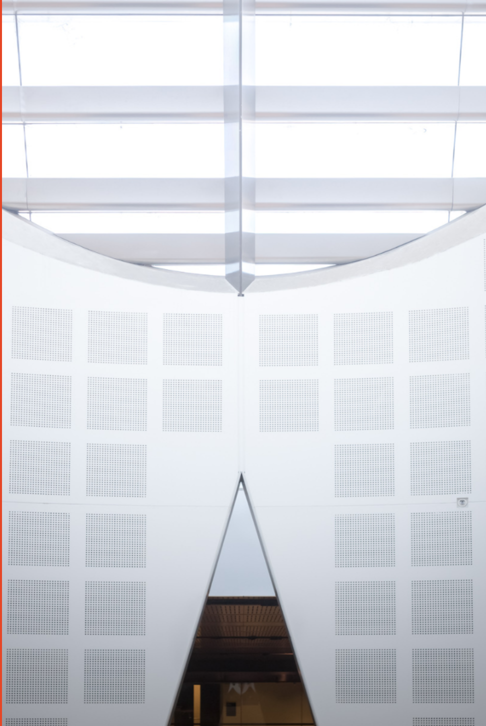
Completing Linear Temporal Logic Objectives Before a Deadline with Reinforcement Learning

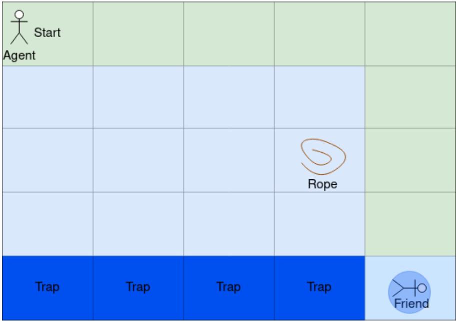
Daniel Braun
(Supervisors: Sasha Rubin, Rafael Oliveira)

November 24, 2021

THE UNIVERSITY OF SYDNEY



Objective-Based Reinforcement Learning



Current Literature: "Take your time, but don't screw it up."
(Infinite Horizon)
Our Problem: "Your friend is drowning, hurry up!"
(Finite Horizon)

Motivation

Examples of objective-based finite horizon problems:

- Manufacturing a high quality product before the factory closes.
- Performing a surgery before the patient bleeds out.
- Getting to school (safely) on time.

Main Contributions

- Created a frozen lake environment suitable for running objective-based reinforcement learning (RL). https://github.com/danbraunai/rl-ltl.
- Introduced *QTRM-learning* to address the objective-based finite horizon problem.
- Analysed the sample-efficiency and update-efficiency of QTRM-learning and similar algorithms.
- Investigated the limitations of reward discounting and step-based penalties for finite horizon problems.

Background: Linear Temporal Logic Objective

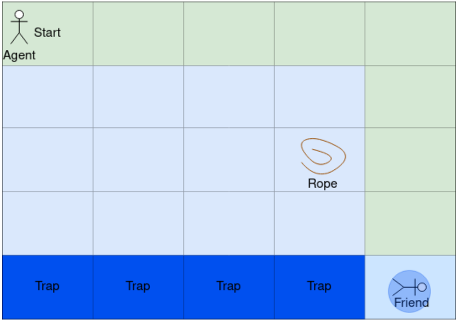


Figure 1: Objective in "finite" LTL[1]: $\varphi = \Diamond(R \land \Diamond F) \land \Box\neg T$

- $\Diamond$ = "eventually"
- $\Box$ = "always"

[1] We define finite LTL to consist of co-safe LTL [1], LTL_f [2, 3] and LDL_f [4]

Background: Reinforcement Learning

Markov Decision Process (MDP):

- Set of **states**.
- Set of **actions** that can be applied in each state.
- A Markov **transition function** mapping states and actions to new states.
- (Optional) **Reward** for taking an action and transitioning to a new state.

Reward-based RL: Learn a *policy* (i.e. a function mapping states to actions) that maximises the expected rewards over an infinite or finite horizon.

An RL agent does not have direct access to the transition function!

Problem statement

Objective-Based Finite Horizon Problem

Given: MDP M without rewards and without direct access to the transition function; deadline H; finite LTL objective φ.
Goal: Learn a policy in M that maximises the probability of achieving φ before H.

Solution Outline

- Reward-based finite horizon methods (QT-learning)
- Objective-based infinite horizon methods (CRM)

$\Rightarrow$

- Objective-based finite horizon method (QTRM-learning)

Solution Outline

- **Reward-based finite horizon methods (QT-learning)**
- Objective-based infinite horizon methods (CRM)

$\Rrightarrow$

- Objective-based finite horizon method (QTRM-learning)

Reward-Based Finite Horizon Q-Learning

1. Initialise Q-values $Q_h(s,a)$ arbitrarily (e.g. to 0) for every state s, action a, and timestep remaining $h \in \{0, 1, ..., H\}$.
2. Take an action and gather experience (h, s, a, s').
3. Update Q-values:

- Value of a in s with h steps remaining

$$Q_h(s,a) \leftarrow Q_h(s,a) + \alpha \left[R(s,a,s') + \max_{a'} Q_{h-1}(s',a') - Q_h(s,a) \right]$$

- Learning rate
- Reward of transition
- Value of best action in s' with $h-1$ steps remaining

Reward-Based Finite Horizon QT-Learning

QT-learning: Environment transitions are independent of the timestep [5].

1. Initialise Q-values $Q_h(s, a)$ arbitrarily (e.g. to 0) for every state s, action a, and timestep remaining $h \in \{0, 1, ..., H\}$
2. Take an action and gather experience (h, s, a, s')
3. Update Q-values **for all timesteps remaining $\boldsymbol{\bar{h} \in \{1, 2, ..., H\}}$ simultaneously**:

$$Q_{\bar{\boldsymbol{h}}}(s,a) \leftarrow Q_{\bar{\boldsymbol{h}}}(s,a) + \alpha\left[R(s,a,s') + \max_{a'} Q_{\bar{\boldsymbol{h}}-\mathbf{1}}(s',a') - Q_{\bar{\boldsymbol{h}}}(s,a)\right]$$

Solution Outline

- Reward-based finite horizon methods (QT-learning)
- **Objective-based infinite horizon methods (CRM)**

$\Rightarrow$

- Objective-based finite horizon method (QTRM-learning)

Reward Machines

- MDP without rewards
- Finite LTL objective

$\Rightarrow$

- Reward machine [6, 7, 8]

Reward Machines

A reward machine (RM) contains:

- Set of **states** U representing different stages of the objective (e.g. when the agent has or doesn't have the rope).
- A function δ_u specifying **transitions** between RM states given environment transitions.
- A function δ_r specifying **rewards** given RM states and environment transitions.

If $\delta_r = 1$ when the objective is completed, and 0 otherwise, then a policy which maximises rewards also maximises the satisfaction probability of φ.

Objective-Based Infinite Horizon Q-Learning

Counterfactual Experience for Reward Machines (CRM) [8]:

1. Create RM from MDP (without rewards) and finite LTL objective.
2. Initialise Q-values $Q^u(s,a)$ arbitrarily (e.g. to 0) for every RM state u, environment state s, and action a.
3. Take an action and gather experience (u, s, a, u', s').
4. Update Q-values **for all RM states $\bar{\boldsymbol{u}}$ simultaneously**:

$$Q^{\bar{\boldsymbol{u}}}(s,a) \leftarrow Q^{\bar{\boldsymbol{u}}}(s,a) + \alpha\left[\delta_r(\bar{\boldsymbol{u}}, s, a, s') + \max_{a'} Q^{\bar{\boldsymbol{u}}'}(s', a') - Q^{\bar{\boldsymbol{u}}}(s,a)\right]$$

Solution Outline

- Reward-based finite horizon methods (QT-learning)
- Objective-based infinite horizon methods (CRM)

$\Rightarrow$

- **Objective-based finite horizon method (QTRM-learning)**

Objective-Based Finite Horizon Q-Learning

Q-Learning for Timesteps and Reward Machines (QTRM-learning):

1. Create RM from MDP (without rewards) and finite LTL objective.
2. Initialise Q-values $Q_h^u(s, a)$ arbitrarily (e.g. to 0) for every RM state u, environment state s, action a, and timestep remaining $h \in \{0, 1, ..., H\}$.
3. Take an action and gather experience (h, u, s, a, u', s').
4. Update Q-values **for all RM states $\bar{\boldsymbol{u}}$ and timesteps remaining $\bar{\boldsymbol{h}} \in \{\mathbf{1}, \mathbf{2}, ..., \boldsymbol{H}\}$ simultaneously**:

$$Q_{\bar{\boldsymbol{h}}}^{\bar{\boldsymbol{u}}}(s, a) \leftarrow Q_{\bar{\boldsymbol{h}}}^{\bar{\boldsymbol{u}}}(s, a) + \alpha \Big[\delta_r(\bar{\boldsymbol{u}}, s, a, s') + \max_{a'} Q_{\bar{\boldsymbol{h}}-\mathbf{1}}^{\bar{\boldsymbol{u}}'}(s', a') - Q_{\bar{\boldsymbol{h}}}^{\bar{\boldsymbol{u}}}(s, a)\Big]$$

Sample Efficiency

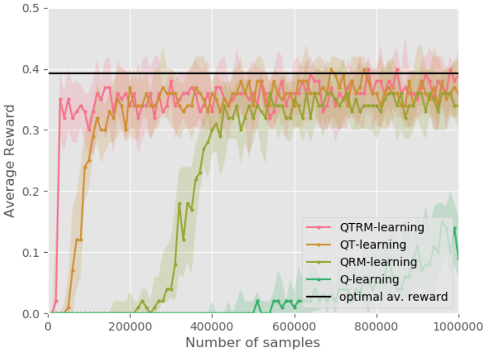


Figure 2: Performance vs number of samples for task with several ropes to be collected in order and horizon $H = 15$.

Discussion

- Same techniques can be applied to RL that uses function approximation [see 8, for infinite horizon].
- *Reward shaping* using RM states can reduce reward sparsity.

Future Work

- Test on different domains (e.g. continuous state spaces, more varied objectives).
- Updating only on a subset of counterfactual experiences.
- Handle the case where environment transitions depend on the RM state.
- Test methods in a real game of frisbee on ice!

Thanks for listening!

Questions?

Figure 3: Man playing frisbee on a frozen lake for some reason [9]

References I

[1] Orna Kupferman and Moshe Y Vardi. “Model checking of safety properties”. In: *Formal Methods in System Design* 19 (3 2001), pp. 291–314.

[2] Thomas Wilke. “Classifying discrete temporal properties”. In: *Annual symposium on theoretical aspects of computer science* (1999), pp. 32–46.

[3] Alfredo Gabaldon. “Precondition Control and the Progression Algorithm.”. In: *ICAPS* (2004), pp. 23–32.

[4] Giuseppe De Giacomo and Moshe Y Vardi. “Linear temporal logic and linear dynamic logic on finite traces”. In: *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence* (2013), pp. 854–860.

[5] Daishi Harada. “Reinforcement learning with time”. In: *AAAI/IAAI* (1997), pp. 577–582.

[6] Rodrigo Toro Icarte et al. “Teaching multiple tasks to an RL agent using LTL”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (2018), pp. 452–461.

[7] Alberto Camacho and Sheila A McIlraith. “Learning interpretable models expressed in linear temporal logic”. In: *Proceedings of the International Conference on Automated Planning and Scheduling* 29 (2019), pp. 621–630.

References II

[8] Rodrigo Toro Icarte et al. "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning". In: *arXiv preprint arXiv:2010.03950* (2020).

[9] Rob McLeod. *Man catching frisbee on ice*. URL: https://frisbeerob.gumlet.io/wp-content/uploads/2016/12/Silver_Skate_Frisbee_Rob_008.jpg?compress=true&quality=80&w=940&dpr=1.0.

Appendix

Some additional slides placed below...

Sample Efficiency Zoomed Out

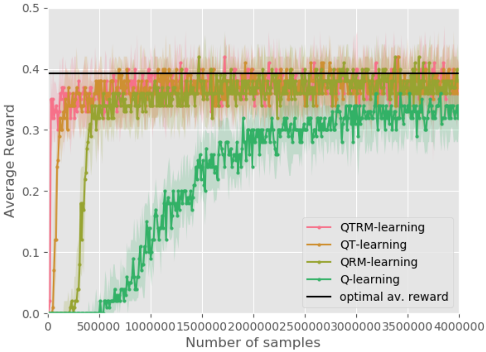


Figure 4: Performance vs number of samples for task with several ropes to be collected in order and horizon $H = 15$.

Update Efficiency

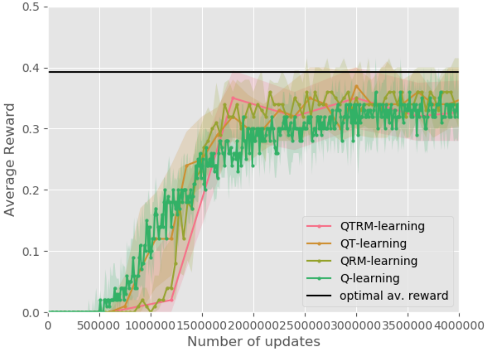
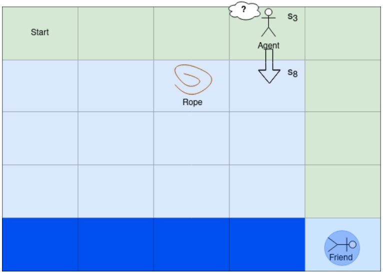


Figure 5: Performance vs number of updates for task with several ropes to be collected in order. Shows that the counterfactual experiences in QTRM-learning are not necessarily less effective than true experiences.

QTRM-learning Example

Suppose $H = 10$, $h = 5$, and in RM state u_0 (i.e. no rope).



Actual experience: $(h, u, s, a, u', s') = (5, u_0, s_3, \downarrow, u_0, s_8)$

QTRM Updates: $$\{(\mathbf{10}, u_0, s_3, \downarrow, u_0, s_8), (\mathbf{9}, u_0, s_3, \downarrow, u_0, s_8), \ldots, \\ (\mathbf{10}, \boldsymbol{u_1}, s_3, \downarrow, \boldsymbol{u_1}, s_8), (\mathbf{9}, \boldsymbol{u_1}, s_3, \downarrow, \boldsymbol{u_1}, s_8), \ldots\}$$

Infinite Horizon Optimal Q-Values

After convergence, *optimal* Q-values $Q^*(s,a)$ satisfy:

- Sum over all possible next states
- Optimal value of a in s

$$Q^*(s,a) = \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \max_{a'} Q^*(s',a') \right]$$

- Probability of transition to s'
- Reward of transition (0 if not goal)
- Optimal value of best action in s' (equivalent to $V^*(s')$)

Infinite Horizon Optimal Policy

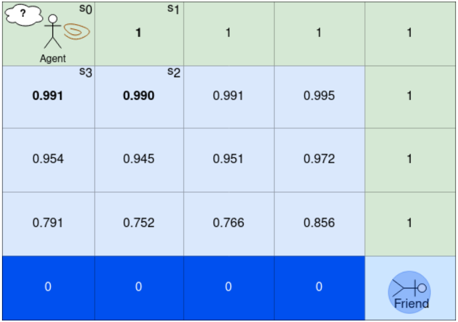


Figure 6: Cell values indicate the optimal values ($\max_a Q^*(s,a)$ or $V^*(s)$)

$$Q^*(s_0, \rightarrow) = 1,\ Q^*(s_0, \searrow) = 0.990,\ Q^*(s_0, \downarrow) = 0.991$$

Finite Horizon Optimal Policy

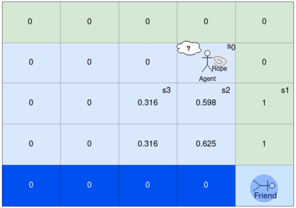


Figure 7: Optimal values with 2 steps left ($\max_a Q_2^*(s, a)$ or $V_2^*(s)$)

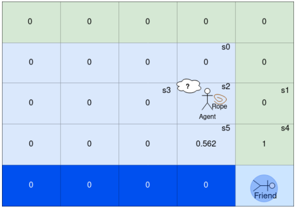


Figure 8: Optimal values with 1 steps left ($\max_a Q_1^*(s, a)$ or $V_1^*(s)$)

$$Q_3^*(s_0, \searrow) = 1,$$
$$Q_3^*(s_0, \downarrow) = 0.598,$$
$$Q_3^*(s_0, \swarrow) = 0.316$$

$$Q_2^*(s_2, \searrow) = 1,$$
$$Q_2^*(s_2, \downarrow) = 0.562,$$
$$Q_2^*(s_2, \swarrow) = 0$$