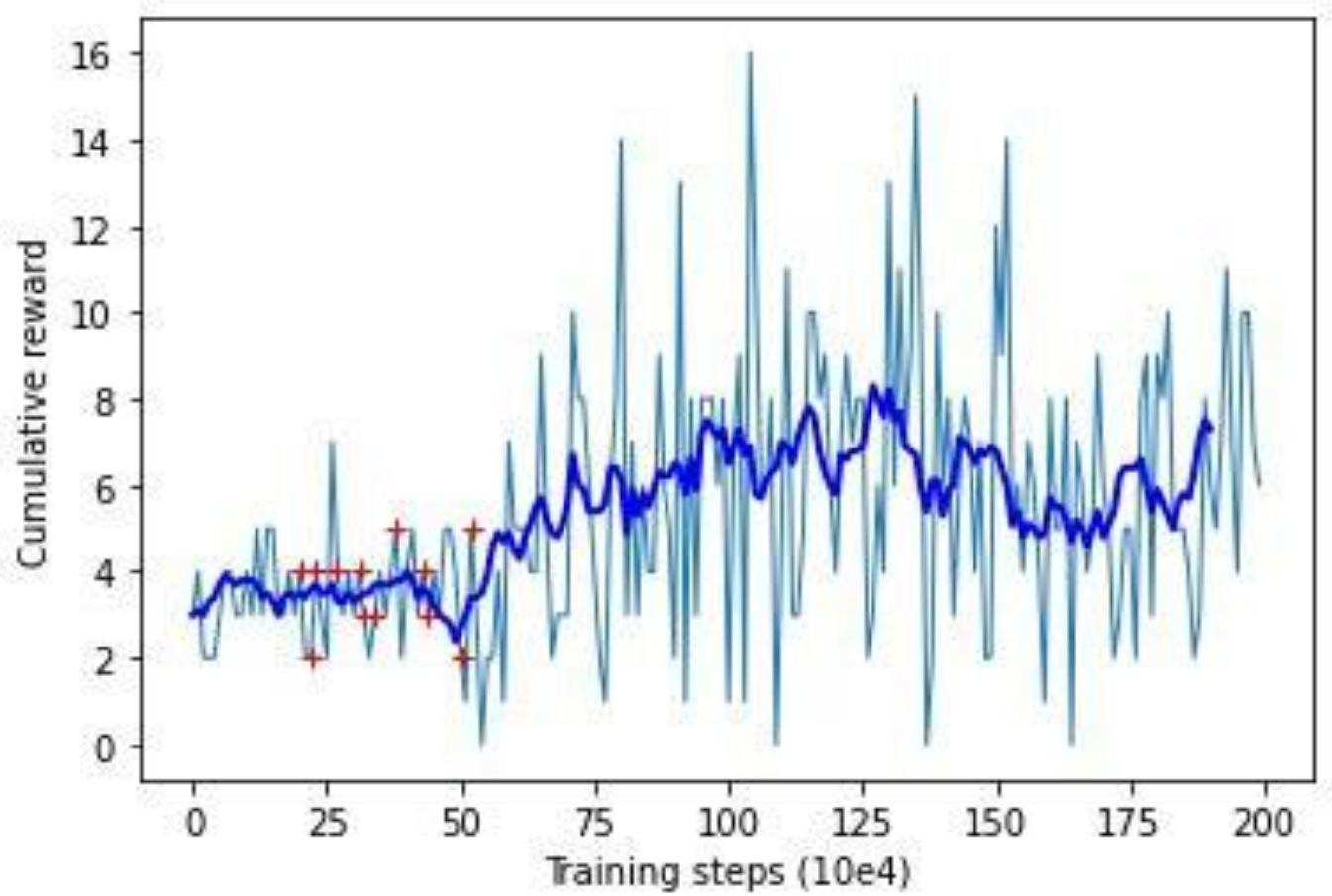




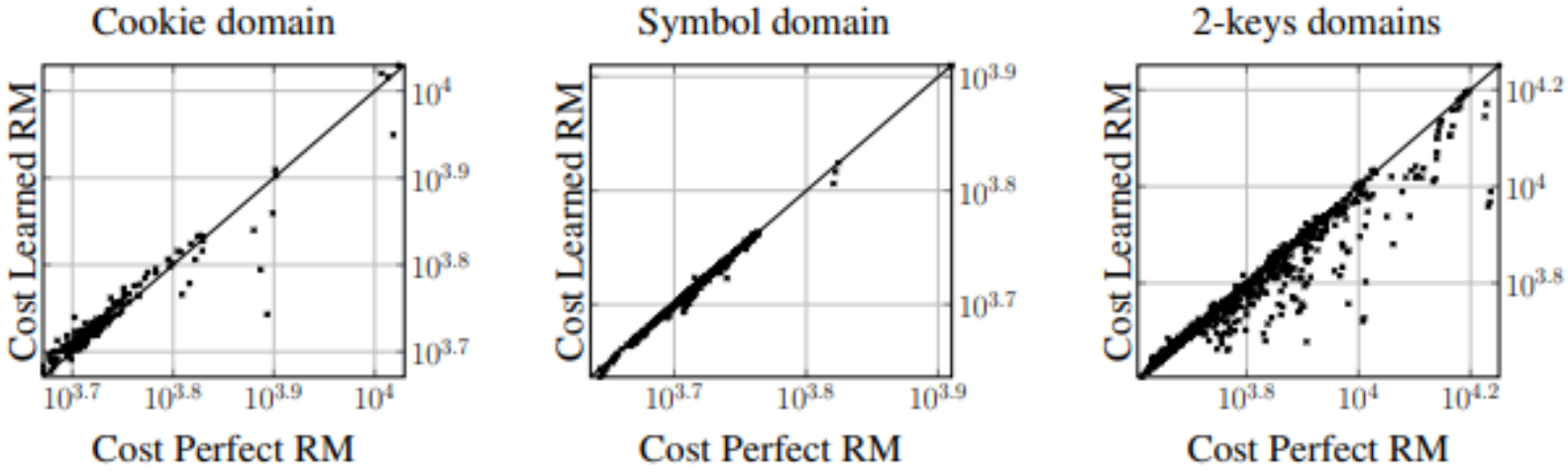
OUR OWN WORLD

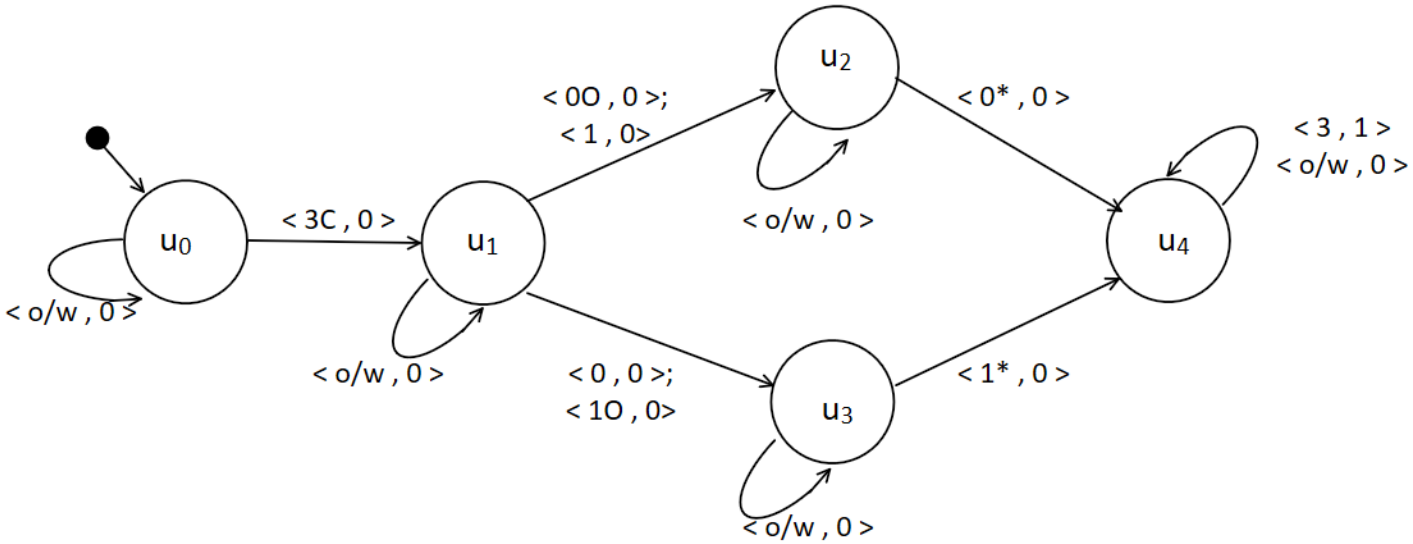
- Waiter world: P=<0,1,2,3,*,C,O>

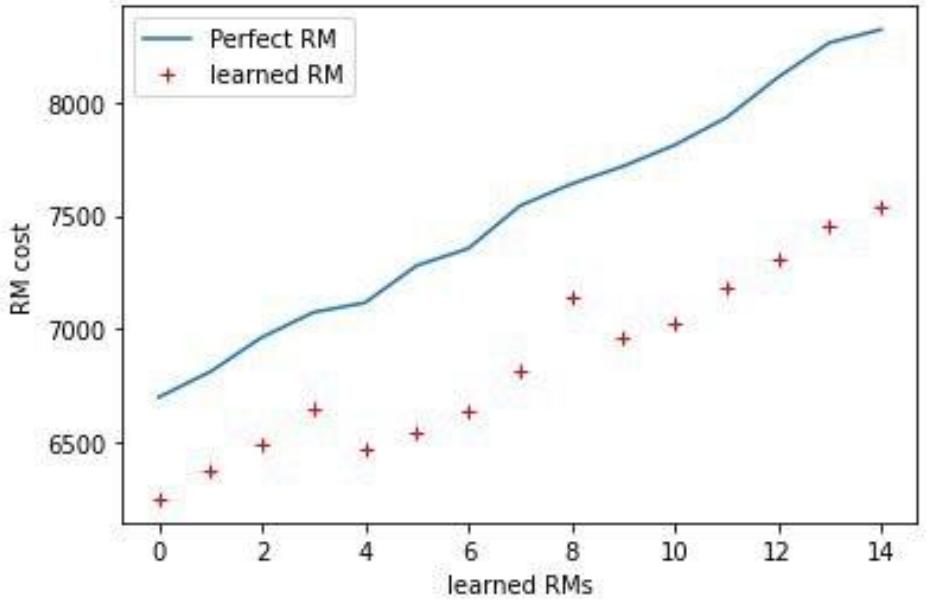




RESULTS


Cookie domain
Symbol domain
2-keys domains
Cost Learned RM
Cost Perfect RM
$10^{3.7}$
$10^{3.8}$
$10^{3.9}$
10^{4}
$10^{4.2}$


u0
u1
u2
u3
u4
< 3C , 0 >
< 0O , 0 >;
< 1 , 0>
< 0* , 0 >
< 3 , 1 >
< o/w , 0 >
< o/w , 0 >
< o/w , 0 >
< o/w , 0 >
< 0 , 0 >;
< 1O , 0>
< 1* , 0 >


Perfect RM
learned RM
RM cost
learned RMs
8000
7500
7000
6500
0
2
4
6
8
10
12
14

CONCLUSIONS

This work showed how RL agents can solve cognitively challenging partially observable tasks.

High level abstraction with RM and low-level policy using deep RL.

The study showed some obstacles regarding:

- Abstraction
- Observability
- Properties of language used for RM construction